# AD-A247 336

## NTATION PAGE

Form Approved
OPM No. 0704-0188

1 hour per response, including the time for reviewing Instructions, searching existing data sources gathering and maintaining the data this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington ~~Headquarters Service, 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.

| 1. AGENCY USE ONLY (Leave Blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | | Final: 21 Jan 1992 to 01 Jun 1993 |

**4. TITLE AND SUBTITLE**

Validation Summary Report: IBM Canada Ltd., AIX Ada/6000 Internal Development Version, RISC System/6000. model 7012-320 (Host & Target), 920121W1.11234

**5. FUNDING NUMBERS**

**6. AUTHOR(S)**

Wright-Patterson AFB, Dayton, OH
USA

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Ada Validation Facility, Language Control Facility ASD/SCEL
Bldg. 676, Rm 135
Wright-Patterson AFB, Dayton, OH 45433

**8. PERFORMING ORGANIZATION REPORT NUMBER**

AVF-VSR-524.0292

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

Ada Joint Program Office
United States Department of Defense
Pentagon, Rm 3E114
Washington, D.C. 20301-3081

DTIC
ELECTE
MAR 1 0 1992
S    D
C

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**

Approved for public release; distribution unlimited.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT (Maximum 200 words)**

IBM Canada Ltd., AIX Ada/6000 Internal Development Version, Wright-Patterson AFB, RISC System/6000. model 7012-320 (Host & Target), ACVC 1.11.

**14. SUBJECT TERMS**

Ada programming language, Ada Compiler Val. Summary Report, Ada Compiler Val. Capability, Val. Testing, Ada Val. Office, Ada Val. Facility, ANSI/MIL-STD-1815A, AJPO.

**15. NUMBER OF PAGES**

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| UNCLASSIFIED | UNCLASSIFED | UNCLASSIFIED | |

NSN 7540-01-280-550

Standard Form 298, (Rev. 2-89)
Prescribed by ANSI Std. 239-128

Ada COMPILER
VALIDATION SUMMARY REPORT:
Certificate Number: 920121W1.11234
IBM Canada Ltd.
AIX Ada/6000 Internal Development Version
RISC System/6000, model 7012-320 => RISC System/6000, model 7012-320

Prepared By:
Ada Validation Facility
ASD/SCEL
Wright-Patterson AFB OH   45433-6503

92-05976

The following Ada implementation was tested and determined to pass ACVC 1.11.  Testing was completed on 21 January 1992.

Compiler Name and Version: AIX Ada/6000 Internal Development Version

Host Computer System:     RISC System/6000, model 7012-320, AIX 3.2
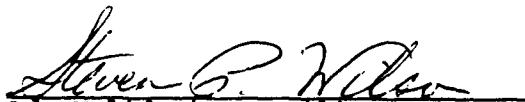
Target Computer System:   RISC System/6000, model 7012-320, AIX 3.2

Customer Agreement Number: 91-11-12-IBM

See section 3.1 for any additional information about the testing environment.

As a result of this validation effort, Validation Certificate 920121W1.11234 is awarded to IBM Canada Ltd.  This certificate expires on 1 June 1993.

This report has been reviewed and is approved.

Ada Validation Facility
Steven P.  Wilson
Technical Director
ASD/SCEL
Wright-Patterson AFB OH   45433-6503

Ada Validation Organization
Director\Computer and Software Engineering Division
Institute for Defense Analyses
Alexandria VA   22311

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

92 3 05 011

## DECLARATION OF CONFORMANCE

Customer:                          IBM Canada Ltd.

Ada Validation Facility:           Wright Patterson AVF
                                   ASD/SCEL
                                   WPAFB OH 45433
                                   USA

ACVC Version:                      1 11

Ada Implementation:
    Compiler Name and Version:     AIX Ada/6000 Internal Development Version
    Host Computer System:          RISC System/6000, model 7012-320, AIX 3.2
    Target Computer System:        RISC System/6000, model 7012-320, AIX 3.2

### Customer's Declaration

I, the undersigned, representing IBM Canada Ltd., declare that IBM Canada Ltd.
has no knowledge of deliberate deviations from the Ada Language Standard
ANSI/MIL-STD-1815A in the implementation listed in this declaration.


Date: Jan 20 /92

Peter Moogk
IBM Canada Ltd. Laboratory
844 Don Mills Road
North York, Ontario
CANADA                    M3C 1V7


Co-Signer:

Date: Jan 20 /992

Paul Kohlbrenner
OC Systems, Inc.
9926 Main St. Suite 200
Fairfax, VA  22031
USA

# TABLE OF CONTENTS

# CHAPTER 1

## INTRODUCTION

The Ada implementation described above was tested according to the Ada
Validation Procedures [Pro90] against the Ada Standard [Ada83] using the
current Ada Compiler Validation Capability (ACVC). This Validation Summary
Report (VSR) gives an account of the testing of this Ada implementation.
For any technical terms used in this report, the reader is referred to
[Pro90]. A detailed description of the ACVC may be found in the current
ACVC User's Guide [UG89].

