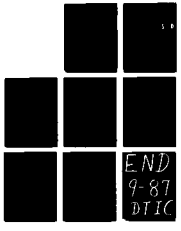
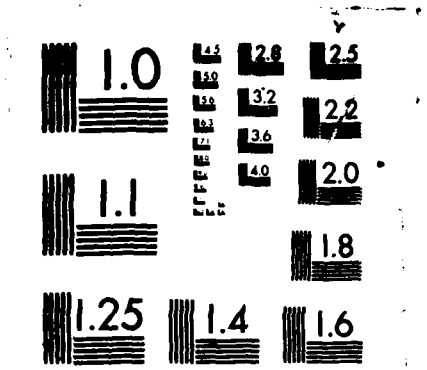
AD-A183 666 ADA (TRADEMARK) COMPILER VALIDATION SUMMARY REPORT 1/1
SOFTECH INC ADAVAX COM. (U) FEDERAL SOFTWARE MANAGEMENT
SUPPORT CENTER FALLS CHURCH VA 30 AUU 85

UNCLASSIFIED F/G 12/5 NL





(2)

DTIC FILE COPY

Ad-* COMPILER VALIDATION SUMMARY REPORT:

SofTech, Inc.
AdaVAX Compiler
Version 2.47
VAX 8600, VAX-11/780, VAX-11/785; and MicroVAX II
using VAX/VMS Version 4.1 and MicroVMS 4.1M



August 30, 1985



Prepared for:

Department of the Army
Communications Electronic Command
Fort Monmouth, New Jersey 07703-5204

Ada Joint Program Office
1211 Fern St.
Arlington, Va 22202

Prepared by:

Federal Software Management
Support Center
Office of Software Development
and Information Technology
Two Skyline Place, Suite 1100
5203 Leesburg Pike
Falls Church, VA 22041-3467



* Ada is a registered trademark of the U.S. Government, (Ada Joint Program Office).

## REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETEING FORM

| | |
|---|---|
| 1. REPORT NUMBER | 2. GOVT ACCESSION NO. AD-A183 666 / 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle) Ada Compiler Validation Summary Report: SofTech, Inc. AdaVAX Compiler Version 2.47 VAX 8600, VAX-11/780, VAX-11/785; and MicroVAX II using VAX/VMS Version 4.1 and microVMS 4.1M | 5. TYPE OF REPORT & PERIOD COVERED 30 AUG 1985 to 30 AUG 1986<br>6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s) Federal Software Management Support Center | 8. CONTRACT OR GRANT NUMBER(s) |
| 9. PERFORMING ORGANIZATION AND ADDRESS Federal Software Management Support Center, Office of Software Development and Information Technology, Two Skyline Place, Suite 1100, 5203 Leesburg Pike , Falls Church, VA 22041-3467 | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS Ada Joint Program Office United States Department of Defense Washington, DC 20301-3081ASD/SIOL | 12. REPORT DATE 30 AUG 1985<br>13. NUMBER OF PAGES 22 |
| 14. MONITORING AGENCY NAME & ADDRESS(If different from Controlling Office) Federal Software Management Support Center | 15. SECURITY CLASS (of this report) UNCLASSIFIED<br>15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20. If different from Report)

UNCLASSIFIED

18. SUPPLEMENTARY NOTES

19. KEYWORDS (Continue on reverse side if necessary and identify by block number)

Ada Programming language, Ada Compiler Validation Summary Report, Ada Compiler Validation Capability, ACVC, Validation Testing, Ada Validation Office, AVO, Ada Validation Facility, AVF, ANSI/MIL-STD-1815A, Ada Joint Program Office, AJPO

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

See Attached.

## ABSTRACT

The purpose of this Validation Summary Report is to present the results and conclusions of performing standardized tests on the SofTech AdaVAX compiler, Version 2.47. On-site testing was performed by the Federal Software Management Support Center (FSMSC)—an Ada Validation Facility—in accordance with current Ada Validation Office policies and procedures. The following are the dates and locations of on-site testing:

23-30 August 1985, SofTech, Waltham, MA (VAX-11/780*) and
Naval Underwater Systems Center (NUSC), Newport, RI (VAX-11/785);
9-10 September 1985, Picatinny Arsenal, Dover, NJ, (VAX 8600); and
15 October 1985, SofTech, Inc., Waltham, MA, (MicroVAX II).

The suite of test known as the Ada Compiler Validation Capability (ACVC), Version 1.6, was used. The ACVC suite of tests is used to validate conformance of the compiler to ANSI/MIL-STD-1815A (Ada). This standard is described in the ANSI Ada Reference Manual, January 1983. Not all tests in the ACVC test suite are applicable to this specific implementation. Also, known test errors in Version 1.6 are present in some tests; these tests were withdrawn. The purpose of the testing is to ensure that the compiler properly implements legal language constructs and that it identifies, rejects from processing, and labels illegal constructs.

