



# REPORT DOCUMENTATION PAGE

Form Approved  
OPM No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Information and Regulatory Affairs, Office of Management and Budget, Washington, DC 20503.

AD-A224 562

1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE	3. REPORT TYPE AND DATES COVERED Final 24 Nov. 1989 to 24 Nov. 1990	
4. TITLE AND SUBTITLE Ada Compiler Validation Summary Report: Alsys Limited, AlsysCOMP_017 V4.3, Micro VAX II (Host) to IMOS T222 transputer implemented on a B416 TRAM (bare) (Target), 891124N1.10202			5. FUNDING NUMBERS	
6. AUTHOR(S) National Computing Centre Limited Manchester, UNITED KINGDOM			8. PERFORMING ORGANIZATION REPORT NUMBER AVF-VSR-90502/59	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) National Computing Centre Limited Oxford Road Manchester M1 7ED UNITED KINGDOM			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Ada Joint Program Office United States Department of Defense Washington, D.C. 20301-3081			11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words)  Alsys Limited, AlsysCOMP_017 V4.3, Manchester, England, MicroVAX II under MicroVMS V4.7 (Host) to IMOS T222 transputer implemented on a B416 TRAM (bare) using an IBM PC/AT under MS-DOS 3.1 running IMOS Iserver V1.41 for file-server support via a CAPLIN QTO board link (Target), ACVC 1.10.				
14. SUBJECT TERMS Ada programming language, Ada Compiler Validation Summary Report, Ada Compiler Validation Capability, Validation Testing, Ada Validation Office, Ada Validation Facility, ANSI/MIL-STD-1815A, Ada Joint Program Office			15. NUMBER OF PAGES	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED			18. PRICE CODE	
18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED		19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED		20. LIMITATION OF ABSTRACT

DTIC  
ELECTE  
JUL 26 1990  
S E D

TABLE OF CONTENTS

CHAPTER 1  
 INTRODUCTION ..... 1  
     1.1 PURPOSE OF THIS VALIDATION SUMMARY REPORT .... 1  
     1.2 USE OF THIS VALIDATION SUMMARY REPORT ..... 2  
     1.3 REFERENCES ..... 2  
     1.4 DEFINITION OF TERMS ..... 3  
     1.5 ACVC TEST CLASSES ..... 4

CHAPTER 2  
 CONFIGURATION INFORMATION ..... 1  
     2.1 CONFIGURATION TESTED ..... 1  
     2.2 IMPLEMENTATION CHARACTERISTICS ..... 1

CHAPTER 3  
 TEST INFORMATION ..... 1  
     3.1 TEST RESULTS ..... 1  
     3.2 SUMMARY OF TEST RESULTS BY CLASS ..... 1  
     3.3 SUMMARY OF TEST RESULTS BY CHAPTER ..... 1  
     3.4 WITHDRAWN TESTS ..... 1  
     3.5 INAPPLICABLE TESTS ..... 2  
     3.6 TEST, PROCESSING, AND EVALUATION MODIFICATIONS .. 6  
     3.7 ADDITIONAL TESTING INFORMATION ..... 7

APPENDIX A  
 DECLARATION OF CONFORMANCE ..... 1

APPENDIX B  
 APPENDIX F OF THE Ada STANDARD ..... 1

APPENDIX C  
 TEST PARAMETERS ..... 1

APPENDIX D  
 WITHDRAWN TESTS ..... 1

Ada COMPILER  
VALIDATION SUMMARY REPORT:  
Certificate Number: #891124N1.10202  
Alsys Limited  
AlsyCOMP\_017 V4.3  
MicroVAX II host and

INMOS T222 transputer implemented on a B416 TRAM (bare),  
using an IBM PC/AT under MS-DOS 3.1 running INMOS  
Iserver V1.41 for file-server support via a CAPLIN QT0 board link



Completion of On-Site Testing:  
24 November 1989

Prepared By:  
Testing Services  
The National Computing Centre Limited  
Oxford Road  
Manchester M1 7ED  
England

Prepared For:  
Ada Joint Program Office  
United States Department of Defense  
Washington DC 20301-3081

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input checked="" type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/ _____	
Availability Codes	
Dist	Avail and/or Special
A-1	

Ada Compiler Validation Summary Report:

Compiler Name: **AlsyCOMP\_017 V4.3**

Certificate Number: **#891124N1.10202**

Host: **MicroVAX II under MicroVMS V4.7**

Target: **INMOS T222 transputer implemented on a B416 TRAM (bare) using an IBM PC/AT under MS-DOS 3.1 running INMOS Iserver V1.41 for file-server support via a CAPLIN QT0 board link**

**Testing Completed 24 November 1989 Using ACVC 1.10**

This report has been reviewed and is approved.

*J. Pink*

---

Jane Pink  
Testing Services Manager  
The National Computing Centre Limited  
Oxford Road  
Manchester M1 7ED  
England

*John F. Kramer*

---

Ada Validation Organization  
Dr. John F. Kramer  
Institute for Defense Analyses  
Alexandria VA 22311

*John P. Solomond*

---

Ada Joint Program Office  
Dr. John Solomond  
Director AJPO  
Department of Defense  
Washington DC 20301

---

**CHAPTER 1****INTRODUCTION**

This Validation Summary Report (VSR) describes the extent to which a specific Ada compiler conforms to the Ada Standard, ANSI/MIL-STD-1815A. This report explains all technical terms used within it and thoroughly reports the results of testing this compiler using the Ada Compiler Validation Capability (ACVC). An Ada compiler must be implemented according to the Ada Standard, and any implementation-dependent features must conform to the requirements of the Ada Standard. The Ada Standard must be implemented in its entirety, and nothing can be implemented that is not in the Standard.

Even though all validated Ada compilers conform to the Ada Standard, it must be understood that some differences do exist between implementations. The Ada Standard permits some implementation dependencies -- for example, the maximum length of identifiers or the maximum values of integer types. Other differences between compilers result from the characteristics of particular operating systems, hardware, or implementation strategies. All the dependencies observed during the process of testing this compiler are given in this report.

The information in this report is derived from the test results produced during validation testing. The validation process includes submitting a suite of standardized tests, the ACVC, as inputs to an Ada compiler and evaluating the results. The purpose of validating is to ensure conformity of the compiler to the Ada Standard by testing that the compiler properly implements legal language constructs and that it identifies and rejects illegal language constructs. The testing also identifies behavior that is implementation dependent, but is permitted by the Ada Standard. Six classes of tests are used. These tests are designed to perform checks at compile time, at link time, and during execution.

**1.1 PURPOSE OF THIS VALIDATION SUMMARY REPORT**

This VSR documents the results of the validation testing performed on an Ada compiler. Testing was carried out for the following purposes:

- o To attempt to identify any language constructs supported by the compiler that do not conform to the Ada Standard
- o To attempt to identify any language constructs not supported by the compiler but required by the Ada Standard
- o To determine that the implementation-dependent behavior is allowed by the Ada Standard

---

Testing of this compiler was conducted by The National Computer Centre Limited according to procedures established by the Ada Joint Program Office and administered by the Ada Validation Organization (AVO). On-site testing was completed 24 November 1989 at Alsys Limited, Partridge House, Newton Road, Henley-on-Thames, Oxon, RG9 1EN, UK.

