THE
ADA COMPILER VALIDATION CAPABILITY
(ACVC)
VERSION 2.0.1
USER'S GUIDE

6 March 1996

Prepared for:

Ada 9X Project : ACVC Program

Prepared by:

Science Applications International Corporation
10770 Wateridge Circle
San Diego, CA 92121

Contents

Typographic Conventions

ACVC Test names are capitalized:

BA1101A

Expected output is quoted in small capitals:

"PASSED"

File names are displayed in sanserif with small caps:

C39006F0.ADA

Text that is expected to appear as part of a test file is displayed in Courier:

```
example of some test text
```

Section 1. Introduction

The Ada Compiler Validation Capability (ACVC) is the official test method used to check conformity of an Ada implementation with the Ada programming language standard (ANSI/ISO/IEC 8652:1995, FIPS PUB 119-1). This User's Guide is part of the ACVC and is distributed with the test programs and testing support packages. It explains the contents and use of the ACVC test suite.

This version of the ACVC is 2.0.1. ACVC versions based on ANSI/MIL-STD-1815A-1983, ISO/8652:1987 (Ada 83) were numbered 1.x where x ranged from 1 to 11. ACVC versions numbered 2.0, 2.0.1, and 2.1 based on ANSI/ISO/IEC 8652:1995 (Ada95) are planned. ACVC 2.0.1 contains test programs to check for conformity to language features shared between Ada83 and Ada95. It contains about half of the test programs that will be written to check for conformity to new language features defined in [Ada95]. ACVC 2.1 will fully cover these features. Subsequent maintenance or enhancement versions of the suite, if they are required, will be numbered 2.2, etc.

The User's Guide is intended to be used by compiler implementers, software developers who maintain a version of the ACVC as a maintenance control tool, and third-party testers (e.g., Ada Validation Facilities).

Section 3 of the User's Guide for ACVC 2.0.1 describes the configuration of the ACVC, including a description of the ACVC software and delivery files. Section 4 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 ACVC 2.0.1 release.

Refer to Sections 1.1 and 4.7 for the definition of an acceptable result and the rules for grading ACVC 2.0.1 test program results. Section 4.8.2 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.

This guide does not discuss specific requirements on processing of the ACVC test suite or submission and grading of results that an Ada Validation Facility (AVF) may impose. These are discussed in [Pro95].

The ACVC test suite is normally available from AVFs or from the National Technical Information System (NTIS).

1.1 Definition of Terms

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

ACVC Implementer's Guide (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 implementation : An Ada compilation system, including any required run-time support software, together with its host and target computer systems.

Ada Joint Program Office (AJPO): The part of the certification body that provides policy and guidance for the Ada certification system.

Ada Maintenance Organization (AMO): The part of the certification body that maintains the ACVC.

Ada programming language : The language defined by reference [Ada95].

Ada Validation Facility (AVF) : The part of the certification body that carries out the procedures required to perform validation testing of an Ada implementation.

Ada Validation Organization (AVO) : The part of the certification body that provides technical guidance for operations of the Ada certification system.

Certification Body : The organizations (AJPO, AMO, AVO, AVFs) collectively responsible for defining and implementing Ada validation policy, including production and maintenance of the ACVC tests, and award of Ada validation certificates.

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

Conforming implementation : An implementation that produces an acceptable result for every applicable test. Any deviation constitutes a non-conformity. [See section 4.7, "Grading Test Results", for the implication of core test failures in ACVC 2.0.1.]

Core language : Sections 2-13 and Annexes A, B, and J of [Ada95]. All implementations are required to implement the core language. Validation tests check to determine whether core language features have been correctly implemented.

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

Foundation Code : Packages 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.

Special Needs Annex : One of annexes C through H of [Ada95]. Validation against one or more special needs annexes is optional. There are sets of validation tests that apply to each of the special needs annexes; the appropriate set of tests must be correctly processed (in addition to all of the core language tests) for an implementation to be validated for a special needs annex.

Test Objectives Document (TOD) : A document describing the test objectives used for the 451 ACVC tests that focus on Ada95-specific features.

1.2 References

1. [Ada83] ANSI/MIL-STD-1815A-1983, ISO 8652:1987, FIPS 119 Reference Manual for the Ada Programming Language--superseded by ISO-8652:95)

2. [Ada95] ANSI/ISO/IEC 8652:1995, FIPS 119-1 (The Reference Manual for the Ada Programming Language), February 1995

3. [Pro95] Ada Compiler Validation Procedures, Version 4.0, Ada Joint Program Office, January 1995

1.3 ACVC Purpose

The purpose of the ACVC 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 validation is to promote Ada software portability by ensuring consistent processing of Ada language features as prescribed by [Ada95]. ACVC tests use language features in accord with contexts and idioms normally found 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.

A compilation system that correctly processes the ACVC tests is not thereby deemed error-free, nor is it thereby deemed capable of correctly processing all software that is submitted to it.

The ACVC tests do not check performance parameters (e.g., compile-time capacities or run-time speed). They do not check for characteristics such as the presence and effectiveness of compiler optimization. They do not investigate compiler or implementation choices in cases where the standard allows options. Most particularly, the ACVC tests do not guarantee compiler correctness.

[This page intentionally blank]

Section 2. Changes for ACVC 2.0.1

2.1 Improved Technology

Ada95 compiler technology has improved significantly since the release of ACVC 2.0. Whereas at that time, it was not feasible to confidently identify all incompatibilities in the 9XBasic tests, it is now possible to identify most or all of them. Improved technology has also made it possible to identify unintentional errors in the Ada95 tests included in ACVC 2.0.

We continue to invite comments about test correctness. Disputes should be submitted through the normal process (see section 4.8.2). Other comments are welcome; please submit them via e-mail to

acvc-comm@sw-eng.falls-church.va.us

cc: to

ada9x-acvc@sw-eng.falls-church.va.us

2.2 Changes in 9XBasic

All known and reported cases of Ada95 incompatibility in 9XBasic tests were addressed. Tests in 9XBasic are intended to be neutral between Ada83 and Ada95. They should be processed correctly by a developing Ada95 compiler in any state of transition from Ada83 to Ada95. Tests that could not reasonably be made neutral were deleted from ACVC 2.0.1. A list of these tests appears in Appendix A. Tests that could be made neutral were appropriately modified; in almost all cases, some code was deleted to achieve neutrality.

All such changes represent transition issues that will be addressed in ACVC 2.1. Tests that were deleted from 9XBasic may form the basis for ACVC 2.1 tests for the correct implementation of Ada95 rules where rules changed from Ada83. Tests that were modified by deletion of subtests will generally restore the subtest in ACVC 2.1, modified to check for Ada95 rules.

To achieve neutrality between Ada83 and Ada95, some files (subtests) of multifile tests (see section 3.4.3) were deleted from ACVC 2.0.1. In general, the file names were not changed, resulting in broken sequences of file names. Deleted files are identified in the accompanying ACVC 2.0.1 release notes. Many of these files will be restored, modified to test Ada95 rules, to ACVC 2.1.

2.3 Changes in Ada95 Tests

All known and reported errors in Ada95 tests were corrected for ACVC 2.0.1. Three tests were determined to be significantly flawed and were deleted from ACVC 2.0.1, to be rewritten for ACVC 2.1.

No new Ada95 tests were added to ACVC 2.0.1.

[This page intentionally blank]

Section 3. Configuration Information

3.1 Introduction

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

ACVC 2.0.1 incorporates 9XBasic, a set of 2861 tests covering Ada95 features that are (upward) compatible with Ada83. The delivery has been structured to maintain as much conformity with the previous deliveries as possible. For example, the delivery structure of 9XBasic is substantially the same as that of previous ACVC suites. Nevertheless, there are differences that reflect new requirements. For example, the naming scheme used in the Ada95 tests is slightly different from that used in 9XBasic.

The baseline for ACVC 2.0 and ACVC 2.0.1 was ACVC 1.12, which was developed and released for public comment but was never required for validation. 9XBasic consists of approximately 75% of the ACVC 1.12 tests and provides the same level of conformance testing as ACVC 1.11. It achieved its smaller size by eliminating tests that were not conformance tests, tests that were substantially duplicative, tests of pathological or non-usage oriented feature combinations, and tests that were not compatible with Ada95.

3.2 Structure

The ACVC 2.0.1 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 ACVC tests). The suite includes tests for the core language and tests for the special needs annexes. There are a total of 3309 tests delivered in 3668 test files. In addition, there are 50 files used to support the installation and execution of tests.

The test suite does not provide tools or scripts that can be used to manage complete test processing, since such tools would normally be site specific.

3.2.1 Physical

The ACVC 2.0.1 test suite consists of 3719 files. Of these, 2 files contain code used to build and run the macro processor support tool, 12 files contain other support code, 1 file contains a list of all delivery contents, and 3704 contain software used to build individual ACVC tests. Note that the number of files containing test software is larger than the number of tests in the ACVC suite because several tests use software included in separate files.

A file name consists of two parts, a name and an extension. When possible, test files use the same name as that of the test they contain. There may be systematic differences in case a test requires software included in multiple files. Files as delivered may also have names that are shorter than the software name because of file name length restrictions (e.g., ENUMCHE rather than ENUMCHECK).

File (and test) names follow conventions that indicate their function in the test suite; naming conventions are explained in section 3.4. The files are organized into distinct directories and subdirectories based on their function in the test suite. The directory organization is explained in section 3.8.

The default delivery format for ACVC is on MS-DOS format high density diskettes; seven diskettes are required. All ACVC files (except as noted later) have been compressed and archived into libraries using PKZip[1]. The libraries have the MS-DOS file extension "`.ZIP`". These files must be decompressed with a compatible decompression utility, as explained in section 4.2.2. PKUnzip[2] or Unzip may be used; Unzip is a shareware decompression utility.

The current ACVC suite is generally available on the AJPO host (sw-eng.falls-church.va.us). Zip archives located there are the same as the archives provided on diskettes, and should be treated exactly as described later (section 4.2).

3.2.2 Logical

The ACVC 2.0.1 suite consists of 3309 tests that check the conformity of an Ada implementation. There are 3250 that check for conformance to the core language and 59 that check for conformance to special needs annexes of [Ada95].

Core tests apply to all implementations. Special needs annex tests are not required for any implementation. Tests for a given special 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 4.5.2. No annex test depends on the implementation of any other annex, except possibly in cases where one annex specifically depends on another in Ada95 (e.g., no test for the Information Processing Annex uses features from any other annex, however Real Time Annex tests depend on Systems Programming Annex features).

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 3.2.4.

Tests are named using conventions that provide limited information about the test. The test naming conventions are explained in section 3.4. 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 3.3.

In addition to test code and foundation code, there is code on which many or all of the 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 4.3.

[1,2] PKZip and PKUnzip are registered trademarks of PKWARE, Inc.

A user might want to focus on a particular language area first by selecting only a subset of the test programs using Appendix A (Version Description) as a guide in the selection of a subset.

3.2.3 9XBasic

The tests included in 9XBasic check only language features that are common to Ada83 and Ada95; they do not use any features unique to Ada95. As noted above, the vast majority of tests included in 9XBasic came unmodified from the ACVC 1.12 suite. Some of these tests have been modified to remove Ada95 incompatibilities.

The 9XBasic tests can be used to verify that a transitional compiler is substantially correct for those (very large) portions of 9X that remain Ada83 compatible.

There are 2861 9XBasic tests in ACVC 2.0.1.

3.2.4 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 a program library 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 3.3). One may think of it as providing utility definitions and routines to a set of tests. Names of foundation units (and therefore names of files containing foundation code) are distinguished as described in Naming Convention, section 3.4.

3.3 Test Classes

Language conformity testing calls for a variety of different kinds of tests. The ACVC tests have been categorized into six different classes, reflecting different testing needs. The test classifications provide an initial indication of the criteria that are used to determine whether a test has been passed or failed. Each ACVC 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.

3.3.1 Class A

Class A tests check for acceptance (compilation) of language constructs that cannot be verified at runtime. They are expected to successfully compile, bind, and execute reporting "PASSED". Any other behavior is a failure.

3.3.2 Class B

Class B tests check that illegal constructs are recognized and are treated as fatal errors. They are not expected to successfully compile, bind, or execute. They are considered to pass if each indicated error in the test is detected. If the test structure is such that a compiler is unable to recover sufficiently to identify all errors, it may be permissible to "split" the test program into separate units for re-processing (see Section 4.3.5 for instructions on modifying tests).

Lines that contain errors are marked "`-- ERROR:`", and generally include a brief description of the illegality.

Ada95 tests may mark some lines as "`-- OK`", indicating that the line must **not** be flagged as an error. Lines so marked are often, but not always, constructs that were errors in Ada83 but are correct in Ada95.

3.3.3 Class C

Class C tests check that verifiable constructs are implemented correctly and check the correct execution of legal Ada programs. 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 fails a class C test 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 an output report or only partially produces one.

The tests CZ1101A, CZ1102A, and CZ1103A are treated separately, as described in section 4.4.2.

3.3.4 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 results of the conditions tested. Some implementations may report errors at compile time for some of them, if the literal numbers exceed compiler limits. Unless a capacity limit is exceeded, an implementation fails a class D test 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 an output report or only partially produces one.

3.3.5 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 of them. The "TENTATIVELY PASSED" message indicates special conditions that must be checked to determine whether the test is passed. A class E test passes if it reports "TENTATIVELY PASSED", and the special conditions noted in the test are satisfied. It may also pass 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 (hangs or crashes), if the reported result is "FAILED", or if they do not produce an output report or only partially produce one.

3.3.6 Class L

Class L tests check that all library unit dependencies within a program are satisfied before the program can be bound and executed, and that circularity among units is detected. These tests are normally expected to compile successfully but not to bind or execute. Some implementations may report errors at compile time. This result is acceptable if the test has flagged potentially illegal constructs with "`-- ERROR:`" and the compiler has identified them. Implementations pass these tests if they do **not** successfully complete the bind phase. Class L tests that run will report "FAILED" (except in rare cases).

3.3.7 Classifications Used in 2.0.1

The 9XBasic part of ACVC 2.0.1 includes tests in classes A, B, C, D, E, and L.

Tests in the Ada95 specific part of ACVC 2.0.1 include tests in classes B, C, and L. These tests may also include files that are foundation code. No 9XBasic tests include foundation code.

There is no separate classification for special needs annex tests. They are classified in the same way as all other tests.

3.4 Naming Convention

This section describes the naming conventions used in ACVC 2.0.1. 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 file <name> is normally the same as the name of the (or a) unit in the file, however, in some cases, the structure of the test requires that the file have a name different from the Ada unit. This section describes name conventions as they apply specifically to files. The application of the conventions to tests is straightforward.

There are two, similar, naming conventions used in ACVC 2.0.1. Tests included in 9XBasic use the naming conventions of previous ACVC versions. New tests based on Ada95 use a slightly different convention. The conventions are consistently distinguishable at the 7th character of the name: names in 9XBasic have a letter in the 7th position; other names have a digit.

3.4.1 9XBasic Naming

A <name> in 9XBasic is composed of seven to nine 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:

Position		
1	Letter	Test class (cf. Section 3.3)
2	Hexadecimal	AIG chapter containing the test objective
3	Hexadecimal	Section within the above AIG chapter
4	Alpha-numeric	Sub-section of the above AIG section
5-6	Decimal	Number of the test objective within the above sub-section
7	Letter	Letter identifier of the sub-objective of the above objective.
8	Alpha-numeric	*optional* - Compilation sequence identifier -- indicates the 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.
9	Character 'M'	*optional* - Main subprogram indication -- indicates the file of a test set that contains the main procedure. This position is used for one file of each multi-file test.

This convention is illustrated in Figure 1.

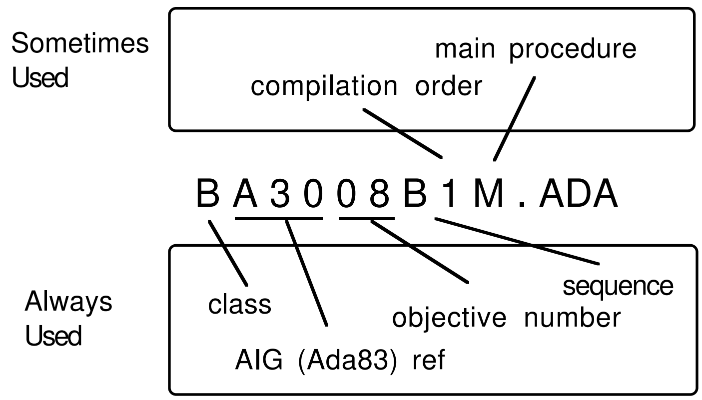

Figure 1. 9XBasic File Name Convention

In 9XBasic, the intended order of compilation of multiple file tests is indicated at character position 8 by a numeral. The first file to be compiled has '0', the second has '1', and so forth. The main subprogram is indicated at character position 9 by an 'M'.

The file <type> is a three character file name extension. Allowable extensions and their meanings are:

- .ADA A file that contains only Ada code. It does not require any processing to prepare it for compilation. It is expected to 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 4.3.2). Once customized, the resulting test must be processed as indicated by its class.

Files not belonging to 9XBasic are easily identified since they use different <type>s.

> Note that 9XBasic tests have **not** been renamed for incorporation into ACVC 2.0.1. Since [Ada95] includes some organizational differences from [Ada83], this means that 9XBasic tests will not, in all cases, correspond to the clause of [Ada95] in which the tested feature is described.

3.4.2 ACVC 2.0.1 Naming

For Ada95 tests, a <name> 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:

Position		
1	Letter	Test class; foundations are marked 'F'
2	Alpha-numeric	Section of [Ada95] describing the feature under test, if other than an 'X'. If an 'X', this test applies to one of the special needs annexes and does not specifically test core features,
3	Alpha-numeric	Core clause or annex letter identifier (whether core or special needs annex)
4	Hexadecimal	Sub-clause (if a core test) or clause (if an annex test)
5	Alpha-numeric	Foundation identifier (alphabetic, unless no foundation is required, in which case a '0')
6-7	Decimal	Sequence number of this test in a series of tests for the same clause; foundation code will have "00" .
8	Alpha-numeric	*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.

This convention is illustrated in Figure 2.

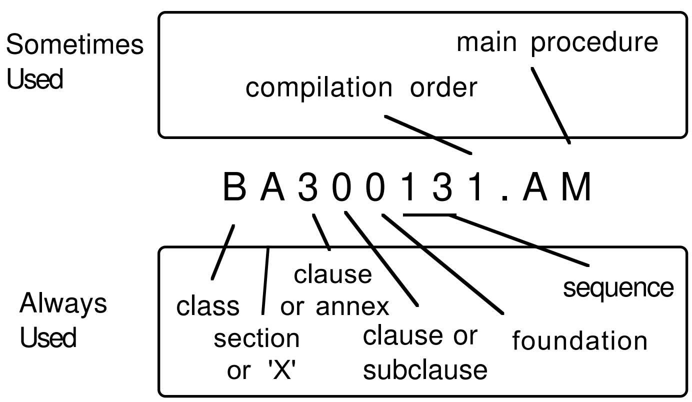

Figure 2. Naming convention in ACVC 2.0.1

The file <type>is a one or two character file name extension. Allowable extensions and their meanings are:

.A A file that contains only Ada code (except for configuration pragmas in the case of some special 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 only Ada code and that contains the main subprogram for a test. This extension is only used for one file of each multi-file 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.]

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, C**123A**xx depends on F**123A**00.

3.4.3 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 related to a single test will be identical. The eighth position will provide a distinguishing alpha-numeric which indicates the suggested or required compilation order. In 9XBasic tests, the ninth position is used to indicate the file that contains the main program. For Ada95 tests, the extension ".AM" indicates the file with the main program.

The convention applied in 9XBasic and consistently applied in Ada95 tests is to name the main subprogram the same as the file, excluding the file extension.

For example, the 9XBasic test, C39006F, is contained in four files, named C39006F0.ADA, C39006F1.ADA, C39006F2.ADA, and C39006F3M.ADA. The main sub-program of the test is contained in C39006F3M.ADA and is named "C390006F3M". The Ada95 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 "C390006".

3.5 Test Program Format

Each test file is composed of a test prologue that documents the test and the test code proper. All prologue content is marked as Ada comments.

The prologue for Ada95 tests is based on that of 9XBasic tests. Tests in 9XBasic are relatively, 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 ACVC 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 all Ada95 tests but not included in 9XBasic tests.

SPECIAL REQUIREMENTS - *optional* - Included if the test has any special requirements for processing. Normally, this section will be found only in special 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.

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

All tests have the next line after the disclaimer marked "`--*`". The Ada95 tests have the line after the last prologue line and 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.

3.6 General Standards

Tests for 9XBasic and for Ada95 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 sub-objectives; in other cases, sub-objectives are checked in separate tests.

Tests in 9XBasic use only the basic 55-character set (26 capital letters, 10 digits, and 19 punctuation marks). Unless there is a specific testing requirement, numeric values are in the range (-2048..2047), which can be represented in 12 bits. Numeric values are often in the range (-128..127). Tests new to ACVC 2.0 and ACVC 2.0.1 use both upper and lower case letters and may use larger numeric values (but within -65536..65535 except in rare cases).

In 9XBasic, tests use as few Ada features as necessary to write a self-checking executable test that can be read and modified. Tests for Ada95 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 tests for Ada95, Ada reserved words are 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.

Executable Ada95 tests visually separate those test elements which focus on conformance issues from those which 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 (user oriented) 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".