The SofTech Compiler AdaVAX, Version 2.47, using VAX/VMS 4.1 and MicroVMS 4.1M, were tested with version 1.6 of the ACVC validation tests. Version 1.6 of the test suite contains 2162 tests of which 56 were withdrawn and 258 were inapplicable to this implementation. All of the 1848 remaining tests were passed.

The SofTech AdaVAX Ada Compiler was tested on those systems listed in the table below. Each system was able to host the compiler and target itself. In addition, testing demonstrated that all systems could target each other. The following table represents the tested target/host relationships:

| Host | Target | Series of Testing | Site of Execution | Compiler Option |
|---|---|---|---|---|
| VAX-11/780* | VAX-11/780* | Full ACVC | Waltham, MA | Optimize |
| VAX-11/780* | VAX-11/780* | Full ACVC | Waltham, MA | No_Optimize |
| VAX-11/780* | VAX-11/785 | Subset | Waltham, MA | Optimize |
| VAX-11/780* | MicroVAX II | Subset | Waltham, MA | Optimize |
| VAX-11/785 | VAX-11/785 | Full ACVC | Newport, RI | Optimize |
| VAX-11/785 | VAX-11/785 | Subset | Waltham, MA | No_Optimize |
| VAX-11/785 | VAX-11/780* | Subset | Waltham, MA | No_Optimize |
| VAX-11/785 | VAX 8600 | Subset | Dover, NJ | No_Optimize |
| VAX-11/780* | VAX 8600 | Subset | Dover, NJ | No_Optimize |
| VAX 8600 | VAX 8600 | Full ACVC | Dover, NJ | Optimize |
| VAX 8600 | VAX 8600 | Subset | Dover, NJ | No_Optimize |
| MicroVAX II | MicroVAX II | Subset | Waltham, MA | Optimize |

(*) The configuration for testing under the VAX-11/780 was five Digital VAX-11/780 systems configured in a VAX Cluster and DECNET architecture. Throughout this document, it will not be necessary to distinguish between the VAX-11/780 Cluster and the VAX-11/780. The term VAX-11/780 will be used to reference the testing performed on the five systems configured under VAX Cluster and DECNET.

for L. Arnold Johnson
Richard Harrison
Director
Federal Software Management Support Center

for John F. Kramer Jr.
Thomas H. Probert, Ph. D.
Institute for Defense Analyses

Virginia Castor
Virginia Castor
Acting Director
Ada Joint Program Office

## ABSTRACT

The purpose of this Validation Summary Report is to present the results and conclusions of performing standardized tests on the SofTech AdaVAX compiler, Version 2.47. On-site testing was performed by the Federal Software Management Support Center (FSMSC)--an Ada Validation Facility--in accordance with current Ada Validation Office policies and procedures. The following are the dates and locations of on-site testing:

23-30 August 1985, SofTech, Waltham, MA (VAX-11/780*) and
Naval Underwater Systems Center (NUSC), Newport, RI (VAX-11/785);
9-10 September 1985, Picatinny Arsenal, Dover, NJ, (VAX 8600); and
15 October 1985, SofTech, Inc., Waltham, MA, (MicroVAX II).

The suite of test known as the Ada Compiler Validation Capability (ACVC), Version 1.6, was used. The ACVC suite of tests is used to validate conformance of the compiler to ANSI/MIL-STD-1815A (Ada). This standard is described in the ANSI Ada Reference Manual, January 1983. Not all tests in the ACVC test suite are applicable to this specific implementation. Also, known test errors in Version 1.6 are present in some tests; these tests were withdrawn. The purpose of the testing is to ensure that the compiler properly implements legal language constructs and that it identifies, rejects from processing, and labels illegal constructs.

The SofTech Compiler AdaVAX, Version 2.47, using VAX/VMS 4.1 and MicroVMS 4.1M, were tested with version 1.6 of the ACVC validation tests. Version 1.6 of the test suite contains 2162 tests of which 56 were withdrawn and 258 were inapplicable to this implementation. All of the 1848 remaining tests were passed.

The SofTech AdaVAX Ada Compiler was tested on those systems listed in the table below. Each system was able to host the compiler and target itself. In addition, testing demonstrated that all systems could target each other. The following table represents the tested target/host relationships:

| Host | Target | Series of Testing | Site of Execution | Compiler Option |
|---|---|---|---|---|
| VAX-11/780* | VAX-11/780* | Full ACVC | Waltham, MA | Optimize |
| VAX-11/780* | VAX-11/780* | Full ACVC | Waltham, MA | No_Optimize |
| VAX-11/780* | VAX-11/785 | Subset | Waltham, MA | Optimize |
| VAX-11/780* | MicroVAX II | Subset | Waltham, MA | Optimize |
| VAX-11/785 | VAX-11/785 | Full ACVC | Newport, RI | Optimize |
| VAX-11/785 | VAX-11/785 | Subset | Waltham, MA | No_Optimize |
| VAX-11/785 | VAX-11/780* | Subset | Waltham, MA | No_Optimize |
| VAX-11/785 | VAX 8600 | Subset | Dover, NJ | No_Optimize |
| VAX-11/780* | VAX 8600 | Subset | Dover, NJ | No_Optimize |
| VAX 8600 | VAX 8600 | Full ACVC | Dover, NJ | Optimize |
| VAX 8600 | VAX 8600 | Subset | Dover, NJ | No_Optimize |
| MicroVAX II | MicroVAX II | Subset | Waltham, MA | Optimize |

(*) The configuration for testing under the VAX-11/780 was five Digital VAX-11/780 systems configured in a VAX Cluster and DECNET architecture. Throughout this document, it will not be necessary to distinguish between the VAX-11/780 Cluster and the VAX-11/780. The term VAX-11/780 will be used to reference the testing performed on the five systems configured under VAX Cluster and DECNET.