## 1.2 USE OF THIS VALIDATION SUMMARY REPORT

Consistent with the national laws of the originating country, the AVO may make full and free public disclosure of this report. In the United States, this is provided in accordance with the "Freedom of Information Act" (5 U.S.C. #552). The results of this validation apply only to the computers, operating systems, and compiler versions identified in this report.

The organizations represented on the signature page of this report do not represent or warrant that all statements set forth in this report are accurate and complete, or that the subject compiler has no nonconformities to the Ada Standard other than those presented. Copies of this report are available to the public from:

**Ada Information Clearinghouse  
Ada Joint Program Office  
OUSDRE  
The Pentagon, Rm 3D-139 (Fern Street)  
Washington DC 20301-3081**

or from:

**Testing Services  
The National Computing Centre Limited  
Oxford Road  
Manchester M1 7ED  
England**

Questions regarding this report or the validation test results should be directed to the AVF listed above or to:

**Ada Validation Organization  
Institute for Defense Analyses  
1801 North Beauregard Street  
Alexandria VA 22311**

## 1.3 REFERENCES

1. Reference Manual for the Ada Programming Language, ANSI/MIL-STD-1815A, February 1983 and ISO 8652-1987.

- 
2. Ada Compiler Validation Procedures and Guidelines,  
Ada Joint Program Office, 1 January 1987.
  3. Ada Compiler Validation Capability Implementers' Guide,  
SofTech, Inc., December 1986.
  4. Ada Compiler Validation Capability User's Guide,  
December 1986.

#### 1.4 DEFINITION OF TERMS

ACVC	The Ada Compiler Validation Capability. The set of Ada programs that tests the conformity of an Ada compiler to the Ada programming language.
Ada Commentary	An Ada Commentary contains all information relevant to the point addressed by a comment on the Ada Standard. These comments are given a unique identification number having the form AI-ddddd.
Ada Standard	ANSI/MIL-STD-1815A, February 1983 and ISO 8652-1987.
Applicant	The agency requesting validation.
AVF	The Ada Validation Facility. The AVF is responsible for conducting compiler validations according to procedures contained in the <u>Ada Compiler Validation Procedures and Guidelines</u> .
AVO	The Ada Validation Organization. The AVO has oversight authority over all AVF practices for the purpose of maintaining a uniform process for validation of Ada compilers. The AVO provides administrative and technical support for Ada validations to ensure consistent practices.
Compiler	A processor for the Ada language. In the context of this report, a compiler is any language processor, including cross-compilers, translators, and interpreters.
Failed test	An ACVC test for which the compiler generates a result that demonstrates nonconformity to the Ada Standard.
Host	The computer on which the compiler resides.
Inapplicable test	An ACVC test that uses features of the language that a compiler is not required to support or may legitimately support in a way other than the one expected by the test.

---

Passed test	An ACVC test for which a compiler generates the expected result.
Target	The computer which executes the code generated by the compiler.
Test	A program that checks a compiler's conformity regarding a particular feature or a combination of features to the Ada Standard. In the context of this report, the term is used to designate a single test, which may comprise one or more files.
Withdrawn test	An ACVC test found to be incorrect and not used to check conformity to the Ada Standard. A test may be incorrect because it has an invalid test objective, fails to meet its test objective, or contains illegal or erroneous use of the language.

### 1.5 ACVC TEST CLASSES

Conformity to the Ada Standard is measured using the ACVC. The ACVC contains both legal and illegal Ada programs structured into six test classes: A, B, C, D, E, and L. The first letter of a test name identifies the class to which it belongs. Class A, C, D, and E tests are executable, and special program units are used to report their results during execution. Class B tests are expected to produce compilation errors. Class L tests are expected to produce errors because of the way in which a program library is used at link time.

Class A tests ensure the successful compilation and execution of legal Ada programs with certain language constructs which cannot be verified at run time. There are no explicit program components in a Class A test to check semantics. For example, a Class A test checks that reserved words of another language (other than those already reserved in the Ada language) are not treated as reserved words by an Ada compiler. A Class A test is passed if no errors are detected at compile time and the program executes to produce a PASSED message.

Class B tests check that a compiler detects illegal language usage. Class B tests are not executable. Each test in this class is compiled and the resulting compilation listing is examined to verify that every syntax or semantic error in the test is detected. A Class B test is passed if every illegal construct that it contains is detected by the compiler.

Class C tests check the run time system to ensure that legal Ada programs can be correctly compiled and executed. Each Class C test is self-checking and produces a PASSED, FAILED, or NOT APPLICABLE message indicating the result when it is executed.

Class D tests check the compilation and execution capacities of a compiler. Since there are no capacity requirements placed on a compiler by the Ada Standard for some parameters -- for example, the number of identifiers permitted in a compilation or the number of units in a library - a compiler may refuse to compile a Class D test and still be a conforming compiler. Therefore, if a Class D test fails to compile because the capacity of the compiler is exceeded, the test is classified as inapplicable. If a Class D test compiles successfully, it is self-checking and produces a PASSED or FAILED message during execution.



Class E tests are expected to execute successfully and check implementation-dependent options and resolutions of ambiguities in the Ada Standard. Each Class E test is self-checking and produces a NOT APPLICABLE, PASSED, or FAILED message when it is compiled and executed. However, the Ada Standard permits an implementation to reject programs containing some features addressed by Class E tests during compilation. Therefore, a Class E test is passed by a compiler if it is compiled successfully and executes to produce a PASSED message, or if it is rejected by the compiler for an allowable reason.

Class L tests check that incomplete or illegal Ada programs involving multiple, separately compiled units are detected and not allowed to execute. Class L tests are compiled separately and execution is attempted. A Class L test passes if it is rejected at link time -- that is, an attempt to execute the main program must generate an error message before any declarations in the main program or any units referenced by the main program are elaborated. In some cases, an implementation may legitimately detect errors during compilation of the test.

Two library units, the package REPORT and the procedure CHECK\_FILE, support the self-checking features of the executable tests. The package REPORT provides the mechanism by which executable tests report PASSED, FAILED, or NOT APPLICABLE results. It also provides a set of identity functions used to defeat some compiler optimizations allowed by the Ada Standard that would circumvent a test objective. The procedure CHECK\_FILE is used to check the contents of text files written by some of the Class C tests for Chapter 14 of the Ada Standard. The operation of REPORT and CHECK\_FILE is checked by a set of executable tests. These tests produce messages that are examined to verify that the units are operating correctly. If these units are not operating correctly, then the validation is not attempted.

The text of each test in the ACVC follows conventions that are intended to ensure that the tests are reasonably portable without modification. For example, the tests make use of only the basic set of 55 characters, contain lines with a maximum length of 72 characters, use small numeric values, and place features that may not be supported by all implementations in separate tests. However, some tests contain values that require the test to be customized according to implementation-specific values -- for example, an illegal file name. A list of the values used for this validation is provided in Appendix C.