3.7 Test Structure

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

```
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. 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 in enclosed in a declare block.

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

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 ACVC 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.

B class are 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, legality rules are invoked in specific ways and expected compiler results are documented as comments to the source code. Code that is legal is included in B class tests as well as code that is illegal. 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.

3.8 Delivery Directory Structure

The delivery of ACVC tests is structured into a directory tree that reflects the organization of the test suite and support code. See Fig. 3.

The top level directory contains two subdirectories: support contains all support packages (Report, ImpDef, TCTouch) and the source code for all test processing tools (Macro expander); Tests contains the files for all tests, separated into 9XBasic and Ada95. Each of the 9XBasic and Ada95 subdirectories contains subdirectories for each class of test, and each of those subdirectories contains subdirectories for each ARM section. Subdirectories that would be empty are not stubbed (e.g., there are no E class tests in Ada95, therefore there is no ETests subdirectory in Ada95).

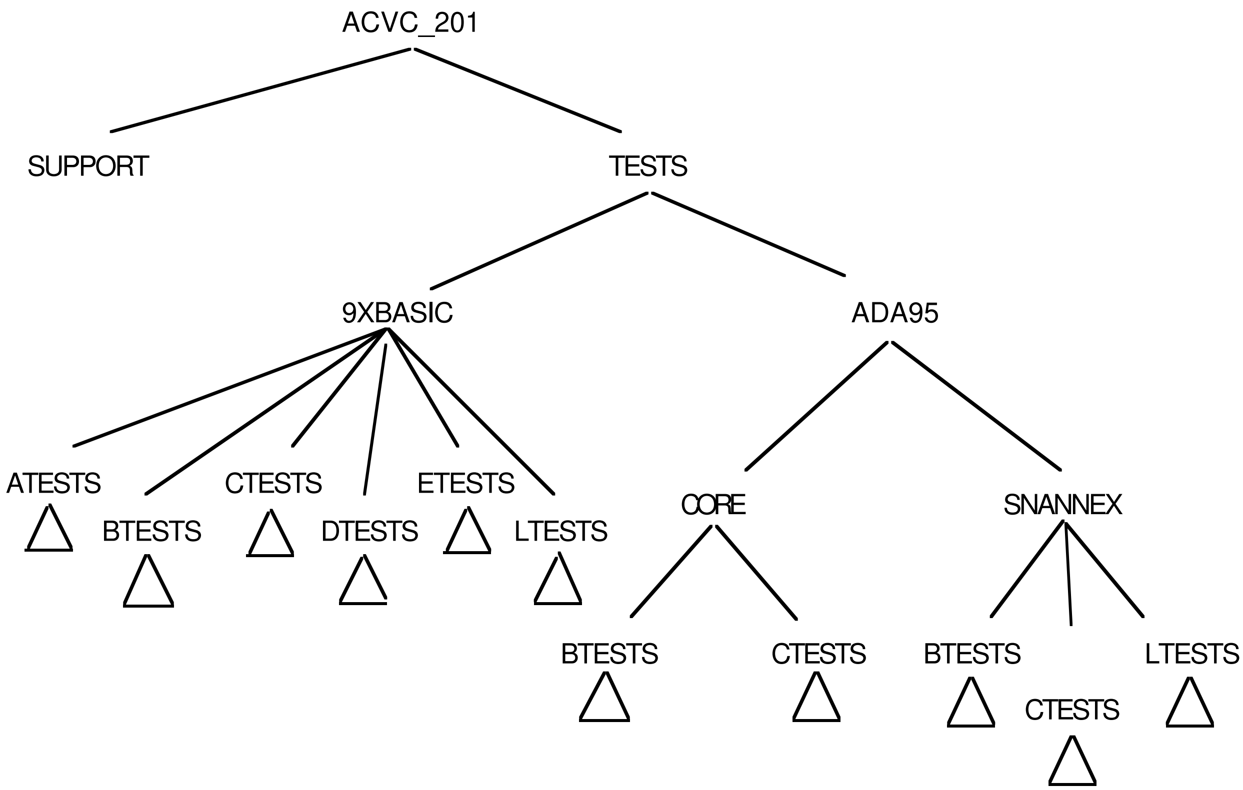

Figure 3. Delivery Directory Structure

3.9 File Format

ACVC 2.0.1 is delivered on 3.5" high-density, MS-DOS formatted diskettes. To conserve space, all files (including test files, foundation files, and support files) except two delivery notes files have been compressed. Decompressed files (see section 4.2.2) use only ASCII characters with <CR><LF> line terminators. No formatting control characters, rulers or other information intended for word processors or editors is included in the files.

[This page intentionally blank]

Section 4 Using The ACVC

4.1 Introduction

There are eight major steps involved in using the ACVC test suite; two of them are not always required. The steps are: installing the software, tailoring the software, processing the support files, establishing command scripts, processing the ACVC 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 ACVC release. Each step is explained in the following sections. The flow from one to the next is illustrated in figure 4.

4.2 Installation of the ACVC Test Suite

The ACVC test suite must be unloaded from the delivery medium before it can be customized for an implementation, run, and graded.

4.2.1 Contents of the ACVC Delivery

The delivery consists of seven diskettes. Each disk contains one or more archives. Each archive contains compressed versions of ACVC software (test, foundation, and/or support code) structured into a directory tree. Files must be extracted from the archives. Archive contents are described later in this section.

Disk 1 also contains the files

ACVC201.LST

TSTTESTS.DAT

CATEGORY.LST

UNZIP.EXE

The file named ACVC201.LST has the names of all test files (including foundations) and support files in ACVC 2.0.1. The file named TSTTESTS.DAT lists all .tst tests. It may be used or modified for use with the macro expander (see section 4.3.2). The file named CATEGORY.LST lists all tests that are included in the ACVC 2.0.1 feature areas (see Appendix B). None are compressed; all may be read in an editor, listed to the screen of a PC, or printed to standard print output. All are ASCII files with <CR><LF> line terminators.

The file named UNZIP.EXE is a decompression tool that may be used to extract all archives.

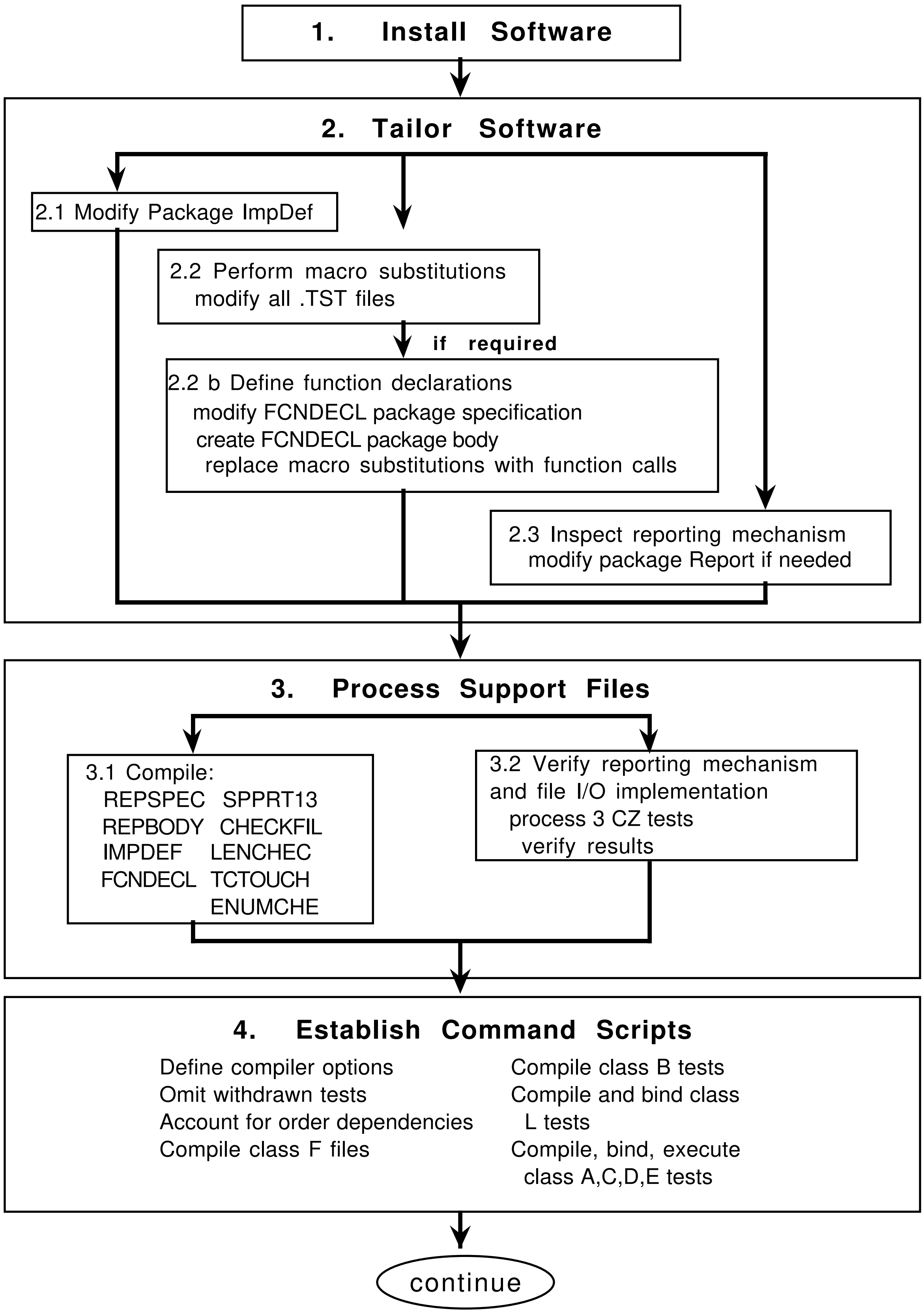

Figure 4. Using the ACVC

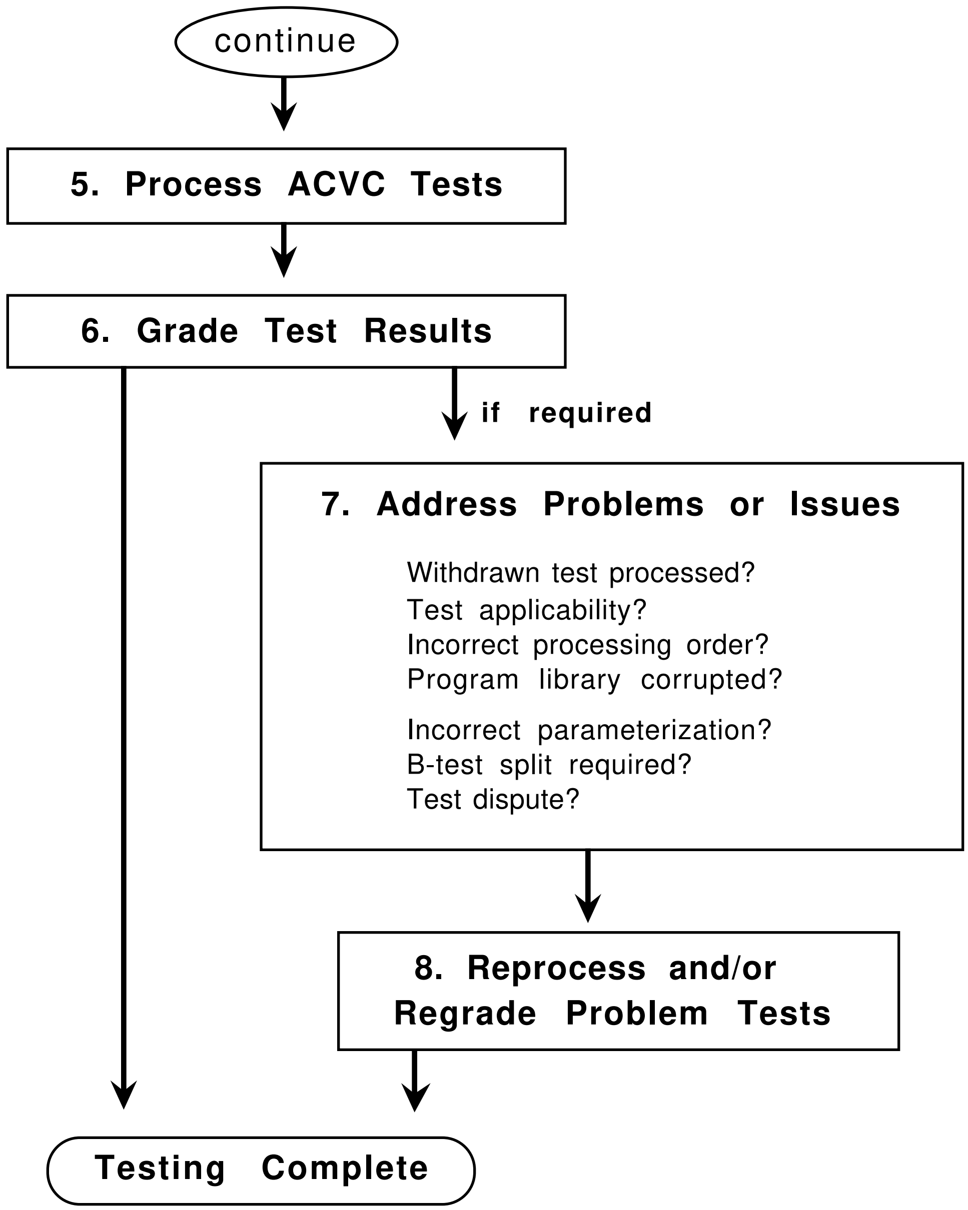

Figure 4, cont. Using the ACVC

4.2.2 Guide to Decompressing Files

[This guide is one approach to decompressing files. Sophisticated users may prefer to implement different approaches.]

All ACVC files other than those noted in section 4.2.1 have been zipped (compressed) into zip-files (compressed archives) that have the MS-DOS file extension ".ZIP". The PC utility PKZip was used to compress them. They must be decompressed before they can be further processed.

The file UNZIP.EXE, included on disk 1, is a compatible decompression utility which may be used to decompress them. All ACVC 2.0.1 files may be decompressed using the following steps. Approximately 35 MB of free space on a DOS machine hard drive will be required to accomplish the decompression using this technique.

Copy UNZIP.EXE to a hard disk. For each library (file with .ZIP extension), copy it to the hard disk. Decompress it using the default settings (no options). For example, if the archive name is ACVC1.ZIP, type

unzip acvc1

Note that the analogous command for PKUnzip is

pkunzip acvc1 -d

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

As it decompresses files, unzip (or PKUnzip) will restore the directory structure of the files, creating all needed subdirectories. When all archives have been extracted, the following subdirectories will have been created:

\ACVC_201\Tests\9XBasic
\ACVC_201\Tests\9XBasic\ATests
\ACVC_201\Tests\9XBasic\BTests
\ACVC_201\Tests\9XBasic\BTests\B2
\ACVC_201\Tests\9XBasic\BTests\B3
\ACVC_201\Tests\9XBasic\BTests\B4
\ACVC_201\Tests\9XBasic\BTests\B5
\ACVC_201\Tests\9XBasic\BTests\B6
\ACVC_201\Tests\9XBasic\BTests\B7
\ACVC_201\Tests\9XBasic\BTests\B8
\ACVC_201\Tests\9XBasic\BTests\B9
\ACVC_201\Tests\9XBasic\BTests\BA
\ACVC_201\Tests\9XBasic\BTests\BB
\ACVC_201\Tests\9XBasic\BTests\BC
\ACVC_201\Tests\9XBasic\BTests\BD
\ACVC_201\Tests\9XBasic\BTests\BE
\ACVC_201\Tests\9XBasic\CTests
\ACVC_201\Tests\9XBasic\CTests\C2
\ACVC_201\Tests\9XBasic\CTests\C3
\ACVC_201\Tests\9XBasic\CTests\C4
\ACVC_201\Tests\9XBasic\CTests\C5
\ACVC_201\Tests\9XBasic\CTests\C6
\ACVC_201\Tests\9XBasic\CTests\C7
\ACVC_201\Tests\9XBasic\CTests\C8
\ACVC_201\Tests\9XBasic\CTests\C9
\ACVC_201\Tests\9XBasic\CTests\CA
\ACVC_201\Tests\9XBasic\CTests\CB
\ACVC_201\Tests\9XBasic\CTests\CC
\ACVC_201\Tests\9XBasic\CTests\CD
\ACVC_201\Tests\9XBasic\CTests\CE
\ACVC_201\Tests\9XBasic\DTests
\ACVC_201\Tests\9XBasic\ETests
\ACVC_201\Tests\9XBasic\LTests
\ACVC_201\Tests\Ada95\core
\ACVC_201\Tests\Ada95\core\BTests
\ACVC_201\Tests\Ada95\core\BTests\B3
\ACVC_201\Tests\Ada95\core\BTests\B4
\ACVC_201\Tests\Ada95\core\BTests\B6
\ACVC_201\Tests\Ada95\core\BTests\B7
\ACVC_201\Tests\Ada95\core\BTests\B8
\ACVC_201\Tests\Ada95\core\BTests\B9
\ACVC_201\Tests\Ada95\core\BTests\BA
\ACVC_201\Tests\Ada95\core\BTests\BB
\ACVC_201\Tests\Ada95\core\BTests\BC
\ACVC_201\Tests\Ada95\core\BTests\BD
\ACVC_201\Tests\Ada95\core\BTests\AnxA
\ACVC_201\Tests\Ada95\core\BTests\AnxB
\ACVC_201\Tests\Ada95\core\CTests
\ACVC_201\Tests\Ada95\core\CTests\C3
\ACVC_201\Tests\Ada95\core\CTests\C4
\ACVC_201\Tests\Ada95\core\CTests\C6
\ACVC_201\Tests\Ada95\core\CTests\C7
\ACVC_201\Tests\Ada95\core\CTests\C8
\ACVC_201\Tests\Ada95\core\CTests\C9
\ACVC_201\Tests\Ada95\core\CTests\CA
\ACVC_201\Tests\Ada95\core\CTests\CB

\ACVC_201\Tests\Ada95\core\CTests\CC	\ACVC_201\Tests\Ada95\SNAnnex\BTests\AnxF
\ACVC_201\Tests\Ada95\core\CTests\CD	\ACVC_201\Tests\Ada95\SNAnnex\BTests\AnxG
\ACVC_201\Tests\Ada95\core\CTests\CE	\ACVC_201\Tests\Ada95\SNAnnex\CTests\AnxC
\ACVC_201\Tests\Ada95\core\CTests\AnxA	\ACVC_201\Tests\Ada95\SNAnnex\CTests\AnxD
\ACVC_201\Tests\Ada95\core\CTests\AnxB	\ACVC_201\Tests\Ada95\SNAnnex\CTests\AnxE
\ACVC_201\Tests\Ada95\SNAnnex	\ACVC_201\Tests\Ada95\SNAnnex\CTests\AnxF
\ACVC_201\Tests\Ada95\SNAnnex\BTests\AnxC	\ACVC_201\Tests\Ada95\SNAnnex\CTests\AnxG
\ACVC_201\Tests\Ada95\SNAnnex\BTests\AnxD	\ACVC_201\Tests\Ada95\SNAnnex\LTests\AnxD
\ACVC_201\Tests\Ada95\SNAnnex\BTests\AnxE	\ACVC_201\support

Some users may prefer to work with ACVC 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. Alternatively, if ***no*** option is used with PKUnzip, all decompressed files go into the local directory. In other words, none of the above subdirectories will be created. Since there are too many ACVC 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 \ACVC**) and unzip all archives there.

The files ACVC201.LST and CATEGORY.LST may be copied from Disk 1 to your preferred location.

4.3 Tailoring the ACVC Test Suite

There are some files in the delivery that require modification before ACVC 2.0.1 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. ImpDef is a package that is new to the suite and all users will have to do this modification. The MACROS.DFS file must similarly be edited to include values suitable for testing. This file is slightly different from previous ACVC suites, so all users will have to modify it, but most changes can be retained from previous versions. All .TST files (including package Spprt13 (SPPRT13S.TST)) must have their macro symbols replaced by implementation specific values. A body for FcnDecl (FCNDECL.ADA) must be provided if necessary. Finally, package Report (REPBODY.ADA) must be modified if necessary; again, previous modifications can generally be carried forward. The required customization is described in the following sections.

4.3.1 ImpDef Customization

All implementations *must* customize IMPDEF.A for ACVC 2.0.1 unless they wish to rely on the defaults provided. ImpDef *must* be part of the program library whenever a test that depends on it is processed.

ACVC tests use the entities in ImpDef to control test execution. Most 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 ACVC 2.0.1. It is are similar to MACRO.DFS in that it must be customized with numbers specific to an implementation and ACVC tests will rely on these numbers. ImpDef is different in the following respects:

- Defaults are provided. Implementations may choose to rely on the default values and subprograms, so no customization would be necessary.
- Some implementations may choose to provide bodies for one procedure and/or one function. Bodies so provided must satisfy requirements stated in ImpDef.
- It is not used for macro expansion of tests. Instead, ImpDef must be available at compile time (often, this means "compiled into the program library") for tests that rely on it.

Specific instructions for the values required by ImpDef are included in IMPDEF.A. A copy is included in Appendix C.

4.3.2 Macro Defs Customization

The details of MACRO.DFS have changed from ACVC 1.11 to ACVC 2.0.1. A MACRO.DFS that worked with ACVC 1.11 will have to be modified for ACVC 2.0.1.

Tests that are type ".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 suitable for compilation.

The required substitutions can be done automatically by using the Macrosub program distributed with the ACVC. 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. A sample MACRO.DFS is included with the ACVC, and is included in Appendix D; 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 sample, using the delivery directory location of files, is included on disk 1.
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.

4.3.3 Package SPPRT13 and Function 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 which 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.

The version of SPPRT13 distributed with ACVC 2.0.1 is slightly different from the version distributed with ACVC 1.11. A body is not required for this package (and would, therefore, be illegal in Ada95).

> **All implementations should verify that package SPPRT13 can be properly customized using the macro substitution technique. Note that in Ada95, a body for SPPRT13 is illegal.**

The specification for package FCNDECL is in the file FCNDECL.ADA. SPPRT13 depends on FCNDECL (in a context clause that both "with"s it and "use"s it). As supplied with the ACVC, 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 AVF (and, if necessary, the AVO) before use in a validation attempt.

4.3.4 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 AVF (and, if necessary, the AVO) before use in a validation attempt.

4.3.5 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 of executable tests, as for example when implementation capacity limitations are 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 a length clause (to alter the default size of a collection), or by the addition of an Pragma Elaborate (to force an elaboration order the test had assumed).

All splits and modifications must be approved in advance by the AVF (and, if necessary, the AVO) before they are used in a validation attempt. 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 alpha-numeric character to the name of the original test. When possible, line numbers of the original test should be preserved in the modification.

For all modified tests, the following procedure must be used: the test as distributed (and customized, if required) must be submitted to the compiler. If the test is executable (class A, C, D, E) and it compiles successfully, then it must be run. Modified tests or split tests may then be processed. The results of the modified tests will be the only results graded.

4.4 Processing the Support Files

After the designated files have been customized as needed and required, the support files can be processed and the reporting mechanism can be verified.

4.4.1 Compile Files

The following files are necessary to many of the ACVC tests, both 9XBasic and Ada95. Implementations that maintain program libraries may wish to compile them into the program library used for validation:

REPSPEC.ADA	REPBODY.ADA
IMPDEF.A	
FCNDECL.ADA	
CHECKFIL.ADA	
LENCHECK.ADA	
ENUMCHEK.ADA	
SPPRT13S.ADT	(after macro substitution)
TCTOUCH.ADA	

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

4.4.2 "CZ" Acceptance Tests

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 CHECKFILE.ADA.

The acceptance test CZ1103A must be processed and run as the first step of a validation attempt to ensure correct operation of CHECKFILE. CZ1103A checks that 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 mean the test has failed. A listing of the expected output for CZ1103A is included in Appendix D (times and dates in the actual output will differ).

4.5 Establishing Command Scripts

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

4.5.1 Command Scripts

All compiler options and switches that are appropriate and necessary to run the ACVC tests must be identified. These options must be included in commands that invoke the compiler. The same is true for the binder or any other post-compilation tools.

The script should compile all class B tests. It should compile and bind all class L tests. It should compile all class F files. It should compile, bind, and execute all class A, C, D, and E tests.

4.5.2 Dependencies

A command script must take account of all required dependencies (both file and test). As noted earlier, some tests are composed of multiple test files. Also, some tests include foundation code, that may be used by other tests. If a foundation is not already in a program library, 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.

4.6 Processing ACVC Tests

After the ACVC tests and support code has been installed and all required modifications have been made, the suite can be processed by an implementation. This section describes the tests required for validation, 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.