TABLE OF CONTENTS

# 1. Introduction

## 1.1 Purpose of the Validation Summary Report

This report describes the results of the validation testing for the compiler designated as AdaVAX, Version 2.47 the following configurations:

| | |
|---|---|
| Host Machines; | VAX 8600, VAX-11/780 and VAX-11/785, and MicroVAX II. |
| Operating System; | VAX/VMS 4.1, MicroVMS 4.1M |
| Host Disk Systems; | RA81, RM05, RP06, RP07, RD53 |
| Target Machines; | VAX 8600, VAX-11/780 and VAX-11/785, and MicroVAX II |
| Operating System; | VAX/VMS 4.1 and MicroVMS 4.1M |
| Language Version; | ANSI/MIL-STD-1815A Ada |
| Translator Name; | AdaVAX 2.47 |
| Validation Test Version: | 1.6 |

Testing of this compiler was conducted by the Federal Software Management Support Center under the supervision of the Ada Validation Office (AVO), at the direction of the Ada Joint Program Office. Testing was conducted from 23-30 August, 1985 at SOFTECH, Waltham, Mass and the Naval Underwater Systems Center (NUSC), R.I. At Picatinny Arsenal, Dover, N.J., testing was conducted 9-10 September, 1985. The MicroVAX II testing was conducted 15 October 1985 at Waltham, Mass. All testing was performed in accordance with AVO policies and procedures.

The purpose of this report is to document the results of the testing performed on the compiler, and in particular, to:

- identify any language constructs supported by the compilers that do not conform to the Ada standard;
- identify any unsupported language constructs required by the Ada standard; and
- describe implementation dependent behavior allowed by the standard.

## 1.2 Host to Target Relationship Table

The SofTech AdaVAX Ada Compiler was tested on those systems listed in the table below. Each system was able to host the compiler and target itself. In addition, testing demonstrated that all systems could target each other. The following table represents the tested target/host relationships:

| Host | Target | Series of Testing | Site of Execution | Compiler Option |
|---|---|---|---|---|
| VAX-11/780* | VAX-11/780* | Full ACVC | Waltham, MA | Optimize |
| VAX-11/780* | VAX-11/780* | Full ACVC | Waltham, MA | No_Optimize |
| VAX-11/780* | VAX-11/785 | Subset | Waltham, MA | Optimize |
| VAX-11/780* | MicroVAX II | Subset | Waltham, MA | Optimize |
| VAX-11/785 | VAX-11/785 | Full ACVC | Newport, RI | Optimize |
| VAX-11/785 | VAX-11/785 | Subset | Waltham, MA | No_Optimize |
| VAX-11/785 | VAX-11/780* | Subset | Waltham, MA | No_Optimize |
| VAX-11/785 | VAX 8600 | Subset | Dover, NJ | No_Optimize |
| VAX-11/780* | VAX 8600 | Subset | Dover, NJ | No_Optimize |
| VAX 8600 | VAX 8600 | Full ACVC | Dover, NJ | Optimize |
| VAX 8600 | VAX 8600 | Subset | Dover, NJ | No_Optimize |
| MicroVAX II | MicroVAX II | Subset | Waltham, MA | Optimize |

(*) The configuration for testing under the VAX-11/780 was five Digital VAX-11/780 systems configured in a VAX Cluster and DECNET architecture. Throughout this document, it will not be necessary to distinguish between the VAX-11/780 Cluster and the VAX-11/780. The term VAX-11/780 will be used to reference the testing performed on the five systems configured under VAX Cluster and DECNET.

## 1.3 Use of the Validation Summary Report

The Ada Validation Office may make full and free public disclosure of this report in accordance with the "Freedom of Information Act" (5 U.S.C. #552). The results of the validation apply only to the computers, operating systems, and compiler version identified in this report.

The Ada Compiler Validation Capability is used to determine insofar as is practical, the degree to which the subject compiler conforms to the Ada standard. Thus, this report is necessarily discretionary and judgemental. The United States Government does not represent or warrant that the statements, or any one of them, set forth in this report are accurate or complete, nor that the subject compiler has no other nonconformances to the Ada standard. This report is not meant to be used for the purpose of publicizing the findings summarized herein.

Any questions regarding this report or the validation tests should be sent to the Ada Validation Office at:

Ada Joint Program Office
1211 Fern Street
Arlington, VA 22202

## 1.4 References

Reference Manual for the Ada Programming Language, ANSI/MIL-STD-1815A, February 1983.