## 1.1 USE OF THIS VALIDATION SUMMARY REPORT

Consistent with the national laws of the originating country, the Ada
Certification Body 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 implementation has no
nonconformities to the Ada Standard other than those presented. Copies of
this report are available to the public from the AVF which performed this
validation or from:

> National Technical Information Service
> 5285 Port Royal Road
> Springfield VA   22161

Questions  regarding  this  report or the validation test results should be
directed to the AVF which performed this validation or to:

> Ada Validation Organization
> Computer and Software Engineering Division
> Institute for Defense Analyses
> 1801 North Beauregard Street
> Alexandria VA   22311-1772

## 1.2 REFERENCES

[Ada83] Reference Manual for the Ada Programming Language, ANSI/MIL-STD-1815A, February 1983 and ISO 8652-1987.

[Pro90] Ada Compiler Validation Procedures, Version 2.1, Ada Joint Program Office, August 1990.

[UG89] Ada Compiler Validation Capability User's Guide, 21 June 1989.

## 1.3 ACVC TEST CLASSES

Compliance of Ada implementations is tested by means of the ACVC. The ACVC contains a collection of test programs structured into six test classes: A, B, C, D, E, and L. The first letter of a test name identifies the class to which it belongs. Class A, C, D, and E tests are executable. Class B and class L tests are expected to produce errors at compile time and link time, respectively.

The executable tests are written in a self-checking manner and produce a PASSED, FAILED, or NOT APPLICABLE message indicating the result when they are executed. Three Ada library units, the packages REPORT and SPPRT13, and the procedure CHECK_FILE are used for this purpose. The package REPORT also provides a set of Identity functions used to defeat some compiler optimizations allowed by the Ada Standard that would circumvent a test objective. The package SPPRT13 is used by many tests for Chapter 13 of the Ada Standard. 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. If these units are not operating correctly, validation testing is discontinued.

Class B tests check that a compiler detects illegal language usage. Class B tests are not executable. Each test in this class is compiled and the resulting compilation listing is examined to verify that all violations of the Ada Standard are detected. Some of the class B tests contain legal Ada code which must not be flagged illegal by the compiler. This behavior is also verified.

Class L tests check that an Ada implementation correctly detects violation of the Ada Standard involving multiple, separately compiled units. Errors are expected at link time, and execution is attempted.

In some tests of the ACVC, certain macro strings have to be replaced by implementation-specific values — for example, the largest integer. A list of the values used for this implementation is provided in Appendix A. In addition to these anticipated test modifications, additional changes may be required to remove unforeseen conflicts between the tests and implementation-dependent characteristics. The modifications required for this implementation are described in section 2.3.

For each Ada implementation, a customized test suite is produced by the AVF. This customization consists of making the modifications described in the preceding paragraph, removing withdrawn tests (see section 2.1), and possibly removing some inapplicable tests (see section 2.2 and [UG89]).

In order to pass an ACVC an Ada implementation must process each test of the customized test suite according to the Ada Standard.

## 1.4 DEFINITION OF TERMS

| | |
|---|---|
| Ada Compiler | The software and any needed hardware that have to be added to a given host and target computer system to allow transformation of Ada programs into executable form and execution thereof. |
| Ada Compiler Validation Capability (ACVC) | The means for testing compliance of Ada implementations, consisting of the test suite, the support programs, the ACVC user's guide and the template for the validation summary report. |
| Ada Implementation | An Ada compiler with its host computer system and its target computer system. |
| Ada Joint Program Office (AJPO) | The part of the certification body which provides policy and guidance for the Ada certification system. |
| Ada Validation Facility (AVF) | The part of the certification body which carries out the procedures required to establish the compliance of an Ada implementation. |
| Ada Validation Organization (AVO) | The part of the certification body that provides technical guidance for operations of the Ada certification system. |
| Compliance of an Ada Implementation | The ability of the implementation to pass an ACVC version. |
| Computer System | A functional unit, consisting of one or more computers and associated software, that uses common storage for all or part of a program and also for all or part of the data necessary for the execution of the program; executes user-written or user-designated programs; performs user-designated data manipulation, including arithmetic operations and logic operations; and that can execute programs that modify themselves during execution. A computer system may be a stand-alone unit or may consist of several inter-connected units. |

