(2)

···CONTROL DATA···

AD-A237 765

Ada Language System/Navy (ALS/N) For Embedded Real Time Systems

91-03985

THE Ada LANGUAGE SYSTEM/NAVY PROJECT

Abstract

The objective of the United States Navy's Ada Language System/ Navy (ALS/N) development is to provide a full production Ada capability for the Navy's AN/UYK-43, AN/UYK-44, and Pre-Planned Product Improvement (PPPI) AN/AYK-14. The ALS/N implements the Ada Programming Language as specified by MIL-STD-1815A and required by Department of Defense (DOD) Directives 3405.1 and 3405.2. This Ada program generation environment will improve programmer efficiency and reduce life cycle costs; while the Ada run-time environment will satisfy Navy-specific embedded application requirements such as performance, reliability, and fault-tolerance. The ALS/N Full-Scale Engineering Development (FSED) program supports the mandate or Ada for the Navy's standard embedded computers.

Introduction

The Navy's approach to providing Ada technology has been accomplished by separate but related developments. The first development has resulted in the early availability of a pilot production Ada capability for the AN/UYK-44, called Ada/M(44). This prototype included an Ada compiler, linker, importer, exporter and run-time operating system (consisting of an executive and library) for the AN/UYK-44. This development was completed in September 1986 and was one of the fastest Ada run-time environments at that time. Subsequent developments, that is the ALS/N FSED program, have resulted in a full production capabilities for the AN/UYK-43 (Ada/L), AN/UYK-44 (Ada/M), and PPPI AN/AYK-14 (Ada/M).

The ALS/N FSED program consists of the Ada/L, Ada/M and Programming Support Environment (PSE) developments which are being performed by multiple FSED contractors in two builds. The first build consists of the "single CPU" version of the AN/UYK-44 or PPPI AN/AYK-14 and "single/dual CPU" versions of the AN/UYK-43. The second build of the ALS/N FSED program adds multiprocessing, multiprogramming, and distributed Ada capabilities to the existing ALS/N products. The schedule is:

UYK-43 & 44 Ada validated
Dec 1988

AYK-14 Ada available
Mar 1989

Ada/L & Ada/M complete
Jun 1990

PSE complete
Sep 1990

Basic ALS/N delivered
Dec 1990

According to Navy and DOD directives, systems required to use the ALS/N are any Navy applications being implemented after the December, 1988 validation (i.e., new starts and any major software upgrades). The final delivery is being revalidated, but note that multiprocessing, multiprogramming, and distributed Ada are being incorporated into the products as part of maintenance releases.

Historical Perspective

The U.S. Navy has historically standardized Mission Critical Computer Resources (MCCR). Their AN/UYK-7, AN/UYK-20, and AN/AYK-14 Standard Embedded Computers have, for years, supplied the computing power for Navy mission critical systems. Almost all of the Navy software for these machines is written in one high-order language, CMS-2, and supported by one programming environment — Machine Transportable AN/ Support Software (MTASS). The newer standard embedded computers that evolved, the AN/UYK-43, AN/UYK-44, and PPPI AN/AYK-14, were in development about the same time as the DOD Standard High Order Language Ada was being adopted.

In keeping with their practice of standardization, the Navy put together a team from the Navy laboratories that represented

Navy users to specify requirements for an Ada program generation and run-time environment targeted to the new machines. The result of this team's efforts, led by Naval Sea Systems Command (NAVSEA) PMS-412, was the ALS/N System Specification. After the ALS/N FSED phase began with the awarding of a competitive contract to the Control Data Corporation/TRW/SYSCON team and a directed task to SofTech Inc., the support from Navy laboratories continued.

