

DTIC FILE COPY

2

UR20 — Process/Environment Integration
Ada Command Environment (ACE)
Version 8.0 SunOS Implementation

Version Description Document

UNISYS

AD-A228 822



STARS-RC-00990/001/00

25 October 1990

DTIC
ELECTE
NOV 14 1990
S-8 D

90 11 13 119

RESTRICTION STATEMENT IN

is not to be publicly released

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204 Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE 25 October 1990	3. REPORT TYPE AND DATES COVERED Version Description Document	
4. TITLE AND SUBTITLE Ada Command Environment (ACE)			5. FUNDING NUMBERS STARS Contract F19628-88-D-0031	
6. AUTHOR(S) William P. Loftus				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Unisys Corporation 12010 Sunrise Valley Drive Reston, VA 22091			8. PERFORMING ORGANIZATION REPORT NUMBER GR-7670-1163(NP)	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Department of the Air Force Headquarters, Electronic Systems Division (AFSC) Hanscom AFB, MA 01731-5000			10. SPONSORING / MONITORING AGENCY REPORT NUMBER 00990	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) The Ada Command Environment (ACE) is an interactive Ada environment coupled with a set of Ada abstract data types (ADTs). The interactive environment allows users to rapidly prototype general Ada applications, while the ADTs allow prototyping of applications for particular domains, such as X Window System applications. In addition, the ADTs provide an Ada view of underlying applications, which when combined with the interactive environment replaces the traditional role of a command language. When using ACE, Ada becomes the command language as well as the programming language. This version of ACE includes support for X Window System prototyping.				
14. SUBJECT TERMS Ada Command Environment (ACE) Abstract Data Types (ADT)			15. NUMBER OF PAGES 42	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT SAR	

TASK: UR20
CDRL: 00990
25 October 1990

VERSION DESCRIPTION DOCUMENT
For The
SOFTWARE TECHNOLOGY FOR ADAPTABLE, RELIABLE SYSTEMS
(STARS)

Ada Command Environment (ACE)
Version 8.0
SunOS Implementation

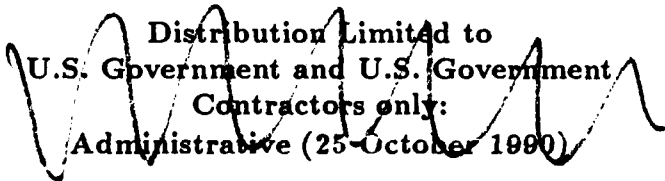
STARS-RC-00990/001/00
Publication No. GR-7670-1163(NP)
25 October 1990

Data Type: A005, Informal Technical Data

CONTRACT NO. F19628-88-D-0031
Delivery Order 0002

Prepared for:
Electronic Systems Division
Air Force Systems Command, USAF
Hanscom AFB, MA 01731-5000

Prepared by:
Unisys Defense Systems
Tactical Systems Division
12010 Sunrise Valley Drive
Reston, VA 22091

 Distribution Limited to
U.S. Government and U.S. Government
Contractors only:
Administrative (25 October 1990)

TASK: UR20
CDRL: 00990
25 October 1990

VERSION DESCRIPTION DOCUMENT
For The
SOFTWARE TECHNOLOGY FOR ADAPTABLE, RELIABLE SYSTEMS
(STARS)

Ada Command Environment (ACE)
Version 8.0
SunOS Implementation

STARS-RC-00990/001/00
Publication No. GR-7670-1163(NP)
25 October 1990

Data Type: A005, Informal Technical Data

CONTRACT NO. F19628-88-D-0031
Delivery Order 0002

Prepared for:
Electronic Systems Division
Air Force Systems Command, USAF
Hanscom AFB, MA 01731-5000

Prepared by:
Unisys Defense Systems
Tactical Systems Division
12010 Sunrise Valley Drive
Reston, VA 22091

PREFACE

This document was prepared by Unisys Corporation, Valley Forge Laboratories, in support of the Unisys STARS Prime contract under the Process/Environment Integration task (UR20). This CDRL, 00990, is type A005 (Informal Technical Data) and is entitled "Ada Command Environment (ACE) Version 8.0, Version Description Document".

Reviewed by:

Teri F. Payton
Teri F. Payton, System Architect

Approved by:

Hans W. Polzer
Hans W. Polzer, Program Manager

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By <u>per letter</u>	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



Contents

1 SCOPE	1
1.1 Identification	1
1.2 System Overview	1
2 RELATED SOFTWARE	1
3 VERSION DESCRIPTION	1
3.1 Inventory of Contents	1
3.1.1 Subdirectory: ace/code	2
3.1.1.1 Subdirectory: ace/code/common	2
3.1.1.2 Subdirectory: ace/code/design	2
3.1.1.2.1 Subdirectory: ace/code/design/telesoft	2
3.1.1.2.2 Subdirectory: ace/code/design/vads	3
3.1.1.3 Subdirectory: ace/code/src	3
3.1.1.3.1 Subdirectory: ace/code/src/telesoft	3
3.1.1.3.2 Subdirectory: ace/code/src/vads	3
3.1.1.3.3 Subdirectory: ace/code/src/C	3
3.1.1.3.4 Subdirectory: ace/code/src/Scanner_specs	3
3.1.1.3.5 Subdirectory: ace/code/src/Parser_specs	3
3.1.2 Subdirectory: /bf ace/bin	3
3.1.3 Subdirectory: ace/startups	4
3.1.4 Subdirectory: docs	4
3.2 Changes Installed	4
3.2.1 Xt Toolkit Support	4
3.2.2 Command Histories	5
3.2.3 Static Semantics Support	6
3.2.4 Expanded String Arguments	6
3.2.5 Miscellaneous	6
3.3 Adaptation Data	7
3.3.1 Operating Environment	7
3.3.2 Development Environment	7
3.3.3 Configuration-Unique Data	8
3.3.3.1 System_Dependent_Routines Package.	8
3.3.3.2 Ace_Universal_Types Package.	8
3.3.3.3 TTY_Operations Package.	9
3.3.3.4 Command Language Scripts.	9
3.4 Interface Compatibility	9
3.5 Installation Instructions	9
3.5.1 VADS Build Procedure	10
3.5.2 Startup Files	11
3.5.3 Execution of ACE	12
3.6 Potential Problems	12
3.7 Enhancements	12

4	NOTES	13
A	Appendix: Inventory of Contents	14
B	Appendix: Build Scripts	21
B.1	File: Build_ACE.var	21
B.2	Script: Build_ACE.VADS	23
B.3	Script: Build_ACE.TeleSoft	33

1 SCOPE

1.1 Identification

Version Description Document,
Ada Command Environment (ACE),
Version 8.0,
SunOS Implementation

1.2 System Overview

A set of Ada abstract data types (ADTs) is the underlying substrate that defines a common, Ada-oriented interface to diverse host environments. The Ada ADTs, in conjunction with the use of Ada as a command language, serve as a unifying concept in the description of a portable Ada command environment. The benefits provided by ADTs and Ada in a software engineering environment are extended into the command language arena. The Ada Command Environment combines the power of Ada as a command language with the description of the host environment through ADTs. ACE presents to the user a consistent Ada-oriented, development environment that supports a uniform interface across a heterogeneous set of development architectures.

This document provides an overview for version 8.0 of the Ada Command Environment (ACE) software. This software was developed in its original form under the STARS Foundations program, administered by the Office of Naval Research. This distribution includes modifications to the ACE software in several key areas, which are briefly described below.

2 RELATED SOFTWARE

The X Window System, Version 11 Release 3 (X11R3), available through the MIT/X Consortium.

The Unisys STARS Ada/Xlib bindings (from Version 2 of the Unisys Ada/Xt Toolkit), available from Unisys STARSCenter.

Sun-3/Unix CAIS-A Implementation, Version 4.5.3, available from Unisys STARSCenter.