1-3

INTRODUCTION

| | |
|---|---|
| Conformity | Fulfillment by a product, process, or service of all requirements specified. |
| Customer | An individual or corporate entity who enters into an agreement with an AVF which specifies the terms and conditions for AVF services (of any kind) to be performed. |
| Declaration of Conformance | A formal statement from a customer assuring that conformity is realized or attainable on the Ada implementation for which validation status is realized. |
| Host Computer System | A computer system where Ada source programs are transformed into executable form. |
| Inapplicable test | A test that contains one or more test objectives found to be irrelevant for the given Ada implementation. |
| ISO | International Organization for Standardization. |
| LRM | The Ada standard, or Language Reference Manual, published as ANSI/MIL-STD-1815A-1983 and ISO 8652-1987. Citations from the LRM take the form "<section>.<subsection>:<paragraph>." |
| Operating System | Software that controls the execution of programs and that provides services such as resource allocation, scheduling, input/output control, and data management. Usually, operating systems are predominantly software, but partial or complete hardware implementations are possible. |
| Target Computer System | A computer system where the executable form of Ada programs are executed. |
| Validated Ada Compiler | The compiler of a validated Ada implementation. |
| Validated Ada Implementation | An Ada implementation that has been validated successfully either by AVF testing or by registration [Pro90]. |
| Validation | The process of checking the conformity of an Ada compiler to the Ada programming language and of issuing a certificate for this implementation. |
| Withdrawn test | A test found to be incorrect and not used in conformity testing. A test may be incorrect because it has an invalid test objective, fails to meet its test objective, or contains erroneous or illegal use of the Ada programming language. |

# CHAPTER 2

## IMPLEMENTATION DEPENDENCIES

## 2.1 WITHDRAWN TESTS

The following tests have been withdrawn by the AVO. The rationale for
withdrawing each test is available from either the AVO or the AVF. The
publication date for this list of withdrawn tests is 2 August 1991.

| | | | | | |
|---|---|---|---|---|---|
| E28005C | B28006C | C32203A | C34006D | C35508I | C35508J |
| C35508M | C35508N | C35702A | C35702B | B41308B | C43004A |
| C45114A | C45346A | C45612A | C45612B | C45612C | C45651A |
| C46022A | B49008A | B49008B | A74006A | C74308A | B83022B |
| B83022H | B83025B | B83025D | C83026A | B83026B | C83041A |
| B85001L | C86001F | C94021A | C97116A | C98003B | BA2011A |
| CB7001A | CB7001B | CB7004A | CC1223A | BC1226A | CC1226B |
| BC3009B | BD1B02B | BD1B06A | AD1B08A | BD2A02A | CD2A21E |
| CD2A23E | CD2A32A | CD2A41A | CD2A41E | CD2A87A | CD2B15C |
| BD3006A | BD4008A | CD4022A | CD4022D | CD4024B | CD4024C |
| CD4024D | CD4031A | CD4051D | CD5111A | CD7004C | ED7005D |
| CD7005E | AD7006A | CD7006E | AD7201A | AD7201E | CD7204B |
| AD7206A | BD8002A | BD8004C | CD9005A | CD9005B | CDA201E |
| CE2107I | CE2117A | CE2117B | CE2119B | CE2205B | CE2405A |
| CE3111C | CE3116A | CE3118A | CE3411B | CE3412B | CE3607B |
| CE3607C | CE3607D | CE3812A | CE3814.. | CE3902B | |

## 2.2 INAPPLICABLE TESTS

A test is inapplicable if it contains test objectives which are irrelevant
for a given Ada implementation. Reasons for a test's inapplicability may
be supported by documents issued by the ISO and the AJPO known as Ada
Commentaries and commonly referenced in the format AI-ddddd. For this
implementation, the following tests were determined to be inapplicable for
the reasons indicated; references to Ada Commentaries are included as
appropriate.

The following 201 tests 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) |

The following 20 tests check for the predefined type LONG_INTEGER; for this implementation, there is no such type:

| | | | | |
|---|---|---|---|---|
| C35404C | C45231C | C45304C | C45411C | C45412C |
| C45502C | C45503C | C45504C | C45504F | C45611C |
| C45613C | C45614C | C45631C | C45632C | B52004D |
| C55B07A | B55B09C | B86001W | C86006C | CD7101F |

C35713B, C45423B, B86001T, and C86006H check for the predefined type SHORT_FLOAT; for this implementation, there is no such type.

C35713D and B86001Z check for a predefined floating-point type with a name other than FLOAT, LONG_FLOAT, or SHORT_FLOAT; for this implementation, there is no such type.

C45531M..P and C45532M..P (8 tests) check fixed-point operations for types that require a SYSTEM.MAX_MANTISSA of 47 or greater; for this implementation, MAX_MANTISSA is less than 47.

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

B86001Y uses the name of a predefined fixed-point type other than type DURATION; for this implementation, there is no such type.

CA2009C and CA2009F check whether a generic unit can be instantiated before its body (and any of its subunits) is compiled; this implementation creates a dependence on generic units as allowed by AI-00408 and AI-00506 such that the compilation of the generic unit bodies makes the instantiating units obsolete. (See section 2.3.)

LA3004A..B, EA3004C..D, and CA3004E..F (6 tests) check pragma INLINE for procedures and functions; this implementation does not support pragma INLINE.

CD2A84A, CD2A84E, CD2A84I..J (2 tests), and CD2A84O use length clauses to specify non-default sizes for access types; this implementation does not support such sizes.

BD8001A, BD8003A, BD8004A..B (2 tests), and AD8011A use machine code insertions; this implementation provides no package MACHINE_CODE.