4.6.1 Required Tests

An implementation may be validated against the core language only or the core language plus one or more special needs annexes. All core tests (except as noted in 4.6.3) must be processed with acceptable results for validation against the core language. This includes all 9XBasic tests and all Ada95 core tests. Validation against a special needs annex requires that all tests for that annex be correctly processed. Note that the core language may be validated without any validations against special needs annexes, but all special needs annexes validations require core language validation.

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.

4.6.2 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; ACVC 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 AVF (and, if necessary, the AVO) to qualify for a validation attempt. It is the responsibility of the user to identify the tests to be bundled and to write a driver for them.

4.6.3 Processing That May be Omitted

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

Many Ada95 tests rely on foundation code. A foundation need not be compiled anew each time a different test uses it. It is reasonable to put the code into the appropriate program library (often, this means "compile the code into the program library") only once and allow the binder to use the processed results for each test that "with"s the foundation.

Some tests may be determined by the user, with AVF concurrence, not to apply to a specific implementation (e.g., tests that rely on file_io need not be processed by implementations that do not support file i/o). Those tests need not be processed.

Tests for the special needs annexes of [Ada95] need not be processed except by implementations that wish to be validated for an annex. In that case, only the tests for the annex in question need be processed.

4.6.4 Tests with Special Processing Requirement

Some annex tests may require special handling. 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. 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 included in Appendix A.

4.6.5 Focus on Specific Areas

The ACVC test suite is structured to allow compiler developers and testers to use parts of the suite to focus on specific compiler feature areas. The 2.0.1 ACVC suite is divided along four axes:

The suite is divided into 9XBasic tests and Ada95 specific tests. The 9XBasic tests check for those features of Ada83 that are compatible with Ada95. All 9XBasic tests should provide the same results whether processed by an Ada83 implementation or an Ada95 implementation. The intent of

ACVC 2.0.1 is to facilitate a smooth technology transition. Tests that are specific to Ada95, of course, are not likely to produce correct results when processed by an Ada83 implementation.

Both the 9XBasic and Ada95 specific 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 9XBasic test names have not changed, but the Ada Reference Manual organization has changed from [Ada83] to [Ada95], so some 9XBasic test names point to the wrong clause of [Ada95]. Further, note that the general style and approach of the Ada95 specific 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.

Additionally, the Ada95 specific tests have been grouped into six major groups: object- oriented tests, child library tests, object-oriented *and* child library tests, real-time tests, predefined library tests and other tests. Tests in the object-oriented, child library, and real-time areas do not include features from the other major areas. For example, tests in the real-time group do not include object-oriented or child library features, and so forth. Tests in the "other" category may include features not included in another category (e.g., generics) or they may incorporate features from more than one of the other groups (i.e., they are "mixed"). A list of the tests in each group is included in Appendix B.

> **Note that the grouping of Ada95 specific tests into six categories is a feature of ACVC 2.0 and ACVC 2.0.1 only; it is intended to aid technology transition; it will not be included in ACVC 2.1.**

Finally, ACVC 2.0.1 tests are divided into core tests and special 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; special needs annex tests have a letter between 'C' and 'H' (inclusive) corresponding to the annex designation, as the third character of the test name.

4.6.6 Ada95 Features in 9XBasic Tests

There is a single test, CC3601A, in 9XBasic that contains a feature that may not be backward compatible with all Ada83 compilers. Consistent with the ISO WG9 ruling, box (<>) is used to indicate unknown discriminants for all formal private types.

4.7 Grading Test Results

Although a single test may examine multiple language issues, ACVC 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 with expectations constitute failures. The only exceptions allowed are discussed above in test splitting and modification; in such cases, processing the approved modified test(s) must produce the expected behavior. Any variances from the general discussion of expected results below for executable or non-executable tests are included as explicit test conditions in test prologues.

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

As an aid to technology transition, during the period when ACVC 2.0.1 is required, Ada95 core tests that fail will not necessarily prevent an implementation from being validated under ACVC 2.0. This policy will change when ACVC 2.1 comes in use. *All* applicable 9XBasic tests must be processed correctly. All tests for a special needs annex must be processed correctly for an implementation to conform to that annex.

Expected results for executable and non-executable tests are discussed in sections 4.7.1 - 4.7.3. Tests that are non-applicable for an implementation are discussed in 4.7.4. Withdrawn tests are discussed in 4.7.5.

4.7.1 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 L of [Ada95] requires implementations to document such implementation-defined characteristics.

A test that reports "FAILED" is graded as "failed" unless the AVF, and possibly the AVO, 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, or raises an unexpected exception). This kind of aberrant behavior may occur, for example, in certain tasking tests, where there are multiple threads of control. A pass / fail 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 "passed" 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 AVF, and possibly the AVO, determine that the test is not applicable for the implementation.

4.7.2 Expected Results for Class B

Class B tests are expected to be compiled but are not subject to any more 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 generally will have errors only in one compilation unit. Ada95 B-tests also include the notation "-- OK" to indicate constructs that must **not** be identified as errors. 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 exercise constructs whose correctness depends on source code that is textually separated (e.g., 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.

A test is graded as "passed" if it reports each error in the test, providing correct characterizations of the errors.

A test is graded as "failed" if it fails to report on each error in the test or if it marks legal code as erroneous, or if it fails to correctly characterize each error it reports.

4.7.3 Expected Results for Class L

Class L tests are expected to receive all pre-run-time processing (compilation, binding, etc.) but are not intended to be executable. These tests consist of multiple units, and there are one or more intentional errors included in their source code. An implementation must correctly report these errors.

Due to the nature of L tests, there is some variability in where the errors may be reported. Often, errors will be detected and reported at bind time. Some implementations may detect and report some errors at compile time. Either strategy is acceptable so long as the documented error(s) in the tests are identified and reported. The error report must provide a correct characterization of the error and its location.

Errors expected to be reported at bind time (or before) are marked "`-- ERROR:`" and have a short indication of the errors. Errors that could be reported at compile time are marked "`-- OPTIONAL ERROR`".

A test is graded as "passed" if it identifies the error(s) before run-time. The error indication must correspond to the nature of the errors marked "`ERROR`" or "`OPTIONAL ERROR`" and must indicate the location of the error.

A test is graded as "failed" if the compilation system does not find any errors, or if it marks legal code as erroneous, if it incorrectly characterizes errors, or if it executes and produces the "FAILED" message. In rare instances, an L test may run and yet not be incorrect for some implementations. In this case, the test reports "TENTATIVELY PASSED", and it does not fail if the special requirements indicated by the test documentation and output are satisfied.

4.7.4 Inapplicable Tests

Each ACVC test has a test objective that is described in the test prologue. Some objectives address Ada language features that are not required to 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.

A test may be inapplicable for an implementation given:

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

Appropriate grading criteria include:

a. whether a test completes execution and reports "NOT APPLICABLE";

b. 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, except that for ACVC 2.0.1, an applicable core test that does not pass will not necessarily prevent an implementation from being validated.

4.7.5 Withdrawn Tests

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

The usual reason for withdrawing a test is the discovery of an error in the test. A withdrawn test is not reissued in the same test suite, although it may be revised and reissued in a later suite.

The AVO maintains a list of withdrawn tests.

4.8 Addressing Problems or Issues

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

4.8.1 Typical Issues

Here is a list of typical causes of unexpected ACVC test failures often resulting from clerical errors:

Processing a test that is on the AVO list of withdrawn tests;

Processing a test that is not applicable to the implementation (as explained in section 4.7.4;

Files (or tests, see section 4.5.2) were processed in an incorrect order;

Program library is corrupted or units required in the program library 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;

Incorrect values in MACRO.DFS which result in incorrectly customized tests;

Need to modify a test (e.g., split a B-test).

Finally, from time to time a user discovers an error in a new ACVC 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 AVO. The dispute process is described in the next section.

4.8.2 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 wish to challenge the applicability or correctness of an ACVC test. A challenger should submit a petition against the test program to an AVF or to the AVO, following the procedure and the format presented in [Pro95]. A petition must clearly state whether it is a claim that the test is erroneous or that the test does not apply to the implementation. The petition must indicate the specific section of code that is disputed and provide a full explanation of the reason for the dispute.

AVFs will forward petitions from their customers to the AVO for adjudication. The AVO will evaluate the petitioner's claims and decide whether

- the test is applicable to the implementation (deviation is *not* allowed);
- the test is not applicable to the implementation (deviation *is* allowed);
- the test should be withdrawn (deviation is allowed).

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

4.9 Reprocessing and Regrading

After all problems have been resolved, tests that failed can be reprocessed and regraded. This step completes the ACVC testing process.

[This page intentionally blank]

Appendix A. Version Description

This version consists of 3309 ACVC 2.0.1 tests in 3704 files.

9XBasic Tests

The following tests constitute the 9XBasic set of tests and use Ada83 features that are upwardly compatible with Ada95.

A22006B	A85007D	B22001H	B28001H	B33102D
A22006C	A85013B	B22001I	B28001I	B33102E
A22006D	A87B59A	B22001J	B28001J	B33201A
A26007A	A95001C	B22001K	B28001K	B33201B
A27003A	A95074D	B22001L	B28001L	B33201C
A29003A	A97106A	B22001M	B28001M	B33201D
A2A031A	A99006A	B22001N	B28001N	B33201E
A33003A	AA2010A	B23002A	B28001O	B33204A
A34017C	AA2012A	B23004A	B28001P	B33205A
A35101B	AC1015B	B23004B	B28001Q	B33302A
A35402A	AC3106A	B24001A	B28001R	B34001B
A35801F	AC3206A	B24001B	B28001S	B34001E
A35902C	AC3207A	B24001C	B28001T	B34002B
A38106D	AD7001B	B24005A	B28001U	B34003B
A38106E	AD7001C	B24005B	B28001V	B34004B
A49027A	AD7001D	B24007A	B28001W	B34005B
A49027B	AD7006A	B24009A	B29001A	B34005E
A49027C	AD7101A	B24009B	B2A003A	B34005H
A54B01A	AD7101C	B24104A	B2A003B	B34005K
A54B02A	AD7102A	B24204A	B2A003C	B34005N
A55B12A	AD7103A	B24204B	B2A003D	B34005Q
A55B13A	AD7103C	B24204C	B2A003E	B34005T
A55B14A	AD7104A	B24204D	B2A003F	B34006B
A71004A	AD7201A	B24204E	B2A005A	B34006E
A73001I	AD7203B	B24204F	B2A005B	B34006H
A73001J	AD7205B	B24205A	B2A007A	B34006K
A74105B	AD8011A	B24206A	B2A010A	B34007B
A74106A	ADA101A	B24206B	B2A021A	B34007E
A74106B	AE2113A	B24211B	B32101A	B34007H
A74106C	AE2113B	B25002A	B32103A	B34007K
A74205E	AE3002G	B25002B	B32104A	B34007N
A74205F	AE3101A	B26001A	B32106A	B34007Q
A83009A	AE3702A	B26002A	B32201A	B34007T
A83009B	AE3709A	B26005A	B32202A	B34008B
A83A02A	B22001A	B28001A	B32202B	B34009B
A83A02B	B22001B	B28001B	B32202C	B34009E
A83A06A	B22001C	B28001C	B33001A	B34009H
A83A08A	B22001D	B28001D	B33101A	B34009K
A83C01C	B22001E	B28001E	B33102A	B34011A
A83C01H	B22001F	B28001F	B33102B	B34014B
A83C01I	B22001G	B28001G	B33102C	B34014D

B34014F
B34014I
B34014M
B34014O
B34014V
B35004A
B35101A
B35103A
B35103B
B35302A
B35401A
B35401B
B35403A
B35501A
B35501B
B35506A
B35506B
B35506C
B35506D
B35701A
B35709A
B35901A
B35901C
B35901D
B35A01A
B35A08A
B36001A
B36002A
B36101A
B36102A
B36103A
B36105C
B36171A
B36171B
B36171C
B36171D
B36171E
B36171F
B36171G
B36171H
B36171I
B36201A
B36307A
B37004A
B37004B
B37004C
B37004D
B37004E
B37004F
B37004G
B37101A
B37102A
B37104A
B37106A
B37201A
B37201B
B37203A
B37301I
B37301J
B37302A
B37303A
B37309B
B37310B
B37311A
B37312B
B37401A
B37409B
B38003A
B38003B
B38003C
B38003D
B38008A
B38008B
B38009A
B38009D
B38101A
B38101B
B38101C
B38103A
B38103B
B38103C
B38103D
B38103E
B38105A
B38105B
B38203A
B41101A
B41101C
B41201A
B41201C
B41202C
B41202D
B41324B
B41325B
B41327B
B43001M
B43002D
B43002E
B43002F
B43002G
B43002H
B43002I
B43002J
B43002K
B43005A
B43005B
B43005F
B43101A
B43102A
B43102B
B43105C
B43201A
B43201C
B43201D
B43202A
B43202C
B43209B
B43221A
B43221B
B43223A
B44001A
B44001B
B44002B
B44002C
B44004A
B44004B
B44004C
B44004D
B44004E
B45102A
B45116A
B45121A
B45204A
B45205A
B45206C
B45207A
B45207B
B45207C
B45207D
B45207G
B45207H
B45207I
B45207J
B45207M
B45207N
B45207O
B45207P
B45207S
B45207T
B45207U
B45207V
B45208A