3 VERSION DESCRIPTION

3.1 Inventory of Contents

The ACE distribution is structured as shown below. The top-level directory `ace` includes PostScript (`VDDace.ps`) and clear ASCII text (`VDDace.tty`) versions of this document,

along with a complete directory listing of the ACE distribution (**Contents.tty**, reproduced herein as **Appendix A**).

```
ace
ace/code
ace/code/common
ace/code/common/vads
ace/code/common/telesoft
ace/code/design
ace/code/design/vads
ace/code/design/telesoft
ace/code/src
ace/code/src/vads
ace/code/src/C
ace/code/src/Parser_specs
ace/code/src/Scanner_specs
ace/code/src/telesoft
ace/bin
ace/docs
ace/startups
```

3.1.1 Subdirectory: **ace/code**

The **code** subdirectory contains C shell scripts and associated support files that are needed to build ACE from this distribution. The Ada source code for ACE is divided into the following **code** subdirectories:

3.1.1.1 Subdirectory: **ace/code/common.** The **code/common** subdirectory contains reusable Ada code (specifications and implementations) which can be used independently of ACE. Included in this directory, in addition to several commonly available packages, is a key binding facility that can be used to provide any Ada application with command line editing and history functions.

3.1.1.2 Subdirectory: **ace/code/design.** The **code/design** subdirectory contains ACE-specific Ada specifications.

3.1.1.2.1 Subdirectory: **ace/code/design/telesoft.** This subdirectory contains TeleSoft-specific versions of certain Ada specifications.

3.1.1.2.2 Subdirectory: ace/code/design/vads. This subdirectory contains VADS-specific versions of certain Ada specifications.

3.1.1.3 Subdirectory: ace/code/src. The code/src subdirectory contains ACE-specific Ada implementations. In addition, three subdirectories contain non-Ada source code and grammar definitions for ACE's lexical scanner and parsers.

3.1.1.3.1 Subdirectory: ace/code/src/telesoft. This subdirectory contains TeleSoft-specific versions of certain Ada implementations.

3.1.1.3.2 Subdirectory: ace/code/src/vads. This subdirectory contains VADS-specific versions of certain Ada implementations.

3.1.1.3.3 Subdirectory: ace/code/src/C. This subdirectory contains "C" support files.

3.1.1.3.4 Subdirectory: ace/code/src/Scanner_specs. This subdirectory contains the specifications processed by the **adalex** lexical scanner generator to produce the ACE lexical scanner implementation in the source files listed below. This information is provided for reference and is not needed for installation or operation of the ACE distribution.

lex.a

3.1.1.3.5 Subdirectory: ace/code/src/Parser_specs. This subdirectory contains the specifications processed by the **ayacc** parser generator to produce the ACE parser implementation in the source code files listed below. This information is provided for reference and is not needed for installation or operation of this ACE distribution.

compilation_unit.a

decl_part.a

seq_of_stmts.a

3.1.2 Subdirectory: /bf ace/bin

The bin subdirectory contains Unix C shell scripts that are used by ACE to execute operating system functions and aid in window placement. The user must modify these scripts to reflect the actual operating environment.

3.1.3 Subdirectory: ace/startups

This subdirectory contains the interpreted Ada code necessary to define the initial Ada environment provided by ACE.

startup.ace - This file should be copied to the user's home directory. It is the file that is read when ACE is invoked, and it specifies what startup environment that user will have.

acvc.ace - Contains routines necessary to execute the ACVC 1.10 test suite.

commands.ace - Contains routines that are analogous to commands normally found in a command environment (e.g., directory list, edit, compile).

cpu_time.ace - Defines routines to record and report CPU time

developer.ace - Defines routines that ACE developers use (this is not a static set of routines).

startup40.ace - Release 4.0's startup.ace file.

standard.ace - Defines standard Ada packages (e.g., Text_IO)

windowing.ace - Defines windowing operations.

cais.ace - Defines CAIS-A tools and operations.

bindings.ace - Defines Key mapping functions.

xt.ace - Defines the ACE interface to the X Window System.

3.1.4 Subdirectory: docs

The **docs** subdirectory contains PostScript (.ps) and clear ASCII text (.tty) versions of the ACE user's manual.

A complete list of files is contained in Appendix A.

3.2 Changes Installed

3.2.1 Xt Toolkit Support

The major enhancement in Version 8.0 over Version 6.0 (Version 7.0 was an internal release) is its extensive support for the X Window System's Xt toolkit. Ada interfaces are provided to the complete Hewlett-Packard widget set, as well as many of the Xt intrinsic functions. This makes it possible to prototype Xt widget applications with ACE.

Files that were added to support Xt prototyping are:

```
src/ace_hp_widgets.a
src/ace_widget.a
src/ace_x_windows.a
design/ace_hp_widgets.a
design/ace_widget.a
design/ace_x_windows.a
common/hp_widgets.a
common/hp_widgets_.a
common/intrinsics.a
common/renamed_xlib_types.a
common/stringdefs.a
common/widget.a
common/widget_.a
src/C/xwc.c
startups/xt.ace
```

Files that were modified to support Xt prototyping are:

```
src/ebuilt.a
src/ace_adt.a
src/system.a
src/C/cflush.c
src/C/xws.c
design/system.a
startups/commands.ace
startups/standard.ace
```

3.2.2 Command Histories

Several enhancements were made to the command history code, including the ability to call the ACE interpreter from within a key binding. This required modification of:

```
common/sinput.a
common/sinput_.a
common/tty_ops.a
common/tty_ops_.a
common/keymap.a
common/keymap_.a
src/ebuilt.a
startups/bindings.ace
```

Changing the package `Sanctified_Input` into a generic caused the addition of:

```
design/ace_input.a
src/ace_input.a
```

3.2.3 Static Semantics Support

Support for processing the static semantics of **Array** declarations and references required the modification of:

```
src/cexpr.a
src/semant.a
src/smgmt.a
src/compilation_unit.a
src/decl_part.a
src/seq_of_stmts.a
src/Parser_specs/compilation_unit.y
src/Parser_specs/decl_part.y
src/Parser_specs/seq_of_stmts.y
design/semant.a
```

3.2.4 Expanded String Arguments

The following files were created or modified in order to support the use of expanded names in string arguments:

```
design/expand.a
src/expand.a
```

3.2.5 Miscellaneous

Many small problems were fixed. These included determining the correct size of an **out** string parameter to a built-in routine, improper exception handling during subprogram parameter elaboration, correctly identifying an expanded name, comment changes, etc. Files that were modified are:

```
src/ace_adt.a
src/euser.a
src/user.a
src/ebuilt.a
src/debug.a
src/atextio.a
src/context.a
```

src/drtns.a
src/dump.a
src/eprag.a
src/error.a
src/liter1.a
src/misc.a
design/debug.a
design/misc.a

File `src/rwace.a` was moved to `design/rwace.a`.

File `src/observe_window.icn` was moved to `src/C/observe_window.icn`.

The following files were removed from the distribution, since they were not directly related to the execution of ACE:

bin/move-if-change
misc/sysenv.a
src/cexpand.a
src/Make.script_unix
src/Makefile.unix_example
src/C/sunws.c

3.3 Adaptation Data

3.3.1 Operating Environment

The ACE prototype operates on a Sun-3 workstation. ACE provides an interface to the X Window System, but may be executed independently, without a supporting windowing system. Window manipulation operations are provided with the X Window System Ada/Xlib bindings from the Unisys Ada/Xt Toolkit.

To execute ACE, the suggested configuration is a Sun-3 workstation running:

SunOS, Version 4.0.3

MIT X Window System, Version 11, Release 3 (if running ACE with X)

3.3.2 Development Environment

To create an ACE executable image, the suggested configuration is a Sun-3 workstation running:

SunOS, Version 4.0.3

MIT X Window System, Version 11, Release 3 (if running ACE with X)

Verdix Ada Development System, Version 5.5t

Ada/Xlib binding to the X Window System, from Version 2 of the Unisys
Ada/Xt Toolkit.