The tests listed in the following table check that USE_ERROR is raised if the given file operations are not supported for the given combination of mode and access method; this implementation supports these operations.

| Test | File Operation | Mode | File Access Method |
|------|----------------|------|--------------------|
| CE2102D | CREATE | IN_FILE | SEQUENTIAL_IO |
| CE2102E | CREATE | OUT_FILE | SEQUENTIAL_IO |
| CE2102F | CREATE | INOUT_FILE | DIRECT_IO |
| CE2102I | CREATE | IN_FILE | DIRECT_IO |
| CE2102J | CREATE | OUT_FILE | DIRECT_IO |
| CE2102N | OPEN | IN_FILE | SEQUENTIAL_IO |
| CE2102O | RESET | IN_FILE | SEQUENTIAL_IO |
| CE2102P | OPEN | OUT_FILE | SEQUENTIAL_IO |
| CE2102Q | RESET | OUT_FILE | SEQUENTIAL_IO |
| CE2102R | OPEN | INOUT_FILE | DIRECT_IO |
| CE2102S | RESET | INOUT_FILE | DIRECT_IO |
| CE2102T | OPEN | IN_FILE | DIRECT_IO |
| CE2102U | RESET | IN_FILE | DIRECT_IO |
| CE2102V | OPEN | OUT_FILE | DIRECT_IO |
| CE2102W | RESET | OUT_FILE | DIRECT_IO |
| CE3102E | CREATE | IN_FILE | TEXT_IO |
| CE3102F | RESET | Any Mode | TEXT_IO |
| CE3102G | DELETE | ———— | TEXT_IO |
| CE3102I | CREATE | OUT_FILE | TEXT_IO |
| CE3102J | OPEN | IN_FILE | TEXT_IO |
| CE3102K | OPEN | OUT_FILE | TEXT_IO |

The following 16 tests check operations on sequential, direct, and text files when multiple internal files are associated with the same external file and one or more are open for writing; USE_ERROR is raised when this association is attempted.

| | | | | |
|---|---|---|---|---|
| CE2107B..E | CE2107G..H | CE2107L | CD2110B | CE2110D |
| CE2111D | CE2111H | CE3111B | CE3111D..E | CE3114B |
| CE3115A | | | | |

CE2203A checks that WRITE raises USE_ERROR if the capacity of an external sequential file is exceeded; this implementation cannot restrict file capacity.

2-3

EE2401D uses an instantiation of DIRECT_IO with an unconstrained array type whose index type is INTEGER; this implementation raises USE_ERROR on the attempt to create a file of such a type, as the array type's size exceeds the implementation's maximum element size.

CE2403A checks that WRITE raises USE_ERROR if the capacity of an external direct file is exceeded; this implementation cannot restrict file capacity.

CE3304A checks that SET_LINE_LENGTH and SET_PAGE_LENGTH raise USE_ERROR if they specify an inappropriate value for the external file; there are no inappropriate values for this implementation.

CE3413B checks that PAGE raises LAYOUT_ERROR when the value of the page number exceeds COUNT'LAST; for this implementation, the value of COUNT'LAST is greater than 150000, making the checking of this objective impractical.


## 2.3  TEST MODIFICATIONS

Modifications (see section 1.3) were required for 26 tests.

The following tests were split into two or more tests because this implementation did not report the violations of the Ada Standard in the way expected by the original tests.

| | | | | | |
|---|---|---|---|---|---|
| B28003A | BA1001A | BA2001C | BA2001E | BA3006A | BA3006B |
| BA3007B | BA3008A | BA3008B | BA3013A | | |

C52008B was graded passed by Test Modification as directed by the AVO. This test uses a record type with discriminants with defaults; this test also has array components whose length depends on the values of some discriminants of type INTEGER.  On elaboration of the type declaration, this implementation raises NUMERIC_ERROR as it attempts to calculate the maximum possible size for objects of the type.  The AVO ruled that this behavior was acceptable, and that the test should be modified to constrain the subtype of the discriminants.  Line 16 was modified to create a constrained subtype of INTEGER, and discriminant specifications in lines 17 and 25 were modified to use that subtype; these lines are given below:

```
16   SUBTYPE SUBINT IS INTEGER RANGE -128 .. 127;
17   TYPE REC1(D1,D2 : SUBINT) IS

25   TYPE REC2(D1,D2,D3,D4 : SUBINT := 0) IS
```

CA2009C and CA2009F were graded inapplicable by Evaluation Modification as directed by the AVO.  These tests contain instantiations of a generic unit prior to the compilation of that unit's body; as allowed by AI-00408 and AI-00506, the compilation of the generic unit bodies makes the compilation unit that contains the instantiations obsolete.