B45208B
B45208C
B45208G
B45208H
B45208I
B45208M
B45208N
B45208S
B45208T
B45209A
B45209B
B45209C
B45209D
B45209E
B45209F
B45209G
B45209H
B45209I
B45209J
B45209K
B45221A
B45261A
B45261B
B45261C
B45261D
B45301A
B45301B
B45301C
B45302A
B45341A
B45501A
B45501B
B45501C
B45522A
B45537A
B45601A
B45625A
B45661A
B46002A
B46003A
B46004A
B46004B
B46004C
B46004D
B46004E
B46005A
B47001A
B48001A
B48001B
B48002A
B48002B
B48002C
B48002D
B48002E
B48002G
B48003A
B48003B
B48003C
B48003D
B48003E
B49002A
B49004A
B49005A
B49007A
B49007B
B49008A
B49008C
B49009B
B49009C
B49010A
B49011A
B4A010C
B4A016A
B51001A
B51004B
B51004C
B52002A
B52002B
B52002C
B52002D
B52002E
B52002F
B52002G
B52004A
B52004B
B52004C
B52004D
B52004E
B53001A
B53001B
B53002A
B53002B
B53009A
B53009B
B53009C
B54A01B
B54A01F
B54A01G
B54A01L
B54A05A
B54A05B
B54A10A
B54A12A
B54A20A
B54A21A
B54A25A
B54A60A
B54A60B
B54B01B
B54B01C
B54B02B
B54B02C
B54B02D
B54B04A
B54B04B
B54B05A
B54B06A
B55A01A

B55A01D	B63001B	B73001B	B83026B	B85001L
B55A01E	B63005A	B73001C	B83027B	B85002A
B55A01J	B63005B	B73001D	B83027D	B85003A
B55A01K	B63006A	B73001E	B83028B	B85003B
B55A01L	B63009A	B73001F	B83029B	B85004A
B55A01N	B63009B	B73001G	B83030B	B85008F
B55A01O	B63009C	B73001H	B83030D	B85008G
B55A01T	B63103A	B73004A	B83031B	B85008H
B55A01U	B64002A	B73004B	B83031D	B85010A
B55A01V	B64002C	B74001A	B83031F	B85010B
B55B01A	B64003A	B74001B	B83032B	B85012A
B55B01B	B64004A	B74101A	B83033B	B85013C
B55B09B	B64004B	B74101B	B83041E	B85013D
B55B09C	B64004C	B74103A	B83A01A	B85015A
B55B09D	B64004D	B74103D	B83A01B	B86001A
B55B12B	B64004E	B74103E	B83A01C	B87B23B
B55B12C	B64004F	B74103G	B83A05A	B87B26A
B55B17A	B64101A	B74103I	B83A06B	B87B48C
B55B17B	B64201A	B74104A	B83A06H	B91001B
B55B17C	B65001A	B74105A	B83A07A	B91001C
B55B18A	B65002A	B74105C	B83A07B	B91001D
B56001A	B65002B	B74201A	B83A07C	B91001E
B56001D	B66001A	B74202A	B83A08B	B91001F
B56001E	B66001B	B74202B	B83A09A	B91001G
B56001F	B66001C	B74202C	B83B01A	B91002A
B56001G	B66001D	B74202D	B83B02C	B91002B
B56001H	B67001A	B74203B	B83E01A	B91002C
B57001A	B67001B	B74203C	B83E01B	B91002D
B57001B	B67001C	B74203D	B83E01C	B91002E
B57001C	B67001D	B74203E	B83E01D	B91002F
B57001D	B67001H	B74205A	B83E01E	B91002G
B58001A	B67001I	B74207A	B83E01F	B91002H
B58002A	B67001J	B74304A	B83E11A	B91002I
B58002B	B67001K	B74304B	B83F02A	B91002J
B58002C	B67004A	B74304C	B83F02B	B91002K
B58003A	B71001A	B74404A	B83F02C	B91002L
B58003B	B71001B	B74404B	B84001A	B91003A
B59001A	B71001C	B74409A	B84002B	B91003B
B59001C	B71001D	B83001A	B84004A	B91003C
B59001D	B71001F	B83003A	B84005B	B91003D
B59001E	B71001G	B83003B	B84006A	B91003E
B59001F	B71001H	B83003C	B84007A	B91004A
B59001G	B71001I	B83004A	B84008B	B91005A
B59001H	B71001J	B83004B	B85001A	B92001A
B59001I	B71001L	B83004C	B85001B	B92001B
B61001F	B71001M	B83004D	B85001C	B95001A
B61005A	B71001N	B83006A	B85001D	B95001B
B61006A	B71001O	B83006B	B85001E	B95001D
B61011A	B71001P	B83008A	B85001F	B95002A
B62001A	B71001R	B83008B	B85001G	B95003A
B62001B	B71001T	B83011A	B85001H	B95004A
B62001C	B71001U	B83023B	B85001I	B95004B
B62001D	B71001V	B83024B	B85001J	B95006A
B63001A	B73001A	B83024F	B85001K	B95006B

B95006C
B95006D
B95007A
B95007B
B95020A
B95020B
B95030A
B95031A
B95032A
B95061A
B95061B
B95061C
B95061D
B95061E
B95061F
B95061G
B95062A
B95063A
B95064A
B95068A
B95070A
B95080A
B95080C
B95081A
B95082A
B95082B
B95082C
B95082D
B95082E
B95082F
B95083A
B95094A
B95094B
B95094C
B96002A
B97102B
B97102C
B97102D
B97102F
B97102G
B97102H
B97102I
B97103A
B97103B
B97103D
B97103E
B97103F
B97103G
B97104A
B97104B
B97104C
B97104D
B97104E
B97104F
B97104G
B97107A
B97108A
B97108B
B97109A
B97110A
B97110B
B97111A
B97206A
B97306A
B99001A
B99001B
B99002A
B99002B
B99002C
B99003A
B9A001A
B9A001B
BA1001A
BA1001D
BA1010A
BA1010B
BA1010C
BA1010D
BA1010E
BA1010F
BA1010G
BA1010H
BA1010I
BA1010J
BA1010K
BA1010L
BA1010M
BA1010N
BA1010P
BA1010Q
BA1011B
BA1011C
BA1020A
BA1020B
BA1020C
BA1020F
BA1101A
BA1101B
BA1101C
BA1101E
BA1101F
BA1101G
BA1109A
BA1110A
BA2001A
BA2001B
BA2001C
BA2001D
BA2001F
BA2003B
BA2011A
BA2013A
BA2013B
BA3001A
BA3001B
BA3001C
BA3001E
BA3001F
BA3006A
BA3006B
BB2001A
BB2002A
BB2003A
BB2003B
BB2003C
BB3001A
BB3002A
BC1001A
BC1002A
BC1005A
BC1008A
BC1008B
BC1008C
BC1009A
BC1011A
BC1011B
BC1011C
BC1012A
BC1013A
BC1014A
BC1014B
BC1016A
BC1016B
BC1101A
BC1102A
BC1103A
BC1106A
BC1107A
BC1109A
BC1109B
BC1109C
BC1109D
BC1110A
BC1201A
BC1201B
BC1201C
BC1201D
BC1201E
BC1201F
BC1201G
BC1201H
BC1201I
BC1201J
BC1201K
BC1201L
BC1202A
BC1202C
BC1202E
BC1202F
BC1202G
BC1203A
BC1205A
BC1206A
BC1207A
BC1208A
BC1226A
BC1230A
BC1303A
BC1303B
BC1303C
BC1303D
BC1303E
BC1303F
BC1303G
BC1306A
BC2001B
BC2001C
BC2001D
BC2001E
BC2004A
BC2004B
BC3001A
BC3002A
BC3002B
BC3002C
BC3002D
BC3002E
BC3005A
BC3005B
BC3005C
BC3006A
BC3009C
BC3011B
BC3013A
BC3016G
BC3018A
BC3101A
BC3101B
BC3102A
BC3102B
BC3103B
BC3123C
BC3201A
BC3201B
BC3201C
BC3202A
BC3202B
BC3202C
BC3202D
BC3301A
BC3301B
BC3302A
BC3302B
BC3303A
BC3304A
BC3401A
BC3401B
BC3402A
BC3402B
BC3403A
BC3403B
BC3403C
BC3404A
BC3404B
BC3404C
BC3404D
BC3404E
BC3404F
BC3405A
BC3405B
BC3405D
BC3405E
BC3405F
BC3501A
BC3501B
BC3501C
BC3501D
BC3501E
BC3501F
BC3501G
BC3501H
BC3501I
BC3501J
BC3501K
BC3502A
BC3502B
BC3502C
BC3502D
BC3502E
BC3502F
BC3502G
BC3502H
BC3502I
BC3502J
BC3502K
BC3502L
BC3502M
BC3502N
BC3502O
BC3503A

BC3503C	BD4007A	C24106A	C34005U	C35502K
BC3503D	BD4009A	C24202D	C34005V	C35502L
BC3503E	BD4011A	C24203A	C34006A	C35502M
BC3503F	BD5001A	C24203B	C34006D	C35502N
BC3604A	BD5005A	C24207A	C34006F	C35502O
BC3604B	BD5005D	C24211A	C34006G	C35502P
BC3607A	BD5102A	C25001A	C34006J	C35503A
BD1B01A	BD5102B	C25001B	C34006L	C35503B
BD1B02B	BD5103A	C26006A	C34007A	C35503C
BD1B03C	BD5104A	C26008A	C34007D	C35503D
BD1B05E	BD7001A	C2A001A	C34007F	C35503E
BD1B06J	BD7101H	C2A001B	C34007G	C35503F
BD2001B	BD7201C	C2A001C	C34007I	C35503G
BD2A01H	BD7203A	C2A002A	C34007J	C35503H
BD2A02A	BD7204A	C2A008A	C34007M	C35503K
BD2A03A	BD7205A	C2A021B	C34007P	C35503L
BD2A03B	BD7301A	C32001A	C34007R	C35503O
BD2A06A	BD7302A	C32001B	C34007S	C35503P
BD2A25A	BD8001A	C32001C	C34007U	C35504A
BD2A35A	BD8002A	C32001D	C34007V	C35504B
BD2A45A	BD8004A	C32001E	C34008A	C35505C
BD2A55A	BD8004B	C32107A	C34009A	C35505E
BD2A55B	BD8004C	C32107C	C34009D	C35505F
BD2A67A	BE2101E	C32108A	C34009F	C35507A
BD2A77A	BE2101J	C32108B	C34009G	C35507B
BD2A85A	BE2114A	C32111A	C34009J	C35507C
BD2A85B	BE2116A	C32111B	C34009L	C35507E
BD2B01C	BE2208A	C32112B	C34011B	C35507G
BD2B02A	BE3002A	C32113A	C34012A	C35507H
BD2B03A	BE3002E	C32115A	C34014A	C35507I
BD2B03B	BE3205A	C32115B	C34014C	C35507J
BD2B03C	BE3301C	C34001A	C34014E	C35507K
BD2C01D	BE3606C	C34001C	C34014G	C35507L
BD2C02A	BE3703A	C34001D	C34014H	C35507M
BD2C03A	BE3802A	C34001F	C34014N	C35507N
BD2D01C	BE3803A	C34002A	C34014P	C35507O
BD2D01D	BE3902A	C34002C	C34014R	C35507P
BD2D02A	BE3903A	C34003A	C34014T	C35508A
BD2D03A	C23001A	C34003C	C34014U	C35508B
BD2D03B	C23003A	C34004A	C34018A	C35508C
BD3001A	C23003B	C34004C	C35003A	C35508E
BD3001B	C23003G	C34005A	C35003B	C35508G
BD3001C	C23003I	C34005C	C35003D	C35508H
BD3002A	C23006A	C34005D	C35102A	C35508K
BD3003A	C23006B	C34005F	C35502A	C35508L
BD3003B	C23006C	C34005G	C35502B	C35508O
BD3012A	C23006D	C34005I	C35502C	C35508P
BD3013A	C23006E	C34005J	C35502D	C35703A
BD4001A	C23006F	C34005L	C35502E	C35704A
BD4002A	C23006G	C34005M	C35502F	C35704B
BD4003A	C24002D	C34005O	C35502G	C35704C
BD4003B	C24003A	C34005P	C35502H	C35704D
BD4003C	C24003B	C34005R	C35502I	C35801D
BD4006A	C24003C	C34005S	C35502J	C35902D

C35904A	C37108B	C38108B	C41309A	C43214B
C35904B	C37206A	C38108C	C41320A	C43214C
C35A02A	C37207A	C38108D	C41321A	C43214D
C35A05A	C37208A	C38202A	C41322A	C43214E
C35A05D	C37208B	C39006A	C41323A	C43214F
C35A05N	C37209A	C39006B	C41324A	C43215A
C35A05Q	C37209B	C39006C	C41325A	C43215B
C35A07A	C37210A	C39006D	C41326A	C43222A
C35A07D	C37211A	C39006E	C41327A	C43224A
C35A08B	C37211B	C39006F	C41328A	C44003A
C36104A	C37211C	C39006G	C41401A	C44003D
C36104B	C37211D	C39007A	C41402A	C44003F
C36172A	C37211E	C39007B	C41404A	C44003G
C36172B	C37213B	C39008A	C42006A	C45112A
C36172C	C37213D	C39008B	C42007E	C45112B
C36174A	C37213F	C39008C	C43003A	C45113A
C36180A	C37213H	C41101D	C43004A	C45114B
C36202C	C37213J	C41103A	C43004C	C45201A
C36203A	C37213K	C41103B	C43103A	C45201B
C36204A	C37213L	C41104A	C43103B	C45202B
C36204B	C37215B	C41105A	C43104A	C45210A
C36204C	C37215D	C41107A	C43105A	C45211A
C36204D	C37215F	C41201D	C43105B	C45220A
C36205A	C37215H	C41203A	C43106A	C45220B
C36205B	C37217A	C41203B	C43107A	C45220C
C36205C	C37217B	C41204A	C43108A	C45220D
C36205D	C37217C	C41205A	C43204A	C45220E
C36205E	C37304A	C41206A	C43204C	C45220F
C36205F	C37305A	C41207A	C43204E	C45231A
C36205G	C37306A	C41301A	C43204F	C45231B
C36205H	C37309A	C41303A	C43204G	C45231C
C36205I	C37310A	C41303B	C43204H	C45231D
C36205J	C37312A	C41303C	C43204I	C45232B
C36205K	C37402A	C41303E	C43205A	C45242B
C36205L	C37403A	C41303F	C43205B	C45251A
C36301A	C37404A	C41303G	C43205C	C45252A
C36301B	C37404B	C41303I	C43205D	C45252B
C36302A	C37405A	C41303J	C43205E	C45253A
C36304A	C37411A	C41303K	C43205G	C45262A
C36305A	C38002A	C41303M	C43205H	C45262B
C37002A	C38002B	C41303N	C43205I	C45262C
C37003A	C38005A	C41303O	C43205J	C45262D
C37003B	C38005B	C41303Q	C43205K	C45264A
C37005A	C38005C	C41303R	C43206A	C45264B
C37006A	C38006A	C41303S	C43207B	C45264C
C37008A	C38102A	C41303U	C43207D	C45265A
C37008B	C38102B	C41303V	C43208A	C45271A
C37009A	C38102C	C41303W	C43208B	C45272A
C37010A	C38102D	C41304A	C43209A	C45273A
C37010B	C38102E	C41304B	C43210A	C45274A
C37102B	C38104A	C41306A	C43211A	C45274B
C37103A	C38107A	C41306B	C43212A	C45274C
C37105A	C38107B	C41306C	C43212C	C45281A
C37107A	C38108A	C41307D	C43214A	C45282A

C45282B	C45532B	C46051B	C4A005B	C52104X
C45291A	C45532C	C46051C	C4A006A	C52104Y
C45303A	C45532D	C46052A	C4A007A	C53007A
C45304A	C45532E	C46053A	C4A010A	C54A03A
C45304B	C45532F	C46054A	C4A010B	C54A04A
C45304C	C45532G	C47002A	C4A011A	C54A07A
C45322A	C45532H	C47002B	C4A012B	C54A13A
C45323A	C45532I	C47002C	C4A013A	C54A13B
C45331A	C45532J	C47002D	C4A013B	C54A13C
C45331D	C45532K	C47003A	C4A014A	C54A13D
C45342A	C45532L	C47004A	C51004A	C54A22A
C45343A	C45532M	C47005A	C52005A	C54A23A
C45344A	C45532N	C47006A	C52005B	C54A24A
C45345B	C45532O	C47007A	C52005C	C54A24B
C45347A	C45532P	C47008A	C52005D	C54A42A
C45347B	C45534A	C47009A	C52005E	C54A42B
C45347C	C45534B	C47009B	C52005F	C54A42C
C45347D	C45536A	C48004A	C52008A	C54A42D
C45411A	C45611A	C48004B	C52008B	C54A42E
C45411B	C45611B	C48004C	C52009A	C54A42F
C45411C	C45611C	C48004D	C52009B	C54A42G
C45411D	C45613A	C48004E	C52010A	C55B03A
C45413A	C45613B	C48004F	C52011A	C55B04A
C45431A	C45613C	C48005A	C52011B	C55B05A
C45502B	C45614A	C48005B	C52101A	C55B06A
C45502C	C45614B	C48006A	C52102A	C55B06B
C45503A	C45614C	C48006B	C52102B	C55B07A
C45503B	C45622A	C48007A	C52102C	C55B07B
C45503C	C45624A	C48007B	C52102D	C55B10A
C45504A	C45624B	C48007C	C52103A	C55B11A
C45504B	C45631A	C48008A	C52103B	C55B11B
C45504C	C45631B	C48008C	C52103C	C55B15A
C45504D	C45631C	C48009A	C52103F	C55B16A
C45504E	C45632A	C48009B	C52103G	C55C02A
C45504F	C45632B	C48009C	C52103H	C55C02B
C45505A	C45632C	C48009D	C52103K	C56002A
C45523A	C45651A	C48009E	C52103L	C57003A
C45531A	C45662A	C48009F	C52103M	C57004A
C45531B	C45662B	C48009G	C52103P	C57004B
C45531C	C45672A	C48009H	C52103Q	C58004C
C45531D	C45701A	C48009I	C52103R	C58004D
C45531E	C46011A	C48009J	C52103X	C58004G
C45531F	C46013A	C48010A	C52104A	C58005A
C45531G	C46014A	C48011A	C52104B	C58005B
C45531H	C46021A	C48012A	C52104C	C58005H
C45531I	C46024A	C49020A	C52104F	C58006A
C45531J	C46031A	C49021A	C52104G	C58006B
C45531K	C46032A	C49022A	C52104H	C59002A
C45531L	C46033A	C49022B	C52104K	C59002B
C45531M	C46041A	C49022C	C52104L	C59002C
C45531N	C46042A	C49023A	C52104M	C61008A
C45531O	C46043B	C49024A	C52104P	C61009A
C45531P	C46044B	C49025A	C52104Q	C61010A
C45532A	C46051A	C49026A	C52104R	C62002A

C62003A	C64202A	C83025A	C85017A	C87B34B
C62003B	C65003A	C83025C	C85018A	C87B34C
C62004A	C65003B	C83027A	C85018B	C87B35C
C62006A	C66002A	C83027C	C85019A	C87B38A
C64002B	C66002C	C83028A	C86003A	C87B39A
C64004G	C66002D	C83029A	C86004A	C87B40A
C64005A	C66002E	C83030A	C86004B	C87B41A
C64005B	C66002F	C83030C	C86004C	C87B42A
C64005C	C66002G	C83031A	C86006I	C87B43A
C64005D	C67002A	C83031C	C86007A	C87B44A
C64103B	C67002B	C83031E	C87A05A	C87B45A
C64103C	C67002C	C83032A	C87A05B	C87B45C
C64103D	C67002D	C83033A	C87B02A	C87B47A
C64103E	C67002E	C83051A	C87B02B	C87B48A
C64103F	C67003F	C83B02A	C87B03A	C87B48B
C64104A	C67005A	C83B02B	C87B04A	C87B50A
C64104B	C67005B	C83E02A	C87B04B	C87B54A
C64104C	C67005C	C83E02B	C87B04C	C87B57A
C64104D	C67005D	C83E03A	C87B05A	C87B62A
C64104E	C72001B	C83F01A	C87B06A	C87B62B
C64104F	C72002A	C83F01B	C87B07A	C87B62C
C64104G	C73002A	C83F01C	C87B07B	C87B62D
C64104H	C74004A	C83F01D	C87B07C	C91004B
C64104I	C74203A	C83F03A	C87B07D	C91004C
C64104J	C74206A	C83F03B	C87B07E	C91006A
C64104K	C74207B	C83F03C	C87B08A	C91007A
C64104L	C74208A	C83F03D	C87B09A	C92002A
C64104M	C74208B	C84002A	C87B09C	C92003A
C64104N	C74209A	C84005A	C87B10A	C92005A
C64104O	C74210A	C84008A	C87B11A	C92005B
C64105A	C74211A	C84009A	C87B11B	C92006A
C64105B	C74211B	C85004B	C87B13A	C93001A
C64105C	C74302A	C85005A	C87B14A	C93002A
C64105D	C74302B	C85005B	C87B14B	C93003A
C64106A	C74305A	C85005C	C87B14C	C93004A
C64106B	C74305B	C85005D	C87B14D	C93004B
C64106C	C74306A	C85005E	C87B15A	C93004C
C64106D	C74307A	C85005F	C87B16A	C93004D
C64107A	C74401D	C85005G	C87B17A	C93004F
C64108A	C74401E	C85006A	C87B18A	C93005A
C64109A	C74401K	C85006B	C87B18B	C93005B
C64109B	C74401Q	C85006C	C87B19A	C93005C
C64109C	C74402A	C85006D	C87B23A	C93005D
C64109D	C74402B	C85006E	C87B24A	C93005E
C64109E	C74406A	C85006F	C87B24B	C93005F
C64109F	C74407B	C85006G	C87B26B	C93005G
C64109G	C74409B	C85007A	C87B27A	C93005H
C64109H	C83007A	C85007E	C87B28A	C93006A
C64109I	C83012D	C85009A	C87B29A	C93007A
C64109J	C83022A	C85011A	C87B30A	C93008A
C64109K	C83022G	C85013A	C87B31A	C93008B
C64109L	C83023A	C85014A	C87B32A	C94001A
C64201B	C83024A	C85014B	C87B33A	C94001B
C64201C	C83024E	C85014C	C87B34A	C94001C

C94001E	C95074C	C97116A	CA1003A	CB4008A
C94001F	C95076A	C97117A	CA1004A	CB4009A
C94001G	C95078A	C97117B	CA1005A	CB4013A
C94002A	C95080B	C97117C	CA1006A	CB5001A
C94002B	C95082G	C97118A	CA1011A	CB5001B
C94002D	C95085A	C97120A	CA1012A	CB5002A
C94002E	C95085B	C97120B	CA1012B	CC1004A
C94002F	C95085C	C97201A	CA1013A	CC1005B
C94002G	C95085D	C97201B	CA1014A	CC1010A
C94004A	C95085E	C97201C	CA1020D	CC1010B
C94004B	C95085F	C97201D	CA1020E	CC1018A
C94004C	C95085G	C97201E	CA1022A	CC1104C
C94005A	C95085H	C97201G	CA1102A	CC1107B
C94005B	C95085I	C97201H	CA1106A	CC1111A
C94006A	C95085J	C97201X	CA1108A	CC1204A
C94007A	C95085K	C97202A	CA1108B	CC1207B
C94007B	C95085L	C97203A	CA2001H	CC1220A
C94008A	C95085M	C97203B	CA2002A	CC1221A
C94008B	C95085N	C97203C	CA2003A	CC1221B
C94008C	C95085O	C97204A	CA2004A	CC1221C
C94008D	C95086A	C97204B	CA2007A	CC1221D
C94010A	C95086B	C97205A	CA2008A	CC1222A
C94011A	C95086C	C97205B	CA2009A	CC1223A
C94020A	C95086D	C97301A	CA2009C	CC1224A
C95008A	C95086E	C97301B	CA2009D	CC1225A
C95009A	C95086F	C97301C	CA2009F	CC1226B
C95010A	C95087A	C97301D	CA2011B	CC1227A
C95011A	C95087B	C97301E	CA3011A	CC1301A
C95012A	C95087C	C97302A	CA5003A	CC1302A
C95021A	C95087D	C97303A	CA5003B	CC1304A
C95022A	C95088A	C97303B	CA5004A	CC1304B
C95022B	C95089A	C97303C	CA5004B	CC1307A
C95033A	C95090A	C97304A	CA5006A	CC1307B
C95033B	C95092A	C97304B	CB1001A	CC1308A
C95034A	C95093A	C97305A	CB1004A	CC1310A