A compiler must correctly process each of the tests in the suite and demonstrate conformity to the Ada Standard by either meeting the pass criteria given for the test or by showing that the test is inapplicable to the implementation. The applicability of a test to an implementation is considered each time the implementation is validated. A test that is inapplicable for one validation is not necessarily inapplicable for a subsequent validation. Any test that was determined to contain an illegal language construct or an erroneous language construct is withdrawn from the ACVC and, therefore, is not used in testing a compiler. The tests withdrawn at the time of this validation are given in Appendix D.

CHAPTER 2

CONFIGURATION INFORMATION

2.1 CONFIGURATION TESTED

The candidate compilation system for this validation was tested under the following configuration:

Compiler: **AlsyCOMP\_017 V4.3**

ACVC Version: **1.10**

Certificate Number: **#891124N1.10202**

Host Computer:

Machine: **MicroVAX II**

Operating System: **MicroVMS V4.7**

Memory Size: **9 Mb**

Target Computer:

Machine: **INMOS T222 transputer implemented on a B416 TRAM (bare) using an IBM PC/AT under MS-DOS 3.1 running INMOS Iserver V1.41 for file-server support via a CAPLIN QT0 board link**

Memory Size: **64 Kb**

Communications Network: **CAPLIN QT0 Board**

2.2 IMPLEMENTATION CHARACTERISTICS

One of the purposes of validating compilers is to determine the behavior of a compiler in those areas of the Ada Standard that permit implementations to differ. Class D and E tests specifically check for such implementation differences. However, tests in other classes also characterize an implementation. The tests demonstrate the following characteristics:

- a. Capacities.

- (1) The compiler correctly processes a compilation containing 723 variables in the same declarative part. (See test D29002K.)
  - (2) The compiler correctly processes tests containing loop statements nested to 65 levels. (See tests D55A03A..H (8 tests).)
  - (3) The compiler correctly processes tests containing block statements nested to 65 levels. (See test D56001B.)
  - (4) The compiler correctly processes tests containing recursive procedures separately compiled as subunits nested to six levels. (See tests D64005E..G (3 tests).)
- b. Predefined types.
- (1) This implementation supports the additional predefined types `SHORT_INTEGER`, `LONG_INTEGER`, and `LONG_FLOAT`, in the package `STANDARD`. (See tests B86001T..Z (7 tests).)
- c. Based literals.
- (1) An implementation is allowed to raise `NUMERIC_ERROR` or `CONSTRAINT_ERROR` when a value exceeds `SYSTEM.MAX_INT`. This implementation raises `NUMERIC_ERROR` during execution. (See test E24201A.)
- d. Expression evaluation.
- The order in which expressions are evaluated and the time at which constraints are checked are not defined by the language. While the ACVC tests do not specifically attempt to determine the order of evaluation of expressions, test results indicate the following:
- (1) All of the default initialization expressions for record components are evaluated before any value is checked for membership in a component's subtype. (See test C32117A.)
  - (2) Assignments for subtypes are performed with the same precision as than the base type. (See test C35712B.)
  - (3) This implementation uses no extra bits for extra precision and uses all extra bits for extra range. (See test C35903A.)
  - (4) `NUMERIC_ERROR` is raised when an integer literal operand in a comparison or membership test is outside the range of the base type. (See test C45232A.)
  - (5) `NUMERIC_ERROR` is raised when a literal operand in a fixed-point comparison or membership test is outside the range of the base type. (See test C45252A.)

(6) Underflow is gradual. (See tests C45524A..Z (26 tests).)

e. Rounding.

The method by which values are rounded in type conversions is not defined by the language. While the ACVC tests do not specifically attempt to determine the method of rounding, the test results indicate the following:

- (1) The method used for rounding to integer is round to even. (See tests C46012A..Z (26 tests).)
- (2) The method used for rounding to longest integer is round to even. (See tests C4601?A..Z (26 tests).)
- (3) The method used for rounding to integer in static universal real expressions is round to even. (See test C4A014A.)

f. Array types.

An implementation is allowed to raise `NUMERIC_ERROR` or `CONSTRAINT_ERROR` for an array having a `'LENGTH` that exceeds `STANDARD.INTEGER'LAST` and/or `SYSTEM.MAX_INT`. For this implementation:

- (1) Declaration of an array type or subtype declaration with more than `SYSTEM.MAX_INT` components raises `NUMERIC_ERROR`. (See test C36003A.)
- (2) `CONSTRAINT_ERROR` is raised when `'LENGTH` is applied to an array type with `INTEGER'LAST + 2` components. (See test C36202A.)
- (3) `CONSTRAINT_ERROR` is raised when an array type with `SYSTEM.MAX_INT + 2` components is declared. (See test C36202E.)
- (4) A packed `BOOLEAN` array having a `'LENGTH` exceeding `INTEGER'LAST` raises `CONSTRAINT_ERROR` when the array type is declared. (See test C52103X.)
- (5) A packed two-dimensional `BOOLEAN` array with more than `INTEGER'LAST` components raises `CONSTRAINT_ERROR` when the array sub types are declared. (See test C52104Y.)
- (6) In assigning one-dimensional array types, the expression is evaluated in its entirety before `CONSTRAINT_ERROR` is raised when checking whether the expression's subtype is compatible with the target's subtype. (See test C52013A.)

- (7) In assigning two-dimensional array types, the expression is not evaluated in its entirety before `CONSTRAINT_ERROR` is raised when checking whether the expression's subtype is compatible with the target's subtype. (See test C52013A.)
  
- g. A null array with one dimension of length greater than `INTEGER'LAST` may raise `NUMERIC_ERROR` or `CONSTRAINT_ERROR` either when declared or assigned. Alternatively, an implementation may accept the declaration. However, lengths must match in array slice assignments. This implementation raises `CONSTRAINT_ERROR` when the array type is declared. (See test E52103Y.)
  
- h. Discriminated types.
  - (1) In assigning record types with discriminants, the expression is evaluated in its entirety before `CONSTRAINT_ERROR` is raised when checking whether the expression's subtype is compatible with the target's subtype. (See test C52013A.)
  
- i. Aggregates.
  - (1) In the evaluation of a multi-dimensional aggregate, the test results indicate that all choices are evaluated before checking against the index type. See test C43207A and C43207B.)
  - (2) In the evaluation of an aggregate containing subaggregates, not all choices are evaluated before being checked for identical bounds. (See test E43212B.)
  - (3) `CONSTRAINT_ERROR` is raised after all choices are evaluated when a bound in a non-null range of a non-null aggregate does not belong to an index subtype. (See test E43211B.)
  
- j. Pragmas.
  - (1) The pragma `INLINE` is supported for functions and procedures, but not for functions called inside a package specification. (See tests LA3004A..B (2 tests), EA3004C..D (2 tests), and CA3004E..F (2 tests).)
  
- k. Generics.
  - (1) Generic specifications and bodies can be compiled in separate compilations. (See tests CA1012A, CA2009C, CA2009F, BC3204C, and BC3205D.)