CD1009A, CD1009I, CD1C03A, CD2A21A..C (3 tests), CD2A22J, CD2A23A..B (2 tests), CD2A24A, CD2A31A, CD2A31B and CD2A31C were graded passed by Evaluation Modification as directed by the AVO.  These tests use instantiations of the support procedure LENGTH_CHECK, which uses Unchecked_Conversion according to the interpretation given in AI-00590. The AVO ruled that this interpretation is not binding under ACVC 1.11; the tests are ruled to be passed if they produce Failed messages only from the instances of LENGTH_CHECK—i.e, the allowed Report.Failed messages have the general form:

    " * CHECK ON REPRESENTATION FOR <TYPE_ID> FAILED."

# CHAPTER 3

## PROCESSING INFORMATION

### 3.1 TESTING ENVIRONMENT

The Ada implementation tested in this validation effort is described adequately by the information given in the initial pages of this report.

For technical information about this Ada implementation, contact:

> Paul Kohlebrenner
> OC Systems, Inc.
> 9926 Main Street, Suite 200
> Fairfax, VA  22031

For sales information about this Ada implementation, contact:

> Peter Moogk
> IBM Canada Ltd.  Laboratory
> 844 Don Mills Road
> North York, Ontario
> Canada  M3C 1V7

Testing  of this Ada implementation was conducted at the customer's site by a validation team from the AVF.

## 3.2 SUMMARY OF TEST RESULTS

An Ada Implementation passes a given ACVC version if it processes each test
of the customized test suite in accordance with the Ada Programming
Language Standard, whether the test is applicable or inapplicable;
otherwise, the Ada Implementation fails the ACVC [Pro90].

For all processed tests (inapplicable and applicable), a result was
obtained that conforms to the Ada Programming Language Standard.

The list of items below gives the number of ACVC tests in various
categories. All tests were processed, except those that were withdrawn
because of test errors (item b; see section 2.1), those that require a
floating-point precision that exceeds the implementation's maximum
precision (item e; see section 2.2), and those that depend on the support
of a file system — if none is supported (item d). All tests passed,
except those that are listed in sections 2.1 and 2.2 (counted in items b
and f, below).

|   |   |   |   |
|---|---|---|---|
| a) | Total Number of Applicable Tests | 3777 | |
| b) | Total Number of Withdrawn Tests | 95 | |
| c) | Processed Inapplicable Tests | 97 | |
| d) | Non-Processed I/O Tests | 0 | |
| e) | Non-Processed Floating-Point Precision Tests | 201 | |
| f) | Total Number of Inapplicable Tests | 298 | (c+d+e) |
| g) | Total Number of Tests for ACVC 1.11 | 4170 | (a+b+f) |

## 3.3 TEST EXECUTION

A magnetic tape containing the customized test suite (see section 1.3) was
taken on-site by the validation team for processing. The contents of the
magnetic tape were {loaded directly onto the host computer.

After the test files were loaded onto the host computer, the full set of
tests was processed by the Ada implementation.

The tests were compiled and linked on the host computer system, as
appropriate. The results were captured on the computer system.

Testing was performed using command scripts provided by the customer and reviewed by the validation team. See Appendix B for a complete listing of the processing options for this implementation. It also indicates the default options. The options invoked explicitly for validation testing during this test were:

Compiler Options:

| | |
|---|---|
| -l | Produces a compiler listing |
| -b BIND_NAME | Binds the object into an executable with BIND_NAME as the main unit. |
| -m | Binds the object into an executable with the first compilation unit in the file as the main unit. |
| -v | Produces compiler banner. |

Test output, compiler and linker 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.

# APPENDIX A

## MACRO PARAMETERS

This appendix contains the macro parameters used for customizing the ACVC.
The meaning and purpose of these parameters are explained in [UG89]. The
parameter values are presented in two tables. The first table lists the
values that are defined in terms of the maximum input-line length, which is
the value for $MAX_IN_LEN—also listed here. These values are expressed
here as Ada string aggregates, where "V" represents the maximum input-line
length.

| Macro Parameter | Macro Value |
|---|---|
| $MAX_IN_LEN | 200 — Value of V |
| $BIG_ID1 | (1..V-1 => 'A', V => '1') |
| $BIG_ID2 | (1..V-1 => 'A', V => '2') |
| $BIG_ID3 | (1..V/2 => 'A') & '3' & (1..V-1-V/2 => 'A') |
| $BIG_ID4 | (1..V/2 => 'A') & '4' & (1..V-1-V/2 => 'A') |
| $BIG_INT_LIT | (1..V-3 => '0') & "298" |
| $BIG_REAL_LIT | (1..V-5 => '0') & "690.0" |
| $BIG_STRING1 | '"' & (1..V/2 => 'A') & '"' |
| $BIG_STRING2 | '"' & (1..V-1-V/2 => 'A') & '1' & '"' |
| $BLANKS | (1..V-20 => ' ') |
| $MAX_LEN_INT_BASED_LITERAL | "2:" & (1..V-5 => '0') & "11:" |
| $MAX_LEN_REAL_BASED_LITERAL | "16:" & (1..V-7 => '0') & "F.E:" |

A-1

The following table lists all of the other macro parameters and their respective values.