Ada Validation Organization: Policies and Procedures, Mitre Corporation, June 1982, PB 83-110601.

Ada Compiler Validation Implementers' Guide, SOFTECH, Inc., October 1980.

The Ada Compiler Validation Capability, Computer, Vol. 14, No. 6, June 1981.

Using the ACVC Tests, SofTech, Inc. November 1981.

Ada Compiler Validation Plans and Procedures, SofTech, Inc. November 1981.

## 1.5 Definitions of Terms

Class A tests are passed if no errors are detected at compile time. Although these tests are constructed to be executable, no checks can be performed at run-time to see if the test objective has been met; this distinguished Class A from Class C tests. For example, a Class A test might check that keywords of other languages (other than those already reserved in Ada) are not treated as reserved words by an Ada implementation.

Class B tests are illegal programs. They are passed if all the errors they contain are detected at compile-time (or link-time) and no legal statements are considered illegal by the compiler.

Class L tests consist of illegal programs whose errors cannot be detected until link time. They are passed if errors are detected prior to beginning execution of the main program.

Class C tests consist of executable self-checking programs. They are passed if they complete execution and do not report failure.

Class D tests are capacity tests. Since there are no firm criteria for the number of identifiers permitted in a compilation, number of units in a library, etc., a compiler may refuse to compile a class D test. However, if such a test is successfully compiled, it should execute without reporting a failure.

Class E tests provide information about an implementation's interpretation of the Standard. Each test has its own pass/fail criterion.

ACVC: Acronym for the Ada Compiler Validation Capability.

AVO: The Ada Validation Office. In the context of this report the AVO is responsible for directing compiler validation.

CHECK or CHECKTEST: An automated tool defined by the Federal Software Management Support Center (FSMSC) and developed by the AVF that produces summary test results by reading compiler output in a spool file. This tool is available on the ACVC distribution tapes from the AVF.

CUSTOMER: The agency requesting the validation Department of the Army, Communications Electronics Command, Fort Monmouth, N.J. 07703-5204.

| | |
|---|---|
| FSMSC: | Federal Software Management Support Center. In the context of this report the FSMSC conducts Ada validations under contract to the AVO as a satellite facility. |
| HOST: | The computer on which the compiler executes. |
| IG: | ACVC Implementors' Guide. |
| RM: | The Ada Language Reference Manual. |
| STANDARD: | The standard for the Ada language, ANSI/MIL-STD-1815A (1983). |
| SUBSET TESTS: | A grouping of ACVC tests selected by the FSMSC. Each chapter in the ACVC is represented in the subset by between 4 to 7 tests. The subset is used for statistical sampling of the various host and target hardware configurations. |
| TARGET: | The computers for which the compiler generates object code. |
| VALIDATION: | The process of testing a compilation system to certify that it conforms to the standard. |
| VALIDATION TESTS: | The set of test programs used to detect non-conformances in compilation systems. In this report, the term will be used (unqualified) to mean the ACVC tests. |

## 2. TEST ANALYSIS

The following table shows that the SofTech, Inc. AdaVAX compiler passed all applicable tests.

| | A | B | C | D | E | L | Total |
|---|---|---|---|---|---|---|---|
| In suite | 61 | 800 | 1273 | 17 | 8 | 3 | 2162 |
| Inapplicable | 1 | 9 | 248 | 0 | 0 | 0 | 258 |
| Withdrawn | 0 | 13 | 43 | 0 | 0 | 0 | 56 |
| Passed | 60 | 778 | 982 | 17 | 8 | 3 | 1848 |
| Failed | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

258 tests in the suite were found to be inapplicable to the SofTech implementation.

In addition, 56 tests were withdrawn from the test suite because they were incorrect.

### 2.1 Class A Testing

Class A tests check that legal Ada programs can be successfully compiled. These tests are executed but contain no executable self-checking capabilities.

#### 2.1.1 Class A Test Procedures

Each class A test was separately compiled and executed. However, the only purpose of execution is to produce a message indicating that the test passed.

#### 2.1.2 Class A Test Results

Successful compilation and execution without any error messages indicates the tests passed. All 60 applicable tests passed.

### 2.2 Class B Testing

Class B tests check the ability to recognize illegal language usage. There were 778 applicable class B tests processed.

#### 2.2.1 Class B Test Procedures

Each Class B test was separately compiled. The resulting test compilation listings are manually examined to see whether every illegal construct in the test is detected. If some errors are not detected, a version of the program test is created that contains only undetected illegal constructs.

This revised version is recompiled and the results analyzed. If some errors are still not detected, the revision process is repeated until a revised test contains only a single previously undetected illegal construct.

A B test is considered to fail only if a version of the test containing a single illegal construct is accepted by the compiler (i.e., an illegal construct is not detected) or a version containing no errors is rejected (i.e., a legal construct is rejected).

### 2.2.2 Class B Test Results

There were 800 class B tests presented to the compiler. Of these tests 9 were found to be inapplicable to this implementation (see Section 4.2.7); 13 tests were found to be incorrect (i.e., a conforming compiler would have failed each of these tests). All 778 remaining class B tests passed.