- (2) Generic subprogram declarations and bodies can be compiled in separate compilations. (See tests CA1012A and CA2009F.)
- (3) Generic library subprogram specifications and bodies can be compiled in separate compilations. (See test CA1012A.)
- (4) Generic non-library package bodies as subunits can be compiled in separate compilations. (See test CA2009C.)
- (5) Generic non-library subprogram bodies can be compiled in separate compilations from their stubs. (See test CA2009F.)
- (6) Generic unit bodies and their subunits can be compiled in separate compilations. (See test CA3011A.)
- (7) Generic package declarations and bodies can be compiled in separate compilations. (See tests CA2009C, BC3204C, and BC3205D.)
- (8) Generic library package specifications and bodies can be compiled in separate compilations. (See tests BC3204C and BC3205D.)
- (9) Generic unit bodies and their subunits can be compiled in separate compilations. (See test CA3011A.)

l. Input and output.

- (1) The package SEQUENTIAL\_IO can be instantiated with unconstrained array types or record types with discriminants without defaults. (See tests AE2101C, EE2201D, and EE2201E.)
- (2) The package DIRECT\_IO can be instantiated with unconstrained array types or record types with discriminants without defaults. (See tests AE2101H, EE2401D, and EE2401G.)
- (3) RESET and DELETE operations are supported for SEQUENTIAL\_IO. (See tests CE2102G and CE2102X.)
- (4) RESET and DELETE operations are supported for DIRECT\_IO. (See tests CE2102K and CE2102Y.)
- (5) RESET and DELETE operations are supported for text files. (See tests CE3102F..G (2 tests), CE3104C, CE3110A, and CE3114A.)

## CONFIGURATION INFORMATION

---

- (6) Overwriting to a sequential file truncates to the last element written. (See test CE2208B.)
- (7) Temporary sequential files are given names and deleted when closed. (See test CE2108A.)
- (8) Temporary direct files are given names and deleted when closed. (See test CE2108C.)
- (9) Temporary text files are given names and deleted when closed. (See test CE3112A.)
- (10) Only one internal file can be associated with each external file for sequential files when writing or reading. (See tests CE2107A..E (5 tests), CE2102L, CE2110B, and CE2111D.)
- (11) Only one internal file can be associated with each external file for direct files when writing or reading. (See tests CE2107F..H (3 tests), CE2110D and CE2111H.)
- (12) Only one internal file can be associated with each external file for text files when writing or reading. (See tests CE3111A..E (5 tests), CE3114B, and CE3115A.)

## CHAPTER 3

## TEST INFORMATION

3.1 TEST RESULTS

Version 1.10 of the ACVC comprises 3717 tests. When this compiler was tested, 44 tests had been withdrawn because of test errors. The AVF determined that 429 tests were inapplicable to this implementation. All inapplicable tests were processed during validation testing except for 201 executable tests that use floating-point precision exceeding that supported by the implementation. Modifications to the code, processing, or grading for 49 tests were required to successfully demonstrate the test objective. (See section 3.6.)

The AVF concludes that the testing results demonstrate acceptable conformity to the Ada Standard.

3.2 SUMMARY OF TEST RESULTS BY CLASS

RESULT	TEST CLASS						TOTAL
	<u>A</u>	<u>B</u>	<u>C</u>	<u>D</u>	<u>E</u>	<u>L</u>	
Passed	129	1134	1897	15	23	46	3244
Inapplicable	0	4	418	2	5	0	429
Withdrawn	1	2	35	0	6	0	44
TOTAL	130	1140	2350	17	34	46	3717

3.3 SUMMARY OF TEST RESULTS BY CHAPTER

RESULT	CHAPTER														TOTAL
	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>	<u>8</u>	<u>9</u>	<u>10</u>	<u>11</u>	<u>12</u>	<u>13</u>	<u>14</u>		
Passed	198	577	555	248	170	99	161	332	137	36	252	250	229	3244	
Inapp	14	72	125	0	2	0	5	0	0	0	0	119	92	429	
Withdrawn	1	1	0	0	0	0	0	2	0	0	1	35	4	44	
TOTAL	213	650	680	248	172	99	166	334	137	36	253	404	325	3717	

3.4 WITHDRAWN TESTS



The following 44 tests were withdrawn from ACVC Version 1.10 at the time of this validation:

E28005C	A39005G	B97102E	C97116A
BC3009B	CD2A62D	CD2A63A..D (4 tests)	CD2A66A..D (4 tests)
CD2A73A..D (4 tests)	CD2A76A..D (4 tests)	CD2A81G	CD2A83G
CD2A84M..N (2 tests)	CD5011O	CD2B15C	CD7205C
CD2D11B	CD5007B	ED7004B	ED7005C..D (2 tests)
ED7006C..D (2 tests)	CD7105A	CD7203B	CD7204B
CD7205D	CE2107I	CE3111C	CE3301A
CE3411B			

See Appendix D for the reason that each of these tests was withdrawn.

### 3.5 INAPPLICABLE TESTS

Some tests do not apply to all compilers because they make use of features that a compiler is not required by the Ada Standard to support. Others may depend on the result of another test that is either inapplicable or withdrawn. The applicability of a test to an implementation is considered each time a validation is attempted. A test that is inapplicable for one validation attempt is not necessarily inapplicable for a subsequent attempt. For this validation attempt, 429 tests were inapplicable for the reasons indicated:

- a. The following 201 tests are not applicable because they have floating-point type declarations requiring more digits than SYSTEM.MAX\_DIGITS:
  - C24113L..Y (14 tests)
  - C35705L..Y (14 tests)
  - C35706L..Y (14 tests)
  - C35707L..Y (14 tests)
  - C35708L..Y (14 tests)
  - C35802L..Z (15 tests)
  - C45241L..Y (14 tests)
  - C45321L..Y (14 tests)
  - C45421L..Y (14 tests)
  - C45521L..Z (15 tests)
  - C45524L..Z (15 tests)
  - C45621L..Z (15 tests)
  - C45641L..Y (14 tests)
  - C46012L..Z (15 tests)
- b. C35702A and B86001T are not applicable because this implementation supports no predefined type SHORT\_FLOAT.
- c. C45531M..P (4 tests) and C45532M..P (4 tests) are inapplicable because the size of a mantissa of a fixed point type is limited to 31 bits.
- d. D64005F..G (2 tests) are inapplicable because the recursive capacity within these tests exceed the capacity for this implementation.
- e. C86001F, is not applicable because, for this implementation, the package TEXT\_IO is dependent upon package SYSTEM. This test recompile package SYSTEM, making package TEXT\_IO, and hence package REPORT, obsolete.

- f. B86001X, C45231D, and CD7101G are not applicable because this implementation does not support any predefined integer type with a name other than INTEGER, LONG\_INTEGER, or SHORT\_INTEGER.
- g. B86001Y is not applicable because this implementation supports no predefined fixed-point type other than DURATION.
- h. B86001Z is not applicable because this implementation supports no predefined floating-point type with a name other than FLOAT, LONG\_FLOAT, or SHORT\_FLOAT.
- i. CD1009C, CD2A41A..E (5 tests) and CD2A42A..J (10 tests) are not applicable because the SIZE clause on type FLOAT is not supported by this implementation.
- j. The following 26 tests are inapplicable because for this implementation a length clause on a type derived from a private type is not supported outside the defined package.