| Macro Parameter | Macro Value |
|---|---|
| $MAX_STRING_LITERAL | '"' & (1..V-2 => 'A') & '"' |
| $ACC_SIZE | 32 |
| $ALIGNMENT | 4 |
| $COUNT_LAST | 2_147_483_646 |
| $DEFAULT_MEM_SIZE | 268_435_456 |
| $DEFAULT_STOR_UNIT | 8 |
| $DEFAULT_SYS_NAME | AIX_6000 |
| $DELTA_DOC | 2#1.0#E-31 |
| $ENTRY_ADDRESS | ENTRY0'ADDRESS |
| $ENTRY_ADDRESS1 | ENTRY1'ADDRESS |
| $ENTRY_ADDRESS2 | ENTRY2'ADDRESS |
| $FIELD_LAST | 1000 |
| $FILE_TERMINATOR | ' ' |
| $FIXED_NAME | NO_SUCH_FIXED_TYPE |
| $FLOAT_NAME | NO_SUCH_TYPE |
| $FORM_STRING | "" |
| $FORM_STRING2 | "CANNOT_RESTRICT_FILE_CAPACITY" |
| $GREATER_THAN_DURATION | 100_000.0 |
| $GREATER_THAN_DURATION_BASE_LAST | 131_073.0 |
| $GREATER_THAN_FLOAT_BASE_LAST | 1.0E308 |
| $GREATER_THAN_FLOAT_SAFE_LARGE | 3.0E38 |
| $GREATER_THAN_SHORT_FLOAT_SAFE_LARGE | |

```
                           1.0E308

$HIGH_PRIORITY             255

$ILLEGAL_EXTERNAL_FILE_NAME1
                           A*

$ILLEGAL_EXTERNAL_FILE_NAME2
                           *B

$INAPPROPRIATE_LINE_LENGTH
                           -1

$INAPPROPRIATE_PAGE_LENGTH
                           -1

$INCLUDE_PRAGMA1           PRAGMA INCLUDE ("A28006D1.TST")

$INCLUDE_PRAGMA2           PRAGMA INCLUDE ("B28006D1.TST")

$INTEGER_FIRST             -2147483648

$INTEGER_LAST              2147483647

$INTEGER_LAST_PLUS_1       2147483648

$INTERFACE_LANGUAGE        FORTRAN

$LESS_THAN_DURATION        -100_000.0

$LESS_THAN_DURATION_BASE_FIRST
                           -131_073.0

$LINE_TERMINATOR           ASCII.LF

$LOW_PRIORITY              0

$MACHINE_CODE_STATEMENT
                           NULL;

$MACHINE_CODE_TYPE         NO_SUCH_TYPE

$MANTISSA_DOC              31

$MAX_DIGITS                15

$MAX_INT                   2147483647

$MAX_INT_PLUS_1            2_147_483_648

$MIN_INT                   -2147483648

$NAME                      SHORT_SHORT_INTEGER
```

A-3

MACRO PARAMETERS

| | |
|---|---|
| $NAME_LIST | AIX_6000 |
| $NAME_SPECIFICATION1 | /dino/val/work/ctests/X2120A |
| $NAME_SPECIFICATION2 | /dino/val/work/ctests/X2120B |
| $NAME_SPECIFICATION3 | /dino/val/work/ctests/X3119A |
| $NEG_BASED_INT | 16#F000000E# |
| $NEW_MEM_SIZE | 65535 |
| $NEW_STOR_UNIT | 16 |
| $NEW_SYS_NAME | AIX_6000 |
| $PAGE_TERMINATOR | ASCII.FF |
| $RECORD_DEFINITION | NEW INTEGER; |
| $RECORD_NAME | NO_SUCH_MACHINE_CODE_TYPE |
| $TASK_SIZE | 32 |
| $TASK_STORAGE_SIZE | 32768 |
| $TICK | 0.00006 |
| $VARIABLE_ADDRESS | VARIABLE'ADDRESS |
| $VARIABLE_ADDRESS1 | VARIABLE1'ADDRESS |
| $VARIABLE_ADDRESS2 | VARIABLE2'ADDRESS |
| $YOUR_PRAGMA | EXPORT |

# APPENDIX B

## COMPILATION SYSTEM OPTIONS

The compiler options of this Ada implementation, as described in this Appendix, are provided by the customer. Unless specifically noted otherwise, references in this appendix are to compiler documentation and not to this report.