Because all errors were not detected when compiling the original tests, the following 12 tests were modified by removing the detected errors; the modified tests were then submitted again to see if the remaining errors would be detected.

| | | | |
|---|---|---|---|
| B33004A.ADA | B37301B.ADA | B44001A.ADA | B45205A.ADA |
| B55A01A.ADA | B67001A.ADA | B67001B.ADA | B67001C.ADA |
| B67001D.ADA | BB10AEB.ADA | BC1202B.ADA | BC1202D.ADA |

All illegal constructs were detected except in some tests that were withdrawn because of errors in the tests (see Section 4.2.8).

## 2.3 Class C Testing

Class C tests check that legal Ada programs are correctly compiled and executed by an implementation. There were 1273 class C tests processed in this validation attempt.

### 2.3.1 Class C Test Procedures

Each Class C test is separately compiled and executed. The tests are self-checking and produce PASS/FAIL messages. Any 'failed' tests are individually checked to see if they are correct and if they are applicable to the implementation. Any tests that are inapplicable or that do not conform to the Ada Standard are withdrawn.

### 2.3.2 Class C Test Results

All class C tests were processed except those tests requiring a floating point precision exceeding SYSTEM.MAX_DIGITS. All applicable 982 tests passes.

## 2.4 Class D Testing

Class D tests are executable tests used to check an implementation's compilation and execution capacities. There were 17 class D tests used in this validation.

### 2.4.1 Class D Test Procedures

Each class D test is separately compiled and executed. The tests are self-checking and produce PASS/FAIL messages.

### 2.4.2 Class D Test Results

Of the 17 applicable class D tests, all tests passed.

## 2.5 Class E Testing

Class E tests provide information about an implementation's interpretation of the Standard where the Standard permits implementations to differ. Each test has its own pass/fail criterion. There were 8 class E tests used in this validation.

### 2.5.1 Class E Testing Procedures

Each class E test is separately compiled and executed. The tests are self-checking and produce pass/fail messages.

### 2.5.2 Class E Test Results

All class E tests passed.

## 2.6 Class L Testing

Three Class L tests check that incomplete or illegal Ada programs involving multiple separately compiled source files are detected at link time and are not allowed to execute. There were 3 Class L tests processed in this validation.

### 2.6.1 Class L Test Procedures

Each Class L test is separately compiled and execution is attempted. The tests produce FAIL messages if executed. Any "failed" tests are checked to see if they are correct and applicable to the implementation. Tests that are inapplicable or that do not conform to the Ada standard are withdrawn.

### 2.6.2 Class L Test Results

Of the 3 class L tests, none were found to be inapplicable to this implementation, and none were withdrawn due to errors in the tests. All three L tests passed.

## 2.7 Subset Testing

A subset of the executable ACVC tests was used by the FSMSC and used during this validation. This subset of tests was used to test multiple configuration combinations during the validation, specifically the VAX 8600, VAX-11/780, VAX-11/785 and MicroVAX II host/target relationships using both optimized and unoptimized versions of the compiler.

The subset comprised the following tests:

| Chapter 2 | Chapter 3 | Chapter 4 | Chapter 5 |
|---|---|---|---|
| C23001A | C34001A | C41101D | C51002A |
| C24102A | C34001H | C42005A | C52001A |
| C26008A | A32203D | C43214A | C53005A |
| A29002A | C35904A | C45101A | C54A03A |
| | C36204A | C48004A | D55A03A |
| | C34002B | | |

| Chapter 6 | Chapter 7 | Chapter 8 | Chapter 9 |
|---|---|---|---|
| C61003B | A71002A | A83A02A | C92002A |
| A62006D | C72001B | C84002A | C93001A |
| C63004A | C74302B | C85007E | C94006A |
| C65003A | C74209A | C86003A | A97106A |
| C66002A | C74409B | C87B48A | C97202A |

| Chapter 10 | Chapter 11 | Chapter 12 | Chapter 14 |
|---|---|---|---|
| CA1003A | CB1001A | CC1004A | AE2101A |
| CA2004A0M | CB2004A | CC3004A | CE2102A |
| CA2004A1 | CB3003A | CC3408A | CE2201A |
| CA2004A2 | CB4001A | CC3504C | CE2401E |
| CA2004A3 | | | CE3102A |
| CA2004A4 | | | CE3901A |

*** CZ ***

CZ1101A
CZ1102A
CZ1103A
CZ1201A
CZ1201B
CZ1201C
CZ1201D

## 3. COMPILER ANOMALIES AND NONCONFORMANCE

There were no nonconformances to the Ada standard detected in this validation. The compiler passed all applicable correct tests.

# 4. ADDITIONAL INFORMATION

This section describes in more detail how the validation was concluded.