C compiler provided with SunOS 4.0.3

Further information on the system dependent parts of ACE and how to port these to other environments is given in section 4.0.3

3.3.3 Configuration-Unique Data

ACE was designed with portability as one of its primary goals. Operating and Ada compilation system dependent routines have been isolated to the **System_Dependent_Routines** package, with the minor exceptions noted below. Of the many features designed into ACE, the least portable is the interface to the host window system. Operating system dependencies, and subsequent symbolic debugger dependencies on the host OS, comprise the remainder of configuration-dependent features.

3.3.3.1 System_Dependent_Routines Package. The **System_Dependent_Routines** package contains the procedures which implement file system manipulations and interrupt handling (see files **system.a** and **system.cais.a**).

ACE's hierarchical file system abstract data type (ADT) and command language facilities dictate that it provide access to the host file system utilities. Porting considerations between different file systems include minor differences in naming conventions (name length and lexical character limitations) and more complex differences between how Unix and other operating systems treat directory structures. In these situations, it is best to use the host OS-supplied routines or the interfaces directly provided by the host Ada compilation system.

Interrupt handling is very operating system (and system architecture) dependent. Interrupt handling is primarily used by the ACE symbolic debugger ADT, but similar mechanisms could be used in the future to implement exception handling or Ada tasking. Depending on the host OS interface provided by the Ada compilation system, somewhat complex rework might be necessary. For example, the Ada compilation systems on the Unisys PC/IT provide an Ada interface to handle interrupts.

3.3.3.2 Ace_Universal_Types Package. Data sizes for types such as **Integer** vary significantly from one Ada compilation system to another. In order to isolate this, ACE defines

its own integer type (**Ace_Integer**) with an explicit range. This forces the compilation system to choose an appropriate machine representation for the **Ace_Integer** type or inform the user of its inability to do so.

3.3.3.3 TTY_Operations Package. The **TTY_Operations** package provides screen manipulation routines, which are necessarily configuration dependent. The current implementation of **TTY_Operations** uses **termcap**, a terminal capability database, to handle screen manipulation. For most systems there is a public domain implementation of **termcap**. If a version of **termcap** is not available for a system that is to run ACE, then the body of **TTY_Operations** would need to be rewritten to remove the references to **termcap**.

3.3.3.4 Command Language Scripts. Other current host OS dependencies include the use of accessory native command language scripts to provide needed, high-level host dependent functionality. This is provided so users can interchange compatible host applications without recompiling ACE itself. The **ace_edit** and **ace_ada** files are two such scripts. However, the need for such scripts is defined by the environment implementor, since all the functionality provided by the scripts could also be provided in the system environment implementation (i.e., the interpreted Ada ADT bodies).

3.4 Interface Compatibility

Changes to ACE appearing in this version will not affect other components of the user's system. This is true even though this version introduces a new interface to the X Window System library.

3.5 Installation Instructions

This section contains the instructions necessary to construct ACE from the source files contained in the version 8.0 distribution. ACE executables have been successfully compiled using the Verdix and TeleSoft compilation systems. The Verdix compilation is the recommended ruggedized version. The following configurations are supported:

Verdix, version 5.5t, on SunOS 4.0.3

- with/without X11R3 support
- with/without CAIS-A support

TeleSoft, version 1.4, on SunOS 4.0.3

- with/without X11R3 support
- without CAIS-A support

Currently, only the Verdex version of ACE supports both CAIS-A interaction and stand-alone operation. All other versions of ACE are stand-alone. Only the source code for ACE is included in this delivery. Contact Unisys STARSCenter for the Ada/X Window System Xlib bindings and/or CAIS-A distributions.

3.5.1 VADS Build Procedure

1. Edit the environment variables in file **Build_ACE.var** to reflect the actual operating environment. The following environment variables must be modified:

VADS_BASE (Optional) - identifies the full pathname of the VADS compilation system (e.g., /mybase/compilers/vads_5.5). Needed only if building with the VADS compilation system.

TELEGEN2 (Optional) - identifies the full pathname of the TeleSoft compilation system (e.g., /mybase/compilers/telegen_1.4). Needed only if building with the TeleSoft compilation system.

ADA_XLIB (Optional) - identifies the full pathname of the Ada libraries for the Ada/Xt Toolkit Xlib bindings (e.g., /mybase/adaxt/code/Xlib). Needed only if building for operation with an X Window System interface.

CAIS_LIB (Optional) - identifies the full pathname of the nonshared portion of the CAIS-A installation (e.g., /mybase/cais-a/src/nonshared). Needed only if you are compiling for operation under the CAIS-A object management system.

LIB_X_SUPPORT (Optional) - identifies the full pathname of the standard C utility library from the X Window System distribution or from the Ada/Xt Toolkit (e.g., /mybase/adaxt/code/C/lib.a). Needed only if building for operation with an X Window System interface.

ACE - identifies the full pathname of the top-level ACE directory (e.g., /mybase/ace).

If the \$ACE directory is structured as described in this document, no further modifications are necessary. If not, the following additional variables in **Build_ACE.var** will have to be modified to indicate which host directory contains each of the major code components of this release:

ACE.CODE
ACE.COMMON
ACE.DESIGN
ACE.SRC
ACE.CLIB
ACE.BIN
LOG

Other environment variables may need to change, depending on the installer's system configuration. A complete listing of the **Build_ACE.var** file is included in Appendix B.

2. Execute **Build_ACE.VADS**, or **Build_ACE.TeleSoft**, as appropriate, providing configuration information when prompted by the script.

3.5.2 Startup Files

There are several startup files provided with this distribution (see the **startups** directory for their source). Users may (and should) configure their individual **startup.ace** file to suit their needs. The **startup.ace** file should reside in a user's home directory. The following code is an example **startup.ace**:

```
1  --pragma echo(on);
2
3  -- get routines for measuring CPU.
4  Interpret_File ("/ace/startups/cpu_time.ace");
5
6  -- Variables for clocking our startup speed.
7
8  Start : Time;
9  Stop  : Time;
10
11 -- Start ticking
12 Start := Clock;
13
14 Interpret_File ("/ace/startups/standard.ace");
15 Interpret_File ("/ace/startups/commands.ace");
16 Interpret_File ("/ace/startups/windowing.ace");
17 Interpret_File ("/ace/startups/bindings.ace");
18
19 -- Stop Ticking
20 Stop := Clock;
21
22 -- How much time?
23
24 Put ("Startup CPU seconds: ");
25 Put_Time(Difference(Stop, Start));
26
27 -- ASCII Terminal clear to EOL.
28 Put_Line(Ascii.Esc & "[K");
```

3.5.3 Execution of ACE

Type **ACE** at the shell prompt to invoke ACE stand-alone, where **ACE** is the name of the executable made from the compiling and linking steps described above. ACE is invoked automatically from CAIS-A as the user's login shell. Upon startup, the stand-alone ACE reads the startup file from the user's home directory, while the CAIS-A version will follow the rules specified in the CAIS-A script for starting ACE. To install the new ACE executable and **startup.ace** into the CAIS-A baseline SEE, replace the existing ACE executable (in **Baseline_SEE/cais-a/bin/ACE.exec**) and the existing ACE startup file (in **Baseline_SEE/cais-a/lib/startup.ace**). See the Unisys STARS Baseline SEE, Virtual Interface Implementation 3, Informal Report, 26 September 1989 for more information.

3.6 Potential Problems

Several problems are known to exist in the ACE 8.0 distribution:

1. Non-terminating recursive functions may cause ACE to lose track of declared objects. Currently, there is no known workaround (other than not writing non-terminating functions).
2. Calling the **Ace_Adt.Interpret** procedure recursively is illegal and can cause ACE to terminate unexpectedly; sometimes this can occur without a direct call by the user, i.e., calling **Edit_And_Interpret** during the interruption of an X callback.
3. Several enumeration values of **Key_Bindings.Commands** will have no effect when used in the **Make_Bindings** command. They are:

Get_Current_Line
Get_Current_Character
Get_Current_Column

They will be removed from the enumeration type in the next release. They cannot be removed in the current version, since there exists an underlying dependency in the **Evaluate_Built_In_Subprogram** subprogram.

