AVF Control Number: AVF-VSR-008

Ada® COMPILER
VALIDATION SUMMARY REPORT:

TeleSoft
TeleGen2 E68, Version 3.11
HOST: MicroVAX II
TARGETS: Motorola 68020, 68010
         Tektronix 8540
         (M68010 CPU)

Completion of On-Site Testing:
86-09-24

Prepared By:
Industrieanlagen-Betriebsgesellschaft mbH
Dept SZT
Einsteinstraße 20
D 8012 Ottobrunn
Federal Republic of Germany

Prepared For:
Ada Joint Program Office
United States Department of Defense
Washington, D.C.

* Ada is a registered trademark of the United States Government (Ada Joint Program Office)
**Ada Compiler Validation Summary Report:**

**Title and Subtitle:**
AdaSoft Telegen2 E68, Version 3.11, Host: MicroVAX II TARGETS; Motorola 68020, 68010, Tektronix 8540 (M68010 CPU)

**Author:**
Industrieanlagen-Betriebsgesellschaft mbH

**Performing Organization and Address:**
Industrieanlagen-Betriebsgesellschaft mbH, Dept 922, Einsteinstrasse 20, D 8012 Ottobrunn Federal Republic of Germany

**Controlling Office Name and Address:**
Ada Joint Program Office
United States Department of Defense
Washington, DC 20301-3081

**Distribution Statement (of Report):**
Approved for public release; distribution unlimited.

**Distribution Statement (of the abstract entered in Block 20, if different from Report):**
UNCLASSIFIED

**Keywords:**

**Abstract:**
See Attached.
EXECUTIVE SUMMARY

This Validation Summary Report (VSR) summarizes the results and conclusions of validation testing performed on the TeleSoft TeleGen2 E68, Version 3.11 Ada Compiler using Version 1.8 of the Ada Compiler Validation Capability (ACVC). This compiler is hosted on a MicroVAX II operating under MicroVMS, Version 4.2. Programs processed by this compiler may be executed on the Motorola 68010, Motorola 68020 and the Tektronix 8540 with M68010 CPU.

On site testing was performed in September 1986 at Nynäshamn, Sweden, under the direction of the Industrieanlagen Betriebsgesellschaft mbH (AVF), according to Ada Validation Organisation (AVO) policies and procedures. The AVF identified 2210 of the 2399 test in ACVC Version 1.8 to be processed during on-site testing of the compiler. 19 tests withdrawn at the time of validation testing, as well as the 170 executable tests that make use of floating-point precision exceeding that supported by the implementation, were not processed. After the 2210 tests were processed, results for Class A, C, D, and E tests were examined for correct execution. Compilation listings for Class B tests were analyzed for correct diagnosis of syntax and semantic errors. Compilation and link results of Class L tests were analyzed for correct detection of errors. There were 202 of the processed tests determined to be inapplicable. The remaining 2008 tests were passed.

The results of validation are summarized in the following table:

<table>
<thead>
<tr>
<th>RESULT</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passed</td>
<td>102</td>
<td>252</td>
<td>334</td>
<td>243</td>
<td>161</td>
<td>97</td>
<td>136</td>
<td>262</td>
<td>107</td>
<td>32</td>
<td>217</td>
<td>65</td>
<td></td>
<td>2008</td>
</tr>
<tr>
<td>Failed</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not Appl.</td>
<td>14</td>
<td>73</td>
<td>86</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>23</td>
<td>0</td>
<td>1</td>
<td>168</td>
<td>372</td>
<td></td>
</tr>
<tr>
<td>Withdrawn</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>116</td>
<td>330</td>
<td>425</td>
<td>247</td>
<td>161</td>
<td>98</td>
<td>140</td>
<td>264</td>
<td>134</td>
<td>32</td>
<td>219</td>
<td>233</td>
<td>2399</td>
<td></td>
</tr>
</tbody>
</table>

The AVF concludes that these results demonstrate acceptable conformity to ANIS/MIL-STD-1815 Ada.
EXECUTIVE SUMMARY

This Validation Summary Report (VSR) summarizes the results and conclusions of validation testing performed on the TeleSoft TeleGen2 E68, Version 3.11 Ada Compiler using Version 1.8 of the Ada Compiler Validation Capability (ACVC). This compiler is hosted on a MicroVAX II operating under MicroVMS, Version 4.2. Programs processed by this compiler may be executed on the Motorola 68010, Motorola 68020 and the Tektronix 8540 with M68010 CPU.

On-site testing was performed in September 1986 at Nynäshamn, Sweden, under the direction of the Industrieanlagen Betriebsgesellschaft mbH (AVF), according to Ada Validation Organization (AVO) policies and procedures. The AVF identified 2210 of the 2399 test in ACVC Version 1.8 to be processed during on-site testing of the compiler. The 19 tests withdrawn at the time of validation testing, as well as the 170 executable tests that make use of floating-point precision exceeding that supported by the implementation, were not processed. After the 2210 tests were processed, results for Class A, C, D, and E tests were examined for correct execution. Compilation listings for Class B tests were analyzed for correct diagnosis of syntax and semantic errors. Compilation and link results of Class L tests were analyzed for correct detection of errors. There were 202 of the processed tests determined to be inapplicable. The remaining 2008 tests were passed.

The results of validation are summarized in the following table:

<table>
<thead>
<tr>
<th>RESULT</th>
<th>CHAPTER</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Passed</td>
<td>102</td>
<td>252</td>
</tr>
<tr>
<td>Failed</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Not Appl.</td>
<td>14</td>
<td>73</td>
</tr>
<tr>
<td>Withdrawn</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>TOTAL</td>
<td>116</td>
<td>330</td>
</tr>
</tbody>
</table>

The AVF concludes that these results demonstrate acceptable conformity to ANIS/MIL-STD-1815 Ada.
# TABLE OF CONTENTS

## CHAPTER 1
**INTRODUCTION**

1.1 PURPOSE OF THIS VALIDATION SUMMARY REPORT . 6
1.2 USE OF THIS VALIDATION SUMMARY REPORT ..... 6
1.3 REFERENCES .............................. 7
1.4 DEFINITION OF TERMS ....................... 8
1.5 ACVC TEST CLASSES ....................... 9

## CHAPTER 2
**CONFIGURATION INFORMATION**

2.1 CONFIGURATION TESTED ........................ 12
2.2 IMPLEMENTATION CHARACTERISTICS ............. 13

## CHAPTER 3
**TEST INFORMATION**

3.1 TEST RESULTS ................................ 17
3.2 SUMMARY OF TEST RESULTS BY CLASS .......... 17
3.3 SUMMARY OF TEST RESULTS BY CHAPTER ......... 18
3.4 WITHDRAWN TESTS .............................. 18
3.5 INAPPLICABLE TESTS ......................... 13
3.6 SPLIT TESTS .................................. 21
3.7 ADDITIONAL TESTING INFORMATION .......... 22
3.7.1 Prevalidation .............................. 22
3.7.2 Test Method .............................. 22

## APPENDIX A
**COMPLIANCE STATEMENT** 25

## APPENDIX B
**IMPLEMENTATION DEPENDENCIES** 28

## APPENDIX C
**TEST PARAMETERS** 36

## APPENDIX D
**WITHDRAWN TESTS** 37

## APPENDIX E
**SUBSET LIST** 39

## APPENDIX F
**SAMPLE SCRIPT** 41

## APPENDIX G
**CONFIGURATION DIAGRAM** 44
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 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 characteristics of particular operating systems, hardware, or implementation strategies. All of 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 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:

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

Testing of this compiler was conducted by the AVF according to policies and procedures established by the Ada Validation Office (AVO). On-site testing was conducted at Nynåshamn and completed on 86-09-24.