The same people that developed the ALS/N System Specification became the Navy Design Review Group (DRG) which reviews all contractual items and attends all the major design reviews. In particular, the Independent Verification and Validation (IV&V) agent for the ALS/N FSED was the Fleet Combat Direction Systems Support Activity, San Diego CA (FCDSSA, SD). Science Applications International Corporation (SAIC), Comsystems was the IV&V contractor for FCDSSA, SD. Government Furnished Information (GFI) support of the ALS/N Common Ada Baseline (i.e., VAX/VMS host) is being performed by the Naval Underwater Systems Center, Newport RI (NUSC, Npt).

When the Navy DRG was preparing the ALS/N System Specification they polled the existing major programs for requirements. The emphasis was on finding performance requirements for the ALS/N Run-Time Environment that would satisfy any existing Navy program and include a factor for growth into the 1990s. Shadow programs of selected functions from some of the major programs were initiated in the laboratories using the AN/UYK-44 prototype run-time operating system to familiarize the Navy's Ada team with its use and to further prove the usefulness of the ALS/N approach.

The performance of the AN/UYK-44 prototype run-time, written in Ada, was measured and compared with the ALS/N System Specification requirements, in conjunction with existing real-time embedded Standard Executives (SDEX) for CMS-2. The results showed the prototype to be one of the fastest Ada run-time operating system in existence at that time. These measurements gave the Navy's Ada team confidence that the ALS/N FSED versions will meet the ALS/N System Specification requirements and support the Navy's mission critical embedded systems through the year 2000.

ALS/N FSED Baseline

The ALS/N FSED Baseline was initially the Kernel Ada Programming Support Environment (KAPSE) based product that was developed by the U.S. Army's Ada Language System (ALS) and transferred to the U.S. Navy in late 1986/early 1987. Since the Navy's mission has always been oriented towards developing an Ada run-time environment for the Navy standard embedded computers with a full Ada Programming Support Environment (APSE) not being required, extensive changes to what is now called the ALS/N Common Ada Baseline (CAB) were initiated.

In particular, the Navy is only providing a Minimal APSE suitable for coding and testing (i.e., implementation phase), therefore we must look to other sources to provide tools/processes for all other development phases (e.g., specification and design). Cost estimates for incorporating a tool used in the ALS/N FSED, which is PSL/PSA, indicated that this would not be cost-effective if the Navy had to bear the burden alone. However, PSL/PSA is already hosted directly on VAX/VMS, as are a large number of other tools/processes that are commercially available.

In a separate, but closely related activity, the ALS/N FSED contractors had been utilizing a VAX/VMS hosted and targeted capability in their implementation of the retargets to Navy standard embedded computers. This capability, called the AdaVAX subsystem, had been the method by which the Ada compilation system for the Army's ALS was maintained between major releases. The FSED contractors requested that the AdaVAX subsystem be utilized for acceptance testing and delivery of the Navy retargets, thus officially changing the ALS/N FSED baseline.

The Navy investigated the ramifications of changing the ALS/N FSED baseline, agreed with the ALS/N FSED contractors' request for the reasons outlined above, and publicly announced this decision in July, 1987. As such, the ALS/N CAB consists of the AdaVAX subsystem and Program Library Manager (PLM) shown in figure (1). The ALS/N FSED products now consist of the Ada/M (AN/UYK-44 and AN/AYK-14), Ada/L (AN/UYK-43), and PSE subsystems being developed by the ALS/N FSED contractors and available in conjunction with the

ALS/N CAB for delivery. The additional benefits of this FSED baseline change in the ALS/N CAB to the Navy are as follows:

- Increased maintainability and transportability based on reduction in the amount of foreign code being utilized.
- Reduced maintenance and rehost costs because a complete layer of interfaces and associated tools have been removed.
- Continued concentration of the ALS/N FSED Contractors resources on the Ada run-time environment capabilities.

Functional Description

The ALS/N is basically a run-time environment for the target machines and a program generation environment currently hosted on the VAX that produces code for the target machines, plus some additional tools as shown by figure (2). The ALS/N was developed in accordance with DOD-STD-2167A, and DOD-STD-2168, designed to support large software development projects, and satisfy real-time embedded application requirements. Many features were added that are the direct result of the Navy DRG's knowledge of Navy mission critical system requirements, such as fast interrupt entries, precisely timed interrupts/delays, asynchronous Input/Output support, and fault-tolerant damage control.