4. A function cannot be used as one of the bounds of a **String** object declaration; the workaround is to assign the function value to an object and use the object in the **String** object declaration.

3.7 Enhancements

Ultimately, ACE will be an interpreter for the entire Ada language. This will provide the foundation for a portable interactive environment for Ada coding. By extending the interpreter with complete support for the X Window System, an environment similar to InterLisp

and Smalltalk could be created for Ada, which would remove many of the barriers that currently exist to a rapid prototyping Ada system.

4 NOTES

A Appendix: Inventory of Contents

NOTE: "*" identifies executables; "/" identifies directories

ace:

Contents.tty

VDDace.ps

VDDace.tty

bin/

code/

docs/

startups/

ace/bin:

ace_ada*

ace_edit*

place_observe_window.csh*

ace/code:

Build_ACE.TeleSoft*

Build_ACE.VADS*

Build_ACE.var

common/

design/

src/

ace/code/common:

btrees.a

btrees_.a

hash_table.a

hash_table_.a

hp_widgets.a

intrinsic.a

keymap.a

keymap_.a

lists.a

lists_.a

renamed_xlib_types.a

sinput.a

sinput_.a

stringdefs.a

telesoft/

tty_ops.a

unix_types.a

vads/

widget.a

ace/code/common/telesoft:

common.alb

common.x.alb

hp_widgets_.a

tty_ops_.a

unix_types_.a

widget_.a

ace/code/common/vads:

hp_widgets_.a

tty_ops_.a

unix_types_.a

widget_.a

ace/code/design:

ace_adt.a

ace_hp_widgets.a

ace_input.a

ace_widget.a

ace_x_windows.a

amain.a

aprgut.a

astd.a

atextio.a

auntyp.a

calendar.a

common_parser.a

compilation_unit.a

context.a

cpu.a

create.a

ctree.a

debug.a

decl_part.a

dir.a

drsup.a

drtns.a

dspprt.a

dsubsup.a

dump.a

error.a

expand.a

files.a

get.a
help.a
lex.a
lexact.a
liter1.a
misc.a
miscs.a
obj.a
oe.a
preprs.a
prsdef.a
rwace.a
sdb.a
semant.a
seq_of_stmts.a
set.a
smgms.a
smgmt.a
stget.a
stmtev.a
string.a
strt.a
sts.a
stset.a
system_environment_.a
telesoft/
tokens_definition.a
treeb.a
user.a
vads/
wndobj.a
ws.a
xadt.a
yyerr.a

ace/code/design/telesoft:
design.alb
design.x.alb
sunview.a
system.a
x.a

ace/code/design/vads:
sunview.a
system.a

x.a

ace/code/src:

C/

Parser_specs/

Scanner_specs/

ace_adt.a

ace_hp_widgets.a

ace_input.a

ace_widget.a

ace_x_windows.a

alloc.a

amain.a

aprgut.a

astd.a

atextio.a

calendar.a

carray.a

cattr.a

cdot.a

cexpr.a

cinfix.a

cmsppt.a

common_parser.a

compilation_unit.a

compilation_unit_goto.a

compilation_unit_shift_reduce.a

context.a

create.a

csmisc.a

cstmt.a

csubpgm.a

ctree.a

debug.a

decl_part.a

decl_part_goto.a

decl_part_shift_reduce.a

denum.a

dir.a

dpkg.a

drecs.a

drsup.a

drtns.a

dspprt.a

dsubprg.a

dsubsup.a
dtyped.a
dump.a
eattr.a
eblock.a
ebuilt.a
ebuilt.cais.a
ebuilt.x.a
ebuilt.xcais.a
eexpr.a
einfix.a
eistmt.a
eobject.a
epkgb.a
eprag.a
error.a
estmt.a
esubprm.a
euser.a
expand.a
files.a
get.a
help.a
initsym.a
lex.a
lexact.a
literl.a
main.a
misc.a
miscs.a
nowin.a
occe.a
pre_parser.a
rwace.a
semant.a
seq_of_stmts.a
seq_of_stmts_goto.a
seq_of_stmts_shift_reduce.a
set.a
snags.a
snagt.a
staddpk.a
stget.a
stmtecs.a
stmtcv.a

strt.a
stset.a
telesoft/
user.a
vads/
xadt.a
yyerr.a

ace/code/src/C:
cflush.c
observe_window.icn
sunws.c
sys_env.c
xwc.c
xws.c

ace/code/src/Parser_specs:
compilation_unit.y
decl_part.y
seq_of_stmts.y

ace/code/src/Scanner_specs:
scan_adalex.l

ace/code/src/telesoft:
onlyx.a
src.alb
src.cais.alb
src.x.alb
src.xcais.alb
sunview.a
system.a
system.cais.a
system_environment.a
ws.a
x.a

ace/code/src/vads:
onlyx.a
sunview.a
system.a
system.cais.a
system_environment.a
ws.a
x.a

ace/docs:
UserManual.ps
UserManual.tty

ace/startups:
acvc.ace
bindings.ace
cais.ace
commands.ace
cpu_time.ace
developer.ace
standard.ace
startup.ace
startup40.ace
windowing.ace
xt.ace

B Appendix: Build Scripts

B.1 File: Build_ACE.var

```
1 #
2 # Establish a path to the VADS compilation system.
3 #
4 setenv VADS_BASE      <path to the VADS compilation system
5                        (e.g. /mybase/compilers/vads5.5)>
6 setenv VADS_BIN       $VADS_BASE/bin
7 set path              = ( $VADS_BIN $path )
8 setenv ADA            " ada -w -O0 "
9
10 #
11 # Establish a path to the TeleSoft compilation system.
12 #
13 setenv TELEGEN2       <path to the TeleSoft compilation system
14                        (e.g., /mybase/compilers/telegen_1.4)>
15 setenv TADA           " $TELEGEN2/bin/ada -v "
16 setenv TALD           " $TELEGEN2/bin/ald -v -V 2000 "
17 setenv TACR           " $TELEGEN2/bin/acr -f -m 32000 "
18
19 #
20 # Define C Language compilation variable
21 #
22 setenv CC             " cc -g -c "
23
24 #
25 # Define the location of the required source code directories
26 # and I11R3 C archive
27 #
28 setenv ADA_XLIB       <path to the Ada libraries for the Ada/It Toolkit
29                        Xlib bindings (e.g. /mybase/adaxt/code/Xlib)>
30
31 setenv CAIS_LIB       <path to the Ada libraries for the nonshared portion of
32                        CAIS-A (e.g. /mybase/cais-a/src/nonshared)>
33
34 setenv LIB_X_SUPPORT  <path to the standard C utility library from the X Window
35                        System distribution or from the Ada/It Toolkit
36                        (e.g. /mybase/adaxt/code/C/lib.a)>
37
38 setenv ACE            <path to the top-level ACE directory (e.g. /mybase/ace)>
39 setenv ACE_CODE       $ACE/code
40 setenv ACE_COMMON     $ACE_CODE/common
41 setenv ACE_DESIGN     $ACE_CODE/design
```

```
42  setenv ACE_SRC      $ACE_CODE/src
43  setenv ACE_CLIB     $ACE_CODE/src/C
44
45  #
46  # Define the location of C support utilities for the X Window interface
47  #
48  setenv LIB_MISC      $ACE_CLIB/cflush.o
49  setenv LIB_ARG       $ACE_CLIB/sys_env.o
50  setenv LIB_WIDGETCLASSES $ACE_CLIB/xwc.o
51  setenv LIB_WINDOW_SUPPORT $ACE_CLIB/xws.o
52
53  #
54  # The following variables locate various components of the X11R3 distribution.
55  # The pathnames shown are typical, but can vary from one installation to
56  # another. Please consult your system administrator.
57  #
58  setenv LIB_HP_WIDGETS    /usr/lib/X11R3/libXw.a
59  setenv LIB_ATHENA_WIDGETS /usr/lib/X11R3/libXaw.a
60  setenv LIB_XT            /usr/lib/X11R3/libXt.a
61  setenv LIB_MU            /usr/lib/X11R3/libXmu.a
62  setenv LIB_X11           /usr/lib/X11R3/libX11.a
63  setenv LIB_RESOLV        /usr/lib/libresolv.a
64  setenv LIB_TERMCAP       /usr/lib/libtermcap.a
65
66  setenv LIBRARIES_WITH_X  " $LIB_MISC          \
67                          $LIB_X_SUPPORT        \
68                          $LIB_WIDGETCLASSES    \
69                          $LIB_WINDOW_SUPPORT    \
70                          $LIB_HP_WIDGETS        \
71                          $LIB_ATHENA_WIDGETS    \
72                          $LIB_XT                \
73                          $LIB_MU                \
74                          $LIB_X11               \
75                          $LIB_RESOLV            \
76                          $LIB_TERMCAP "
77
78  setenv LIBRARIES_NO_X    " $LIB_MISC $LIB_RESOLV $LIB_TERMCAP "
```