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. no.522). 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 any statement or statements set forth in this report are accurate or complete, or that the subject compiler has no nonconformances 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:
IABG m.b.H., Dept SZT
Einsteinstrasse 20
D-8012 Ottobrunn
Federal Republic of Germany

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


1.4 DEFINITION OF TERMS

ACVC  The Ada Compiler Validation Capability. A set of programs that evaluates the conformity of a compiler to the Ada language specification, ANSI/MIL-STD-1815A.


Applicant  The agency requesting validation.

AVF  The Ada Validation Facility. In the context of this report, the AVF is responsible for conducting compiler validations according to established policies and procedures.

AVO  The Ada Validation Organization. In the context of this report, the AVO is responsible for setting policies and procedures for compiler validations.

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  A test for which the compiler generates a result that demonstrates nonconformance to the Ada Standard.

Host  The computer on which the compiler resides.

Inapplicable test  A 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  A test for which a compiler generates the expected result.

Target  The computer for which a compiler generates code.

Test  A program that checks a compiler's conformity regarding a particular feature or 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
A test found to be incorrect and not used to
check conformity to the Ada language specification. 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 link errors.

Class A tests check that legal Ada programs can be successfully compiled and executed. However, no checks are performed during execution to see if the test objective has been met. 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 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.

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.

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 these units 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 the tests in the ACVC follow 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 are listed 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. 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 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 (see Appendix G for a diagram).

Compiler: TeleSoft TeleGen2 E68, Version 3.11
ACVC: 1.8
Certificate Expiration Date: 87-12-17

Host Compiler:
Machine: MicroVAX II
Operating System: MicroVMS, Version 4.2
Memory Size: 7 Mbytes

DEC-NET connection to VAX 11/750 for tape reading and line printing.

Target Computers

(1) Target System Motorola 68010

The 68010 target system consists of an MC 68010 CPU mounted on a VME bus printed circuit board with 256 Kb of memory. Floating point operations are handled by software, written in assembler and interfaced from the code-generator as part of the run-time system.

(2) Target System Motorola 68020

The 68020 target system consists of an MC 68020 CPU mounted on a VME bus printed circuit board with 256 Kb of memory. Floating point operations are handled by software, written in assembler and interfaced from the code-generator as part of the run-time system.
(3) Target System Tektronix 8540

In the Tektronix 8540 Emulator Station the probe contains an MC 68010 CPU. Floating point operations are handled by software, written in assembler and interfaced from the code-generator as part of the run-time system. This system had 192 Kb of memory.

The targets were run without operating systems. The Motorola machines contain basic service routines for host target communication on PROM storage. On the Tektronix system the M68010 Emulator V2.0 is used for host target communication.

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. This compiler is characterized by the following interpretations of the Ada Standard:

- Capacities.

The compiler correctly processes tests containing loop statements nested to 65 levels, block statements nested to 65 levels, and recursive procedures separately compiled as subunits nested to 17 levels. It correctly processes a compilation containing 722 variables in the same declarative part. (See tests D55A03A..H (2 tests), D56001B, D64005E..G (3 tests), and D29002K.)

- Universal integer calculations.

An implementation is allowed to reject universal integer calculations having values that exceed SYSTEM.MAX_INT. This implementation does not reject such calculations and processes them correctly. (See tests D4A002A, D4A002B, D4A004A, and D4A004B.)
Predefined types.

This implementation supports the additional predefined types LONG_INTEGER and LONG_FLOAT in the package STANDARD. (See tests B86001C and B86001D.)

Based literals.

An implementation is allowed to reject a based literal with a value exceeding SYSTEM.MAX_INT during compilation, or it may raise NUMERIC_ERROR or CONSTRAINT_ERROR during execution. This implementation raises NUMERIC_ERROR during execution. (See test E24101A.)

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.

A packed BOOLEAN array having a 'LENGTH exceeding INTEGER'LAST raises no exception when declared or used. (See test C52103X.)

A packed two-dimensional BOOLEAN array with more than INTEGER'LAST components raises CONSTRAINT_ERROR when the length of a dimension is calculated and exceeds INTEGER'LAST. (See test C52104Y.)

A null array with one dimension of length greater than INTEGER'LAST may raise NUMERIC_ERROR or CONSTRAINT_ERROR either when declared or assigned. Alternately, an implementation may accept the declaration. However, lengths must match in array slice assignments. This implementation raises CONSTRAINT_ERROR when array objects are assigned. (See test E52103Y.)

In assigning one-dimensional array types, the expression appears to be evaluated in its entirety before CONSTRAINT_ERROR is raised when checking whether the expression's subtype is compatible with the target's subtype. In assigning two-dimensional array types, the expression does not appear to be 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.)
Discriminated types.

During compilation, an implementation is allowed to either accept or reject an incomplete type with discriminants that is used in an access type definition with a compatible discriminant constraint. This implementation accepts such subtype indications. (See test E38104A.)

In assigning record types with discriminants, the expression appears to be 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.)

Aggregates.

In the evaluation of a multi-dimensional aggregate, the order in which choices are evaluated and index subtype checks are made appears to depend upon the aggregate itself. (See tests C43207A and C43207B.)

In the evaluation of an aggregate containing subaggregates, all choices are not evaluated before being checked for identical bounds. (See test E43212B.)

All choices are evaluated before CONSTRAINT_ERROR is raised if a bound in a nonnull range of a nonnull aggregate does not belong to an index subtype. (See test E43211B.)

Functions.

An implementation may allow the declaration of a parameterless function and an enumeration literal having the same profile in the same immediate scope, or it may reject the function declaration. If it accepts the function declarations, the use of the enumeration literal's identifier denotes the function. This implementation accepts the declarations. (See test E66001D.)

Representation clauses.

The Ada Standard does not require an implementation to support representation clauses. If a representation clause is not supported, then the implementation must reject it. While the operation of representation clauses is not checked by Version 1.8 of the ACVC, they are used in testing other language features. This implementation accepts 'SIZE and 'STORAGE_SIZE for tasks; it rejects
STORAGE_SIZE for collections, and 'SMALL clauses. Enumeration representation clauses appear not to be supported. (See tests C55B16A, C87B62A, C87B62B, C87B62C, and BC1002A.)

Pragmas.

The pragma INLINE is not supported for procedures. The pragma INLINE is not supported for functions. (See tests CA3004E and CA3004F.)

Input/output.

The package SEQUENTIAL_IO cannot be instantiated with unconstrained array types and record types with discriminants. The package DIRECT_IO cannot be instantiated with unconstrained array types and record types with discriminants without defaults. (See tests AE2101C, AE2101H, CE2201D, CE2201E, and CE2401D.)

The target computers are not equipped with permanent mass storage and the Ada Run Time Systems do not provide file systems. No files - permanent or temporary - can be created. Any attempt to create a file yields USE_ERROR.

Generics.

Generic subprogram declarations and bodies cannot be compiled in separate compilations. (See test CA2009F.)