C95034B	C95095A	C97305B	CB1005A	CC1311A
C95035A	C95095B	C97305C	CB1010A	CC1311B
C95040A	C95095C	C97305D	CB1010B	CC2002A
C95040B	C95095D	C97307A	CB1010C	CC3004A
C95040C	C95095E	C99004A	CB1010D	CC3007A
C95040D	C96001A	C99005A	CB2004A	CC3007B
C95041A	C96004A	C9A003A	CB2005A	CC3011A
C95065A	C96005A	C9A004A	CB2006A	CC3011D
C95065B	C96005B	C9A005A	CB2007A	CC3012A
C95065C	C96005D	C9A007A	CB3003A	CC3015A
C95065D	C96005F	C9A008A	CB3003B	CC3016B
C95065E	C96006A	C9A009A	CB3004A	CC3016C
C95065F	C96007A	C9A009C	CB4001A	CC3016F
C95066A	C96008A	C9A009F	CB4002A	CC3016I
C95067A	C96008B	C9A009G	CB4003A	CC3017B
C95071A	C97112A	C9A009H	CB4004A	CC3017C
C95072A	C97113A	C9A010A	CB4005A	CC3019A
C95072B	C97114A	C9A011A	CB4006A	CC3019B
C95073A	C97115A	C9A011B	CB4007A	CC3019C

CC3106B
CC3120A
CC3120B
CC3121A
CC3123A
CC3125A
CC3125B
CC3125C
CC3125D
CC3126A
CC3127A
CC3128A
CC3203A
CC3207B
CC3220A
CC3221A
CC3222A
CC3223A
CC3224A
CC3225A
CC3230A
CC3231A
CC3232A
CC3233A
CC3234A
CC3235A
CC3236A
CC3240A
CC3305A
CC3305B
CC3305C
CC3305D
CC3601A
CC3601C
CC3602A
CC3603A
CC3605A
CC3606A
CC3606B
CC3607B
CD1009A
CD1009B
CD1009C
CD1009D
CD1009E
CD1009F
CD1009G
CD1009H
CD1009I
CD1009J
CD1009K
CD1009L
CD1009M
CD1009N
CD1009O
CD1009P
CD1009Q
CD1009R
CD1009S
CD1009T
CD1009U
CD1009V
CD1009W
CD1009X
CD1009Y
CD1009Z
CD1C03A
CD1C03B
CD1C03C
CD1C03E
CD1C03F
CD1C03G
CD1C03H
CD1C03I
CD1C04A
CD1C04D
CD1C04E
CD1C06A
CD2A21A
CD2A21C
CD2A21E
CD2A22A
CD2A22E
CD2A22I
CD2A22J
CD2A23A
CD2A23E
CD2A24A
CD2A24E
CD2A24I
CD2A24J
CD2A31A
CD2A31C
CD2A31E
CD2A32A
CD2A32C
CD2A32E
CD2A32G
CD2A32I
CD2A32J
CD2A51A
CD2A53A
CD2A53E
CD2A83C
CD2A91C
CD2B11A
CD2B11B
CD2B11D
CD2B11E
CD2B11F
CD2B15C
CD2B16A
CD2C11A
CD2C11B
CD2C11D
CD2D11A
CD2D13A
CD3014A
CD3014C
CD3014D
CD3014F
CD3015A
CD3015C
CD3015E
CD3015F
CD3015G
CD3015H
CD3015I
CD3015K
CD3021A
CD4031A
CD4041A
CD4051A
CD4051B
CD4051C
CD4051D
CD5003A
CD5003B
CD5003C
CD5003D
CD5003E
CD5003F
CD5003G
CD5003H
CD5003I
CD5011A
CD5011C
CD5011E
CD5011G
CD5011I
CD5011K
CD5011M
CD5011Q
CD5011S
CD5012A
CD5012B
CD5012E
CD5012F
CD5012I
CD5012M
CD5013A
CD5013C
CD5013E
CD5013G
CD5013I
CD5013K
CD5013M
CD5013O
CD5014A
CD5014C
CD5014E
CD5014G
CD5014I
CD5014K
CD5014M
CD5014O
CD5014T
CD5014V
CD5014X
CD5014Y
CD5014Z
CD7002A
CD7007B
CD7101D
CD7101E
CD7101F
CD7101G
CD7103D
CD7202A
CD7204B
CD7204C
CD7305A
CDA201A
CDA201B
CDA201C
CDA201E
CE2102A
CE2102B
CE2102C
CE2102D
CE2102E
CE2102F
CE2102G
CE2102H
CE2102I
CE2102J
CE2102K
CE2102L
CE2102M
CE2102N
CE2102O
CE2102P
CE2102Q
CE2102R
CE2102S
CE2102T
CE2102U
CE2102V
CE2102W
CE2102X
CE2102Y
CE2103A
CE2103B
CE2103C
CE2103D
CE2104A
CE2104B
CE2104C
CE2104D
CE2106A
CE2106B
CE2108E
CE2108F
CE2108G
CE2108H
CE2109A
CE2109B
CE2109C
CE2110A
CE2110C
CE2111A
CE2111B
CE2111C
CE2111E
CE2111F
CE2111G
CE2111I
CE2120A
CE2120B
CE2201A
CE2201B
CE2201C
CE2201D
CE2201E
CE2201F
CE2201G
CE2201H
CE2201I
CE2201J
CE2201K
CE2201L
CE2201M
CE2201N
CE2202A
CE2203A
CE2204A
CE2204B
CE2204C
CE2204D
CE2205A

CE2206A
CE2208B
CE2401A
CE2401B
CE2401C
CE2401E
CE2401F
CE2401H
CE2401I
CE2401J
CE2401K
CE2401L
CE2402A
CE2403A
CE2404A
CE2404B
CE2405B
CE2406A
CE2407A
CE2407B
CE2408A
CE2408B
CE2409A
CE2409B
CE2410A
CE2410B
CE2411A
CE3002B
CE3002C
CE3002D
CE3002F
CE3102A
CE3102B
CE3102D
CE3102E
CE3102F
CE3102G
CE3102H
CE3102I
CE3102J
CE3102K
CE3103A
CE3104A
CE3104B
CE3104C
CE3106A
CE3106B
CE3107A
CE3107B
CE3108A
CE3108B
CE3110A
CE3112C
CE3112D
CE3114A
CE3115A
CE3119A
CE3201A
CE3202A
CE3206A
CE3207A
CE3301A
CE3302A
CE3303A
CE3304A
CE3305A
CE3306A
CE3401A
CE3402A
CE3402C
CE3402D
CE3402E
CE3403A
CE3403B
CE3403C
CE3403D
CE3403E
CE3403F
CE3404A
CE3404B
CE3404C
CE3404D
CE3405A
CE3405C
CE3405D
CE3406A
CE3406B
CE3406C
CE3406D
CE3407A
CE3407B
CE3407C
CE3408A
CE3408B
CE3408C
CE3409A
CE3409B
CE3409C
CE3409D
CE3409E
CE3410A
CE3410B
CE3410C
CE3410D
CE3410E
CE3411A
CE3411C
CE3412A
CE3413A
CE3413B
CE3413C
CE3414A
CE3601A
CE3602A
CE3602B
CE3602C
CE3602D
CE3603A
CE3604A
CE3604B
CE3605A
CE3605B
CE3605C
CE3605D
CE3605E
CE3606A
CE3606B
CE3701A
CE3704A
CE3704B
CE3704C
CE3704D
CE3704E
CE3704F
CE3704M
CE3704N
CE3704O
CE3705A
CE3705B
CE3705C
CE3705D
CE3705E
CE3706C
CE3706D
CE3706F
CE3706G
CE3707A
CE3708A
CE3801A
CE3801B
CE3804A
CE3804B
CE3804C
CE3804D
CE3804E
CE3804F
CE3804G
CE3804H
CE3804I
CE3804J
CE3804M
CE3804O
CE3804P
CE3805A
CE3805B
CE3806A
CE3806B
CE3806C
CE3806D
CE3806E
CE3806F
CE3806G
CE3806H
CE3809A
CE3809B
CE3810A
CE3810B
CE3815A
CE3901A
CE3902B
CE3904A
CE3904B
CE3905A
CE3905B
CE3905C
CE3905L
CE3906A
CE3906B
CE3906C
CE3906D
CE3906E
CE3906F
CE3907A
CE3908A
D4A002A
D4A002B
D4A004A
D4A004B
E28002B
E28005D
E52103Y
EA3004G
EB4011A
EB4012A
EB4014A
EE3203A
EE3204A
EE3402B
EE3409F
EE3412C
LA1001F
LA5001A
LA5007A
LA5007B
LA5007C
LA5007D
LA5007E
LA5007F
LA5007G
LA5008A
LA5008B
LA5008C
LA5008D
LA5008E
LA5008F
LA5008G

Ada95 Tests

The following tests use features from Ada95 that are not upwardly compatible from Ada83.

B354001	B952003	BC54A01	C392C07	C760001
B360001	B954001	BC54A02	C392D01	C760002
B390001	B960001	BC54A03	C392D02	C760007
B391001	BA11001	BC54A04	C392D03	C761001
B391002	BA11002	BC54A05	C393001	C761002
B392001	BA11003	BC54A06	C393007	C761003
B392002	BA11004	BC70001	C393008	C761004
B392003	BA11005	BC70002	C393009	C761005
B392004	BA11007	BC70003	C393011	C840001
B392005	BA11008	BC70004	C393012	C854001
B392006	BA11009	BC70005	C393A02	C910001
B392007	BA11010	BC70006	C393A03	C940001
B392008	BA12001	BC70007	C393A05	C940002
B392009	BA12002	BC70008	C393A06	C940005
B393001	BA12003	BC70009	C393B12	C940006
B393002	BA12004	BC70010	C393B13	C940007
B393003	BA12005	BDD2001	C393B14	C940010
B393004	BA12007	BXA8001	C3A0001	C940011
B393006	BA12008	BXAC001	C3A0002	C940012
B3A0001	BA13B01	BXAC002	C3A0003	C940013
B3A0003	BA13B02	BXAC003	C3A0004	C940A03
B3A2002	BB10001	BXAC004	C3A0005	C951001
B3A2003	BC30001	BXAC005	C3A0006	C951002
B3A2004	BC40001	C340001	C3A0007	C954001
B3A2006	BC50001	C340A01	C3A0008	C954010
B3A2007	BC50002	C340A02	C3A0009	C954011
B3A2009	BC51002	C341A01	C3A0010	C954012
B3A2010	BC51003	C341A02	C3A0011	C954013
B460001	BC51004	C341A03	C3A0012	C954014
B641001	BC51005	C341A04	C3A0013	C954015
B730001	BC51006	C390001	C3A0014	C954016
B730002	BC51007	C390002	C3A2001	C954017
B730003	BC51011	C390003	C3A2002	C954018
B730004	BC51012	C390004	C3A2003	C954019
B731A01	BC51013	C390005	C431001	C954020
B731A02	BC51015	C390006	C432001	C954021
B740001	BC51016	C390A01	C432002	C954022
B840001	BC51017	C390A02	C432003	C954023
B940001	BC51018	C390A03	C432004	C954024
B940002	BC51019	C391001	C452001	C954A01
B940003	BC51020	C391002	C460001	C954A02
B940004	BC51B01	C392002	C460002	C954A03
B940005	BC51B02	C392003	C640001	C960001
B940006	BC51C01	C392004	C730001	C960002
B940007	BC51C02	C392005	C730002	C960004
B951001	BC54001	C392008	C730003	C974001
B952001	BC54002	C392A01	C730A01	C974002
B952002	BC54003	C392C05	C730A02	C974003

C974004
C974005
C974006
C974007
C974008
C974009
C974010
C974011
C974012
C974013
C974014
CA11001
CA11002
CA11003
CA11004
CA11005
CA11006
CA11007
CA11008
CA11009
CA11010
CA11011
CA11012
CA11013
CA11014
CA11015
CA11016
CA11017
CA11018
CA11019
CA11020
CA11021
CA11022
CA11A01
CA11A02
CA11B01
CA11B02
CA11C01
CA11C02
CA11C03
CA11D01
CA11D02
CA11D03
CA13001
CA13002
CA13003
CA13A01
CA13A02
CB20001
CB20003
CB20004
CB20005
CB20006
CB20007
CB20A02
CB40A01
CB40A02
CB40A03
CB40A04
CB41001
CB41002
CB41003
CB41004
CC30001
CC50001
CC50A01
CC50A02
CC51001
CC51002
CC51003
CC51004
CC51006
CC51007
CC51A01
CC51B03
CC51D01
CC51D02
CC54001
CC54002
CC54003
CC54004
CC70001
CC70002
CC70003
CC70A01
CC70A02
CC70B01
CC70B02
CC70C01
CC70C02
CXA3001
CXA3002
CXA3003
CXA3004
CXA4001
CXA4002
CXA4003
CXA4004
CXA4005
CXA4006
CXA4007
CXA4008
CXA4009
CXA4010
CXA4011
CXA4012
CXA4013
CXA4014
CXA4015
CXA4016
CXA4017
CXA4018
CXA4019
CXA4020
CXA4021
CXA4022
CXA4023
CXA8001
CXA8002
CXA8003
CXA9001
CXA9002
CXAA001
CXAA002
CXAA003
CXAA004
CXAA005
CXAA006
CXAA007
CXAA008
CXAA009
CXAA010
CXAA011
CXAA012
CXAA013
CXAA014
CXAA015
CXAB001
CXAC001
CXAC002
CXAC003
CXACA01
CXACA02
CXACB01
CXACB02
CXACC01
CXB3001
CXB3002
CXB3003
CXB4001
CXB5001

Special Needs Annex Tests

BXC3001	CXC6002	CXD4001	CXDB003	CXG1004
BXC5001	CXC7001	CXD4002	CXDB004	CXG1005
BXC6001	CXD1001	CXD4003	CXE1001	LXD7001
BXE2A01	CXD1002	CXD4004	CXE5001	LXD7003
BXE2A02	CXD1003	CXD4005	CXF1001	LXD7004
BXE2A03	CXD1004	CXD4006	CXF2001	LXD7005
BXE2A04	CXD1005	CXD5001	CXF3A01	LXD7006
BXE2A05	CXD2001	CXD8001	CXF3A02	LXD7007
BXE2A06	CXD2003	CXDA001	CXF3A03	LXD7008
BXF1001	CXD2004	CXDA002	CXG1001	
CXC3001	CXD3001	CXDB001	CXG1002	
CXC6001	CXD3002	CXDB002	CXG1003	

Tests With Special Requirements

BXC5001	CXD2003	CXD4001	CXD4006	CXG1003
CXD1004	CXD2004	CXD4003	CXE1001	CXG1004
CXD1005	CXD3001	CXD4004	CXG1001	CXG1005
CXD2001	CXD3002	CXD4005	CXG1002	

Tests That Depend on ImpDef

C940001	C954012	C954024	C974009	CXD4003
C940002	C954013	C954A01	C974010	CXD4004
C940005	C954014	C954A02	C974011	CXD4005
C940007	C954015	C954A03	C974012	CXD4006
C940010	C954016	C960001	C974013	CXD5001
C940013	C954017	C960002	CB20001	CXDA002
C940A03	C954018	C974001	CXC3001	CXDB001
C951001	C954019	C974002	CXD2001	CXDB003
C951002	C954020	C974003	CXD2003	CXDB004
C954001	C954021	C974004	CXD2004	CXG1005
C954010	C954022	C974005	CXD4001	LXD7008
C954011	C954023	C974008	CXD4002	

Tests Deleted From 2.0

B23003D	B63102A	BD2A11A	C35711A	C37215A
B23003E	B74103B	BD2A13B	C35711B	C37215C
B23003F	B74103C	BD4008B	C35A03C	C37215E
B34014Q	B74103F	BD5002A	C35A07N	C37215G
B34014S	B74103H	BD5002F	C35A07Q	C37216A
B34014Z	B74205B	BD5101A	C36105B	C393010
B35505D	B74303A	BD5101B	C37213A	C42005A
B38204A	B95074E	BD5101C	C37213C	C43004B
B48002F	BA11006	BD8003A	C37213E	C43213A
B54A08A	BC3205C	C35003F	C37213G	C44003E
B55B14B	BC3220B	C35505D	C37214A	C45232A

C45242A	C52012B	CC3208A	CC3407C	CD1D02A
C45345A	C52103S	CC3208B	CC3407D	CD1D03A
C45423A	C55B08A	CC3208C	CC3407E	CDA101B
C45652A	C64105E	CC3406A	CC3407F	CE2401D
C46023A	C64105F	CC3406B	CC3408A	CE2401G
C48005C	C87B09B	CC3406C	CC3408B	CE3208A
C48008B	C87B35B	CC3406D	CC3408C	CXACC02
C48008D	C96005C	CC3407A	CC3408D	
C52012A	CA3009A	CC3407B	CD1C04B	

Tests Renamed From 2.0

Old Name	New Name
BC51014	B393006
C392B04	C392008

Test Files Split from 2.0

BA1101E was split into BA1101E0 and BA1101E1

BA1101C5 was split into BA1101C5 and BA1101C6

CA5004B was split into CA5004B0 and CA5004B1

[This page intentionally blank]

Appendix B. Ada95 Feature Categories

Ada95 tests have been grouped into five categories to aid technology transition:

- **Real-Time**: This subset is composed of tests for the new Ada 95 features from Section 9: Tasks and Synchronization. These features include protected objects, modifications to task types, select statements, and delay alternatives.

- **OOP**: This subset of tests focuses on some necessary facilities for achieving object-oriented programming in Ada 95. Features validated include tagged types, class attributes, and abstract types and subprograms. Other Ada 95 facilities commonly used in object-oriented programs are included in subsequent subsets.

- **Type Extensions in Child Units**: Tests that focus on the interaction of the two new Ada features of type extensions of tagged types and child library units. This includes the related semantics of visibility, accessibility, and calls on primitive operations of tagged types.

- **Child Library Units**: Tests that focus on the support for the new Ada capability to provide a hierarchical organization of the compilation units of an Ada program with the associated capabilities of granting access to the contents of private declarations and of hiding selected units within subsystems.

- **Pre-defined Language Environment**: This subset of tests includes some Ada 83 facilities and some new features defined in Annex A. Annex A provides specifications for root library units for Ada, Interface, and System, character and string handling, and input/output.

- **Mixed Features**: This relatively large subset of tests focuses on the interaction of Ada features that are a mixture of familiar Ada 83 and new Ada 95 features.

Real-time Tests:

B940001	C910001	C954011	C954023	C974011
B940002	C940001	C954013	C954024	C974012
B940003	C940002	C954014	C974001	CB20001
B940004	C940006	C954015	C974003	CB20004
B940006	C940010	C954016	C974004	CB20005
B940007	C940011	C954017	C974005	CB20006
B951001	C940012	C954018	C974006	CB20007
B952001	C951001	C954019	C974007	
B952002	C951002	C954020	C974008	
B952003	C954001	C954021	C974009	
B954001	C954010	C954022	C974010	

OOP Tests:

B360001	B392003	B393002	B3A2004	B3A2010
B390001	B392004	B393003	B3A2006	B460001
B391001	B392009	B3A2002	B3A2007	B730001
B392002	B393001	B3A2003	B3A2009	B730002

C340A01	C392C05	C3A0002	C3A0009	C460002
C340A02	C392C07	C3A0003	C3A0010	C640001
C392002	C393009	C3A0004	C3A0012	C730001
C392003	C393012	C3A0005	C3A0013	C730A01
C392004	C393A03	C3A0006	C3A2003	C730A02
C392A01	C393B12	C3A0007	C432003	CB20003
C392008	C3A0001	C3A0008	C432004	

Type Extensions in Child Unit Tests:

B392005	BA13B02	C390A03	CA11006	CA11A01
B730003	C390003	C393B13	CA11007	CA11A02
B730004	C390004	C393B14	CA11008	CA11C01
B731A01	C390005	C730003	CA11009	CA11C02
B731A02	C390006	C760007	CA11010	CA11C03
BA11001	C390A01	C761001	CA11011	CC51007
BA13B01	C390A02	C761002	CA11019	

Child Library Tests:

BA11002	BA12001	CA11012	CA11B01	CA13A02
BA11003	BA12002	CA11013	CA11B02	CB40A01
BA11004	BA12003	CA11015	CA11D01	CB40A02
BA11005	BA12004	CA11016	CA11D02	CB40A03
BA11007	BA12005	CA11017	CA11D03	CB40A04
BA11008	BA12008	CA11020	CA13002	
BA11009	CA11001	CA11021	CA13003	
BA11010	CA11002	CA11022	CA13A01	

Predefined Environment Tests:

CXA3001	CXA4003	CXA4009	CXA4015	CXA4021
CXA3002	CXA4004	CXA4010	CXA4016	CXA4022
CXA3003	CXA4005	CXA4011	CXA4017	CXA4023
CXA3004	CXA4006	CXA4012	CXA4018	
CXA4001	CXA4007	CXA4013	CXA4019	
CXA4002	CXA4008	CXA4014	CXA4020	

Mixed Feature Tests:

B354001	BC30001	BC51019	BC70003	C341A02
B391002	BC40001	BC51020	BC70004	C341A03
B392001	BC50001	BC51B01	BC70005	C341A04
B392006	BC50002	BC51B02	BC70006	C390001
B392007	BC51002	BC51C01	BC70007	C390002
B392008	BC51003	BC51C02	BC70008	C391001
B393004	BC51004	BC54001	BC70009	C391002
B393006	BC51005	BC54002	BC70010	C392005
B3A0001	BC51006	BC54003	BDD2001	C392D01
B3A0003	BC51007	BC54A01	BXA8001	C392D02
B641001	BC51011	BC54A02	BXAC001	C392D03
B740001	BC51012	BC54A03	BXAC002	C393001
B840001	BC51013	BC54A04	BXAC003	C393007
B940005	BC51015	BC54A05	BXAC004	C393008
B960001	BC51016	BC54A06	BXAC005	C393011
BA12007	BC51017	BC70001	C340001	C393A02
BB10001	BC51018	BC70002	C341A01	C393A05

C393A06	C940A03	CB41004	CC70A01	CXAA011
C3A0011	C954012	CC30001	CC70A02	CXAA012
C3A0014	C954A01	CC50001	CC70B01	CXAA013
C3A2001	C954A02	CC50A01	CC70B02	CXAA014
C3A2002	C954A03	CC50A02	CC70C01	CXAA015
C431001	C960001	CC51001	CC70C02	CXAB001
C432001	C960002	CC51002	CXA8001	CXAC001
C432002	C960004	CC51003	CXA8002	CXAC002
C452001	C974002	CC51004	CXA8003	CXAC003
C460001	C974013	CC51006	CXA9001	CXACA01
C730002	C974014	CC51A01	CXA9002	CXACA02
C760001	CA11003	CC51B03	CXAA001	CXACB01
C760002	CA11004	CC51D01	CXAA002	CXACB02
C761003	CA11005	CC51D02	CXAA003	CXACC01
C761004	CA11014	CC54001	CXAA004	CXB3001
C761005	CA11018	CC54002	CXAA005	CXB3002
C840001	CA13001	CC54003	CXAA006	CXB3003
C854001	CB20A02	CC54004	CXAA007	CXB4001
C940005	CB41001	CC70001	CXAA008	CXB5001
C940007	CB41002	CC70002	CXAA009	
C940013	CB41003	CC70003	CXAA010	

[This page intentionally blank]

Appendix C. Parameterization Files

MACRO.DFS