USAGE: ada [Option | FileName]

| OPTION/SWITCH | EFFECT |
|---|---|
| -a | Produce assembler code. |
| -b UnitName | Bind and link unit. |
| -d | Keep debug info in the Ada sublibrary. |
| -e | Suppress creation of executable file. |
| -G | Do partial optomization. |
| -h | Display info about the ada command. |
| -i FileName | Include archive or object file FileName when binding. |
| -I | Read a list of files to compile from standard input. |
| -l | Generate a compilation listing. |
| -L Library | Use this library list file. |
| -m | Compile and bind the last unit as the main program. |
| -o FileName | Specify the name of the executable is to be FileName. |

COMPILATION SYSTEM OPTIONS

| OPTION/SWITCH | EFFECT |
|---|---|
| -O | Produce fully optomized code. |
| -p | Turn on execution profiling option. |
| -Q | Consistently detect single precision floating point overflow. |
| -q Parameter=Value | Special parameters as follows:. task_stack=Bytes time_slice=Milliseconds. |
| -S | Suppress run-time checks for Ada exceptions. |
| -u | Unlock the working sublibrary. |
| -v | Verbose - display informative messages. |
| -V Pages | Specify the number of 1K memory pages to be used. |

## LINKER OPTIONS

The linker options of this Ada implementation, as described in this
Appendix, are provided by the customer.  Unless specifically noted
otherwise, references in this appendix are to linker documentation and not
to this report.



There is no separate linker.

# APPENDIX C

## 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 this Ada implementation, as described in this Appendix,
are provided by the customer.  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 -2147483648 .. 2147483647;
  type SHORT_INTEGER is range -32768 .. 32767;
  type SHORT_SHORT_INTEGER is range -128 .. 127;

  type FLOAT is digits 6 range
      -3.40282E+38 .. 3.40282E+38;
  type LONG_FLOAT is digits 15 range
      -1.79769313486232E+308 .. 1.79769313486232E+308;

  type DURATION is delta 2**(-14) range
      -86400.0 .. 86400.0;
    ..........
end STANDARD;
```

# Appendix 5 - Appendix F of the Ada Language Reference Manual

Appendix F. Implementation-Dependent Characteristics

The Ada language definition allows for certain target dependencies in a controlled manner. This appendix, called Appendix F as prescribed in the Ada Language Reference, describes implementation-dependent characteristics of the AIX Ada/6000 compiler running under the AIX operating system Version 3.

Implementation-Defined Pragmas

Implementation dependent pragmas are:

pragma COMMENT(String_Literal);

Imbeds String_Literal into object code.

pragma IMAGES(Enumeration_Type, immediate | deferred);

Generates a table of images for the enumeration type. deferred causes the table to be generated only if the enumeration type is used in a compilation unit.

pragma INTERFACE_INFORMATION
   (Interfaced_Subprogram_Name, Link_Name);

When used in conjunction with pragma INTERFACE, provides access to any routine whose name can be specified by an Ada string literal.

pragma LINKNAME (Interfaced_Subprogram_Name, Link_Name);

When used in conjunction with pragma INTERFACE, provides access to any routine whose name can be specified by an Ada string literal. Pragma INTERFACE_INFORMATION is the preferred method for naming non-Ada routines.

pragma OS_TASK (Priority);

Appears within the declaration for a task or task type (in the same context as pragma PRIORITY), and causes the task or task type to be placed into a separate AIX process. The priority value is of type SYSTEM.PRIORITY, and is not currently acted upon. To maintain upward compatibility, always use a 0 for this parameter.

pragma NO_SUPPRESS(Condition_Name);

Prevents checks for a specified condition from being suppressed. It has the same scope as pragma SUPPRESS.

pragma PRESERVE_LAYOUT (Record_Type);

Prevents the compiler from re-ordering record components. It must occur in a declarative region, following the declaration of the record type to which it is applied.

```
pragma SUPPRESS_ALL;
```

Suppresses all error checks, including elaboration checks. It has the same scope as pragma SUPPRESS.

```
pragma EXPORT(object, String_Literal, Language);
```

Makes the named object visible to other programs. The object can be a procedure or a function or a data object. The string_literal is the name the linker will make global. The Language must be C, FORTRAN, or Assembler, all of which have the same effect.

### Predefined Pragmas

Supported pragmas are INTERFACE, ELABORATE, SUPPRESS, PACK, PAGE, LIST, INLINE, and PRIORITY.

All pragmas have conventional meanings except LIST, which suppresses listings prior to pragma LIST(ON) regardless of the user request.

Pragma INTERFACE supports the interface languages C, FORTRAN, and ASSEMBLY.

Unrecognized and unsupported pragmas are ignored with the appropriate warning message.

### Representation Clauses

Supported representation clauses include:

w       Length clauses

w       Enumeration representation clauses, except for Boolean types

w       Record representation clauses

w       Address clauses, including those for interrupt entries.

Records are aligned by default on 32-bit boundaries. You can use a representation clause to force them to be aligned on 64-bit boundaries.

### Restrictions on Unchecked Conversion

The only restriction on unchecked conversion is that the two types (or subtypes) A and B must be the same static size, and that neither A nor B are private.

### Package System

The package SYSTEM has the following characteristics:

```
PACKAGE system IS
-- for integer use 32;
TYPE Memory is private;
TYPE Address is access Memory;
TYPE name IS (AIX_6000);
```