Generic package declarations and bodies cannot be compiled in separate compilations. (See tests CA2009C and BC3205D.)
CHAPTER 3
TEST INFORMATION

3.1 TEST RESULTS

Version 1.8 of the ACVC contains 2399 tests. When validation testing of the TeleSoft TeleGen2 EG8 was performed, 19 tests had been withdrawn. The remaining 2380 tests were potentially applicable to this validation. The AVF determined that 372 tests were inapplicable to this implementation, and that the 2008 applicable tests were passed by the implementation.

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

3.2 SUMMARY OF TEST RESULTS BY CLASS

<table>
<thead>
<tr>
<th>RESULT</th>
<th>TEST CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>Passed</td>
<td>66</td>
</tr>
<tr>
<td>Failed</td>
<td>0</td>
</tr>
<tr>
<td>Not Appl.</td>
<td>3</td>
</tr>
<tr>
<td>Withdrawn</td>
<td>0</td>
</tr>
<tr>
<td>TOTAL</td>
<td>69</td>
</tr>
</tbody>
</table>
3.3 SUMMARY OF TEST RESULTS BY CHAPTER

<table>
<thead>
<tr>
<th>RESULT</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passed</td>
<td>102</td>
<td>252</td>
<td>334</td>
<td>243</td>
<td>161</td>
<td>97</td>
<td>136</td>
<td>262</td>
<td>107</td>
<td>32</td>
<td>217</td>
<td>65</td>
</tr>
<tr>
<td>Failed</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Not Appl.</td>
<td>14</td>
<td>73</td>
<td>86</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>23</td>
<td>0</td>
<td>1</td>
<td>168</td>
<td>372</td>
</tr>
<tr>
<td>Withdrawn</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>TOTAL</td>
<td>116</td>
<td>330</td>
<td>425</td>
<td>247</td>
<td>161</td>
<td>98</td>
<td>140</td>
<td>264</td>
<td>134</td>
<td>32</td>
<td>219</td>
<td>233</td>
</tr>
</tbody>
</table>

3.4 WITHDRAWN TESTS

The following 19 tests were withdrawn from ACVC Version 1.2 at the time of this validation:

- C32114A
- B33203C
- C34018A
- C35904A
- B37401A
- C41404A
- B45116A
- C48008A
- B49006A
- B4A010C
- B74101B
- C87850A
- C92005A
- CA3005A..D (4 tests)
- C940ACA

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. For this validation attempt, 372 tests were inapplicable for the reasons indicated:

- C34001D, B52004E, B55B09D and C55B07B use SHORT_INTEGER which is not supported by this compiler.
C34001F and C35702A use SHORT_FLOAT which is not supported by this compiler.

C55B16A makes use of an enumeration representation clause containing noncontiguous values which is not supported by this compiler.

B86001D requires a predefined numeric type other than those defined by the Ada language in package STANDARD. There is no such type for this implementation.

C86001F redefines package SYSTEM, but the test cannot be executed since the package REPORT is dependent on the package SYSTEM.

C87B62B uses a length clause to specify 'STORAGE_SIZE. The length clause is rejected during compilation.

CA1012A compiles generic subroutine declarations and bodies in separate compilations. Separate compilation of generic specifications and bodies is not supported by this compiler.

CA2009C, CA2009F, and BC2005D compile generic subunits in separate compilation files. Separate compilation of generic specifications and bodies is not supported by this compiler.

BA1011C, LA5008A..K (11 tests), LA5008M..N (2 tests) require generic specifications and bodies to be in the different compilation files which is not supported by this compiler.

CA3004E, EA3004C, and LA3004A use INLINE pragma for procedures which is not supported by this compiler.

CA3004F, EA3004D, and LA3004B use INLINE pragma for functions which is not supported by this compiler.

AE2101C, CE2201D, and CE2201E use instantiation of package SEQUENTIAL_IO with unconstrained array types which is not supported by this compiler.

AE2101H and CE2401D use instantiation of package DIRECT_IO with unconstrained array types which is not supported by this compiler.
The following 163 tests attempt to check operations on files. The checking is impossible because every create operation raises USE_ERROR.

<table>
<thead>
<tr>
<th>AE3101A</th>
<th>CE2102C</th>
<th>CE2102G</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE2104A</td>
<td>CE2104B</td>
<td>CE2104C</td>
</tr>
<tr>
<td>CE2104D</td>
<td>CE2105A</td>
<td>CE2106A</td>
</tr>
<tr>
<td>CE2107A</td>
<td>CE2107B</td>
<td>CE2107C</td>
</tr>
<tr>
<td>CE2107D</td>
<td>CE2107E</td>
<td>CE2107F</td>
</tr>
<tr>
<td>CE2108A</td>
<td>CE2108B</td>
<td>CE2108C</td>
</tr>
<tr>
<td>CE2108D</td>
<td>CE2109A</td>
<td>CE2110A</td>
</tr>
<tr>
<td>CE2110B</td>
<td>CE2110C</td>
<td>CE2111A</td>
</tr>
<tr>
<td>CE2111B</td>
<td>CE2111C</td>
<td>CE2111D</td>
</tr>
<tr>
<td>CE2111E</td>
<td>CE2111G</td>
<td>CE2111H</td>
</tr>
<tr>
<td>CE2201A</td>
<td>CE2201B</td>
<td>CE2201C</td>
</tr>
<tr>
<td>CE2201F</td>
<td>CE2202A</td>
<td>CE2204A</td>
</tr>
<tr>
<td>CE2204B</td>
<td>CE2210A</td>
<td>CE2401A</td>
</tr>
<tr>
<td>CE2401B</td>
<td>CE2401C</td>
<td>CE2401E</td>
</tr>
<tr>
<td>CE2401F</td>
<td>CE2404A</td>
<td>CE2405B</td>
</tr>
<tr>
<td>CE2406A</td>
<td>CE2407A</td>
<td>CE2409A</td>
</tr>
<tr>
<td>CE2409A</td>
<td>CE2410A</td>
<td>CE3102A</td>
</tr>
<tr>
<td>CE3102B</td>
<td>CE3103A</td>
<td>CE3104A</td>
</tr>
<tr>
<td>CE3107A</td>
<td>CE3108A</td>
<td>CE3108B</td>
</tr>
<tr>
<td>CE3109A</td>
<td>CE3110A</td>
<td>CE3111A</td>
</tr>
<tr>
<td>CE3111B</td>
<td>CE3111C</td>
<td>CE3111D</td>
</tr>
<tr>
<td>CE3111E</td>
<td>CE3112A</td>
<td>CE3112B</td>
</tr>
<tr>
<td>CE3114A</td>
<td>CE3114B</td>
<td>CE3115A</td>
</tr>
<tr>
<td>CE3203A</td>
<td>CE3208A</td>
<td>CE3301A</td>
</tr>
<tr>
<td>CE3301B</td>
<td>CE3301C</td>
<td>CE3302A</td>
</tr>
<tr>
<td>CE3305A</td>
<td>CE3402A</td>
<td>CE3402B</td>
</tr>
<tr>
<td>CE3402C</td>
<td>CE3402D</td>
<td>CE3403A</td>
</tr>
<tr>
<td>CE3403B</td>
<td>CE3403C</td>
<td>CE3403E</td>
</tr>
<tr>
<td>CE3403F</td>
<td>CE3404A</td>
<td>CE3404B</td>
</tr>
<tr>
<td>CE3404C</td>
<td>CE3405A</td>
<td>CE3405B</td>
</tr>
<tr>
<td>CE3405C</td>
<td>CE3405D</td>
<td>CE3406A</td>
</tr>
<tr>
<td>CE3406B</td>
<td>CE3406C</td>
<td>CE3406D</td>
</tr>
<tr>
<td>CE3407A</td>
<td>CE3407B</td>
<td>CE3407C</td>
</tr>
<tr>
<td>CE3408A</td>
<td>CE3408B</td>
<td>CE3408C</td>
</tr>
<tr>
<td>CE3409A</td>
<td>CE3409C</td>
<td>CE3409D</td>
</tr>
<tr>
<td>CE3409E</td>
<td>CE3409F</td>
<td>CE3409G</td>
</tr>
<tr>
<td>CE3410C</td>
<td>CE3411D</td>
<td>CE3410E</td>
</tr>
<tr>
<td>CE3410F</td>
<td>CE3411A</td>
<td>CE3412A</td>
</tr>
<tr>
<td>CE3413A</td>
<td>CE3413C</td>
<td>CE3602A</td>
</tr>
<tr>
<td>CE3602B</td>
<td>CE3602C</td>
<td>CE3602D</td>
</tr>
<tr>
<td>CE3603A</td>
<td>CE3604A</td>
<td>CE3605A</td>
</tr>
<tr>
<td>CE3605B</td>
<td>CE3605C</td>
<td>CE3605D</td>
</tr>
<tr>
<td>CE3605E</td>
<td>CE3606A</td>
<td>CE3606B</td>
</tr>
<tr>
<td>CE3704A</td>
<td>CE3704B</td>
<td>CE3704D</td>
</tr>
</tbody>
</table>
The following 170 tests make use of floating-point precision that exceeds the maximum of 15 supported by the implementation:

C24113L..Y (14 tests)
C35705L..Y (14 tests)
C35706L..Y (14 tests)
C35707L..Y (14 tests)
C35708L..Y (14 tests)
C35802L..Y (14 tests)
C45241L..Y (14 tests)
C45321L..Y (14 tests)
C45421L..Y (14 tests)
C45424L..Y (14 tests)
C45521L..Z (15 tests)
C45621L..Z (15 tests)

3.6 SPLIT TESTS

If one or more errors do not appear to have been detected in a Class B test because of compiler error recovery, then the test is split into a set of smaller tests that contain the undetected errors. These splits are then compiled and examined. The splitting process continues until all errors are detected by the compiler or until there is exactly one error per split. Any Class A, Class C, or Class E test that cannot be compiled and executed because of its size is split into a set of smaller subtests that can be processed.

19 splits were required for Class B tests. No splits were required for tests in the other classes.
3.7 ADDITIONAL TESTING INFORMATION

3.7.1 Prevalidation

Prior to validation, a set of test results for ACVC Version 1.8 produced by the TeleSoft TeleGen2 E68 Ada 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 TeleSoft TeleGen2 E68 using ACVC Version 1.8 was conducted on-site by a validation team from the AVF. The configuration consisted of a MicroVAX II operating under MicroVMS Version 4.2 and a Motorola 68010 without operating system. The host and target computers were linked via RS232. The following configurations involving the same host computer and the same communications network were also tested using a subset (see Appendix E) of the ACVC:

Motorola 68020
Tektronix 8540 with M68010 CPU

The second diagram of Appendix G shows how the Ada Systems for these configurations are related to the Ada System which was fully tested.

A set of magnetic tapes containing all tests except for withdrawn tests and tests requiring unsupported floating-point precisions was taken on-site by the validation team for processing. The set of magnetic tapes contained tests that make use of implementation-specific values were customized before being written to the set of magnetic tapes. Tests requiring splits were included in their split form on the set of magnetic tapes except for B4900CB.
The report package body was modified in order to avoid downloading code for text I/O for every executable test. The modified package passed the CZ-tests. CZ1103A was ruled inapplicable because it attempts to create a file.

The magnetic tapes were read on a VAX 11/750. They were then transferred to the host computer's disk system via DEC-NET.

After the test files were loaded to disk, the full set of tests was compiled, and all executable tests were linked on the host computer. Executable images were transferred to the target computers via RS232. Results were picked up by the host computer via RS232 and transferred to a VAX 11/750 via DEC-NET where they were printed.

The compiler was tested using command scripts provided by TeleLOGIC and reviewed by the validation team. A more detailed explanation of the command script is given in Appendix F. The following options were in effect for testing:

3-tests:
TSADA/E68/PROCEED/MONITOR/VIRTUAL=3000/LIST

Other Tests:
TSADA/E68/PROCEED/MONITOR/VIRTUAL=3000

Linker options for M68010 and M68020 targets

LOCATE/AT=%X100500
INPUT/OFM ENVCLERC

PRESENT/ADA VECTOR_BASE=%X1000000
PRESENT/ADA_BG3TRAP=%XEE766

PRESENT/ADA_USER_STACKLOCATION=%X100000
PRESENT/ADA_USER_STACKSIZE=%X11F00

PRESENT/ADA_INTERRUPT_STACKLOCATION=%X12DF00
PRESENT/ADA_INTERRUPT_STACKSIZE=%X100

PRESENT/ADA_HEAPLOCATION=%X12E000
PRESENT/ADA_HEAPSIZE=%X12000

A slightly different set of options was used for two CE-tests and one CZ-test.

Linker options for Tektronix system
Tests were compiled, linked, and executed (as appropriate) using a single host computer and 4 identical target computers. 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.
APPENDIX A

COMPLIANCE STATEMENT

TeleLOGIC has submitted the following compliance statement concerning the TeleSoft TeleGen2 E68 Ada Compiler.
Validation of Telegen2 for Cross-Compilation from VAX/VMS to embedded Motorola 68010, Motorola 68020 and Tektronix 8540

Dear Sirs:

TeleLOGIC hereby certifies that this Ada compiler passes all applicable ACVC 1.8 tests required for formal validation. All pertinent information is enclosed.

The details of the validation are as follows:

1. Environment
The validation is run on the Host/Target configuration specified below.
The equipment is on the premises at TeleLOGIC in Nynashamn and will be available for your inspection during the validation visit.

2. Host/Target Configuration
We plan to run the validation tests in parallel from one host system to 3 different target systems with the following configurations:

Host System MINADA:
- MicroVAX II running Micro VMS 4.2.
- 7 megabytes of main memory.
- 730 Mb of disk memory.
- Cassette tape station TK50.
- DECNet connection to VAX 750 system with line-printer.
- No local printing facility.

Target System HERA:
- 1 Motorola 68010 CPU
- 256 kilobytes of RAM memory.
- Resident monitor in PROM storage.
- RS232 connection to 1 host tty port for down-line loading and reporting results.
Target System JUNO:

1 Motorola 68020 CPU
256 kilobytes of RAM memory.
Resident monitor in PROM storage.
RS232 connection to host tty port for down-line loading and reporting results.

Target System KIRKE:

Tektronix 8540 with Motorola 68010 CPU.
192 kilobytes of RAM memory.
Host-target connection using Tektronix ICOM40.

The host is connected to the targets as depicted in figure 1. The host is placed in a room adjacent to the laboratory where the targets are located. Each target is mounted in a separate desktop cabinet.

The compiler system is configured according to figure 2. The differences are in the linkers, the run-time support and in the target monitor systems.

A full prevalidation run was performed on system HERA. For JUNO and KIRKE a selected set of tests was run according to your directives.