Ada/L and Ada/M are the designations for the ALS/N subsystems that include Ada program generation and run-time environments for the 32-bit AN/UYK-43 and the 16-bit AN/AYK-14 and AN/UYK-44; respectively. A common specification and design approach was taken for the Ada/L and Ada/M subsystems, with the objective to implement a run-time operating system that is usable by many different Ada programs simply by tailoring the run-time executive and configuring the run-time library to the application. A standard run-time executive, in conjunction with an extensible run-time library, will reduce costs throughout the life cycle of any Navy application.

The architecture of the Navy standard computers present a challenge for the Ada language in that these machines all have a small virtual address space. The AN/UYK-44 and AN/AYK-14 can only address 64K words, and although the AN/UYK-43 can address 512K words using indirection and indexing, only 64K words can be directly addressed in a static manner. However, this problem of limited addressability was solved with two different approaches: (1) a software approach, called the "phase model," that is described below;

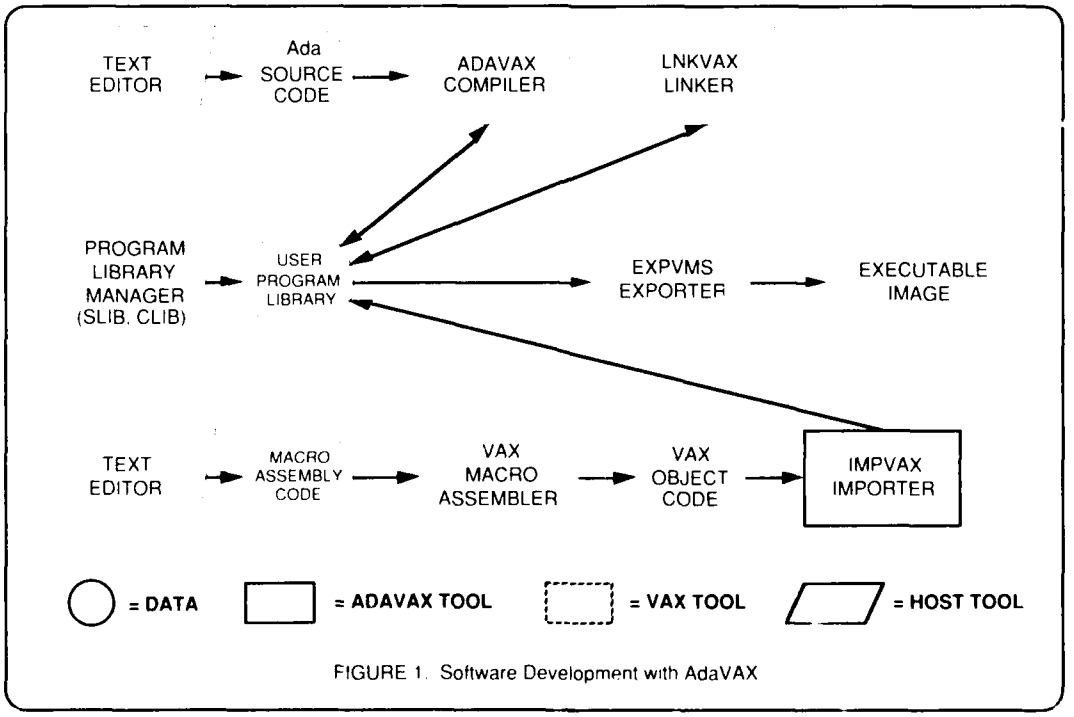


FIGURE 1. Software Development with AdaVAX

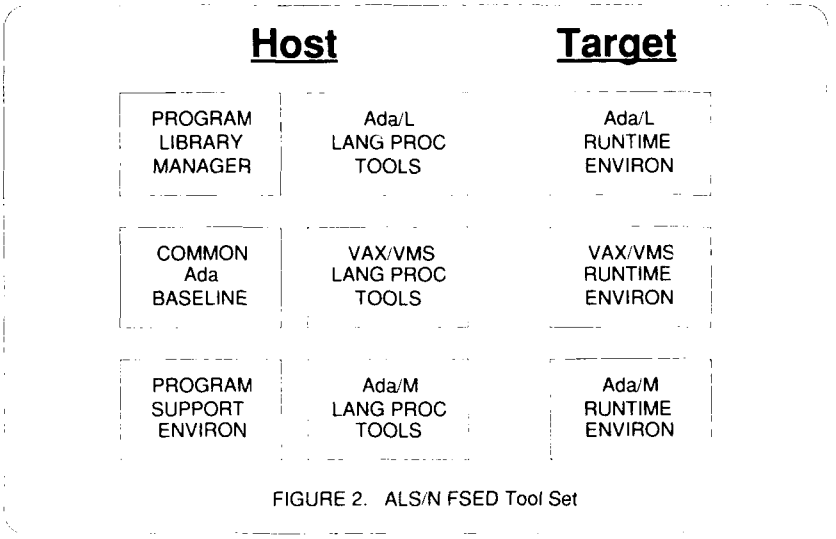


FIGURE 2. ALS/N FSED Tool Set

and (2) the development of Extended Memory Reach (EMR) versions of the AN/UYK-43 and the AN/UYK-44. These EMR machines support the phase model, but also provide specific hardware enhancements for "phasing" that includes multiple sets of memory mapping registers (called "memory groups") which are addressable in the task state (unprivileged) and, therefore, do not require executive service to be changed by the application.

Phase Model

An analytical model was developed that divides the total physical address space into a set of virtual address spaces, each called a "phase." A single phase is made of two address regions: a shared memory area, termed "phase common," and a "phase unique" portion. The Ada program can be specified and designed without regard to this "phase model". The application designer thus does not have to consider partitioning