B.2 Script: Build_ACE.VADS

```
1  #! /bin/csh -f
2  echo ""
3  echo "Defining installation-dependent variables"
4  echo ""
5  source Build_ACE.var
6
7  setenv TARGET $ACE_CODE/Build_VADS
8  setenv LOG    $TARGET/Build_ACE.Log
9
10 #
11 set Caisop="n"
12 set Xop="n"
13 echo -n "Are you building for operation under CAIS-A? [y n](n) "
14 set Caisop=$<
15 if ( $Caisop == "y" || $Caisop == "Y" ) then
16   set Caisop="y"
17   echo "Building for operation under CAIS-A.  Thank you."
18 else
19   set Caisop="n"
20   echo "Building for stand-alone operation.  Thank you."
21 endif
22 echo ""
23
24 echo -n "Are you building for operation with an X Window System interface? [y n](n) "
25 set Xop=$<
26 if ( $Xop == "y" || $Xop == "Y" ) then
27   set Xop="y"
28   echo "Building for operation with an X Window System interface.  Thank you."
29 else
30   set Xop="n"
31   echo "Building for operation without an X Window System interface.  Thank you."
32 endif
33 echo ""
34
35 #
36 if ( ! -d $TARGET ) mkdir $TARGET
37 if ( ! -d $TARGET/common ) mkdir $TARGET/common
38 if ( ! -d $TARGET/design ) mkdir $TARGET/design
39 if ( ! -d $TARGET/src ) mkdir $TARGET/src
40 if ( ! -d $TARGET/bin ) mkdir $TARGET/bin
41
42 #
43 echo "Building Ada libraries in each sub-directory"
```

```
44 echo ""
45 #
46 foreach dir (common design src)
47   a.mklib -f $TARGET/$dir $VADS_BASE/verdirxlib
48 end
49
50 #
51 echo "Establishing dependencies"
52 echo ""
53 #
54 cd $TARGET/common
55 if ( $Xop == "y" ) then
56   a.path -a $ADA_XLIB
57 endif
58
59 cd $TARGET/design
60 a.path -a $TARGET/common
61 if ( $Xop == "y" ) then
62   a.path -a $ADA_XLIB
63 endif
64 a.path -a $TARGET/src
65
66 cd $TARGET/src
67 a.path -a $TARGET/design
68 if ( $Xop == "y" ) then
69   a.path -a $ADA_XLIB
70 endif
71 a.path -a $TARGET/common
72
73 if ( $Caisop == "y" ) then
74   a.path -a $CAIS_LIB
75 endif
76
77 #
78 echo "Creating source code links in $TARGET"
79 echo ""
80 cd $TARGET/common
81 foreach file ($ACE_COMMON/*.a)
82   ln -s $file ${file:t}
83 end
84 foreach file ($ACE_COMMON/vads/*.a)
85   ln -s $file ${file:t}
86 end
87
88 cd $TARGET/design
```

```
89  foreach file ($ACE_DESIGN/*.a)
90      ln -s $file ${file:t}
91  end
92  foreach file ($ACE_DESIGN/vads/*.a)
93      ln -s $file ${file:t}
94  end
95
96  cd $TARGET/src
97  foreach file ($ACE_SRC/*.a)
98      ln -s $file ${file:t}
99  end
100  foreach file ($ACE_SRC/vads/*.a)
101      ln -s $file ${file:t}
102  end
103
104  #
105  echo "Initializing the build log, file $LOG"
106  echo ""
107  #
108  if -e $LOG rm -f $LOG
109
110  #
111  echo "Compiling the code"
112  echo ""
113  #
114  echo "Subdirectory:  common"
115  echo ""
116  #
117  cd $TARGET/common
118      date                                >>& $LOG
119
120      $ADA unix_types.a                  >>& $LOG
121      $ADA unix_types_.a                  >>& $LOG
122      $ADA tty_ops.a                      >>& $LOG
123      $ADA tty_ops_.a                    >>& $LOG
124      $ADA lists.a                       >>& $LOG
125      $ADA lists_.a                      >>& $LOG
126      $ADA hash_table.a                  >>& $LOG
127      $ADA hash_table_.a                 >>& $LOG
128      $ADA btrees.a                      >>& $LOG
129      $ADA btrees_.a                    >>& $LOG
130      $ADA keymap.a                      >>& $LOG
131      $ADA keymap_.a                    >>& $LOG
132      $ADA sinput.a                      >>& $LOG
133      $ADA sinput_.a                    >>& $LOG
```

```
134
135 if ( $Xop == "y" ) then
136     $ADA renamed_xlib_types.a    >>& $LOG
137     $ADA stringdefs.a           >>& $LOG
138     $ADA intrinsics.a           >>& $LOG
139     $ADA widget.a               >>& $LOG
140     $ADA widget_.a              >>& $LOG
141     $ADA hp_widgets.a            >>& $LOG
142     $ADA hp_widgets_.a           >>& $LOG
143 endif
144
145 #
146 echo "Subdirectory:  design"
147 echo ""
148 #
149 cd $TARGET/design
150     date                        >>& $LOG
151
152     $ADA aotyp.a                >>& $LOG
153     $ADA misc.a                 >>& $LOG
154     $ADA lex.a                  >>& $LOG
155     $ADA ws.a                   >>& $LOG
156     $ADA literl.a               >>& $LOG
157
158 #
159 echo "filename src/literl.a"
160 echo ""
161 #
162 cd $TARGET/src
163     $ADA literl.a                >>& $LOG
164
165 #
166 echo "Continuing with subdirectory:  design"
167 echo ""
168 #
169 cd $TARGET/design
170     $ADA string.a                >>& $LOG
171     $ADA smgms.a                 >>& $LOG
172     $ADA sunview.a               >>& $LOG
173     $ADA atextio.a               >>& $LOG
174     $ADA ooe.a                   >>& $LOG
175     $ADA files.a                 >>& $LOG
176     $ADA user.a                  >>& $LOG
177     $ADA dir.a                   >>& $LOG
178
```

```
179 if ( $Xop == "n" ) then
180     $ADA system_environment.a    >>& $LOG
181 endif
182
183     $ADA system.a                >>& $LOG
184     $ADA help.a                  >>& $LOG
185     $ADA sts.a                   >>& $LOG
186     $ADA sdb.a                   >>& $LOG
187     $ADA context.a               >>& $LOG
188     $ADA amain.a                 >>& $LOG
189     $ADA treeb.a                 >>& $LOG
190     $ADA stntev.a                >>& $LOG
191     $ADA debug.a                 >>& $LOG
192     $ADA stset.a                 >>& $LOG
193     $ADA stget.a                 >>& $LOG
194     $ADA preprs.a                >>& $LOG
195     $ADA ctrees.a                >>& $LOG
196     $ADA dspprt.a                >>& $LOG
197     $ADA create.a                >>& $LOG
198     $ADA miscs.a                 >>& $LOG
199     $ADA drtns.a                 >>& $LOG
200     $ADA prsdef.a                >>& $LOG
201     $ADA yyerr.a                 >>& $LOG
202     $ADA tokens_definition.a     >>& $LOG
203     $ADA common_parser.a         >>& $LOG
204     $ADA seq_of_stmts.a          >>& $LOG
205     $ADA get.a                   >>& $LOG
206     $ADA set.a                   >>& $LOG
207     $ADA ace_adt.a               >>& $LOG
208     $ADA dump.a                  >>& $LOG
209     $ADA dsubsup.a               >>& $LOG
210     $ADA semant.a                >>& $LOG
211     $ADA smgmt.a                 >>& $LOG
212     $ADA strt.a                  >>& $LOG
213     $ADA error.a                 >>& $LOG
214     $ADA compilation_unit.a      >>& $LOG
215     $ADA decl_part.a             >>& $LOG
216     $ADA drsup.a                 >>& $LOG