3. The information relevant for "Appendix F of the LRM" can be found in the User's Guide (chapter 14 - LRM annotations). The list of values used as parameters to the '.TST' tests are given in Attachment A.

4. No deliberate extensions have been made to the Ada language standard.

5. Public release of the validation results is welcomed.

We wish that in all forthcoming references to this validation, the keyword should be "Telesoft", since this is a compiler in the family of portable Ada-technology from Telesoft.

6. We agree to continue to comply with the Ada trademark policy, as defined by the Ada Joint Program Office.

7. All applicable tests in the 1.8 ACVC test suite have been passed, with the exception of a small number of disputed tests, listed in Attachment B. Since no input-output to external files is supported on the targets, most of the tests in group CE have induced raise of "use error".

8. All tests were run using the same set of options. The setting of some options used in running the tests varies with the nature of the compilation module. A list of these options is contained in Attachment C.

9. Payment for the validation services will be made as invoiced by you.

10. TeleLOGIC appreciates your cooperation and consultation in achieving this milestone in our validation efforts.

Stefan Bjornson
APPENDIX B
IMPLEMENTATION DEPENDENCIES

The only allowed implementation dependencies correspond to implementation-dependent pragmas, to certain machine-dependent conventions as mentioned in chapter 13 of MIL-STD-1815A, and to certain allowed restrictions on representation classes. The implementation-dependent characteristics of the TeleSoft TeleGen2 E68, Version 3.11, are described in the following sections which discuss topics in Appendix F of the Ada Language Reference Manual (ANSI/MIL-STD-1315A); they are taken from the compiler's User Manual. Implementation-specific portions of the package STANDARD are also included in this appendix.

The following additional quantities should be noted:

- `DURATION'SMALL`: \(2^{(-14)}\)
- `FLOAT'EPSILON`: \(1.2\times10^{(-38)}\)
- `FLOAT'LARGE`: \(3.4\times10^38\)
- `FLOAT'SMALL`: \(-\text{FLOAT'LARGE}\)
- `LONG_FLOAT'EPSILON`: \(2.2\times10^{(-308)}\)
- `LONG_FLOAT'LARGE`: \(18\times10^307\)
- `LONG_FLOAT'SMALL`: \(-\text{LONG_FLOAT'LARGE}\)
### APPENDIX C: LRM ANNOTATIONS

#### APPENDIX CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.1</td>
<td>LRM Chapter 2</td>
<td>C-1</td>
</tr>
<tr>
<td>C.2</td>
<td>LRM Chapter 3</td>
<td>C-1</td>
</tr>
<tr>
<td>C.3</td>
<td>LRM Chapter 4</td>
<td>C-2</td>
</tr>
<tr>
<td>C.4</td>
<td>LRM Chapter 9</td>
<td>C-2</td>
</tr>
<tr>
<td>C.5</td>
<td>LRM Chapter 10</td>
<td>C-3</td>
</tr>
<tr>
<td>C.6</td>
<td>LRM Chapter 11</td>
<td>C-3</td>
</tr>
<tr>
<td>C.7</td>
<td>LRM Chapter 13</td>
<td>C-3</td>
</tr>
<tr>
<td>C.8</td>
<td>LRM Appendix A</td>
<td>C-4</td>
</tr>
<tr>
<td>C.9</td>
<td>LRM Appendix F</td>
<td>C-4</td>
</tr>
<tr>
<td>C.9.1</td>
<td>Implementation-Defined Pragmas.</td>
<td>C-4</td>
</tr>
<tr>
<td>C.9.2</td>
<td>Implementation-Dependent Attributes.</td>
<td>C-4</td>
</tr>
<tr>
<td>C.9.3</td>
<td>Package SYSTEM.</td>
<td>C-5</td>
</tr>
<tr>
<td>C.9.4</td>
<td>Representation Clauses.</td>
<td>C-5</td>
</tr>
<tr>
<td>C.9.5</td>
<td>Implementation-Generated Names.</td>
<td>C-5</td>
</tr>
<tr>
<td>C.9.6</td>
<td>Address Clause Expression Interpretation.</td>
<td>C-5</td>
</tr>
<tr>
<td>C.9.7</td>
<td>Unchecked Conversion Restrictions.</td>
<td>C-5</td>
</tr>
<tr>
<td>C.9.8</td>
<td>Implementation-Dependent Characteristics of the I/O Packages.</td>
<td>C-6</td>
</tr>
<tr>
<td>C.9.9</td>
<td>Compilation of Generic Units.</td>
<td>C-6</td>
</tr>
</tbody>
</table>
C. LRM ANNOTATIONS

TeleGen2 for VAX/VMS to embedded MC680X0 targets compiles the full ANSI Ada language as defined by the Reference Manual for the Ada Programming Language (LRM) (ANSI/MIL-STD-1815A). This appendix describes the sections of the language that are designated by the LRM as implementation dependent for the compiler and run-time environment. These language-related issues are presented in the order in which they appear in the LRM. Each section answers the corresponding section of questions presented in the document Ada-Europe Guidelines for Ada Compiler Specification and Selection (J. Nissen and B. Wichmann, MPL Report DITC 10/82).

C.1. LRM Chapter 2

[LRM 2.1] The host and target character set is the ASCII character set.

[LRM 2.2] The maximum number of characters on an Ada source line is 200.

[LRM 2.8] TeleGen2 implements the language-defined pragmas: ELABORATE, INLINE, INTERFACE, PRIORITY, and SUPPRESS. This release of TeleGen2 does not support the following language-defined pragmas: CONTROLLED, LIST, MEMORY_SIZE, OPTIMIZE, PACK, PAGE, SHARED, STORAGE_UNIT, and SYSTEM_NAME. If included in Ada source, the pragma will have no effect.

The one implementation-defined pragma for the TeleSoft's Ada compilers, COMMENT, is useful for embedding a comment into the object code. The form of the pragma is

pragma COMMENT ("Comment to be embedded");

Pragma COMMENT may appear at any location within the source code of a compilation unit except within the generic formal part of a generic unit. Any number of comments may be entered into the object code using this method.

C.2. LRM Chapter 3

[LRM 3.2.1] This release of TeleGen2 does not produce warning messages about the use of uninitialized variables. The compiler will not reject a program for this reason. In a later release, the Global Optimizer will identify uninitialized variables.

[LRM 3.5.1] The maximum number of elements in an enumeration type is 32767.

[LRM 3.5.4] There are two predefined integer types: INTEGER and LONG_INTEGER. The attributes of these types are shown in Table C-1.
Table C-1. Attributes of Predefined Types Integer and Long_Integer.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Integer</th>
<th>Long_Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>'First</td>
<td>-32768</td>
<td>-2147483648</td>
</tr>
<tr>
<td>'Last</td>
<td>32767</td>
<td>2147483647</td>
</tr>
<tr>
<td>'Size</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>'Width</td>
<td>6</td>
<td>11</td>
</tr>
</tbody>
</table>

The type Short_Integer is not implemented. Note that better object code will be produced and portability will be enhanced by using explicit integer type definitions rather than these predefined integer types.

[LRM 3.5.8] There are two predefined floating point types: FLOAT and LONG_FLOAT. The floating point types yield:

- \( \text{FLOAT\_DIGITS} = 6 \)
- \( \text{LONG\_FLOAT\_DIGITS} = 15 \)

for the number of decimal digits in the decimal mantissa of the model numbers. These floating point facilities are based on the IEEE standard for 32-bit and 64-bit numbers. Explicit real type definitions should lead to more portable code. The type SHORT_FLOAT is not implemented.