```
-- 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. 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.
-- B. 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.

-- $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

-- $MAX_STRING_LITERAL
-- A STRING LITERAL CONSISTING OF $MAX_IN_LEN CHARACTERS (INCLUDING THE
-- QUOTE CHARACTERS).
-- USED IN:  A26007A
MAX_STRING_LITERAL
          "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

-- $BIG_ID1
```

```
-- AN IDENTIFIER IN WHICH THE NUMBER OF CHARACTERS IS $MAX_IN_LEN.
-- THE MACROSUB PROGRAM WILL SUPPLY AN IDENTIFIER IN WHICH THE
-- LAST CHARACTER IS '1' AND ALL OTHERS ARE 'A'.
-- USED IN:  C23003A  C23003B  C23003G  C23003I
--           C35502D  C35502F
BIG_ID1  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA1

-- $BIG_ID2
-- AN IDENTIFIER IN WHICH THE NUMBER OF CHARACTERS IS $MAX_IN_LEN,
-- DIFFERING FROM $BIG_ID1 ONLY IN THE LAST CHARACTER.  THE MACROSUB
-- PROGRAM WILL USE '2' AS THE LAST CHARACTER.
-- USED IN:  C23003A  C23003B  B23003F  C23003G  C23003I
BIG_ID2  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA2

-- $BIG_ID3
-- AN IDENTIFIER IN WHICH THE NUMBER OF CHARACTERS IS $MAX_IN_LEN.
-- MACROSUB WILL USE '3' AS THE "MIDDLE" CHARACTER; ALL OTHERS ARE 'A'.
-- USED IN:  C23003A  C23003B  C23003G  C23003I
BIG_ID3  AAAAAAAAAAAAAAAAAAAAAAAAAAAAA3AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

-- $BIG_ID4
-- AN IDENTIFIER IN WHICH THE NUMBER OF CHARACTERS IS $MAX_IN_LEN,
-- DIFFERING FROM $BIG_ID3 ONLY IN THE MIDDLE CHARACTER.  MACROSUB
-- WILL USE '4' AS THE MIDDLE CHARACTER.
-- USED IN:  C23003A  C23003B  C23003G  C23003I
BIG_ID4  AAAAAAAAAAAAAAAAAAAAAAAAAAAAA4AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

-- $BIG_STRING1
-- A STRING LITERAL (WITH QUOTES) WHOSE CATENATION WITH $BIG_STRING2
-- ($BIG_STRING1 & $BIG_STRING2) PRODUCES THE IMAGE OF $BIG_ID1.
-- USED IN:  C35502D  C35502F
BIG_STRING1  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

-- $BIG_STRING2
-- A STRING LITERAL (WITH QUOTES) WHOSE CATENATION WITH $BIG_STRING1
-- ($BIG_STRING1 & $BIG_STRING2) PRODUCES THE IMAGE OF $BIG_ID1.
-- USED IN:  C35502D  C35502F
BIG_STRING2  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAA1"

-- $BLANKS
-- A SEQUENCE OF ($MAX_IN_LEN - 20) BLANKS.
-- USED IN:  B22001A  B22001B  B22001C  B22001D  B22001E  B22001F
--           B22001G  B22001I  B22001J  B22001K  B22001L  B22001M
--           B22001N
--      <       LIMITS OF SAMPLE SHOWN BY ANGLE BRACKETS           >
BLANKS

-- $ACC_SIZE
-- AN INTEGER LITERAL WHOSE VALUE IS THE MINIMUM NUMBER OF BITS
-- SUFFICIENT TO HOLD ANY VALUE OF AN ACCESS TYPE.
-- USED IN:  CD2A83C  BD2A02A
ACC_SIZE                  32

-- $ALIGNMENT
-- A VALUE THAT IS LEGITIMATE FOR USE IN A RECORD ALIGNMENT CLAUSE.
-- USED IN:  CD4041A BD4006A
ALIGNMENT 4
```

```
-- $COUNT_LAST
-- AN INTEGER LITERAL WHOSE VALUE IS TEXT_IO.COUNT'LAST.
-- USED IN:  CE3002B
COUNT_LAST                32_767

-- $ENTRY_ADDRESS
-- AN EXPRESSION YIELDING A LEGAL ADDRESS FOR A TASK ENTRY
-- (I.E., FOR AN INTERRUPT) FOR THIS IMPLEMENTATION.
-- USED IN:  SPPRT13SP
ENTRY_ADDRESS  16#40#

-- $ENTRY_ADDRESS1
-- AN EXPRESSION YIELDING A LEGAL ADDRESS FOR A TASK ENTRY
-- (I.E., FOR AN INTERRUPT) FOR THIS IMPLEMENTATION.  THE ADDRESS
-- MUST BE DISTINCT FROM THAT USED IN $ENTRY_ADDRESS.
-- USED IN:  SPPRT13SP
ENTRY_ADDRESS1  16#80#

-- $ENTRY_ADDRESS2
-- AN EXPRESSION YIELDING A LEGAL ADDRESS FOR A TASK ENTRY
-- (I.E., FOR AN INTERRUPT) FOR THIS IMPLEMENTATION.  THE ADDRESS
-- MUST BE DISTINCT FROM THOSE USED IN $ENTRY_ADDRESS
-- AND $ENTRY_ADDRESS1.
-- USED IN:  SPPRT13SP
ENTRY_ADDRESS2  16#100#

-- $FIELD_LAST
-- AN INTEGER LITERAL WHOSE VALUE IS TEXT_IO.FIELD'LAST.
-- USED IN:  CE3002C
FIELD_LAST                32_767

-- $FORM_STRING
-- A STRING LITERAL SPECIFYING THAT THE EXTERNAL FILE MEETS BOTH
-- CONDITIONS: (1) THERE IS A VALUE OF TYPE TEXT_IO.COUNT THAT IS NOT
-- AN APPROPRIATE LINE-LENGTH FOR THE FILE, (2) THERE IS A VALUE
-- OF TYPE TEXT_IO.COUNT THAT IS NOT AN APPROPRIATE PAGE-LENGTH
-- FOR THE FILE.
-- IF IT IS NOT POSSIBLE TO SATISFY BOTH CONDITIONS, THEN SUBSTITUTE
-- A STRING LITERAL SPECIFYING THAT THE EXTERNAL FILE SATISFIES ONE
-- OF THE CONDITIONS.  IF IT IS NOT POSSIBLE TO SATISFY EITHER CONDITION,
-- THEN SUBSTITUTE THE NULL STRING ("").
-- USED IN:  CE3304A
FORM_STRING               ""

-- $FORM_STRING2
-- A STRING LITERAL SPECIFYING THAT THE CAPACITY OF THE FILE IS
-- RESTRICTED TO 4096 CHARACTERS OR LESS. IF THE IMPLEMENTATION
-- CANNOT RESTRICT FILE CAPACITY, $FORM_STRING2 SHOULD EQUAL
-- "CANNOT_RESTRICT_FILE_CAPACITY".
-- USED IN:  CE2203A  CE2403A
FORM_STRING2              "CANNOT_RESTRICT_FILE_CAPACITY"

-- $GREATER_THAN_DURATION
-- A REAL LITERAL WHOSE VALUE (NOT SUBJECT TO ROUND-OFF ERROR
-- IF POSSIBLE) LIES BETWEEN DURATION'BASE'LAST AND DURATION'LAST.  IF
-- NO SUCH VALUES EXIST, USE A VALUE IN DURATION'RANGE.
-- USED IN:  C96005B
GREATER_THAN_DURATION     75_000.0
```

```
-- $GREATER_THAN_FLOAT_SAFE_LARGE
-- A REAL LITERAL HAVING A VALUE WHICH IS GREATER THAN FLOAT'SAFE_LARGE,
-- BUT WITHIN THE RANGE OF THE BASE TYPE.
-- USED IN:  C45423A
GREATER_THAN_FLOAT_SAFE_LARGE  1.0E308

-- $ILLEGAL_EXTERNAL_FILE_NAME1
-- AN ILLEGAL EXTERNAL FILE NAME (E.G., TOO LONG, CONTAINING INVALID
-- CHARACTERS, CONTAINING WILD-CARD CHARACTERS, OR SPECIFYING A
-- NONEXISTENT DIRECTORY).
-- USED IN:  CE2102C  CE2102H  CE2103A  CE2103B  CE3102B  CE3107A
ILLEGAL_EXTERNAL_FILE_NAME1       \NODIRECTORY\FILENAME

-- $ILLEGAL_EXTERNAL_FILE_NAME2
-- AN ILLEGAL EXTERNAL FILE NAME, DIFFERENT FROM $ILLEGAL_EXTERNAL_FILE_NAME1.
-- USED IN:  CE2102C  CE2102H  CE2103A  CE2103B  CE3102B
ILLEGAL_EXTERNAL_FILE_NAME2       THIS-FILE-NAME-IS-TOO-LONG-FOR-MY-SYSTEM

-- $INAPPROPRIATE_LINE_LENGTH
-- A LITERAL OF TYPE COUNT THAT IS INAPPROPRIATE AS THE LINE-LENGTH
-- FOR THE EXTERNAL FILE.  IF THERE IS NO SUCH VALUE, THEN USE -1.
-- USED IN:  CE3304A
INAPPROPRIATE_LINE_LENGTH -1

-- $INAPPROPRIATE_PAGE_LENGTH
-- A LITERAL OF TYPE COUNT THAT IS INAPPROPRIATE AS THE PAGE-LENGTH
-- FOR THE EXTERNAL FILE.  IF THERE IS NO SUCH VALUE, THEN USE -1.
-- USED IN:  CE3304A
INAPPROPRIATE_PAGE_LENGTH -1

-- $INTEGER_FIRST
-- AN INTEGER LITERAL, WITH SIGN, WHOSE VALUE IS INTEGER'FIRST.
-- THE LITERAL MUST NOT INCLUDE UNDERSCORES OR LEADING OR TRAILING
-- BLANKS.
-- USED IN:  C35503F  B54B01B
INTEGER_FIRST             -32768

-- $INTEGER_LAST
-- AN INTEGER LITERAL WHOSE VALUE IS INTEGER'LAST.  THE LITERAL MUST
-- NOT INCLUDE UNDERSCORES OR LEADING OR TRAILING BLANKS.
-- USED IN:  C35503F  B54B01B
INTEGER_LAST              32767


-- $LESS_THAN_DURATION
-- A REAL LITERAL (WITH SIGN) WHOSE VALUE (NOT SUBJECT TO
-- ROUND-OFF ERROR IF POSSIBLE) LIES BETWEEN DURATION'BASE'FIRST AND
-- DURATION'FIRST.  IF NO SUCH VALUES EXIST, USE A VALUE IN
-- DURATION'RANGE.
-- USED IN:  C96005B
LESS_THAN_DURATION        -75_000.0


-- $MACHINE_CODE_STATEMENT
-- A VALID MACHINE CODE STATEMENT AS SPECIFIED IN THE PACKAGE
-- MACHINE_CODE.  IF THE IMPLEMENTATION DOES NOT SUPPORT MACHINE
-- CODE THEN USE THE ADA NULL STATEMENT (I.E. NULL; ).
```

```
-- USED IN:  AD8011A  BD8001A  BD8002A  BD8004A  BD8004B
MACHINE_CODE_STATEMENT    NULL;

-- $MAX_INT
-- AN INTEGER LITERAL WHOSE VALUE IS SYSTEM.MAX_INT.
-- THE LITERAL MUST NOT INCLUDE UNDERSCORES OR LEADING OR TRAILING
-- BLANKS.
-- USED IN:  C35503D  C35503F  C4A007A
MAX_INT                   2147483647


-- $MIN_INT
-- AN INTEGER LITERAL, WITH SIGN, WHOSE VALUE IS SYSTEM.MIN_INT.
-- THE LITERAL MUST NOT CONTAIN UNDERSCORES OR LEADING OR TRAILING
-- BLANKS.
-- USED IN:  C35503D  C35503F
MIN_INT                   -2147483648

-- $NAME
-- THE NAME OF A PREDEFINED INTEGER TYPE OTHER THAN INTEGER,
-- SHORT_INTEGER, OR LONG_INTEGER.
-- (IMPLEMENTATIONS WHICH HAVE NO SUCH TYPES SHOULD USE AN UNDEFINED
-- IDENTIFIER SUCH AS   NO_SUCH_TYPE_AVAILABLE.)
-- USED IN:  C45231D  CD7101G
NAME  SHORT_SHORT_INTEGER

-- $NAME_SPECIFICATION1
-- THE FULL SPECIFICATION FOR THE FILENAME RETURNED FROM THE FUNCTION
-- LEGAL_FILE_NAME WHEN CALLED FROM TEST CE2120A.
-- USED IN:  CE2120A
NAME_SPECIFICATION1  DISK$AWC_2:[CROCKETTL.ACVC20.DEVELOPMENT]X2120A.;1

-- $NAME_SPECIFICATION2
-- THE FULL SPECIFICATION FOR THE FILENAME RETURNED FROM THE FUNCTION
-- LEGAL_FILE_NAME WHEN CALLED FROM TEST CE2120B.
-- USED IN:  CE2120B
NAME_SPECIFICATION2   DISK$AWC_2:[CROCKETTL.ACVC20.DEVELOPMENT]X2120B.;1

-- $NAME_SPECIFICATION3
-- THE FULL SPECIFICATION FOR THE FILENAME RETURNED FROM THE FUNCTION
-- LEGAL_FILE_NAME WHEN CALLED FOM TEST CE3119A.
-- USED IN:  CE3119A
NAME_SPECIFICATION3   DISK$AWC_2:[CROCKETTL.ACVC20.DEVELOPMENT]X2120C.;1

-- $OPTIONAL_DISC
-- A DISCRIMINANT USED AS THE DISCRIMINANT PART OF $RECORD_NAME.
-- IF MACHINE CODE INSERTIONS ARE NOT SUPPORTED THEN SUBSTITUTE
-- NO_SUCH_MACHINE_CODE_DISC.
-- USED IN:  BD8002A
OPTIONAL_DISC             NO_SUCH_MACHINE_CODE_DISC

-- $RECORD_DEFINITION
-- THE RECORD TYPE DEFINITION (WITH FINAL SEMICOLON) FOR THE TYPE THAT
-- WAS USED IN THE MACRO $RECORD_NAME, AS DECLARED IN PACKAGE
-- MACHINE_CODE.  IF THE IMPLEMENTATION DOES NOT SUPPORT MACHINE CODE,
-- THEN USE A NULL RECORD DEFINITION
-- USED IN:  BD8002A
RECORD_DEFINITION         RECORD  NULL; END RECORD;
```

```
-- $RECORD_NAME
-- A VALID RECORD TYPE NAME THAT IS DEFINED IN PACKAGE MACHINE_CODE.
-- IF THE IMPLEMENTATION DOES NOT SUPPORT MACHINE CODE THEN
-- USE THE NAME "NO_SUCH_MACHINE_CODE_TYPE"
-- USED IN:  BD8002A
RECORD_NAME    NO_SUCH_MACHINE_CODE_TYPE

-- $TASK_SIZE
-- AN INTEGER LITERAL WHOSE VALUE IS THE NUMBER OF BITS REQUIRED TO
-- HOLD A TASK OBJECT.
-- USED IN:  CD2A91C
TASK_SIZE                 128

-- $TASK_STORAGE_SIZE
-- THE NUMBER OF STORAGE UNITS REQUIRED FOR A TASK ACTIVATION.
-- USED IN:  BD2C01D  BD2C02A  BD2C03A  C87B62D  CD1009K  CD1009T
--           CD1009U  CD1C03E  CD1C06A  CD2C11A  CC1225A  CD2C11D
TASK_STORAGE_SIZE 1024

-- $VARIABLE_ADDRESS
-- AN EXPRESSION YIELDING A LEGAL ADDRESS FOR A VARIABLE FOR THIS
-- IMPLEMENTATION.
-- USED IN:  SPPRT13SP
VARIABLE_ADDRESS  16#0020#

-- $VARIABLE_ADDRESS1
-- AN EXPRESSION YIELDING A LEGAL ADDRESS FOR A VARIABLE FOR THIS
-- IMPLEMENTATION.  THE ADDRESS MUST BE DISTINCT FROM THAT USED IN
-- THE MACRO $VARIABLE_ADDRESS.
-- USED IN:  SPPRT13SP
VARIABLE_ADDRESS1  16#0024#

-- $VARIABLE_ADDRESS2
-- AN EXPRESSION YIELDING A LEGAL ADDRESS FOR A VARIABLE FOR THIS
-- IMPLEMENTATION.  THE ADDRESS MUST BE DISTINCT FROM THOSE USED IN
-- THE MACROS $VARIABLE_ADDRESS AND $VARIABLE_ADDRESS1.
-- USED IN:  SPPRT13SP
VARIABLE_ADDRESS2  16#0028#
```

TSTTESTS.DAT

There are 65 .TST files that require macro substitution:

\ACVC_201\9XBasic\ATests\A2\A26007A.TST
\ACVC_201\9XBasic\ATests\AD\AD8011A.TST
\ACVC_201\9XBasic\BTests\B2\B22001A.TST
\ACVC_201\9XBasic\BTests\B2\B22001B.TST
\ACVC_201\9XBasic\BTests\B2\B22001C.TST
\ACVC_201\9XBasic\BTests\B2\B22001D.TST
\ACVC_201\9XBasic\BTests\B2\B22001E.TST
\ACVC_201\9XBasic\BTests\B2\B22001F.TST
\ACVC_201\9XBasic\BTests\B2\B22001G.TST
\ACVC_201\9XBasic\BTests\B2\B22001I.TST
\ACVC_201\9XBasic\BTests\B2\B22001J.TST
\ACVC_201\9XBasic\BTests\B2\B22001K.TST
\ACVC_201\9XBasic\BTests\B2\B22001L.TST
\ACVC_201\9XBasic\BTests\B2\B22001M.TST
\ACVC_201\9XBasic\BTests\B2\B22001N.TST
\ACVC_201\9XBasic\BTests\B5\B54B01B.TST
\ACVC_201\9XBasic\BTests\BD\BD2A02A.TST
\ACVC_201\9XBasic\BTests\BD\BD2C01D.TST
\ACVC_201\9XBasic\BTests\BD\BD2C02A.TST
\ACVC_201\9XBasic\BTests\BD\BD2C03A.TST
\ACVC_201\9XBasic\BTests\BD\BD4006A.TST
\ACVC_201\9XBasic\BTests\BD\BD8001A.TST
\ACVC_201\9XBasic\BTests\BD\BD8002A.TST
\ACVC_201\9XBasic\BTests\BD\BD8004A.TST
\ACVC_201\9XBasic\BTests\BD\BD8004B.TST
\ACVC_201\9XBasic\BTests\BD\BD8004C.TST
\ACVC_201\9XBasic\CTests\C2\C23003A.TST
\ACVC_201\9XBasic\CTests\C2\C23003B.TST
\ACVC_201\9XBasic\CTests\C2\C23003G.TST
\ACVC_201\9XBasic\CTests\C2\C23003I.TST
\ACVC_201\9XBasic\CTests\C3\C35502D.TST
\ACVC_201\9XBasic\CTests\C3\C35502F.TST
\ACVC_201\9XBasic\CTests\C3\C35503D.TST
\ACVC_201\9XBasic\CTests\C3\C35503F.TST
\ACVC_201\9XBasic\CTests\C4\C45231D.TST
\ACVC_201\9XBasic\CTests\C4\C4A007A.TST
\ACVC_201\9XBasic\CTests\C8\C87B62D.TST
\ACVC_201\9XBasic\CTests\C9\C96005B.TST
\ACVC_201\9XBasic\CTests\CC\CC1225A.TST
\ACVC_201\9XBasic\CTests\CD\CD1009K.TST
\ACVC_201\9XBasic\CTests\CD\CD1009T.TST
\ACVC_201\9XBasic\CTests\CD\CD1009U.TST
\ACVC_201\9XBasic\CTests\CD\CD1C03E.TST
\ACVC_201\9XBasic\CTests\CD\CD1C06A.TST
\ACVC_201\9XBasic\CTests\CD\CD2A83C.TST
\ACVC_201\9XBasic\CTests\CD\CD2A91C.TST
\ACVC_201\9XBasic\CTests\CD\CD2C11A.TST
\ACVC_201\9XBasic\CTests\CD\CD2C11D.TST
\ACVC_201\9XBasic\CTests\CD\CD4041A.TST
\ACVC_201\9XBasic\CTests\CD\CD7101G.TST
\ACVC_201\9XBasic\CTests\CE\CE2102C.TST
\ACVC_201\9XBasic\CTests\CE\CE2102H.TST
\ACVC_201\9XBasic\CTests\CE\CE2103A.TST
\ACVC_201\9XBasic\CTests\CE\CE2103B.TST
\ACVC_201\9XBasic\CTests\CE\CE2120A.TST
\ACVC_201\9XBasic\CTests\CE\CE2120B.TST
\ACVC_201\9XBasic\CTests\CE\CE2203A.TST
\ACVC_201\9XBasic\CTests\CE\CE2403A.TST
\ACVC_201\9XBasic\CTests\CE\CE3002B.TST
\ACVC_201\9XBasic\CTests\CE\CE3002C.TST
\ACVC_201\9XBasic\CTests\CE\CE3102B.TST
\ACVC_201\9XBasic\CTests\CE\CE3107A.TST
\ACVC_201\9XBasic\CTests\CE\CE3119A.TST
\ACVC_201\9XBasic\CTests\CE\CE3304A.TST
\ACVC_201\Support\SPPRT13S.TST

Package ImpDef

```
-- IMPDEF.A

with Report;
-- the following With (and associated constants) may be deleted
-- for implementations not validating the Systems Programming Annex
with Ada.Interrupts.Names;

Package ImpDef is

   -- This package provides the tailorable constants for a particular
   -- implementation.  Each constant may be modified to suit the needs
   -- of the implementation and default values are provided to act as
   -- a guide.
   --
   -- The descriptions are necessarily vague: how long does it take a
   -- task to run to its first accept statement?  Obviously it depends
   -- on many factors.  In this case we are considering a simple task
   -- with very few Ada statements before the accept.  An
   -- implementation is free to specify a delay of several seconds, or
   -- even minutes if need be.  The main effect of specifying a longer
   -- delay than necessary will be an extension of the time needed to
   -- run the suite of tests.