CD1C04A	CD2A21C..D (2 tests)	CD2A22C..D (2 tests)
CD2A22G..H (2 tests)	CD2A31C..D (2 tests)	CD2A32C..D (2 tests)
CD2A32G..H (2 tests)	CD2A51C..D (2 tests)	CD2A52C..D (2 tests)
CD2A52G..H (2 tests)	CD2A53D	CD2A54D
CD2A54H	CD2A72A..B (2 tests)	CD2A75A..B (2 tests)

- k. CD1C04B, CD1C04E, CD4051A..B (2 tests) and CD4051C..D (2 tests) are not applicable because this implementation does not support representation clauses in derived records or derived tasks.
- l. The following 53 tests are inapplicable for this implementation because they exceeded the capacity limit for the target configuration.

CD2A53C	CE2401A..C (3 tests)	CE3305A
CE3402C	CD3405A	CE3405D
CE3409C	CE3410C	CE3602B
CE3605D..E (2 tests)	CE3606A..B (2 tests)	CE3704F
CE3706F	CE3801A..B (2 tests)	CE3804A..P (16 tests)
CE3805A..B (2 tests)	CE3806A..H (8 tests)	CE3809A..B (2 tests)
CE3906A	CE3906E..F (2 tests)	EE3203A
EE3301B	EE3402B	

- m. The following 25 tests are inapplicable because a LENGTH clause on an array or record would require a change to the representation of the components or elements.

CD2A61A..D (4 tests)	CD2A61F	CD2A61H..L (5 tests)
CD2A62A..C (3 tests)	CD2A71A..D (4 tests)	CD2A72C..D (2 tests)
CD2A74A..D (4 tests)	CD2A75C..D (2 tests)	

- 
- n. CD2A84B..I (8 tests) and CD2A84K..L (2 tests) are not applicable because the 'SIZE clause applied to the access type is less than the minimum (32 bits) required.
  - o. The following 30 tests are inapplicable because an ADDRESS clause for a constant is not supported.
 

CD5011B	CD5011D	CD5011F
CD5011H	CD5011L	CD5011N
CD5011R..S (2 tests)	CD5012C..D (2 tests)	CD5012G..H (2 tests)
CD5012L	CD5013B	CD5013D
CD5013F	CD5013H	CD5013L
CD5013N	CD5013R	CD5014B
CD5014D	CD5014F	CD5014H
CD5014J	CD5014L	CD5014N
CD5014R	CD5014U	CD5014W
  - p. CD5012J, CD5013S and CD5014S are not applicable because ADDRESS clauses for tasks are not supported.
  - q. CE2102D is inapplicable because this implementation supports CREATE with IN\_FILE mode for SEQUENTIAL\_IO.
  - r. CE2102E is inapplicable because this implementation supports CREATE with OUT\_FILE mode for SEQUENTIAL\_IO.
  - s. CE2102F is inapplicable because this implementation supports CREATE with INOUT\_FILE mode for DIRECT\_IO.
  - t. CE2102I is inapplicable because this implementation supports CREATE with IN\_FILE mode for DIRECT\_IO.
  - u. CE2102J is inapplicable because this implementation supports CREATE with OUT\_FILE mode for DIRECT\_IO.
  - v. CE2102N is inapplicable because this implementation supports OPEN with IN\_FILE mode for SEQUENTIAL\_IO.
  - w. CE2102O is inapplicable because this implementation supports RESET with IN\_FILE mode for SEQUENTIAL\_IO.
  - x. CE2102P is inapplicable because this implementation supports OPEN with OUT\_FILE mode for SEQUENTIAL\_IO.
  - y. CE2102Q is inapplicable because this implementation supports RESET with OUT\_FILE mode for SEQUENTIAL\_IO.

- 
- z. CE2102R is inapplicable because this implementation supports OPEN with INOUT\_FILE mode for DIRECT\_IO.
  - za. CE2102S is inapplicable because this implementation supports RESET with INOUT\_FILE mode for DIRECT\_IO.
  - zb. CE2102T is inapplicable because this implementation supports OPEN with IN\_FILE mode for DIRECT\_IO.
  - zc. CE2102U is inapplicable because this implementation supports RESET with IN\_FILE mode for DIRECT\_IO.
  - zd. CE2102V is inapplicable because this implementation supports OPEN with OUT\_FILE mode for DIRECT\_IO.
  - ze. CE2102W is inapplicable because this implementation supports RESET with OUT\_FILE mode for DIRECT\_IO.
  - zf. EE2401D, EE2401G and CE2401H use instantiations of package DIRECT\_IO with unconstrained array types and record types with discriminants with defaults. These instantiations cause USE\_ERROR to be raised without a FORM parameter.
  - zg. CE2107B..E (4 tests), CE2107L, and CE2110B are not applicable because multiple internal files cannot be associated with the same external file when one or more files is writing for sequential files. The proper exception is raised when *multiple access is attempted*.
  - zh. CE2107G..H (2 tests), CE2110D, and CE2111H are not applicable because multiple internal files cannot be associated with the same external file when one or more files is writing for direct files. The proper exception is raised when *multiple access is attempted*.
  - zi. CE3102E is inapplicable because text file CREATE with IN\_FILE mode is supported by this implementation.
  - zj. CE3102F is inapplicable because text file RESET is supported by this implementation.
  - zk. CE3102G is inapplicable because text file deletion of an external file is supported by this implementation.
  - zl. CE3102I is inapplicable because text file CREATE with OUT\_FILE mode is supported by this implementation.
  - zm. CE3102J is inapplicable because text file OPEN with IN\_FILE mode is supported by this implementation.
  - zn. CE3102K is inapplicable because text file OPEN with OUT\_FILE mode is not supported by this implementation.

- ao. CE3111B, CE3111D..E (2 tests), CE3114B, and CE3115A are not applicable because multiple internal files cannot be associated with the same external file when one or more files is writing for text files. The proper exception is raised when multiple access is attempted.
- ap. CE3202A requires association of a name with the standard input/output files, but this is not supported by this implementation which raises USE\_ERROR. This behaviour is accepted by the AVO pending a ruling by the language maintenance body.
- aq. CE3605A is inapplicable because this test attempts to output a string of 517 characters which exceeds the maximum allowed for this implementation.

### 3.6 TEST, PROCESSING, AND EVALUATION MODIFICATIONS

It is expected that some tests will require modifications of code, processing, or evaluation in order to compensate for legitimate implementation behavior. Modifications are made by the AVF in cases where legitimate implementation behavior prevents the successful completion of an (otherwise) applicable test. Examples of such modifications include: adding a length clause to alter the default size of a collection; splitting a Class B test into subtests so that all errors are detected; and confirming that messages produced by an executable test demonstrate conforming behavior that was not anticipated by the test (such as raising one exception instead of another).

Modifications were required for 49 tests.