## 4.1 Compiler Parameters

Certain tests do not apply to all Ada compilers, e.g., compilers are not required to support several predefined floating point types, and so tests must be selected based on the predefined types an implementation actually supports. In addition, some tests are parameterized according to the maximum length allowed by an implementation for an identifier (or other lexical element; this is also the maximum line length), the maximum floating point precision supported, etc. The implementation dependent parameters used in performing this validation were:

- maximum lexical element length: 120 characters.
- maximum digits value for floating point types: 9 (nine)
- SYSTEM.MIN_INT: -2147483648
- SYSTEM.MAX_INT: 2147483647
- predefined numeric types: INTEGER, FLOAT, LONG_INTEGER and LONG_FLOAT.

  INTEGER_FIRST: -32_768

  INTEGER_LAST: 32_768
- LONG_INTEGER'FIRST: -2147483648
- LONG_INTEGER'LAST: 2147483647
- Source character set: ASCII
- Extended ASCII chars: abcdefghijklmnopqrstuvwxyz !$%?@[\]^`{}~#
- non-ascii char type: (NON_NULL)
- TEXT_IO.COUNT'LAST: 32_767
- TEXT_IO.FIELD'LAST: 32_767
- illegal external file name1: "badcharacter*O"
- illegal external file name2: "much__too__long__name for_a_file"
- SYSTEM.PRIORITY'FIRST: 1
- SYSTEM.PRIORITY'LAST: 15

## 4.2 Testing Information

Complete ACVC tests runs were compiled/executed at SofTech, Inc., Waltham, MA, at the Naval Underwater Systems Center (NUSC), Newport RI, and at Picatiny Arsenal, New Jersey

### 4.2.1 Pre-Test Procedures

Prior to testing, appropriate values for the compiler-dependent parameters were determined. These values were used to adapt tests that depend on the values. A magnetic tape containing the adapted tests was prepared and brought to the testing site. Spilt B tests were not prepared on this tape and had to be split on-site.

For pre-validation, SofTech processed the entire ACVC on the VAX-11/780, in the VAX Cluster, DECNET configuration. Additionally, the subset (see section 2.7) was compiled on the above configuration and executed on the MicroVAX II.

### 4.2.2 Control Files

SofTech, Inc. provided command procedures that compiled and executed tests automatically at all sites. All control files were duplicated at the NUSC site through transported tapes prior to the on-site validation.

4.2.3 On-site Data Collection

The complete ACVC was run on the VAX 8600 (Picatinny Arsenal, NJ), the VAX-11/785 (NUSC) and the VAX-11/780 (SofTech).

The results from the VAX-11/780 were analyzed; they were determined to be correct. These results were thus used as the "test basis" against which the results from running the ACVC on other configurations would be compared.

The results from the VAX 8600 and VAX-11/785 were compared against the test basis; no significant differences were detected.

The ACVC was again processed by the VAX-11/780, but this time without optimization. The class B tests were excluded, since the do not enter the optimization stage (being illegal programs that must fail in compilation). The results were compared against the test basis, and determined to be correct.

A subset of ACVC executables tests (see 2.7) was compiled on the VAX-11/780 and the VAX-11/785*--the former with optimization, the later without. The VAX-11/780 executed the subset compiled by the VAX-11/785*; the VAX-11/785* executed the subset compiled by the VAX-11/780. The results were examined and determined to be correct.

The VAX 8600 compiled and executed the subset, without optimization. In addition, the VAX 8600 executed the two subsets compiled by the VAX-11/780 and VAX-11/785 (described above). All results were compared with each other and against the test basis; they were found to be correct.

The MicroVAX II compiled and executed the subset, with optimization. The MicroVAX II also executed the optimized subset load module from the VAX-11/780. The results were checked against those of the previous testing with the subset (described above); no significant differences were detected.

* This was a VAX-11/785 resident at SofTech, Waltham, MA.

### 4.2.4 Test Analysis Procedures

On completion of testing the base system, all results were analyzed for failed Class A, C, D, E, or L programs, and all class B compilation results were individually analyzed. Analysis procedures are described for each test class in chapter 2.

### 4.2.5 Timing Information

The real (wall clock) times required for compiling the non-executable tests and compiling, linking, and running the executable tests--with optimization--were:

| | | |
|---|---|---|
| VAX 11/780 | 49:03 | Full run of validation suite under VAX/VMS - single batch streams on 5 separate systems |
| VAX 11/785 | 120:00 | Full run of validation suite under VAX/VMS - two batch streams on a single machine run in parallel |
| VAX 8600 | 72:30 | Full run of validation suite under VAX/VMS - three batch streams on a single machine run in parallel |
| MicroVAX II | 15:44 | Subset run of validation suite under MicroVMS 4.1M - single batch steam on a single MicroVAX II system. |

### 4.2.6 Description of Errors in Withdrawn Tests

The following tests in version 1.6 of the ACVC did not conform to the ANSI Ada standard and were withdrawn for the reasons given below:

- B66001A-B: Test checks (in section G) that a parameterless function that is equivalent to an enumeration literal in the same declarative region is a redeclaration and, as such, is forbidden. According to RM 8.3(17), the explicit declaration of such a function is allowed if an enumeration literal is considered to be an implicitly declared predefined operation. The RM is not clear on this point. This issue has been referred to the Language Maintenance Committee for resolution. Since the issue cannot be resolved at this time, the test is withdrawn from Version 1.6. (Please note that this test may be considered correct and may appear in the future Versions of the ACVC, including Version 1.6.)