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

   -- This is the minimum time required to allow another task to get
   -- control.  It is expected that the task is on the Ready queue.
   -- A duration of 0.0 would normally be sufficient but some number
   -- greater than that is expected.
   --
   Minimum_Task_Switch : constant duration := 0.1;
   --                                         ^^^ --- MODIFY HERE AS NEEDED

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

   -- This is the time required to activate another task and allow it
   -- to run to its first accept statement.
   --
   Switch_To_New_Task : constant duration := 1.0;
   --                                        ^^^ -- MODIFY HERE AS NEEDED

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

   -- This is the time which will clear the queues of other tasks
   -- waiting to run.  It is expected that this will be about five
   -- times greater than Switch_To_New_Task.
   --
   Clear_Ready_Queue : constant duration := 5.0;
   --                                       ^^^ --- MODIFY HERE AS NEEDED

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

   -- Some implementations will boot with the time set to 1901/1/1/0.0
```

```
   -- This constant is such that when a delay of Delay_For_Time_Past is given,
   -- the implementation guarantees that a subsequent call to
   -- Ada.Calendar.Time_Of(1901,1,1) will yield a time that has already passed
   -- (for example, when used in a delay_until statement)
   --
   Delay_For_Time_Past : constant duration := 0.1;
   --                                         ^^^ --- MODIFY HERE AS NEEDED

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

   -- The following constant only applies to implementations validating
   -- against the Real-Time Annex.
   -- This constant is the maximum storage size that can be specified
   -- for a task.  A single task that has this size must be able to
   -- run.  Ideally, this value is large enough that two tasks of this
   -- size cannot run at the same time.  If the value is too small then
   -- test CXDC001 may take longer to run.  See the test for further
   -- information.
   Maximum_Task_Storage_Size : constant := 16_000_000;

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

   -- Minimum time interval between calls to the time dependent Reset
   -- procedures in Float_Random and Discrete_Random packages that is
   -- guaranteed to initiate different sequences.
   -- RM A.5.2(45);6.0
   --
   Time_Dependent_Reset : constant duration := 0.3;
   --                                          ^^^ --- MODIFY HERE AS NEEDED

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

   -- Test CXA5013 will loop, trying to generate the required sequence
   -- of random numbers.  If the RNG is faulty, the required sequence
   -- will never by generated.  Delay_Per_Random_Test is a time-out value
   -- which allows the test to run for a period of time after which the
   -- test is failed if the required sequence has not been produced.
   -- This value should be the time allowed for the test to run before it
   -- times out.  It should be long enough to allow multiple (independent)
   -- runs of the testing code, each generating up to 1000 random
   -- numbers.
   --
   Delay_Per_Random_Test : constant duration := 1.0;
   --                                           ^^^ --- MODIFY HERE AS NEEDED

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

   -- Test CXA5014 determines whether a Random Number Generator produces
   -- a uniform distribution of random numbers.  Discrete_Sample_Size
   -- and Float_Sample_Size should be large enough so that a sequence of
   -- that many random numbers will exhibit approximately uniform
   -- distribution.
   --
   -- Note: Each value *must* be at least 100 (Test failure if less).
   --
   --
   Discrete_Sample_Size : constant := 10000;
   --                                 ^^^^^ --- MODIFY HERE AS NEEDED
```

```
   --
   Float_Sample_Size    : constant :=  1000;
   --                                  ^^^^ --- MODIFY HERE AS NEEDED

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

   -- This is a procedure who's execution is guaranteed to be greater
   -- than the time slice unit on implementations which use time slicing
   -- For those which do not implement time slicing this could be
   -- null;
   --
   Procedure Exceed_Time_Slice;

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

   -- Indicates the type of processor on which the tests are running.
   --    Time_Slice indicates a uniprocessor with an operating system
   --    that simulates a multi-processor by using time slicing
   --
   type Processor_Type is (Uni_Processor, Time_Slice, Multi_Processor);
   --
   Processor : constant Processor_Type := Uni_Processor;
   --                                     ^^^^^^^^^ --- MODIFY HERE AS NEEDED

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

   -- The next two constants are used for interrupt testing in the
   -- Systems Annex.  If the Systems Annex tests are not going to be
   -- used, it is allowed to delete these two constants, along with the
   -- above reference to Ada.Interrupts.Names;
   --
   -- Interrupt_To_Use_For_Testing should be set to the name of an
   -- interrupt ID that will ideally cause interrupts within the time
   -- interval specified by:
   -- A_Reasonable_Amount_Of_Time_To_Wait_For_An_Interrupt.

   Interrupt_To_Use_For_Testing : constant Ada.Interrupts.Interrupt_ID
     := Ada.Interrupts.Interrupt_ID'First;  -- to allow trivial compilation
   --   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^  --- MODIFY HERE AS NEEDED

   A_Reasonable_Amount_Of_Time_To_Wait_For_An_Interrupt : constant := 10.0;
   --                                       MODIFY HERE AS NEEDED --- ^^^^

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

   -- The Max_RPC_Call_Time value is the longest time a test needs to wait for
   -- an RPC to complete.  Included in this time is the time for the called
   -- procedure to make a task entry call where the task is ready to accept
   -- the call.
   --
   Max_RPC_Call_Time : constant Duration := 2.0;
   --                                       ^^^  --- MODIFY HERE AS NEEDED

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

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

```
   function Negative_Zero return Float;

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

   -- This constant must not depict a random number generator state value.
   -- Using this string in a call to function Value from either the
   -- Discrete_Random or Float_Random packages will result in
   -- Constraint_Error (expected result in CXA5012.A).
   --
   Non_State_String : constant String := "By No Means A State";
   --           MODIFY HERE AS NEEDED --- ^^^^^^^^^^^^^^^^^^^

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

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

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

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

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

    CD30005_1_Foreign_Address : constant := 16#0000_0000#;
  --          MODIFY HERE IF NEEDED --- ^
  --            MODIFY HERE AS REQUIRED --- ^^^^^^^^^^^^^

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

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

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

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

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

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

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

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

  -- The following boolean constant indicates that this validation will
  -- include the Systems Programming Annex (Annex C).
  -- The value of this boolean alters the behavior of the reporting software
  -- (TCTouch) to report certain Chapter 13 tests.  Setting it to True will
```

```
  -- cause Pass/Fail results;
  -- setting it to False will cause Pass/Not_Applicable results.
  --  False means Annex C is NOT to be included
  --  True means Annex C IS to be included

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

end ImpDef;

Package body ImpDef is

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

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

   -- This is a procedure who's execution is guaranteed to be greater
   -- than the time slice unit on implementations which use time slicing
   -- For those which do not implement time slicing this could be null;
   --
   Procedure Exceed_Time_Slice is
      t : integer := 0;
      Loop_Max : constant integer := 4_000;
   begin
      for i in 1..Loop_Max loop
         t := Report.Ident_Int (1) * Report.Ident_Int (2);
      end loop;
   end Exceed_Time_Slice;

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

   -- This function must return a negative zero value for implementations
   -- for which Float'Signed_Zeros is True.
   -- The body provided returns a negative zero.  If this default body
   -- does not provide the appropriate value of negative zero for the
   -- implementation under test, the body must be replaced by one which
   -- returns the appropriate value; the replacement body must return
   -- negative zero.
   --
   function Negative_Zero return Float is
   begin
      return -0.0;      -- Note: If this value is not negative zero for the
                        --       implementation, use of this "default" value
                        --       could result in false failures in
                        --       implementations where Float'Signed_Zeros
                        --       is True.
   end Negative_Zero;

end ImpDef;
```

Appendix D. Output of CZ Tests

Sample expected output from CZ1101A:

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

,.,. PASS_TEST ACVC 2.0.1 95-12-06 14:42:53
---- 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 ACVC 2.0.1 95-12-06 14:42:53
---- FAIL_TEST CHECKING 'FAILED' AND 'RESULT' FOR 'FAILED'.
   * FAIL_TEST 'RESULT' SHOULD NOW BE 'FAILED'.
**** FAIL_TEST FAILED ****************************.

,.,. NA_TEST ACVC 2.0.1 95-12-06 14:42:53
---- NA_TEST CHECKING 'NOT-APPLICABLE'.
   + NA_TEST 'RESULT' SHOULD NOW BE 'NOT-APPLICABLE'.
++++ NA_TEST NOT-APPLICABLE ++++++++++++++++++++.

,.,. FAIL_NA_TEST ACVC 2.0.1 95-12-06 14:42:53
---- 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 ACVC 2.0.1 95-12-06 14:42:53
---- 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 ACVC 2.0.1 95-12-06 14:42:53
---- 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 ACVC 2.0.1 95-12-06 14:42:53
---- CZ1101A CHECKING 'SPECIAL_ACTION' ALONE.
   ! CZ1101A 'RESULT' BECOMES 'TENTATIVELY PASSED'.
```

```
!!!! CZ1101A TENTATIVELY PASSED !!!!!!!!!!!!!!!!.
!!!!         SEE '!' COMMENTS FOR SPECIAL NOTES!!
```

Sample expected output from CZ1102A:

```
,.,. CZ1102A ACVC 2.0.1 95-12-06 14:44:58
---- CZ1102A CHECK THAT THE DYNAMIC VALUE ROUTINES OF THE REPORT PACKAGE
                WORK CORRECTLY.
==== CZ1102A PASSED ============================.
```

Sample expected output from CZ1103A (note that this test passes even though it reports unexpected input and reports "failed" -- see the pass/fail criteria in CZ1103A):

```
,.,. CZ1103A ACVC 2.0.1 95-12-06 14:46:17
---- 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 FROM CHECK_FILE: THIS IMPLEMENTATION PADS LINES WITH
                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 FROM CHECK_FILE: LAST CHARACTER IN FOLLOWING STRING
                REVEALED ERROR: THIS LINE WILL CONTAIN AN #.
   - CZ1103A BEGIN TEST WITH FILE WITH END OF PAGE ERROR.
   * CZ1103A FROM CHECK_FILE: END_OF_PAGE NOT WHERE EXPECTED.
   - CZ1103A FROM CHECK_FILE: LAST CHARACTER IN FOLLOWING STRING
                REVEALED ERROR: THIS LINE WILL CONTAIN AN @.
   - CZ1103A BEGIN TEST WITH FILE WITH END OF FILE ERROR.
   * CZ1103A FROM CHECK_FILE: END_OF_FILE NOT WHERE EXPECTED.
   - CZ1103A FROM CHECK_FILE: LAST CHARACTER IN FOLLOWING STRING
                REVEALED ERROR: THIS LINE WILL CONTAIN AN %.
   - CZ1103A BEGIN TEST WITH 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 ACVC 2.0.1 95-12-06 14:46:18
---- CZ1103A THE LINE ABOVE SHOULD REPORT FAILURE.
   ! CZ1103A COMPARE THIS OUTPUT TO THE EXPECTED RESULT.
!!!! CZ1103A TENTATIVELY PASSED !!!!!!!!!!!!!!!!.
!!!!         SEE '!' COMMENTS FOR SPECIAL NOTES!!
```

Appendix E. Translating DOS File Names

DOS limits file names to a total length of eight characters plus a three character extension. While all Ada95 tests satisfy this restriction, some 9XBasic tests have names that are longer than eight characters (tests which use multiple files; see the naming conventions described in 3.4). This appendix describes how to take file names as included on a DOS delivery diskette and reconstruct file names that match the test names.

The following algorithm is used to generate ACVC test file names consistent with DOS restrictions:

> When a file name is nine characters long (three character file name extensions are not considered), the final character is deleted.

Since the ninth character is always an 'M' (see section 3.4), the only information that is lost is the indication of the file containing the test main subprogram. The following is a list of DOS file names and their translation to full names:

DOS name	Original name	DOS name	Original name
ad7001c0.ada	ad7001c0m.ada	ba1011c0.ada	ba1011c0m.ada
ad7001d0.ada	ad7001d0m.ada	ba1020a0.ada	ba1020a0m.ada
b38103c3.ada	b38103c3m.ada	ba1020b6.ada	ba1020b6m.ada
b38103e0.ada	b38103e0m.ada	ba1020c0.ada	ba1020c0m.ada
b63009c3.ada	b63009c3m.ada	ba1020f2.ada	ba1020f2m.ada
b73004b0.ada	b73004b0m.ada	ba1101b0.ada	ba1101b0m.ada
b83003b0.ada	b83003b0m.ada	ba1101c2.ada	ba1101c2m.ada
b83004b0.ada	b83004b0m.ada	ba1109a2.ada	ba1109a2m.ada
b83004c2.ada	b83004c2m.ada	ba1110a1.ada	ba1110a1m.ada
b83004d0.ada	b83004d0m.ada	ba2001f0.ada	ba2001f0m.ada
b83024f0.ada	b83024f0m.ada	ba2003b0.ada	ba2003b0m.ada
b83e01e0.ada	b83e01e0m.ada	ba2011a1.ada	ba2011a1m.ada
b83e01f0.ada	b83e01f0m.ada	ba3001a0.ada	ba3001a0m.ada
b86001a1.ada	b86001a1m.ada	ba3001b0.ada	ba3001b0m.ada
b95020b2.ada	b95020b2m.ada	ba3001c0.ada	ba3001c0m.ada
ba1001a0.ada	ba1001a0m.ada	ba3001e0.ada	ba3001e0m.ada
ba1010a0.ada	ba1010a0m.ada	ba3001f0.ada	ba3001f0m.ada
ba1010b0.ada	ba1010b0m.ada	ba3006a6.ada	ba3006a6m.ada
ba1010c0.ada	ba1010c0m.ada	ba3006b4.ada	ba3006b4m.ada
ba1010d0.ada	ba1010d0m.ada	c38108c1.ada	c38108c1m.ada
ba1010e0.ada	ba1010e0m.ada	c38108d0.ada	c38108d0m.ada
ba1010f0.ada	ba1010f0m.ada	c39006c0.ada	c39006c0m.ada
ba1010g0.ada	ba1010g0m.ada	c39006f3.ada	c39006f3m.ada
ba1010h0.ada	ba1010h0m.ada	c64005d0.ada	c64005d0m.ada
ba1010i0.ada	ba1010i0m.ada	c83022g0.ada	c83022g0m.ada
ba1010j0.ada	ba1010j0m.ada	c83024e1.ada	c83024e1m.ada
ba1010k0.ada	ba1010k0m.ada	c83f01c2.ada	c83f01c2m.ada
ba1010l0.ada	ba1010l0m.ada	c83f01d0.ada	c83f01d0m.ada
ba1010m0.ada	ba1010m0m.ada	c83f03c2.ada	c83f03c2m.ada
ba1010n0.ada	ba1010n0m.ada	c83f03d0.ada	c83f03d0m.ada
ba1010p0.ada	ba1010p0m.ada	c86004b2.ada	c86004b2m.ada
ba1010q0.ada	ba1010q0m.ada	c86004c2.ada	c86004c2m.ada
ba1011b0.ada	ba1011b0m.ada	ca1011a6.ada	ca1011a6m.ada

DOS name	Original name	DOS name	Original name
ca1012a4.dep	ca1012a4m.dep	ca5004b1.ada	ca5004b1m.ada
ca1012b4.ada	ca1012b4m.ada	cc3019b2.ada	cc3019b2m.ada
ca1013a6.ada	ca1013a6m.ada	cc3019c2.ada	cc3019c2m.ada
ca1014a0.ada	ca1014a0m.ada	ea3004g4.ada	ea3004g4m.ada
ca1020d3.ada	ca1020d3m.ada	la5001a7.ada	la5001a7m.ada
ca1020e3.ada	ca1020e3m.ada	la5007a1.ada	la5007a1m.ada
ca1022a6.ada	ca1022a6m.ada	la5007b1.ada	la5007b1m.ada
ca1102a2.ada	ca1102a2m.ada	la5007c1.ada	la5007c1m.ada
ca2001h3.ada	ca2001h3m.ada	la5007d1.ada	la5007d1m.ada
ca2002a0.ada	ca2002a0m.ada	la5007e1.ada	la5007e1m.ada
ca2003a0.ada	ca2003a0m.ada	la5007f1.ada	la5007f1m.ada
ca2004a0.ada	ca2004a0m.ada	la5007g1.ada	la5007g1m.ada
ca2007a0.ada	ca2007a0m.ada	la5008a1.ada	la5008a1m.ada
ca2008a0.ada	ca2008a0m.ada	la5008b1.ada	la5008b1m.ada
ca2009c0.dep	ca2009c0m.dep	la5008c1.ada	la5008c1m.ada
ca2009f0.dep	ca2009f0m.dep	la5008d1.ada	la5008d1m.ada
ca3011a4.dep	ca3011a4m.dep	la5008e1.ada	la5008e1m.ada
ca5003a6.ada	ca5003a6m.ada	la5008f1.ada	la5008f1m.ada
ca5003b5.ada	ca5003b5m.ada	la5008g1.ada	la5008g1m.ada

Appendix F. 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 AVF may require pre-validation evidence that the tests are properly rejected.)

If file I/O is not supported, then the tests listed in section F.2.14 will not be part of the customized test suite for bare target validations and will not be run on-site.

F.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 must be rejected at compile time 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 Appendix F of the implementation's documentation.

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

F.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	C45304C.DEP

C45411C.DEP	C45504C.DEP	C45613C.DEP	C45632C.DEP
C45502C.DEP	C45504F.DEP	C45614C.DEP	C55B07A.DEP
C45503C.DEP	C45611C.DEP	C45631C.DEP	CD7101F.DEP

F.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
```

F.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	C45532M.DEP	C45532O.DEP
C45531N.DEP	C45531P.DEP	C45532N.DEP	C45532P.DEP

F.1.5 Non-binary, Non-decimal Values of 'SMALL

If 'SMALL representation clauses which are not powers of two are not supported, then the test contained in the following file is not applicable:

```
C45536A.DEP
```

F.1.6 Compiler Rejection of Supposedly Static Expression

Consider the following declarations:

```
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
```

F.1.7 Separate Compilation with Generics

If a generic library-subprogram body cannot be compiled in a separate file from its specification, then the tests contained in the following files are not applicable:

```
BA1010D*.ADA       CA1012A*.DEP
```

If a generic non-library package body cannot be compiled as a subunit in a separate file from its specification, then the test contained in the following files is not applicable:

```
CA2009C*.DEP
```

If a generic non-library subprogram body cannot be compiled as a subunit in a separate file from its specification, then the test contained in the following files is not applicable:

```
CA2009F*.DEP
```

If a generic procedure cannot have subunits which are compiled in separate files from the generic unit, then the test contained in the following files is not applicable:

CA3011A*.DEP

If a generic library-package body cannot be compiled in a separate file from its specification, then the tests contained in the following files are not applicable:

B83004B*.ADA	B83004D*.ADA	B83024F*.ADA

F.1.8 Non-default Size Rep Clauses for Floating Point Types

If representation clauses specifying non-default sizes for floating point types are not supported, then the following tests may be not applicable:

CD1009C.DEP

F.1.9 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	BD8004B.TST
BD8001A.TST	BD8004A.TST	BD8004C.TST

F.1.10 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

F.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 objectives and with the implementation's Appendix F.