The following tests were split because syntax errors at one point resulted in the compiler not detecting other errors in the test:

B23004A	B24007A	B24009A	B28003A	B28003C
B32202A	B32202B	B32202C	B33001A	B37004A
B45102A	B61012A	B62001B	B62001C	B62001D
B74304A	B74401F	B74401R	B91004A	B95069A
B95069B	B97103E	BA1101B2	BA1101B4	BC2001D
BC3009C	BD5005B			

C35A06O and C35A06Q..R (2 tests) were split to allow the executable code, for these multiple generic tests, generated by this compiler to remain within the bounds of the target memory restrictions.

The following tests were split to prove the not-applicability criteria:

CD2A62A	CD2A62B	CD2A72A	CD2A72B	CD2A75A
CD2A75B	CD2A84B	CD2A84C	CD2A84D	CD2A84E
CD2A84F	CD2A84G	CD2A84H	CD2A84I	

EA3004D, when processed, produces only two of the expected three errors: the implementation fails to detect an error on line 27 of file EA3004D6M. This is because the pragma INLINE has

no effect when its object is within a package specification. The task was reordered to compile files D2 and D3 after file D5 ( the re-compilation of the "with"ed package that makes the various earlier units obsolete), the re-ordered test executed and produced the expected NOT\_APPLICABLE result (as though INLINE were not supported at all). The re-ordering of EA3004D test files was: 0-1-4-5-2-3-6. The AVO ruled that the test should be counted as passed.

This implementation implements REPORT based on I/O mechanisms other than TEXT\_IO. For this reason modifications were required for 4 tests as follows:

CE3201A required modification to the analysis of the output because the order that output is passed for logging is not necessarily the order that the output is actually logged.

EE3405B required the addition of an additional TEXT\_IO output to ensure that the new page command does actually initiate a new page.

EE3401F and EE3412C require that the output to be logged in the correct order and hence the AVO authorised that the calls for "SPECIAL\_ACTION" were replaced with "PUT\_LINE".

### 3.7 ADDITIONAL TESTING INFORMATION

#### 3.7.1 Prevalidation

Prior to validation, a set of test results for ACVC Version 1.10 produced by the **AlsYCOMP\_017 V4.3** compiler was submitted to the AVF by the applicant for review. Analysis of these results demonstrated that the compiler successfully passed all applicable tests, and the compiler exhibited the expected behavior on all inapplicable tests.

#### 3.7.2 Test Method

Testing of the **AlsYCOMP\_017 V4.3** compiler using ACVC Version 1.10 was conducted on-site by a validation team from the AVF. The configuration in which the testing was performed is described by the following designations of hardware and software components:

Host computer	: MicroVAX II
Host operating system	: MicroVMS V4.7
Target computer	: INMOS T222 transputer implemented on a B416 TRAM (bare) using an IBM PC/AT under MS-DOS 3.1 running INMOS Iserver V1.41 for file-server support via a CAPLIN QT0 board link
Compiler	: AlsYCOMP_017 V4.3
Pre-linker	: AlsYCOMP_017 V4.3
Linker	: IMS D705B ILINK V2.1
Loader/Downloader	: IMS D705B IBOOT V1.1
Runtime System	: AlsYCOMP_017 V4.3

---

The host and target computers were linked via **CAPLIN QT0 Board**.

A magnetic tape containing all tests for withdrawn tests and tests requiring unsupported floating-point precisions was taken on-site by the validation team for processing. Tests that make use of implementation-specific values were customized before being written to the magnetic tape. Tests requiring modifications during the prevalidation testing were not included in their modified form on the magnetic tape.

The contents of the magnetic tape were not loaded directly onto the host computer, but loaded on to a hard disc via a VAX 11.780. The disc was then manually switched to allow the MicroVAX to access the test files.

After the test files were loaded to disc, the full set of tests was compiled and linked on the MicroVAX, then all executable images were transferred to the INMOS T222 via the CAPLINK and run. Results were transferred from the host computer to the VAX 11/750 via FTP software from where they were printed.

The compiler was tested using command scripts provided by **Alsys Limited** and reviewed by the validation team. The compiler was tested using all the following default option settings:

<u>OPTION</u>	<u>EFFECT</u>
<b>CALLS=INLINE</b>	Allows inline insertion of code for subprograms.
<b>OBJECT=NONE</b>	No peephole optimisations are performed, this is done for compilation speed improvements.
<b>OUTPUT=&lt;file&gt;</b>	<file> specifies the name of compilation listing generated.
<b>SHOW=NONE</b>	Do not print a header and do not include an error summary in the compilation listing.
<b>ERROR=999</b>	Set the maximum number of compilation errors permitted before compilation is terminated to 999.
<b>FILE_WIDTH=80</b>	Set width for listing file to 80 columns.
<b>FILE_LENGTH=9999</b>	Disable insertion of form feeds in the output.

In addition the following option was used to produce full compiler listings:

**TEXT** Print a compilation listing including full source text.

Tests were compiled, linked, and executed (as appropriate) using a single host and target computer. Test output, compilation listings, and job logs were captured on magnetic tape and archived at the AVF. The listings examined on-site by the validation team were also archived.

3.7.3 Test Site

Testing was conducted at Alsys Limited, Partridge House, Newtown Road, Henley-on-Thames, Oxfordshire, RG9 1EN, UNITED KINGDOM and was completed on 24 November 1989.



DECLARATION OF CONFORMANCE

---

APPENDIX A

DECLARATION OF CONFORMANCE

**Alsys Limited** has submitted the following Declaration of Conformance concerning the **AlsyCOMP\_017 V4.3** compiler.

---

DECLARATION OF CONFORMANCE

Compiler Implementor:        Alsys Limited

Ada Validation Facility:     The National Computing Centre Limited,  
   Oxford Road  
   Manchester  
   M1 7ED  
   UNITED KINGDOM

Ada Compiler Validation Capability (ACVC) Version: 1.10

Base Configuration

Base Compiler Name:        AlsyCOMP\_017 V4.3

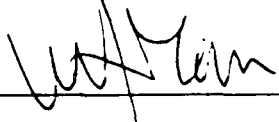
Host Architecture:         MicroVAX II

Host OS and Version:      MicroVMS V4.7

Target Architecture:      INMOS T222 transputer implemented on a B416 TRAM,  
   (bare), using an IBM PC/AT under MS-DOS 3.1 running  
   INMOS lserver V1.41 for file-server support via a CAPLIN  
   QTO board link

Implementor's Declaration

I, the undersigned, representing **Alsys Limited**, have implemented no deliberate extensions to the Ada Language Standard ANSI/MIL-STD-1815A in the compiler(s) listed in this declaration. I declare that **Alsys Limited** is the owner of record of the Ada language compiler(s) listed above and, as such, is responsible for maintaining said compiler(s) in conformance to ANSI/MIL-STD-1815A. All certificates and registrations for Ada language compiler(s) listed in this declaration shall be made only in the owner's corporate name.

  
\_\_\_\_\_ Date : 6/6/90

**Martyn Jordan**  
**Marketing Director**

DECLARATION OF CONFORMANCE

---

Owner's Declaration

I, the undersigned, representing **Alsys Limited**, take full responsibility for implementation and maintenance of the Ada compiler(s) listed above, and agree to the public disclosure of the final Validation Summary Report. I declare that all of the Ada language compilers listed, and their host/target performance, are in compliance with the Ada Language Standard ANSI/MIL-STD-1815A.