- BC1013A-B: The declaration of equality in lines 86-87 is illegal because the parameter type T declared in line 11 is not a limited type (LRM 6.7-4).

- C45521A, C45521B, ... C45521Y (25 tests) : Cases C and I define the model interval for the result too narrowly.

- C48005C-B: Lines 38 and 63 of this test should check that the value of the designated object is null.

- C64103C-B: This test should raise CONSTRAINT_ERROR during the conversion at line 179.

- C64103D-B: This test involves a CONSTRAINT_ERROR vs. NUMERIC__ERROR issue that is to be resolved by the Language Maintenance Committee.

- C64105E-AB: For case E, ensure that non-null dimensions of formal and actual parameters belong to both index subtypes (see AI-00313).

- C64105F-AB: For case E, ensure that non-null dimensions of formal and actual parameters belong to both index subtypes (See AI-00313).

- B67001A-B: Line 414 is missing the "BEGIN NULL; END;" needed to complete the block beginning at line 389 (case H).

- B67004A-B: The default name for a formal generic equality function should not be allowed to be "/=" unless an expanded name is used.

- C93005A, C93005B, and C93005C: These tests contain a declaration of an interger variable whose initialization is solely for the purpose of raising an exception. Some compilers will not raise this exception due to their optimization.

- C93007B-B: This test should check for PROGRAM_ERROR rather than TASKING_ERROR (See AI-000149).

- CA1011A*-B: The test objective should be reversed to be consistent with AI-00199.

- CA1108A-B: A pragma ELABORATE is needed for OTHER_PKG at line 25.

- CA1108B-B: A pragma ELABORATE is needed for FIRST_PKG at line 39 and for LATER_PKG at line 49.

- CA2009B-B, CA2009E-B: The repetition of the main procedure after the subunit body makes the subunit body obsolete; therefore, an attempt to execute the main procedure will fail.

- CA2009F*-B: The file CA2009F2-B is missing from the test suite.

- BC3204A-B, BC3204B-B, BC3204C*-B, BC3204D-B, BC3205A-B, BC3205B-B, BC3205C-B, BC3205D*-B, BC3405B-B: Instantiations with types that have default discriminants are now legal (AI-00037).

- CE3603A-B: The last case is inconsistent with AI-00050. If the string argument is null, no attempt to read is made and END_ERROR is not raised.

- CE3604A-B: Cases 5, 8, 9, and 11 are inconsistent with AI-00050. SKIP_LINE is called only if the end of the output string has not been met.

- CE3704M-B: A superfluous SKIP_LINE causes the input and output operations to be out of synchronization.

### 4.2.7 Description of Inapplicable Tests

AE2101C was inapplicable because the implementation's version of SEQUENTIAL_IO, DIRECT_IO, and TEXT_IO did not allow for instantiation with unconstrained array and record types.

B52004E, B55B09D, B86001CR, C34001D, and C55B07B were inapplicable because the implementation does not support SHORT_INTEGER.

B86001CP, C34001F, and C35702A were inapplicable because the implementation does not support SHORT_FLOAT.

B86001DT was inapplicable because the implementation does not support LONG_LONG_INTEGER.

BA2001E is inapplicable because Ada RM 10 2/5.4 states that the "simple names of all subunits that have the same ancestor library unit must be distinct identifiers". The test expects that the above condition be checked at the point of the declaration of the stub. The implementation detects a duplicate subunit name under a single ancestor library unit when the subunit itself is being compiled. No program library will contain duplicate subunits since the second of the subunits will be rejected.

BC3009A, BC3009B, BC3009D are inapplicable because of detection of generic instantiation only on an instantiation of a "real" entity - an instantiation outside of a generic entity. Since no subprogram or package is instantiated outside of a generic unit in these tests, the circularity is not detected. In essence, since generics are treated as templates, only a "real" instantiation actually brings a copy into being: circularity within a template is tolerated through no instantiation of this template will be legal.

The following tests were inapplicable because they exceed the accuracy of the floating-point definition for the target implementation:

C35705F through C35705Y (20)
C35706F through C35706Y (20)
C35707F through C35707Y (20)
C35708F through C35707Y (20)
C35802F through C35802Y (20)
C45241F through C45241Y (20)
C45321F through C45321Y (20)
C45421F through C45421Y (20)
C45424F through C45424Y (20)
C45621F through C45621Z (21). (=201 tests, total)

C24113F through C24113Y, and CE3605A were in applicable because the implementation does not support the line length specified in the tests. (21 tests)

C64103A was inapplicable because line 66 value of LARGE not big enough to cause VMS exception "floating error".

C86001F was inapplicable because "SYSTEM" is recompiled, requiring all other library packages to be compiled.

C96005B - This test checks to find a difference between DURATION'BASE'FIRST and DURATION'FIRST. If no difference exists (as is the case in the implementation) the test is inapplicable.