F.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:

C45523A.ADA	C45622A.ADA

F.2.2 Value of MACHINE_OVERFLOWS is True

If MACHINE_OVERFLOWS is TRUE for floating point types, then the tests contained in the following files should report NOT_APPLICABLE:

C45624A.ADA	C45624B.ADA

F.2.3 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
```

F.2.4 Floating Point Overflow

Consider the declaration

```
TYPE F IS DIGITS SYSTEM.MAX_DIGITS;
```

If F'MACHINE_OVERFLOWS is 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
```

F.2.5 Type DURATION

If

```
DURATION'FIRST = DURATION'BASE'FIRST or
DURATION'LAST = DURATION'BASE'LAST,
```

then the test contained in the following file should report NOT_APPLICABLE:

```
C96005B.TST
```

F.2.6 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	CE3302A.ADA	CE3408A.ADA	CE3605A.ADA
CE3102A.ADA	CE3304A.TST	CE3408B.ADA	CE3605B.ADA
CE3102B.TST	CE3305A.ADA	CE3408C.ADA	CE3605C.ADA
CE3102F.ADA	CE3401A.ADA	CE3409A.ADA	CE3605D.ADA
CE3102G.ADA	CE3402A.ADA	CE3409C.ADA	CE3605E.ADA
CE3102H.ADA	CE3402C.ADA	CE3409D.ADA	CE3606A.ADA
CE3102J.ADA	CE3402D.ADA	CE3409E.ADA	CE3606B.ADA
CE3102K.ADA	CE3403A.ADA	CE3410A.ADA	CE3704A.ADA
CE3103A.ADA	CE3403B.ADA	CE3410C.ADA	CE3704B.ADA
CE3104A.ADA	CE3403C.ADA	CE3410D.ADA	CE3704C.ADA
CE3104B.ADA	CE3403E.ADA	CE3410E.ADA	CE3704D.ADA
CE3104C.ADA	CE3403F.ADA	CE3411A.ADA	CE3704E.ADA
CE3106A.ADA	CE3404B.ADA	CE3411C.ADA	CE3704F.ADA
CE3106B.ADA	CE3404C.ADA	CE3412A.ADA	CE3704M.ADA
CE3107B.ADA	CE3404D.ADA	CE3413A.ADA	CE3704N.ADA
CE3108A.ADA	CE3405A.ADA	CE3413B.ADA	CE3704O.ADA
CE3108B.ADA	CE3405C.ADA	CE3413C.ADA	CE3705A.ADA
CE3110A.ADA	CE3405D.ADA	CE3414A.ADA	CE3705B.ADA
CE3112C.ADA	CE3406A.ADA	CE3602A.ADA	CE3705C.ADA
CE3112D.ADA	CE3406B.ADA	CE3602B.ADA	CE3705D.ADA
CE3114A.ADA	CE3406C.ADA	CE3602C.ADA	CE3705E.ADA
CE3115A.ADA	CE3406D.ADA	CE3602D.ADA	CE3706D.ADA
CE3119A.TST	CE3407A.ADA	CE3603A.ADA	CE3706F.ADA
CE3207A.ADA	CE3407B.ADA	CE3604A.ADA	CE3706G.ADA
CE3301A.ADA	CE3407C.ADA	CE3604B.ADA	CE3804A.ADA

CE3804B.ADA	CE3806A.ADA	CE3906B.ADA	CXAA011.A
CE3804C.ADA	CE3806B.ADA	CE3906C.ADA	CXAA012.A
CE3804D.ADA	CE3806D.ADA	CE3906E.ADA	CXAA013.A
CE3804E.ADA	CE3806E.ADA	CE3906F.ADA	CXAA014.A
CE3804F.ADA	CE3806G.ADA	CXAA001.A	CXAA015.A
CE3804G.ADA	CE3806H.ADA	CXAA002.A	CXG1003.A
CE3804H.ADA	CE3902B.ADA	CXAA003.A	EE3203A.ADA
CE3804I.ADA	CE3904A.ADA	CXAA004.A	EE3204A.ADA
CE3804J.ADA	CE3904B.ADA	CXAA005.A	EE3402B.ADA
CE3804M.ADA	CE3905A.ADA	CXAA006.A	EE3409F.ADA
CE3804O.ADA	CE3905B.ADA	CXAA007.A	EE3412C.ADA
CE3804P.ADA	CE3905C.ADA	CXAA008.A	
CE3805A.ADA	CE3905L.ADA	CXAA009.A	
CE3805B.ADA	CE3906A.ADA	CXAA010.A	

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

CE3102F.ADA	CE3408A.ADA	CE3704C.ADA	CE3804M.ADA
CE3102H.ADA	CE3408C.ADA	CE3704D.ADA	CE3804P.ADA
CE3104B.ADA	CE3409C.ADA	CE3704E.ADA	CE3805A.ADA
CE3106A.ADA	CE3409D.ADA	CE3704F.ADA	CE3805B.ADA
CE3108A.ADA	CE3409E.ADA	CE3704M.ADA	CE3806A.ADA
CE3108B.ADA	CE3410C.ADA	CE3704N.ADA	CE3806B.ADA
CE3112D.ADA	CE3410D.ADA	CE3704O.ADA	CE3806D.ADA
CE3301A.ADA	CE3410E.ADA	CE3705B.ADA	CE3806G.ADA
CE3302A.ADA	CE3411A.ADA	CE3705C.ADA	CE3902B.ADA
CE3402A.ADA	CE3411C.ADA	CE3705D.ADA	CE3904B.ADA
CE3403B.ADA	CE3412A.ADA	CE3705E.ADA	CE3905A.ADA
CE3403C.ADA	CE3413A.ADA	CE3706D.ADA	CE3905C.ADA
CE3403E.ADA	CE3413B.ADA	CE3706G.ADA	CE3905L.ADA
CE3403F.ADA	CE3413C.ADA	CE3804A.ADA	CE3906B.ADA
CE3404B.ADA	CE3602A.ADA	CE3804B.ADA	CE3906C.ADA
CE3404C.ADA	CE3602B.ADA	CE3804D.ADA	EE3203A.ADA
CE3404D.ADA	CE3602D.ADA	CE3804E.ADA	EE3204A.ADA
CE3406A.ADA	CE3603A.ADA	CE3804F.ADA	EE3412C.ADA
CE3406C.ADA	CE3604A.ADA	CE3804G.ADA	
CE3406D.ADA	CE3604B.ADA	CE3804H.ADA	
CE3407A.ADA	CE3605C.ADA	CE3804I.ADA	
CE3407C.ADA	CE3704A.ADA	CE3804J.ADA	

If USE_ERROR or NAME_ERROR is raised by every attempt to create or open a text file with mode OUT_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	CE3102H.ADA	CE3104A.ADA	CE3106B.ADA
CE3102A.ADA	CE3102J.ADA	CE3104B.ADA	CE3107B.ADA
CE3102B.TST	CE3102K.ADA	CE3104C.ADA	CE3108A.ADA
CE3102F.ADA	CE3103A.ADA	CE3106A.ADA	CE3108B.ADA
CE3102G.ADA			CE3110A.ADA

CE3112C.ADA	CE3408B.ADA	CE3704B.ADA	CE3806D.ADA
CE3112D.ADA	CE3408C.ADA	CE3704C.ADA	CE3806E.ADA
CE3114A.ADA	CE3409A.ADA	CE3704D.ADA	CE3806G.ADA
CE3115A.ADA	CE3409C.ADA	CE3704E.ADA	CE3806H.ADA
CE3119A.TST	CE3409D.ADA	CE3704F.ADA	CE3902B.ADA
CE3207A.ADA	CE3409E.ADA	CE3704M.ADA	CE3904A.ADA
CE3301A.ADA	CE3410A.ADA	CE3704N.ADA	CE3904B.ADA
CE3302A.ADA	CE3410C.ADA	CE3704O.ADA	CE3905A.ADA
CE3304A.TST	CE3410D.ADA	CE3705A.ADA	CE3905B.ADA
CE3305A.ADA	CE3410E.ADA	CE3705B.ADA	CE3905C.ADA
CE3401A.ADA	CE3411A.ADA	CE3705C.ADA	CE3905L.ADA
CE3402A.ADA	CE3411C.ADA	CE3705D.ADA	CE3906A.ADA
CE3402C.ADA	CE3412A.ADA	CE3705E.ADA	CE3906B.ADA
CE3402D.ADA	CE3413A.ADA	CE3706D.ADA	CE3906C.ADA
CE3403A.ADA	CE3413B.ADA	CE3706F.ADA	CE3906E.ADA
CE3403B.ADA	CE3413C.ADA	CE3706G.ADA	CE3906F.ADA
CE3403C.ADA	CE3414A.ADA	CE3804A.ADA	CXAA003.A
CE3403E.ADA	CE3602A.ADA	CE3804B.ADA	CXAA004.A
CE3403F.ADA	CE3602B.ADA	CE3804C.ADA	CXAA005.A
CE3404B.ADA	CE3602C.ADA	CE3804D.ADA	CXAA009.A
CE3404C.ADA	CE3602D.ADA	CE3804E.ADA	CXAA010.A
CE3404D.ADA	CE3603A.ADA	CE3804F.ADA	CXAA011.A
CE3405A.ADA	CE3604A.ADA	CE3804G.ADA	CXAA012.A
CE3405C.ADA	CE3604B.ADA	CE3804H.ADA	CXAA014.A
CE3405D.ADA	CE3605A.ADA	CE3804I.ADA	CXG1003.A
CE3406A.ADA	CE3605B.ADA	CE3804J.ADA	EE3203A.ADA
CE3406B.ADA	CE3605C.ADA	CE3804M.ADA	EE3204A.ADA
CE3406C.ADA	CE3605D.ADA	CE3804O.ADA	EE3402B.ADA
CE3406D.ADA	CE3605E.ADA	CE3804P.ADA	EE3409F.ADA
CE3407A.ADA	CE3606A.ADA	CE3805A.ADA	EE3412C.ADA
CE3407B.ADA	CE3606B.ADA	CE3805B.ADA	
CE3407C.ADA	CE3704A.ADA	CE3806A.ADA	
CE3408A.ADA		CE3806B.ADA	

If USE_ERROR or NAME_ERROR is raised by every attempt to create or open a text file with mode APPEND_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:

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

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

CE3104C.ADA   CE3115A.ADA

If DELETE for text files is not supported, 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 150000 when checking that PAGE raises LAYOUT_ERROR, then the following test should report NOT_APPLICABLE:

CE3413B.ADA

F.2.7 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 open 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

F.2.8 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 the 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	CE2102O.ADA	CE2103C.ADA	CE2108E.ADA
CE2102C.TST	CE2102P.ADA	CE2104A.ADA	CE2108F.ADA
CE2102G.ADA	CE2102Q.ADA	CE2104B.ADA	CE2109A.ADA
CE2102N.ADA	CE2102X.ADA	CE2106A.ADA	CE2110A.ADA

CE2111A.ADA	CE2201F.ADA	CE2201N.ADA	CE2206A.ADA
CE2111C.ADA	CE2201G.ADA	CE2203A.TST	CE2208B.ADA
CE2111F.ADA	CE2201H.ADA	CE2204A.ADA	CXA8001.A
CE2111I.ADA	CE2201I.ADA	CE2204B.ADA	CXA8002.A
CE2120B.TST	CE2201J.ADA	CE2204C.ADA	
CE2201A.ADA	CE2201K.ADA	CE2204D.ADA	
CE2201B.ADA	CE2201L.ADA	CE2205A.ADA	
CE2201C.ADA	CE2201M.ADA		

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

CE2102G.ADA	CE2111C.ADA	CE2201G.ADA	CE2201M.ADA
CE2102O.ADA	CE2111F.ADA	CE2201H.ADA	CE2201N.ADA
CE2104A.ADA	CE2201A.ADA	CE2201I.ADA	CE2204A.ADA
CE2104B.ADA	CE2201B.ADA	CE2201J.ADA	CE2205A.ADA
CE2108F.ADA	CE2201C.ADA	CE2201K.ADA	CE2206A.ADA
CE2111A.ADA	CE2201F.ADA	CE2201L.ADA	CE2208B.ADA

If USE_ERROR or NAME_ERROR is raised by every attempt to create or open a sequential file with mode OUT_FILE (this is the 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	CE2201A.ADA	CE2201N.ADA
CE2102C.TST	CE2108E.ADA	CE2201B.ADA	CE2203A.TST
CE2102G.ADA	CE2108F.ADA	CE2201C.ADA	CE2204A.ADA
CE2102N.ADA	CE2109A.ADA	CE2201F.ADA	CE2204B.ADA
CE2102O.ADA	CE2110A.ADA	CE2201G.ADA	CE2204C.ADA
CE2102P.ADA	CE2111A.ADA	CE2201H.ADA	CE2204D.ADA
CE2102Q.ADA	CE2111C.ADA	CE2201I.ADA	CE2205A.ADA
CE2102X.ADA	CE2111F.ADA	CE2201J.ADA	CE2206A.ADA
CE2103C.ADA	CE2111I.ADA	CE2201K.ADA	CE2208B.ADA
CE2104A.ADA	CE2120B.TST	CE2201L.ADA	CXA8001.A
CE2104B.ADA		CE2201M.ADA	

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

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:

CE2111C.ADA	CE2111F.ADA	CE2111I.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:

```
CE2111F.ADA        CE2111I.ADA        CE2204C.ADA
```

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

```
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:

```
CE2203A.TST
```

F.2.9 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:

```
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 following test should report NOT_APPLICABLE:

```
CE2102E.ADA
```

If open with mode OUT_FILE is supported, then the following test 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
```

F.2.10 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 the appropriate behavior for an implementation which does not support direct access files), then the tests contained in the following files should report NOT_APPLICABLE:

```
CE2102B.ADA        CE2102S.ADA        CE2102W.ADA        CE2104D.ADA
CE2102H.TST        CE2102T.ADA        CE2102Y.ADA        CE2106B.ADA
CE2102K.ADA        CE2102U.ADA        CE2103D.ADA        CE2108G.ADA
CE2102R.ADA        CE2102V.ADA        CE2104C.ADA
```

CE2108H.ADA	CE2401C.ADA	CE2404A.ADA	CE2409A.ADA
CE2109B.ADA	CE2401E.ADA	CE2404B.ADA	CE2409B.ADA
CE2110C.ADA	CE2401F.ADA	CE2405B.ADA	CE2410A.ADA
CE2111B.ADA	CE2401H.ADA	CE2406A.ADA	CE2410B.ADA
CE2111E.ADA	CE2401I.ADA	CE2407A.ADA	CE2411A.ADA
CE2111G.ADA	CE2401J.ADA	CE2407B.ADA	CXA8003.A
CE2120A.TST	CE2401K.ADA	CE2408A.ADA	CXA9001.A
CE2401A.ADA	CE2401L.ADA	CE2408B.ADA	CXA9002.A
CE2401B.ADA	CE2403A.TST		

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

CE2102K.ADA	CE2111B.ADA	CE2401E.ADA	CE2406A.ADA
CE2102U.ADA	CE2111E.ADA	CE2401F.ADA	CE2407A.ADA
CE2104C.ADA	CE2401A.ADA	CE2401H.ADA	CE2411A.ADA
CE2104D.ADA	CE2401B.ADA	CE2401I.ADA	
CE2108H.ADA	CE2401C.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 (this is the appropriate behavior for an implementation which does not support direct access files), then the tests contained in the following files should report NOT_APPLICABLE:

CE2102B.ADA	CE2106B.ADA	CE2404A.ADA	CE2410B.ADA
CE2102K.ADA	CE2108G.ADA	CE2404B.ADA	CE2411A.ADA
CE2102R.ADA	CE2108H.ADA	CE2405B.ADA	CXA8003.A
CE2102W.ADA	CE2110C.ADA	CE2407A.ADA	CXA9001.A
CE2102Y.ADA	CE2111B.ADA	CE2407B.ADA	CXA9002.A
CE2103D.ADA	CE2111E.ADA	CE2408A.ADA	
CE2104C.ADA	CE2120A.TST	CE2409B.ADA	
CE2104D.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 (this is the 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	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

F.2.11 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 to 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

F.2.12 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.

CXAC001.A	CXAC003.A	CXACA02.A	CXACB02.A
CXAC002.A	CXACA01.A	CXACB01.A	CXACC01.A

F.2.13 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 test contained in the following file should report NOT_APPLICABLE.

CXAB001.A

F.2.14 File I/O Tests

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

CE2102A.ADA	CE2109C.ADA	CE2401B.ADA	CE3106A.ADA
CE2102B.ADA	CE2110A.ADA	CE2401C.ADA	CE3106B.ADA
CE2102C.TST	CE2110C.ADA	CE2401E.ADA	CE3107B.ADA
CE2102G.ADA	CE2111A.ADA	CE2401F.ADA	CE3108A.ADA
CE2102H.TST	CE2111B.ADA	CE2401H.ADA	CE3108B.ADA
CE2102K.ADA	CE2111C.ADA	CE2401I.ADA	CE3110A.ADA
CE2102N.ADA	CE2111E.ADA	CE2401J.ADA	CE3112C.ADA
CE2102O.ADA	CE2111F.ADA	CE2401K.ADA	CE3112D.ADA
CE2102P.ADA	CE2111G.ADA	CE2401L.ADA	CE3114A.ADA
CE2102Q.ADA	CE2111I.ADA	CE2403A.TST	CE3115A.ADA
CE2102R.ADA	CE2120A.TST	CE2404A.ADA	CE3119A.TST
CE2102S.ADA	CE2120B.TST	CE2404B.ADA	CE3207A.ADA
CE2102T.ADA	CE2201A.ADA	CE2405B.ADA	CE3301A.ADA
CE2102U.ADA	CE2201B.ADA	CE2406A.ADA	CE3302A.ADA
CE2102V.ADA	CE2201C.ADA	CE2407A.ADA	CE3304A.TST
CE2102W.ADA	CE2201F.ADA	CE2407B.ADA	CE3305A.ADA
CE2102X.ADA	CE2201G.ADA	CE2408A.ADA	CE3401A.ADA
CE2102Y.ADA	CE2201H.ADA	CE2408B.ADA	CE3402A.ADA
CE2103C.ADA	CE2201I.ADA	CE2409A.ADA	CE3402C.ADA
CE2103D.ADA	CE2201J.ADA	CE2409B.ADA	CE3402D.ADA
CE2104A.ADA	CE2201K.ADA	CE2410A.ADA	CE3403A.ADA
CE2104B.ADA	CE2201L.ADA	CE2410B.ADA	CE3403B.ADA
CE2104C.ADA	CE2201M.ADA	CE2411A.ADA	CE3403C.ADA
CE2104D.ADA	CE2201N.ADA	CE3102A.ADA	CE3403E.ADA
CE2106A.ADA	CE2203A.TST	CE3102B.TST	CE3403F.ADA
CE2106B.ADA	CE2204A.ADA	CE3102F.ADA	CE3404B.ADA
CE2108E.ADA	CE2204B.ADA	CE3102G.ADA	CE3404C.ADA
CE2108F.ADA	CE2204C.ADA	CE3102H.ADA	CE3404D.ADA
CE2108G.ADA	CE2204D.ADA	CE3102J.ADA	CE3405A.ADA
CE2108H.ADA	CE2205A.ADA	CE3102K.ADA	CE3405C.ADA
CE2109A.ADA	CE2206A.ADA	CE3103A.ADA	CE3405D.ADA
CE2109B.ADA	CE2208B.ADA	CE3104A.ADA	CE3406A.ADA
	CE2401A.ADA	CE3104B.ADA	CE3406B.ADA
		CE3104C.ADA	CE3406C.ADA

CE3406D.ADA
CE3407A.ADA
CE3407B.ADA
CE3407C.ADA
CE3408A.ADA
CE3408B.ADA
CE3408C.ADA
CE3409A.ADA
CE3409C.ADA
CE3409D.ADA
CE3409E.ADA
CE3410A.ADA
CE3410C.ADA
CE3410D.ADA
CE3410E.ADA
CE3411A.ADA
CE3411C.ADA
CE3412A.ADA
CE3413A.ADA
CE3413B.ADA
CE3413C.ADA
CE3414A.ADA
CE3602A.ADA
CE3602B.ADA
CE3602C.ADA
CE3602D.ADA
CE3603A.ADA
CE3604A.ADA
CE3604B.ADA
CE3605A.ADA
CE3605B.ADA
CE3605C.ADA
CE3605D.ADA
CE3605E.ADA
CE3606A.ADA
CE3606B.ADA
CE3704A.ADA
CE3704B.ADA
CE3704C.ADA
CE3704D.ADA
CE3704E.ADA
CE3704F.ADA
CE3704M.ADA
CE3704N.ADA
CE3704O.ADA
CE3705A.ADA
CE3705B.ADA
CE3705C.ADA
CE3705D.ADA
CE3705E.ADA
CE3706D.ADA
CE3706F.ADA
CE3706G.ADA
CE3804A.ADA
CE3804B.ADA
CE3804C.ADA
CE3804D.ADA
CE3804E.ADA
CE3804F.ADA
CE3804G.ADA
CE3804H.ADA
CE3804I.ADA
CE3804J.ADA
CE3804M.ADA
CE3804O.ADA
CE3804P.ADA
CE3805A.ADA
CE3805B.ADA
CE3806A.ADA
CE3806B.ADA
CE3806D.ADA
CE3806E.ADA
CE3806G.ADA
CE3806H.ADA
CE3902B.ADA
CE3904A.ADA
CE3904B.ADA
CE3905A.ADA
CE3905B.ADA
CE3905C.ADA
CE3905L.ADA
CE3906A.ADA
CE3906B.ADA
CE3906C.ADA
CE3906E.ADA
CE3906F.ADA
CXA8001.A
CXA8002.A
CXA8003.A
CXA9001.A
CXA9002.A
CXAA001.A
CXAA002.A
CXAA003.A
CXAA004.A
CXAA005.A
CXAA006.A
CXAA007.A
CXAA008.A
CXAA009.A
CXAA010.A
CXAA011.A
CXAA012.A
CXAA013.A
CXAA014.A
CXAA015.A
CXAB001.A
CXAC001.A
CXAC002.A
CXAC003.A
CXACA01.A
CXACA02.A
CXACB01.A
CXACB02.A
CXACC01.A
CXG1003.A
EE3203A.ADA
EE3204A.ADA
EE3402B.ADA
EE3409F.ADA
EE3412C.ADA

F.3 Special Case of Unhandled Exception

If files are not supported and USE_ERROR is raised by an attempt to create a file, then the tests contained in the following files will apparently fail. In this specific case, the tests are not applicable:

CE2103A.TST
CE2103B.TST
CE3107A.TST