Date : 6/6/90

**Martyn Jordan**  
**Marketing Director**

## APPENDIX B

## APPENDIX F OF THE Ada STANDARD

The only allowed implementation dependencies correspond to implementation-dependent pragmas, to certain machine-dependent conventions as mentioned in chapter 13 of the Ada Standard, and to certain allowed restrictions on representation clauses. The implementation-dependent characteristics of the **AlsysCOMP\_017 V4.3** compiler, as described in this Appendix, are provided by **Alsys Limited**. Unless specifically noted otherwise, references in this appendix are to compiler documentation and not to this report. Implementation-specific portions of the package STANDARD, which are not a part of Appendix F, are:

```
package STANDARD is
```

```
...
```

```
type INTEGER is range  $-2^{15}$  ..  $2^{15}-1$ ;
```

```
type SHORT_INTEGER is range  $-2^7$  ..  $2^7-1$ ;
```

```
type LONG_INTEGER is range  $-2^{31}$  ..  $2^{31}-1$ ;
```

```
type FLOAT is digits 6 range  $-(2.0-2.0^{-(23)}) \cdot 2.0^{127}$  ..  $(2.0-2.0^{-(23)}) \cdot 2.0^{127}$ ;
```

```
type LONG_FLOAT is digits 15 range  $-(2.0-2.0^{-(51)}) \cdot 2.0^{1023}$  ..  
                                 $(2.0-2.0^{-(51)}) \cdot 2.0^{1023}$ ;
```

```
type DURATION is delta  $2.0^{14}$  range -86400.0 .. 86400.0;
```

```
...
```

```
end STANDARD;
```

## APPENDIX C

## TEST PARAMETERS

Certain tests in the ACVC make use of implementation-dependent values, such as the maximum length of an input line and invalid file names. A test that makes use of such values is identified by the extension .TST in its file name. Actual values to be substituted are represented by names that begin with a dollar sign. A value must be substituted for each of these names before the test is run. The values used for this validation are given below:

<u>Name and Meaning</u>	<u>Value</u>
\$ACC_SIZE An integer literal whose value is the number of bits sufficient to hold any value of an access type.	16
\$BIG_ID1 Identifier the size of the maximum input line length with varying last character.	(1..254=>'A', 255=>1)
\$BIG_ID2 Identifier the size of the maximum input line length with varying last character.	(1..254=>'A', 255=>2)
\$BIG_ID3 Identifier the size of the maximum input line length with varying middle character.	(1..127=>'A', 128=>3, 129..255=>'A')
\$BIG_ID4 Identifier the size of the maximum input line length with varying middle character.	(1..127=>'A', 128=>4, 129..255=>'A')
\$BIG_INT_LIT An integer literal of value 298 with enough leading zeroes so that it is the size of the maximum line length	(1..252=>0, 253..255=>298)

## TEST PARAMETERS

---

\$BIG_REAL_LIT	(1..249=>0, 250..255=>69.0E1)
A universal real literal of value 690.0 with enough leading zeroes to be the size of the maximum line length.	
\$BIG_STRING1	(1..127=>'A')
A string literal which when catenated with BIG_STRING2 yields the image of BIG_ID1.	
\$BIG_STRING2	(1..127=>'A', 128=>1)
A string literal which when catenated to the end of BIG_STRING1 yields the image of BIG_ID1.	
\$BLANKS	(1..235=>' ')
A sequence of blanks twenty characters less than the size of the maximum line length.	
\$COUNT_LAST	2147483647
A universal integer literal whose value is TEXT_IO.COUNT'LAST.	
\$DEFAULT_MEM_SIZE	65536
An integer literal whose value is SYSTEM.MEMORY_SIZE.	
\$DEFAULT_STOR_UNIT	8
An integer literal whose value is SYSTEM.STORAGE_UNIT.	
\$DEFAULT_SYS_NAME	TRANSPUTER
The value of the constant SYSTEM.SYSTEM_NAME.	
\$DELTA_DOC	2#1.0#E-31
A real literal whose value is SYSTEM.FINE_DELTA.	
\$FIELD_LAST	255
A universal integer literal whose value is TEXT_IO.FIELD'LAST.	

---

\$FIXED_NAME	NO_SUCH_TYPE
The name of a predefined fixed-point type other than DURATION.	
\$FLOAT_NAME	NO_SUCH_TYPE
The name of a predefined floating-point type other than FLOAT, SHORT_FLOAT, or LONG_FLOAT.	
\$GREATER_THAN_DURATION	100000.0
A universal real literal that lies between DURATION'BASE'LAST and DURATION'LAST or any value in the range of DURATION.	
\$GREATER_THAN_DURATION_BASE_LAST	10000000.0
A universal real literal that is greater than DURATION'BASE'LAST.	
\$HIGH_PRIORITY	10
An integer literal whose value is the upper bound of the range for the subtype SYSTEM.PRIORITY.	
\$ILLEGAL_EXTERNAL_FILE_NAME1	?#~@[()] +=
An external file name which contains invalid characters.	
\$ILLEGAL_EXTERNAL_FILE_NAME2	[()] +=?#~@'
An external file name which is too long.	
\$INTEGER_FIRST	-32768
A universal integer literal whose value is INTEGER'FIRST.	
\$INTEGER_LAST	32767
A universal integer literal whose value is INTEGER'LAST.	
\$INTEGER_LAST_PLUS_1	32768
A universal integer literal whose value is INTEGER'LAST+1.	

TEST PARAMETERS

---

\$LESS_THAN_DURATION	-100000.0
A universal real literal that lies between DURATION'BASE'FIRST and DURATION'FIRST or any value in the range of DURATION.	
\$LESS_THAN_DURATION_BASE_FIRST	-10000000.0
A universal real literal that is less than DURATION'BASE'FIRST.	
\$LOW_PRIORITY	1
An integer literal whose value is the lower bound of the range for the subtype SYSTEM.PRIORITY.	
\$MANTISSA_DOC	31
An integer literal whose value is SYSTEM.MAX_MANTISSA.	
\$MAX_DIGITS	15
Maximum digits supported for floating-point types.	
\$MAX_IN_LEN	255
Maximum input line length permitted by the implementation.	
\$MAX_INT	2147483647
A universal integer literal whose value is SYSTEM.MAX_INT.	
\$MAX_INT_PLUS_1	2147483648
A universal integer literal whose value is SYSTEM.MAX_INT+1.	
\$MAX_LEN_INT_BASED_LITERAL	(1..2=>'2:', 3..252=>'0', 253..255=>'11:')
A universal integer based literal whose value is 2#11# with enough leading zeroes in the mantissa to be MAX_IN_LEN long.	
\$MAX_LEN_REAL_BASED_LITERAL	(1..3=>'16:', 4..251=>'0', 252..255=>'F.E:')
A universal real based literal whose value is 16:F.E: with enough leading zeroes in the mantissa to be MAX_IN_LEN long.	