217     $ADA x.a                     >>& $LOG
218     $ADA lexact.a                >>& $LOG
219     $ADA calendar.a              >>& $LOG
220     $ADA obj.a                   >>& $LOG
221     $ADA wndobj.a                >>& $LOG
222     $ADA xadt.a                  >>& $LOG
223     $ADA astd.a                  >>& $LOG
```

```
224      $ADA aprgut.a          >>& $LOG
225      $ADA rwace.a           >>& $LOG
226      $ADA expand.a          >>& $LOG
227
228  if ( $Xop == "y" ) then
229      $ADA ace_x_windows.a    >>& $LOG
230      $ADA ace_widget.a      >>& $LOG
231      $ADA ace_hp_widgets.a   >>& $LOG
232  endif
233
234      $ADA ace_input.a        >>& $LOG
235
236  #
237  echo "Subdirectory:  src"
238  echo ""
239  #
240  cd $TARGET/src
241      date                    >>& $LOG
242
243      $ADA atextio.a          >>& $LOG
244      $ADA context.a         >>& $LOG
245      $ADA dir.a             >>& $LOG
246      $ADA main.a           >>& $LOG
247      $ADA user.a           >>& $LOG
248      $ADA ooe.a            >>& $LOG
249      $ADA miscs.a          >>& $LOG
250      $ADA debug.a          >>& $LOG
251      $ADA dump.a           >>& $LOG
252      $ADA dspprt.a         >>& $LOG
253      $ADA smgmt.a          >>& $LOG
254      $ADA stmtcv.a         >>& $LOG
255      $ADA estmt.a          >>& $LOG
256      $ADA eistmt.a         >>& $LOG
257      $ADA stmtcvs.a        >>& $LOG
258      $ADA esubprm.a        >>& $LOG
259      $ADA eexpr.a          >>& $LOG
260      $ADA eattr.a          >>& $LOG
261      $ADA amain.a          >>& $LOG
262      $ADA set.a            >>& $LOG
263      $ADA create.a         >>& $LOG
264      $ADA get.a            >>& $LOG
265      $ADA stset.a          >>& $LOG
266      $ADA stget.a          >>& $LOG
267      $ADA help.a           >>& $LOG
268      $ADA semant.a         >>& $LOG
```

```
269      $ADA cexpr.a                >>& $LOG
270      $ADA cattr.a                >>& $LOG
271      $ADA csubpgm.a              >>& $LOG
272      $ADA csmisc.a               >>& $LOG
273      $ADA cmsppt.a               >>& $LOG
274
275  if ( $Xop == "n" ) then
276      $ADA system_environment.a    >>& $LOG
277  endif
278
279  if ( $Caisop == "y" ) then
280      $ADA system.cais.a          >>& $LOG
281  else
282      $ADA system.a               >>& $LOG
283  endif
284
285      $ADA files.a                >>& $LOG
286      $ADA cstmt.a                >>& $LOG
287      $ADA drtns.a                >>& $LOG
288      $ADA denum.a                >>& $LOG
289      $ADA drecs.a                >>& $LOG
290      $ADA dpkg.a                 >>& $LOG
291      $ADA dsubprg.a              >>& $LOG
292      $ADA error.a                >>& $LOG
293      $ADA strt.a                 >>& $LOG
294      $ADA staddpk.a              >>& $LOG
295      $ADA initsym.a              >>& $LOG
296      $ADA ace_adt.a              >>& $LOG
297      $ADA dsubsup.a              >>& $LOG
298      $ADA drsup.a                >>& $LOG
299      $ADA x.a                    >>& $LOG
300      $ADA lexact.a               >>& $LOG
301      $ADA yyerr.a                >>& $LOG
302      $ADA lex.a                  >>& $LOG
303      $ADA misc.a                 >>& $LOG
304      $ADA ctrees.a               >>& $LOG
305      $ADA common_parser.a        >>& $LOG
306      $ADA decl_part_shift_reduce.a >>& $LOG
307      $ADA decl_part_goto.a       >>& $LOG
308      $ADA decl_part.a            >>& $LOG
309      $ADA pre_parser.a           >>& $LOG
310      $ADA compilation_unit_shift_reduce.a >>& $LOG
311      $ADA compilation_unit_goto.a >>& $LOG
312      $ADA compilation_unit.a     >>& $LOG
313      $ADA seq_of_stmts_shift_reduce.a >>& $LOG
```

```
314      $ADA seq_of_stmts_goto.a      >>& $LOG
315      $ADA seq_of_stmts.a           >>& $LOG
316      $ADA calendar.a               >>& $LOG
317      $ADA main.a                   >>& $LOG
318      $ADA rwace.a                  >>& $LOG
319      $ADA astd.a                   >>& $LOG
320      $ADA expand.a                  >>& $LOG
321      $ADA aprgut.a                  >>& $LOG
322      $ADA alloc.a                  >>& $LOG
323      $ADA cinfix.a                  >>& $LOG
324      $ADA cdot.a                   >>& $LOG
325      $ADA carray.a                 >>& $LOG
326
327  if ( ( $Caisop == "y" ) && ( $Xop == "y" ) ) then
328      $ADA ebuilt.xcais.a            >>& $LOG # ACE under CAIS-A with X
329  else if ( $Xop == "y" ) then
330      $ADA ebuilt.x.a                >>& $LOG # ACE under Unix with X
331  else if ( $Caisop == "y" ) then
332      $ADA ebuilt.cais.a             >>& $LOG # ACE under CAIS-A without X
333  else
334      $ADA ebuilt.a                  >>& $LOG # ACE under Unix without X
335  endif
336
337      $ADA einfix.a                  >>& $LOG
338      $ADA euser.a                   >>& $LOG
339      $ADA eprag.a                   >>& $LOG
340      $ADA eblock.a                  >>& $LOG
341      $ADA epkgb.a                   >>& $LOG
342      $ADA dtyped.a                  >>& $LOG
343
344  if ( $Xop == "y" ) then
345      $ADA onlyx.a                   >>& $LOG
346  else
347      $ADA nowin.a                   >>& $LOG
348  endif
349
350      $ADA eobject.a                 >>& $LOG
351
352  if ( $Xop == "y" ) then
353      $ADA xadt.a                    >>& $LOG
354  endif
355
356      $ADA smgms.a                   >>& $LOG
357      $ADA ace_input.a               >>& $LOG
358
```

```
359 if ( $Xop == "y" ) then
360     $ADA ace_x_windows.a          >>& $LOG
361     $ADA ace_widget.a             >>& $LOG
362     $ADA ace_hp_widgets.a         >>& $LOG
363 endif
364
365 #
366 echo "Finally, the C subroutines"
367 echo ""
368 #
369 cd $ACE_CLIB
370     date                          >>& $LOG
371
372     $CC cflush.c                  >>& $LOG
373
374     $CC sys_env.c                 >>& $LOG
375
376 if ( $Xop == "y" ) then
377     $CC xwc.c                     >>& $LOG
378     $CC xws.c                     >>& $LOG
379 endif
380
381 #
382 echo "Linking the objects"
383 echo ""
384 #
385 cd $TARGET/src
386     date                          >>& $LOG
387
388 if ( $Xop == "y" ) then
389     a.ld main -o ACE $LIBRARIES_WITH_X >>& $LOG
390 else
391     a.ld main -o ACE $LIBRARIES_NO_X   >>& $LOG
392 endif
393
394     date                          >>& $LOG
395
396 #
397 echo "Placing executable in $TARGET/bin"
398 echo ""
399 #
400     mv -f ACE $TARGET/bin          >>& $LOG
401
402 #
403 echo "Build complete"
```