C.3. LRM Chapter 4

[LRM 4.10] There is no limit on the range of literal values for TeleGen2.

[LRM 4.10] There is no limit on the accuracy of real literal expressions. Real literal expressions are computed using an arbitrary precision universal arithmetic package.

C.4. LRM Chapter 9

[LRM 9.6] This implementation uses 32-bit fixed point numbers to represent the type DURATION. The attributes of the type DURATION are shown in Table C-2.

Table C-2. Attributes of Type Duration.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delta</td>
<td>( 2^{15} \times (-14) )</td>
</tr>
<tr>
<td>'First</td>
<td>-86400</td>
</tr>
<tr>
<td>'Last</td>
<td>86400</td>
</tr>
</tbody>
</table>

UG-1002N-V1.2(VAX.E68) © 1988 TeleSoft. 15JUL88
[LRM 9.8] Sixty-four levels of priority are available to associate with tasks through `pragma PRIORITY`. The predefined subtype `PRIORITY` is specified in the package `SYSTEM` as

```lang-ada
subtype PRIORITY is INTEGER range 0..63;
```

Currently the priority assigned to tasks without a `pragma PRIORITY` specification is `PRIORITY'First`. Later optimizations may change this value.

[LRM 9.11] The restrictions on shared variables are only those specified in the LRM.

C.5. LRM Chapter 10

[LRM 10] All main programs are assumed to be procedures without parameters or functions with an integer result type.

[LRM 10.5] A task which was initiated in an imported library unit terminates when the parent program terminates.

C.6. LRM Chapter 11

[LRM 11.1] `NUMERIC_ERROR` is raised for integer or floating point overflow and for divide-by-zero situations. Floating point underflow yields a result of zero without raising an exception.

`PROGRAM_ERROR` and `STORAGE_ERROR` are raised by those situations that are specified in LRM Section 11.1.

C.7. LRM Chapter 13

Only the Chapter 13 facilities explicitly mentioned here are supported in the first release of TeleGen2.

[LRM 13.1] TeleGen2 allows user specification of storage for a task activation using the `STORAGE_SIZE` attribute in a length clause.

[LRM 13.5] Address clauses applied to objects and to single entries are supported. For objects, a simple expression of type `Address` is interpreted as a position within the linear address space of the MC680X0. Unchecked `Conversion` to the private type `System.Address` must be used to specify address constants. For interrupt entries, the address of a TeleSoft-defined `interrupt descriptor` can be given. See Chapter 6 for details.

[LRM 13.7.1] There are no system-generated names for system-dependent components.

[LRM 13.7.3] For a predefined floating point type `F`, the attribute values are as follows:
The TeleSoft Linker can link such Assembly routines into an Ada program when they are represented in TeleSoft Object Form. Third party assemblers can be used to process the Assembly language for these routines, as long as an OMImport utility is available to convert the object modules produced by those assemblers to TeleSoft Object Form.

There are no restrictions on unchecked deallocations. Instantiated versions of this procedure behave as specified in the LRM.

Unchecked conversions are allowed between types (or subtypes) T1 and T2 provided that 1) they have the same static size, 2) they are not unconstrained array types, and 3) they are not private (unless they are subtypes of or are derived from type SYSTEM.ADDRESS).

There are no implementation-defined attributes in the TeleGen2 compiler.

The Ada language definition allows for certain target dependencies in a controlled manner. This appendix, called Appendix F as prescribed in the LRM, describes implementation-dependent characteristics of TeleGen2 for VAX/VMS to embedded MC680X0 targets.

Pragmas INTERFACE is supported for the language Assembly. This pragma can link such Assembly routines into an Ada program when they are represented in TeleSoft Object Form. Third party assemblers can be used to process the Assembly language for these routines, as long as an OMImport utility is available to convert the object modules produced by those assemblers to TeleSoft Object Form.

There are no restrictions on unchecked deallocations. Instantiated versions of this procedure behave as specified in the LRM.

Unchecked conversions are allowed between types (or subtypes) T1 and T2 provided that 1) they have the same static size, 2) they are not unconstrained array types, and 3) they are not private (unless they are subtypes of or are derived from type SYSTEM.ADDRESS).

There are no implementation-defined attributes in the TeleGen2 compiler.

This appendix, called Appendix F as prescribed in the LRM, describes implementation-dependent characteristics of TeleGen2 for VAX/VMS to embedded MC680X0 targets.

Implementation-Defined Pragmas. There is one implementation-defined pragma, pragma COMMENT. This pragma is used for embedding a sequence of characters into the object code. The syntax is:

```
pragma COMMENT( <string_literal> );
```

where:

- `<string_literal>` represents the characters to be embedded in the object code.

Pragma COMMENT may appear at any location within the source code of a compilation unit except within the generic formal part of a generic unit. Any number of comments may be entered into the object code using this method.

Implementation-Dependent Attributes. There are no implementation-dependent attributes.

There are no implementation-defined attributes in the TeleGen2 compiler.
C.9.3. Package SYSTEM.

The current specification of the package is provided below. Note that the named number Tick is not used by any component of the TeleGen2 compiler or run-time system. Similarly, Memory_Size is not used.

```plaintext
type ADDRESS is access INTEGER;

type NAME is (TeleGen2);

SYSTEM_NAME : constant NAME := TeleGen2;

STORAGE_UNIT : constant := 8;

MEMORY_SIZE : constant := (2 ** 31) - 1;

-- 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 - 1));

TICK : constant := 10.0E-3;

-- Other System-Dependent Declarations:

subtype PRIORITY is INTEGER range 0 .. 63;

-- Other TeleSoft Declarations:

type SUBPROGRAM_VALUE is private;

private

end SYSTEM;
```

C.9.4. Representation Clauses. The TeleGen2 supports the following representation clause:

- Address Clauses: for objects and entries [LRM 13.5]
- Length Clauses: specifying storage for a task at activation [LRM 13.1]

C.9.5. Implementation-Generated Names. There are no implementation-generated names denoting implementation-dependent components.

C.9.6. Address Clause Expression Interpretation. Expressions that appear in Address specifications are interpreted as the address of the first storage unit of the object.

C.9.7. Unchecked Conversion Restrictions. Unchecked conversions are allowed between types (or subtypes) T1 and T2 provided that 1) they have the same static size, 2) they are not unconstrained array types, and 3) they are not private (unless they are subtypes of or are derived...