---

\$MAX_STRING_LITERAL	(1=>"", 2..254=>'A', 255=>"")
A string literal of size MAX_IN_LEN, including the quote characters.	
\$MIN_INT	-2147483648
A universal integer literal whose value is SYSTEM.MIN_INT.	
\$MIN_TASK_SIZE	16
An integer literal whose value is the number of bits required to hold a task object which has no entries, no declarations, and "NULL;" as the only statement in its body.	
\$NAME	NO_SUCH_TYPE
A name of a predefined numeric type other than FLOAT, INTEGER, SHORT_FLOAT, SHORT_INTEGER, LONG_FLOAT, or LONG_INTEGER.	
\$NAME_LIST	TRANSPUTER
A list of enumeration literals in the type SYSTEM.NAME, separated by commas.	
\$NEG_BASED_INT	16#FFFFFFFF#
A based integer literal whose highest order nonzero bit falls in the sign bit position of the representation for SYSTEM.MAX_INT.	
\$NEW_MEM_SIZE	65536
An integer literal whose value is a permitted argument for pragma memory_size, other than \$DEFAULT_MEM_SIZE. If there is no other value, then use \$DEFAULT_MEM_SIZE.	

---

\$NEW_STOR_UNIT	8
An integer literal whose value is a permitted argument for pragma <code>storage_unit</code> , other than <code>\$DEFAULT_STOR_UNIT</code> . If there is no other permitted value, then use value of <code>SYSTEM.STORAGE_UNIT</code> .	
\$NEW_SYS_NAME	TRANSPUTER
A value of the type <code>SYSTEM.NAME</code> , other than <code>\$DEFAULT_SYS_NAME</code> . If there is only one value of that type, then use that value.	
\$TASK_SIZE	16
An integer literal whose value is the number of bits required to hold a task object which has a single entry with one inout parameter.	
\$TICK	1.0E-6
A real literal whose value is <code>SYSTEM.TICK</code> .	

## APPENDIX D

## WITHDRAWN TESTS

Some tests are withdrawn from the ACVC because they do not conform to the Ada Standard. The following <TOTAL\_WITHDRAWN> tests had been withdrawn at the time of validation testing for the reasons indicated. A reference of the form AI-ddddd is to an Ada Commentary.

- E28005C This test expects that the string "-- TOP OF PAGE. --63" of line 204 will appear at the top of the listing page due to a pragma PAGE in line 203; but line 203 contains text that follows the pragma, and it is this that must appear at the top of the page.
- A39005G This test unreasonably expects a component clause to pack an array component into a minimum size (line 30).
- B97102E This test contains an unintended illegality: a select statement contains a null statement at the place of a selective wait alternative (line 31).
- C97116A This test contains race conditions, and it assumes that guards are evaluated indivisibly. A conforming implementation may use interleaved execution in such a way that the evaluation of the guards at lines 50 & 54 and the execution of task CHANGING\_OF\_THE\_GUARD results in a call to REPORT.FAILED at one of lines 52 or 56.
- BC3009B This test wrongly expects that circular instantiations will be detected in several compilation units even though none of the units is illegal with respect to the units it depends on; by AI-00256, the illegality need not be detected until execution is attempted (line 95).
- CD2A62D This test wrongly requires that an array object's size be no greater than 10 although its subtype's size was specified to be 40 (line 137).
- CD2A63A..D, CD2A66A..D, CD2A73A..D, CD2A76A..D [16 tests]  
These tests wrongly attempt to check the size of objects of a derived type (for which a 'SIZE length clause is given) by passing them to a derived subprogram (which implicitly converts them to the parent type (Ada standard 3.4:14)). Additionally, they use the 'SIZE length clause and attribute, whose interpretation is considered problematic by the WG9 ARG.
- CD2A81G, CD2A83G, CD2A84N & M, & CD5011O [5 tests]  
These tests assume that dependent tasks will terminate while the main program executes a loop that simply tests for task termination; this is not the case, and the main program may loop indefinitely (lines 74, 85, 86 & 96, 86 & 96, and 58, resp.).

---

**CD2B15C & CD7205C**

These tests expect that a 'STORAGE\_SIZE length clause provides precise control over the number of designated objects in a collection; the Ada standard 13.2:15 allows that such control must not be expected.

**CD2D11B**

This test gives a SMALL representation clause for a derived fixed-point type (at line 30) that defines a set of model numbers that are not necessarily represented in the parent type; by Commentary AI-00099, all model numbers of a derived fixed-point type must be representable values of the parent type.

**CD5007B**

This test wrongly expects an implicitly declared subprogram to be at the the address that is specified for an unrelated subprogram (line 303).

**ED7004B, ED7005C & D, ED7006C & D [5 tests]**

These tests check various aspects of the use of the three SYSTEM pragmas; the AVO withdraws these tests as being inappropriate for validation.

**CD7105A**

This test requires that successive calls to CALENDAR.CLOCK change by at least SYSTEM.TICK; however, by Commentary AI-00201, it is only the expected frequency of change that must be at least SYSTEM.TICK--particular instances of change may be less (line 29).

**CD7203B, & CD7204B**

These tests use the 'SIZE length clause and attribute, whose interpretation is considered problematic by the WG9 ARG.

**CD7205D**

This test checks an invalid test objective: it treats the specification of storage to be reserved for a task's activation as though it were like the specification of storage for a collection.

**CE2107I**

This test requires that objects of two similar scalar types be distinguished when read from a file--DATA\_ERROR is expected to be raised by an attempt to read one object as of the other type. However, it is not clear exactly how the Ada standard 14.2.4:4 is to be interpreted; thus, this test objective is not considered valid. (line 90)

**CE3111C**

This test requires certain behavior, when two files are associated with the same external file, that is not required by the Ada standard.

**CE3301A**

This test contains several calls to END\_OF\_LINE & END\_OF\_PAGE that have no parameter: these calls were intended to specify a file, not to refer to STANDARD\_INPUT (lines 103, 107, 118, 132, & 136).

**CE3411B**

This test requires that a text file's column number be set to COUNT'LAST in order to check that LAYOUT\_ERROR is raised by a subsequent PUT operation. But the former operation will generally raise an exception due to a lack of available disk space, and the test would thus encumber validation testing.

## NCC VSR ADDENDUM

This Addendum to the ACVC 1.10 VSR clarifies some items which are contained within the standard pro-forma Validation Summary Report as supplied by the Ada Maintenance Organisation (AMO).

In line with AJPO regulations the contents of the VSR have not been altered in order to keep consistency between the different AVF's.

The points raised in this addendum are being addressed by the AMO in future issued of the VSR.

- 1 The last paragraph of Chapter 1 contains the following statement 'Any test that was determined to contain an illegal language constructed or an erroneous language construct is withdrawn from the ACVC...'

This is incorrect since illegal constructs are legitimately contained within Class B tests.

- 2 Both the terms 'inapplicable' and 'not applicable' are used within the VSR. These terms are identical.

- 3 Chapter 1 of the VSR does not indicate how 'inapplicable' tests are to be analysed. The analysis is undertaken as follows:

'Each inapplicable test is checked to ensure that this behaviour is consistent with the given reasons for its inapplicability'.