CC3407A, CC3407D, CC3407E and CC3407F are inapplicable because the tests raise NUMERIC_ERROR for array declarations should similarly be considered inapplicable for this implementation.

CE2107B, CE2107C, CE2107D, CE2107E, CE2110B, CE3111B, CE3111C, CE3111D, CE3111D, CE3111E, CE3114B are inapplicable because the implementation does not support associating two internal files with the same external file for writing (Section F.8.1 of Appendix F).

CE3115A was inapplicable because more than one internal file was associated with the same external file.

CE2201D, CE2201E and CE2401E were inapplicable because a constraint was required for instantiated type.

### 4.2.8 Information Derived from the Tests

Processing of the following tests indicated support as described below for a variety of implementation options examined by the tests.

- E24101A-B.TST: If a based integer literal has a value exceeding SYSTEM.MAX_INT, an implementation may either reject the compilation unit at compile time or raise NUMERIC_ERROR at run time. This test showed that the compiler did not reject the compilation unit at compile time.

- B26005A.ADA: This test contains all the ASCII control characters in string literals. The system replaced the control characters corresponding to format effectors with a space in the listing file. All occurrences were identified with a diagnostic message by the compiler.

- D29002K-B.ADA: This test declares 713 identifiers and was passed by the compiler.

- E36202A-B.ADA and E36202B-B.ADA: These tests declare multidimensional null BOOLEAN arrays in which LENGTH of one dimension exceeds INTEGER'LAST and SYSTEM.MAX_INT, respectively. An implementation can accept this, or it can raise NUMERIC_ERROR or STORAGE_ERROR at run time. The compiler did accept the declarations and raised NUMERIC_ERROR during execution.

- D4A002A-AB.ADA and D4A002B.ADA: These tests contain universal integer calculations requiring 32 and 64 bits of accuracy, i.e., values that exceed SYSTEM.MAX_INT are used. An implementation is allowed to reject programs requiring such calculations. The compiler passed these tests.

- E43211B-B.ADA: If a bound in a non-null range of a non-null aggregate does not belong to an index subtype, then all choices may or may not be evaluated before CONSTRAINT_ERROR is raised. The compiler did not evaluate all choices before CONSTRAINT_ERROR is raised.

- E43212B-B.ADA: This test examines whether or not all choices are evaluated before subaggregates are checked for identical bounds. The compiler evaluates all subaggregates for identical bounds.

- E52103Y-B.ADA, C52104X-B.ADA, C52104Y-B.ADA: These tests declare BOOLEAN arrays with INTEGER'LAST+3 components. An implementation may raise NUMERIC_ERROR at the type declaration or STORAGE_ERROR when array objects of these types are declared, or it may accept the type and object declarations. The compiler did not raise NUMERIC_ERROR for null array with one dimension of length greater than INTEGER'LAST in E52103Y-B.

- A series of tests (D55A03*-AB.ADA) checks to see what level of loop nesting is allowed by an implementation. Tests containing up to 65 nested loops passed without exceeding the implementation's capacity.

- D56001B-AB.ADA contains blocks nested 65 levels deep. This test was passed.

- C94004A-B.ADA: This test checks to see what happens when a library unit initiates a task and a main program terminates without ensuring that the library unit's task is terminated. This test showed that such library tasks continued to execute even after the main program terminates and then terminated appropriately by themselves.

- CA1012A4M-B.DEP: This test checks whether an implementation requires generic library unit bodies to be compiled in the same compilation as the generic declaration. The compiler does allow generic declarations and bodies to be compiled in completely separate compilations.

- BC3204C*-B.ADA and BC3205D*-B.ADA: These tests contain a separately compiled generic declaration, some instantiations, and a body. An implementation must reject either the instantiations or the body. The compiler generated errors when compiling the generic package body.

- CE2106A-B.DEP and CE3110A-B.DEP: These tests confirm that dynamic creation and deletion of files is not supported.

- CE2107A.DEP: These tests showed that more than one internal file may be associated with the same external file.

- CE2110B-B.DEP: This test confirmed that an external file associated with more than one internal file can not be deleted.

- EE3102C-B.ADA: This test confirmed that an Ada program can open an existing file in mode OUT_FILE but can not create an existing file in either mode OUT_FILE or IN FILE.

- CE3111A-B.DEP showed that two internal files may read the same external file.

- CE3111B-B.DEP and CE3111C-B.DEP showed that the compiler does not allow two internal TEXT_IO files to be associated with the same external file when one or both internal files are opened for writing.

## 5. SUMMARY AND CONCLUSIONS

ACVC version 1.6 comprises 2162 tests, of which 56 were withdrawn due to errors. The Federal Software Management Support Center (FSMSC), an AVF, identified 1848 of the remaining tests to be applicable to SofTech, Inc.'s AdaVAX compiler. The compiler passed all of these tests, operating on the VAX 8600, VAX-11/785, VAX-11/780, and MicroVAX II under VMS 4.1 and MicroVMS 4.1M.

The FSMSC considers these results to show acceptable conformity to the Ada Standard.

END

9-87

DTIC