1. Only I/O to a system console is supported in Text IO.
2. Direct IO and Sequential IO are not supported.
3. In Text IO, the type Count is defined as follows:
   \[
   \text{type Count is range } 0..32767; 
   \]
4. In Text IO, the type Field is defined as follows:
   \[
   \text{subtype Field is integer range } 0..1000; 
   \]
5. The standard library contains preinstantiated versions of Text IO.Integer IO for type Integer and Long Integer and of Text IO.Float IO for type Float and Long Float. It is suggested that the following be used to eliminate multiple instantiations of these packages:
   \[
   \begin{align*}
   &\text{Integer Text IO} \\
   &\text{Long Integer Text IO} \\
   &\text{Float Text IO} \\
   &\text{Long Float Text IO}
   \end{align*}
   \]

C.9.9. Compilation of Generic Units. The declaration and body of a generic unit must be submitted as a single compilation (i.e., must be in the same source file).
APPENDIX C

TEST PARAMETERS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMAX_IN_LEN</td>
<td>200</td>
</tr>
<tr>
<td>$BIG_ID1</td>
<td>String(1..200) := (1..199 =&gt; 'A', 200 =&gt; '1')</td>
</tr>
<tr>
<td>$BIG_ID2</td>
<td>String(1..200) := (1..199 =&gt; 'A', 200 =&gt; '2')</td>
</tr>
<tr>
<td>$BIG_ID3</td>
<td>String(1..200) := (1..100 =&gt; 'A', 101 =&gt; '3', 102..200 =&gt; 'A')</td>
</tr>
<tr>
<td>$BIG_ID4</td>
<td>String(1..200) := (1..100 =&gt; 'A', 101 =&gt; '4', 102..200 =&gt; 'A')</td>
</tr>
<tr>
<td>$NEG_BASED_INT</td>
<td>16#$FFFFFFFE#</td>
</tr>
<tr>
<td>$BIG_INT_LIT</td>
<td>String(1..200) := (1..197 =&gt; '0', 198..200 =&gt; '298')</td>
</tr>
<tr>
<td>$BIG_REAL_LIT</td>
<td>String(1..200) := (1..194 =&gt; '0', 195..200 =&gt; '69.0E1')</td>
</tr>
<tr>
<td>$EXTENDED_ASCII_CHARS</td>
<td><em>abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ%?@$%^&amp;</em>()-<em>.{[^}]*</em>)*</td>
</tr>
<tr>
<td>$NON_ASCII_CHAR_TYPE</td>
<td>(NON_NULL)</td>
</tr>
<tr>
<td>$BLANKS</td>
<td>String(1..180) := (1..180 =&gt; ' ')</td>
</tr>
<tr>
<td>$MAX_DIGITS</td>
<td>15</td>
</tr>
<tr>
<td>$NAME</td>
<td>No such numeric type - long integer used</td>
</tr>
<tr>
<td>$INTEGER_FIRST</td>
<td>2**31;</td>
</tr>
<tr>
<td>$INTEGER_LAST</td>
<td>2**31-1;</td>
</tr>
<tr>
<td>$MAX_INT</td>
<td>2**31-1;</td>
</tr>
<tr>
<td>$LESS_THAN_DURATION</td>
<td>-86.401.0</td>
</tr>
<tr>
<td>$GREATER_THAN_DURATION</td>
<td>86.401.0</td>
</tr>
<tr>
<td>$LESS_THAN_DURATION_BASE_FIRST</td>
<td>131.072.0</td>
</tr>
<tr>
<td>$GREATER_THAN_DURATION_BASE_LAST</td>
<td>131.072.0</td>
</tr>
<tr>
<td>$COUNT_LAST</td>
<td>2147483645</td>
</tr>
<tr>
<td>$FIELD_LAST</td>
<td>1000</td>
</tr>
<tr>
<td>$FILENAME_WITH_BAD_CHARS</td>
<td><em>X]%'Y</em>$</td>
</tr>
<tr>
<td>$FILE_NAME_WITH_WILD_CARD_CHAR</td>
<td><em>XYZ</em>*</td>
</tr>
<tr>
<td>$ILLEGAL_EXTERNAL_FILE_NAME1</td>
<td><em>BAD-CHARACTER</em>/%?*</td>
</tr>
<tr>
<td>$ILLEGAL_EXTERNAL_FILE_NAME2</td>
<td>String(1..120) := (1..120 =&gt; 'A')</td>
</tr>
</tbody>
</table>
APPENDIX D
WITHDRAWN TESTS

Some tests are withdrawn from the ACVC because they do not conform to the Ada Standard. The following 19 tests had been withdrawn at the time of validation testing for the reasons indicated. A reference of the form "AI-dddd" is to an Ada Commentary.

- C32114A: An unterminated string literal occurs at line 62.
- B33203C: The reserved word "IS" is misspelled at line 45.
- C34018A: The call of function G at line 114 is ambiguous in the presence of implicit conversions.
- C35904A: The elaboration of subtype declarations SFX3 and SFX4 may raise NUMERIC_ERROR instead of CONSTRAINT_ERROR as expected in the test.
- B37401A: The object declarations at lines 126 through 135 follow subprogram bodies declared in the same declarative part.
- C41404A: The values of 'LAST and 'LENGTH are incorrect in the if statements from line 74 to the end of the test.
- B45116A: ARRPRIBL1 and ARRPRIBL2 are initialized with a value of the wrong type--PRIBOOL_TYPE instead of ARRPRIBOOL_TYPE--at line 41.
- C48008A: the assumption that evaluation of default initial values occurs when an exception is raised by an allocator is incorrect according to AI-00397.
- B49006A: Object declarations at lines 41 and 50 are terminated incorrectly with colons, and end case is missing from line 42.
- B4A010C: The object declaration in line 18 follows a subprogram body of the same declarative part.
- B74101B: The begin at line 9 causes a declarative part to be treated as a sequence of statements.
C87B50A: The call of "/=" at line 31 requires a .use clause for package A.

C92005A: The "/=" for type PACK.BIG_INT at line 40 is not visible without a use clause for the package PACK.

C940ACA: The assumption that allocated task TT1 will run prior to the main program, and thus assign SPY.NUM the value checked for by the main program, is erroneous.

CA3005A..D (4 tests): No valid elaboration order exists for these tests.