25 October 1990

STARS-RC-00990/001/00

404 echo ""

B.3 Script: Build_ACE.TeleSoft

```
1  #! /bin/csh -f
2  echo ""
3  echo "Defining installation-dependent variables"
4  echo ""
5  source Build_ACE.var
6
7  setenv TARGET $ACE_CODE/Build_TeleSoft
8  setenv LOG     $TARGET/Build_ACE.Log
9
10 #
11 set Caisop="n"
12 set Xop="n"
13 echo -n "Are you building for operation under CAIS-A? [y n](n) "
14 set Caisop=$<
15 if ( $Caisop == "y" || $Caisop == "Y" ) then
16     set Caisop="y"
17     echo "Building for operation under CAIS-A.  Thank you."
18 else
19     set Caisop="n"
20     echo "Building for stand-alone operation.  Thank you."
21 endif
22 echo ""
23
24 echo -n "Are you building for operation with an X Window System interface? [y n](n) "
25 set Xop=$<
26 if ( $Xop == "y" || $Xop == "Y" ) then
27     set Xop="y"
28     echo "Building for operation with an X Window System interface.  Thank you."
29 else
30     set Xop="n"
31     echo "Building for operation without an X Window System interface.  Thank you."
32 endif
33 echo ""
34
35 #
36 if ( ! -d $TARGET ) mkdir $TARGET
37 if ( ! -d $TARGET/common ) mkdir $TARGET/common
38 if ( ! -d $TARGET/design ) mkdir $TARGET/design
39 if ( ! -d $TARGET/src ) mkdir $TARGET/src
40 if ( ! -d $TARGET/bin ) mkdir $TARGET/bin
41
42 #
43 echo "Building Ada libraries in each sub-directory"
```

```
44 echo ""
45 #
46 foreach dir (common design src)
47   cd $TARGET/$dir
48   $TACR $dir
49 end
50
51 #
52 echo "Establishing dependencies"
53 echo ""
54 #
55 cd $TARGET/common
56 if ( $Xop == "y" ) then
57   ln -s $ACE_COMMON/telesoft/common.x.alb liblst.alb
58 else
59   ln -s $ACE_COMMON/telesoft/common.alb liblst.alb
60 endif
61
62 cd $TARGET/design
63 if ( $Xop == "y" ) then
64   ln -s $ACE_DESIGN/telesoft/design.x.alb liblst.alb
65 else
66   ln -s $ACE_DESIGN/telesoft/design.alb liblst.alb
67 endif
68
69 cd $TARGET/src
70 if ( $Xop == "y" ) then
71   if ( $Caisop == "y" ) then
72     ln -s $ACE_SRC/telesoft/src.xcais.alb liblst.alb
73   else
74     ln -s $ACE_SRC/telesoft/src.x.alb liblst.alb
75   endif
76 else if ( $Caisop == "y" ) then
77   ln -s $ACE_SRC/telesoft/src.cais.alb liblst.alb
78   else
79     ln -s $ACE_SRC/telesoft/src.alb liblst.alb
80   endif
81 endif
82
83 #
84 echo "Creating source code links in $TARGET"
85 echo ""
86 cd $TARGET/common
87 foreach file ($ACE_COMMON/*.a)
88   ln -s $file ${file:t}
```

```
89  end
90  foreach file ($ACE_COMMON/telesoft/*.a)
91    ln -s $file ${file:t}
92  end
93
94  cd $TARGET/design
95  foreach file ($ACE_DESIGN/*.a)
96    ln -s $file ${file:t}
97  end
98  foreach file ($ACE_DESIGN/telesoft/*.a)
99    ln -s $file ${file:t}
100 end
101
102 cd $TARGET/src
103 foreach file ($ACE_SRC/*.a)
104   ln -s $file ${file:t}
105 end
106 foreach file ($ACE_SRC/telesoft/*.a)
107   ln -s $file ${file:t}
108 end
109
110 #
111 echo "Initializing the build log, file $LOG"
112 echo ""
113 #
114 if -e $LOG rm -f $LOG
115
116 #
117 echo "Compiling the code"
118 echo ""
119 #
120 echo "Subdirectory:  common"
121 echo ""
122 #
123 cd $TARGET/common
124     date                                >>& $LOG
125
126     $TADA unix_types.a                >>& $LOG
127     $TADA unix_types_.a               >>& $LOG
128     $TADA tty_ops.a                   >>& $LOG
129     $TADA tty_ops_.a                  >>& $LOG
130     $TADA lists.a                     >>& $LOG
131     $TADA lists_.a                    >>& $LOG
132     $TADA hash_table.a                >>& $LOG
133     $TADA hash_table_.a               >>& $LOG
```

```
134      $TADA btrees.a          >>& $LOG
135      $TADA btrees_.a         >>& $LOG
136      $TADA keymap.a          >>& $LOG
137      $TADA keymap_.a         >>& $LOG
138      $TADA sinput.a           >>& $LOG
139      $TADA sinput_.a          >>& $LOG
140
141  if ( $Xop == "y" ) then
142      $TADA renamed_xlib_types.a >>& $LOG
143      $TADA stringdefs.a        >>& $LOG
144      $TADA intrinsics.a        >>& $LOG
145      $TADA widget.a            >>& $LOG
146      $TADA widget_.a           >>& $LOG
147      $TADA hp_widgets.a        >>& $LOG
148      $TADA hp_widgets_.a       >>& $LOG
149  endif
150
151  #
152  echo "Subdirectory:  design"
153  echo ""
154  #
155  cd $TARGET/design
156      date                      >>& $LOG
157
158      $TADA auntyp.a            >>& $LOG
159      $TADA misc.a              >>& $LOG
160      $TADA lex.a               >>& $LOG
161      $TADA ws.a                >>& $LOG
162      $TADA literl.a            >>& $LOG
163
164  #
165  echo "filename src/literl.a"
166  echo ""
167  #
168  cd $TARGET/src
169      $TADA literl.a            >>& $LOG
170
171  #
172  echo "Continuing with subdirectory:  design"
173  echo ""
174  #
175  cd $TARGET/design
176      $TADA string.a            >>& $LOG
177      $TADA smgms.a             >>& $LOG
178      $TADA sunview.a           >>& $LOG
```

```
179      $TADA atextio.a          >>& $LOG
180      $TADA ooe.a             >>& $LOG
181      $TADA files.a           >>& $LOG
182      $TADA user.a            >>& $LOG
183      $TADA dir.a             >>& $LOG
184
185      if ( $Xop == "n" ) then
186          $TADA system_environment_.a >>& $LOG
187      endif
188
189      $TADA system.a           >>& $LOG
190      $TADA help.a            >>& $LOG
191      $TADA sts.a             >>& $LOG
192      $TADA sdb.a             >>& $LOG
193      $TADA context.a         >>& $LOG
194      $TADA amain.a           >>& $LOG
195      $TADA treeb.a           >>& $LOG
196      $TADA stmtev.a          >>& $LOG
197      $TADA debug.a           >>& $LOG
198      $TADA stset.a           >>& $LOG
199      $TADA stget.a           >>& $LOG
200      $TADA preprs.a          >>& $LOG
201      $TADA ctrees.a          >>& $LOG
202      $TADA dspprt.a          >>& $LOG
203      $TADA create.a          >>& $LOG
204      $TADA miscs.a           >>& $LOG
205      $TADA drtns.a           >>& $LOG
206      $TADA prsdef.a          >>& $LOG
207      $TADA yyerr.a           >>& $LOG
208      $TADA tokens_definition.a >>& $LOG
209      $TADA common_parser.a    >>& $LOG
210      $TADA seq_of_stmts.a     >>& $LOG
211      $TADA get.a             >>& $LOG
212      $TADA set.a             >>& $LOG
213      $TADA ace_adt.a         >>& $LOG
214      $TADA dump.a            >>& $LOG
215      $TADA dsubsup.a         >>& $LOG
216      $TADA semant.a          >>& $LOG
217      $TADA smgmt.a           >>& $LOG
218      $TADA strt.a            >>& $LOG
219      $TADA error.a           >>& $LOG