```
System_Name : CONSTANT name := AIX_6000;
Storage_Unit : CONSTANT := 8;
Memory_Size : CONSTANT := 1024*1024*256;
-- 256 Mb.
-- System-Dependent Named Numbers:
Min_Int
   : CONSTANT := -(2**31);
Max_Int   : CONSTANT := (2 ** 31) -1;
Max_Digits : CONSTANT := 15;
Max_Mantissa : CONSTANT := 31;
Fine_Delta : CONSTANT := 1.0 / (2 ** MAX_MANTISSA);
Tick      : CONSTANT := 0.00006;
-- Other System-Dependent Declarations
SUBTYPE Priority IS integer RANGE 0..255;
END System;
```

Representation Attributes

All defined representation attributes shall be supported.

Implementation-Generated Names

There are no implementation-generated names denoting implementation-dependent components. Component names generated by the compiler shall not interfere with programmer-defined names.

Implementation-Dependent Characteristics of the I/O Packages

w       Packages SEQUENTIAL_IO, DIRECT_IO, and TEXT_IO are supported.

w       Package LOW_LEVEL_IO is not supported.

w       Unconstrained array types and unconstrained types with discriminants may be instantiated for I/O.

w       File names follow the conventions and restrictions of the target operating system, except that non-printing characters, blank(' ') and asterisk ('*') are disallowed.

w       In TEXT_IO, the type FIELD is defined as follows:

subtype Field is integer range 0..1000;

w       In TEXT_IO, the type COUNT is defined as follows:

type Count is range 0..2_147_483_646;

Form Parameters for File Operations

Section 14.2 of the Language Reference describes the Ada functions for manipulating files. As stated in that section, the form string parameter allows you to set file protections when you create a file. The details of file protections and privileges under the AIX operating system are described under the chmod call in the AIX Calls and Subroutines Reference for IBM RISC System/6000 (SC23-2198).

If you do not specify a form string, the default file protection is both read and write privileges for the owner, group, and all others. If you do specify a form string, it is interpreted in the following way:

w       The form string consists of a series of substrings, separated by blanks.

w       The substrings are not case sensitive.

w       The order of the substrings does not matter.

w       Some substrings control the file protection settings.

w       Some substrings enable special AIX behavior for file opening, for example opening with no delay and opening a file for append.

w       The list of recognized substrings is contained in the section entitled #Input/Output" in the User's Guide for IBM AIX Ada/6000 (SC09-1321).

Predefined Numeric Types

The current specification of package STANDARD includes:

type SHORT_SHORT_INTEGER
  is range -128 .. 127;

type SHORT_INTEGER
  is range -32768 .. 32767;

type INTEGER is
  range -2147483648 .. 2147483647;

type FLOAT is
  digits 6 range -3.40282E+38 .. 3.40282E+38;

type LONG_FLOAT is
  digits 15 range
    -1.79769313486232E+308 .. 1.79769313486232E+308;

type DURATION is
  delta 2**(-14) range -86400.0 .. 86400.0;

SHORT_SHORT_INTEGER

'First  = -128

'Last   = 127

'Size   = 8

SHORT_INTEGER

'First  = -32768

'Last   = 32767

```
'Size   = 16
```

INTEGER

```
'First  = -2147483648

'Last   = 2147483647

'Size   = 32
```

FLOAT

```
'Digits = 6

'Emax   = 84

'Epsilon         = 9.53674E-07

'Large  = 1.93428E+25

'Machine_Emax    = 128

'Machine_Emin    = -125

'Machine_Mantissa        = 24

'Machine_Overflows       = TRUE

'Machine_Radix   = 2

'Machine_Rounds  = TRUE

'Mantissa        = 21

'Safe_Emax       = 125

'Safe_Large      = 4.25353E+37

'Safe_Small      = 1.17549E-38

'Size   = 32

'Small  = 2.58494E-26
```

LONG_FLOAT

```
'Digits = 15

'Emax   = 204

'Epsilon         = 8.88178419700125E-16

'Large  = 2.57110087081438E+61

'Machine_Emax    = 1024
```

```
'Machine_Emin    = -1021

'Machine_Mantissa       = 53

'Machine_Overflows      = TRUE

'Machine_Radix  = 2

'Machine_Rounds = TRUE

'Mantissa        = 51

'Size   = 64

'Safe_Emax       = 1021

'Safe_Large      = 2.24711641857789E+307

'Safe_Small      = 2.22507385850720E-308

'Small  = 1.94469227433161E-62
```

DURATION

```
'Delta   = 2**(-14)

'First  = -86400.0

'Last    = 86400.0

'Machine_Overflows      = FALSE

'Machine_Rounds = FALSE
```

## Restrictions on Machine Code Insertions

Machine code insertions are not supported.

## Parameters to a Main Subprogram

You can code an Ada/6000 main subprogram to accept parameters, or to return a value if it is a function. Any parameters to the main subprogram have undefined values when the program is run; use the subprograms in package COMMAND_LINE if you need to pass information to an Ada program when it is run. The operating system always ignores any function value it returns; use COMMAND_LINE.SET_RETURN_CODE if you need to return a value when the program is completed.