BC3204C: The body of BC3204CO is missing.
APPENDIX E
SUBSET LIST

<p>| CZ1101A.ADA | CZ1102A.ADA | CZ1103A.ADA |
| CZ1201A.ADA | CZ1201B.ADA | CZ1201C.ADA |
| B22001E.TST | B22001E.TST | B22001E.TST |
| C23003A.TST | C23003A.TST | C23003A.TST |
| C24101A.TST | C24101A.TST | C24101A.TST |
| A29002E.ADA | C32107B.ADA | B33201D.ADA |
| C34001H.ADA | C35104A.ADA | C35711A.ADA |
| E36171H.ADA | C36205K.ADA | C37012A.ADA |
| B37301G.ADA | B38001D.ADA | E38104A.ADA |
| C38105A.ADA | C41105A.ADA | C41204A.ADA |
| C41303C.ADA | C41306C.ADA | C42006A.ADA |
| C43107A.ADA | C43205D.ADA | C43208A.ADA |
| E43211B.ADA | E43212B.ADA | E43212C.ADA |
| C45101I.ADA | C45123B.ADA | B45206B.ADA |
| B45302G.ADA | C45220B.ADA | C45264B.ADA |
| C45327C.ADA | C45342A.ADA | C45401A.ADA |
| C45505A.ADA | B45522A.ADA | C45526A.ADA |
| C45672A.ADA | B48002E.ADA | C48004A.ADA |
| C48007B.ADA | C48009C.ADA | C48010A.ADA |
| D4A002A.ADA | C4A005A.ADA | C4A011A.ADA |
| C4A014A.ADA | C51004A.ADA | C52001B.ADA |
| C52005D.ADA | C52102A.ADA | C52103P.ADA |
| E52103Y.ADA | C52104B.ADA | C53005A.ADA |
| C54A02A.ADA | C54A24A.ADA | C54A42A.ADA |
| A54B01A.ADA | D55A03B.ADA | A55B12A.ADA |
| C55C02B.ADA | D56001B.ADA | C57003A.ADA |
| B58003B.ADA | C58006A.ADA | B59001I.ADA |
| C59902B.ADA | B61001N.ADA | A62006D.ADA |
| B63009B.ADA | C64005C.ADA | C64103A.ADA |
| C64104L.ADA | C64105E.ADA | C64109E.ADA |
| E66001D.ADA | C66002F.ADA | C67005A.ADA |
| E71002A.ADA | A73001J.ADA | A74106A.ADA |
| C74305A.ADA | C74409B.ADA | A83001F.ADA |
| C870C3A.ADA | A850.3B.ADA | C86001E.ADA |
| C87A05B.ADA | C87B11B.ADA | C87B20A.ADA |
| C87B42A.ADA | B91001E.ADA | C91085B.ADA |
| C93005F.ADA | E94004A.ADA | E94004B.ADA |
| E94004C.ADA | C94007A.ADA | B95006C.ADA |
| E95031A.ADA | C95065B.ADA | C95076A.ADA |
| E95087A.ADA | C96005A.ADA | B97108B.ADA |</p>
<table>
<thead>
<tr>
<th>C97304A.ADA</th>
<th>EA1003B.ADA</th>
<th>CA1008A0.ADA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA1008A1M.ADA</td>
<td>CA1012B0.ADA</td>
<td>CA1012B2.ADA</td>
</tr>
<tr>
<td>CA1012B4M.ADA</td>
<td>LA5001A0.ADA</td>
<td>LA5001A1.ADA</td>
</tr>
<tr>
<td>LA5001A2.ADA</td>
<td>LA5001A3.ADA</td>
<td>LA5001A4.ADA</td>
</tr>
<tr>
<td>LA5001A5.ADA</td>
<td>LA5001A6.ADA</td>
<td>LA5001A7M.ADA</td>
</tr>
<tr>
<td>CA5002A.ADA</td>
<td>CA5003B0.ADA</td>
<td>CA5003B1.ADA</td>
</tr>
<tr>
<td>CA5003B2.ADA</td>
<td>CA5003B3.ADA</td>
<td>CA5003B4.ADA</td>
</tr>
<tr>
<td>CA5003B5M.ADA</td>
<td>CA5004B.ADA</td>
<td>LA5007M0.ADA</td>
</tr>
<tr>
<td>LA5007M1.ADA</td>
<td>LA5007M2.ADA</td>
<td>LA5007M3M.ADA</td>
</tr>
<tr>
<td>LA5007S0.ADA</td>
<td>LA5007S1.ADA</td>
<td>LA5007S2M.ADA</td>
</tr>
<tr>
<td>LA5007S3.ADA</td>
<td>LA5008C0.ADA</td>
<td>LA5008C1M.ADA</td>
</tr>
<tr>
<td>LA5008W0.ADA</td>
<td>LA5008W1.ADA</td>
<td>LA5008W2.ADA</td>
</tr>
<tr>
<td>LA5008W3.ADA</td>
<td>LA5008W4M.ADA</td>
<td>LA5008W5.ADA</td>
</tr>
<tr>
<td>CB1002A.ADA</td>
<td>CB2006A.ADA</td>
<td>CB3004A.ADA</td>
</tr>
<tr>
<td>CB4006A.ADA</td>
<td>CB5001B.ADA</td>
<td>CC2002A.ADA</td>
</tr>
<tr>
<td>CC3011D.ADA</td>
<td>CC3120B.ADA</td>
<td>CC3208B.ADA</td>
</tr>
<tr>
<td>CC3407A.ADA</td>
<td>CC3504C.ADA</td>
<td>AE2101A.ADA</td>
</tr>
<tr>
<td>AE2101F.ADA</td>
<td>CE21021.ADA</td>
<td>CE2102J.ADA</td>
</tr>
</tbody>
</table>
APPENDIX F
SAMPLE SCRIPT

---

Validation of the TeleSoft Cross-Compilation System

Sample VMS/CCL script for compiling, binding, linking and downloading a test program in the cross compiler validation environment.

This is a model of the basic steps taken. A more elaborate set of scripts were used for the actual validation runs to provide an automated test driver.

In this example C1234.S134 is used as test program name. This nonexistent test thus contains the main compilation unit "Procedure C1234.S134".

All steps in this script are done for a class 1, 2, 3 or 4 test.

For class 1 tests linking, linking and downloading are unnecessary.

For class 2 tests downloading is performed if an executable file is produced.

---

Initialize variables

\[\begin{align*}
\text{testname} &= \text{name of test source file} \\
\text{mainname} &= \text{name of main compilation unit} \\
\text{n_test} &= \text{TRUE if test is a class 2 test, FALSE otherwise} \\
\text{port} &= \text{name of terminal port to use for host/target communication}
\end{align*}\]

\[\begin{align*}
\text{testname} &= \text{"C1234.S134"} \\
\text{mainname} &= \text{"C1234.S134"} \\
\text{n_test} &= \text{FALSE} \\
\text{DEF14} &= \text{port TA011}
\end{align*}\]

---

Create an Ada library, i.e. in TeleSoft Ada terminology a list of Ada sublibraries. The list contains entries for the working sublibrary, any environment-specific libraries and the Ada runtime library.

\[\begin{align*}
\text{CREATE library.lib} \\
\text{name: working} \\
\text{name: env_dir:31301/env} \\
\text{name: std_dir:31310} \\
\text{name: std_dir:Ada310}
\end{align*}\]
Create the actual working sublibrary

```
TSDA /CREATE /Iso Worklib
```

Choose compilation command depending on test type. If the test is a test
a compilation listing must be supplied.

```
IF b_test THEN -
    compile_command = "TSDA /666 /PROCEED /MONITOR /VIRT=1000 /LIST"
ELSE -
    compile_command = "TSDA /666 /PROCEED /MONITOR /VIRT=1000"
```

Compile the test program

```
'compile_command' 'testname'
```

Link the main procedure. This step produces the actual main program and a
link command file for the linking.

```
TSDA /LINK -
/LOADLIB = 'TSDA'.LIB
/LOADLIB = 'testname'.LIB -
'mainname'
```

Link the main procedure. This step produces a file suitable for downloading
to the target computer. The options file contains target specific information
eg. how to divide memory in the target into code areas, heap and different
stacks.

```
TSDA /LINK -
/LOADLIB = 'TSDA'.LIB
/LOADLIB = 'testname'.LIB -
'mainname'
```
Download to the target. The target is in a ready-state, starts receiving the
SI code and boots the origina when the downloading is finished. The test
results are written to the typeahead buffer on the host computer using the
same line that were used for downloading.

TSIQA
DOWNLOAD -
LINE = port -
SOUT -
'mainname',.EX

In this example the file will be named C123456.RES

COPY 'mainname',.RES

TYPE 'mainname',.RES

The target computer is now ready for the next test, waiting as a download
receiver again.
APPENDIX G

CONFIGURATION DIAGRAM

Figure 1
Host-Target Communication Configuration for Cross-Validation
MicroVAX II/
Micro VMS

ACVC tests

Front End +
Middle Pass

"Low Form"

Back-End

TS format
generator

Tek format
generator

68020
TS OBJ
TS RTS

68010
Tek OBJ
Tek RTS

Telesoft
Linker

Tektronix
Linker

68010
Tek 40
68020
EXE
EXE

TS DLL
Tek LANDS

TS DLR
M68010
ACVC appl
RPT

TS DLR
M68020
ACVC appl
RPT

Tek 8540
ACVC appl

Figure 2
Development System Configuration