220      $TADA compilation_unit.a >>& $LOG
221      $TADA decl_part.a       >>& $LOG
222      $TADA drsup.a           >>& $LOG
223      $TADA x.a               >>& $LOG
```

```
224      $TADA lexact.a          >>& $LOG
225      $TADA calendar.a       >>& $LOG
226      $TADA obj.a            >>& $LOG
227      $TADA wndobj.a         >>& $LOG
228      $TADA xadt.a           >>& $LOG
229      $TADA astd.a           >>& $LOG
230      $TADA aprgut.a         >>& $LOG
231      $TADA rwace.a          >>& $LOG
232      $TADA expand.a          >>& $LOG
233
234  if ( $Xop == "y" ) then
235      $TADA ace_x_windows.a    >>& $LOG
236      $TADA ace_widget.a      >>& $LOG
237      $TADA ace_hp_widgets.a  >>& $LOG
238  endif
239
240      $TADA ace_input.a        >>& $LOG
241
242  #
243  echo "Subdirectory:  src"
244  echo ""
245  #
246  cd $TARGET/src
247      date                    >>& $LOG
248
249      $TADA atextio.a          >>& $LOG
250      $TADA context.a         >>& $LOG
251      $TADA dir.a             >>& $LOG
252      $TADA main.a            >>& $LOG
253      $TADA user.a            >>& $LOG
254      $TADA ooe.a             >>& $LOG
255      $TADA miscs.a           >>& $LOG
256      $TADA debug.a           >>& $LOG
257      $TADA dump.a            >>& $LOG
258      $TADA dspprt.a          >>& $LOG
259      $TADA smgmt.a           >>& $LOG
260      $TADA stntev.a          >>& $LOG
261      $TADA estnt.a           >>& $LOG
262      $TADA eistnt.a          >>& $LOG
263      $TADA stntecs.a         >>& $LOG
264      $TADA esubprm.a         >>& $LOG
265      $TADA eexpr.a           >>& $LOG
266      $TADA eattr.a           >>& $LOG
267      $TADA amain.a           >>& $LOG
268      $TADA set.a             >>& $LOG
```

```
269      $TADA create.a          >>& $LOG
270      $TADA get.a             >>& $LOG
271      $TADA stset.a           >>& $LOG
272      $TADA stget.a           >>& $LOG
273      $TADA help.a            >>& $LOG
274      $TADA semant.a          >>& $LOG
275      $TADA cexpr.a           >>& $LOG
276      $TADA cattr.a           >>& $LOG
277      $TADA csubpgm.a         >>& $LOG
278      $TADA csmisc.a          >>& $LOG
279      $TADA cmsppt.a          >>& $LOG
280
281  if ( $Xop == "n" ) then
282      $TADA system_environment.a >>& $LOG
283  endif
284
285  if ( $Caisop == "y" ) then
286      $TADA system.cais.a       >>& $LOG
287  else
288      $TADA system.a            >>& $LOG
289  endif
290
291      $TADA files.a            >>& $LOG
292      $TADA cstmt.a            >>& $LOG
293      $TADA drtns.a            >>& $LOG
294      $TADA denum.a            >>& $LOG
295      $TADA dreca.a            >>& $LOG
296      $TADA dpgk.a             >>& $LOG
297      $TADA dsubprg.a          >>& $LOG
298      $TADA error.a            >>& $LOG
299      $TADA strt.a             >>& $LOG
300      $TADA staddpk.a          >>& $LOG
301      $TADA initsym.a          >>& $LOG
302      $TADA ace_adt.a          >>& $LOG
303      $TADA dsubsup.a          >>& $LOG
304      $TADA drsup.a            >>& $LOG
305      $TADA x.a                >>& $LOG
306      $TADA lexact.a           >>& $LOG
307      $TADA yyerr.a            >>& $LOG
308      $TADA lex.a              >>& $LOG
309      $TADA misc.a             >>& $LOG
310      $TADA ctrees.a           >>& $LOG
311      $TADA common_parser.a     >>& $LOG
312      $TADA decl_part_shift_reduce.a >>& $LOG
313      $TADA decl_part_goto.a    >>& $LOG
```

```
314      $TADA decl_part.a          >>& $LOG
315      $TADA pre_parser.a          >>& $LOG
316      $TADA compilation_unit_shift_reduce.a >>& $LOG
317      $TADA compilation_unit_goto.a >>& $LOG
318      $TADA compilation_unit.a     >>& $LOG
319      $TADA seq_of_stmts_shift_reduce.a >>& $LOG
320      $TADA seq_of_stmts_goto.a     >>& $LOG
321      $TADA seq_of_stmts.a          >>& $LOG
322      $TADA calendar.a             >>& $LOG
323      $TADA main.a                 >>& $LOG
324      $TADA rwace.a                >>& $LOG
325      $TADA astd.a                 >>& $LOG
326      $TADA expand.a               >>& $LOG
327      $TADA aprgut.a              >>& $LOG
328      $TADA alloc.a               >>& $LOG
329      $TADA c infix.a             >>& $LOG
330      $TADA cdot.a                >>& $LOG
331      $TADA carray.a              >>& $LOG
332
333      if ( ( $Caisop == "y" ) && ( $Xop == "y" ) ) then
334          $TADA ebuilt.xcais.a      >>& $LOG
335      else if ( $Xop == "y" ) then
336          $TADA ebuilt.x.a          >>& $LOG
337      else if ( $Caisop == "y" ) then
338          $TADA ebuilt.cais.a       >>& $LOG
339      else
340          $TADA ebuilt.a            >>& $LOG
341      endif
342
343      $TADA einfix.a               >>& $LOG
344      $TADA euser.a               >>& $LOG
345      $TADA eprag.a               >>& $LOG
346      $TADA eblock.a              >>& $LOG
347      $TADA epkgb.a               >>& $LOG
348      $TADA dtyped.a              >>& $LOG
349
350      if ( $Xop == "y" ) then
351          $TADA onlyx.a             >>& $LOG
352      else
353          $TADA nowin.a            >>& $LOG
354      endif
355
356      $TADA eobject.a             >>& $LOG
357
358      if ( $Xop == "y" ) then
```

```
359      $TADA xadt.a                >>& $LOG
360  endif
361
362      $TADA smgms.a                >>& $LOG
363      $TADA ace_input.a           >>& $LOG
364
365  if ( $Xop == "y" ) then
366      $TADA ace_x_windows.a        >>& $LOG
367      $TADA ace_widget.a          >>& $LOG
368      $TADA ace_hp_widgets.a      >>& $LOG
369  endif
370
371  #
372  echo "Finally, the C subroutines"
373  echo ""
374  #
375  cd $ACE_CLIB
376      date                        >>& $LOG
377
378      $CC cflush.c                >>& $LOG
379
380      $CC sys_env.c               >>& $LOG
381
382  if ( $Xop == "y" ) then
383      $CC xwc.c                   >>& $LOG
384      $CC xws.c                   >>& $LOG
385  endif
386
387  #
388  echo "Linking the objects"
389  echo ""
390  #
391  cd $TARGET/src
392      date                        >>& $LOG
393
394  if ( $Xop == "y" ) then
395      $TALD -p '$LIBRARIES_WITH_X' -o ACE main >>& $LOG
396  else
397      $TALD -p '$LIBRARIES_NO_X $LIB_ARG' -o ACE main >>& $LOG
398  endif
399
400      date                        >>& $LOG
401
402  #
403  echo "Placing executable in $TARGET/bin"
```

25 October 1990

STARS-RC-00990/001/00

```
404 echo ""
405 #
406     mv -f ACE $TARGET/bin      >>& $LOG
407
408 #
409 echo "Build complete"
410 echo ""
```