into phases as part of the design because the partitioning is performed during implementation. The Ada program is mapped onto this virtual "phased machine" as part of the linking process. The objective of the ALS/N approach is to create a program generation environment for software-first development that is independent of any underlying hardware.

Additionally, a stack model was developed to reduce overhead associated with "cross-phase" calls in the limited address space machines. Each task is assigned a separate stack, and the stack "travels" from phase to phase enabling cross-phase calls, cross-phase data reference, and cross-phase rendezvous with a minimum performance penalty. The user is able to specify which packages share phase space with other packages to optimize address space usage and efficiency of operation. This stack model is an elegant solution to the phase model implementation of Ada programs.

Input/Output Support

The Input/Output (I/O) supported within the ALS/N includes: low-level, synchronous, and asynchronous support for a variety of peripherals such as the AN/USQ-69 terminal, the AN/USH-26 and RD-358 tape drive, or the AN/UYK-3 disk drive. Support for MIL-STD-1553B, MIL-STD-1397 (NTDS) Fast/Slow Channel, and RS232 bus protocol has been provided. If the required I/O support is not available, a template device driver is provided as a model from which the user can develop application specific I/O support.

Fast Interrupt Entries

The Ada language provides for interrupt entries. An interrupt entry allows an Ada task to associate an entry with some external activity (e.g., a device) that will cause an interrupt. Interrupt entries, however, are not always capable of supporting real-time embedded systems due to the overhead imposed by Ada semantics. Fast Interrupt Entries (FIEs) allow efficient support for time critical interrupt processing and does not ignore Ada semantics. The concept of FIEs is to impose restrictions on the Ada features that are supported within the rendezvous so that the overhead associated with delivering the interrupt to the application can be significantly reduced. Also, by requiring the FIE to always be mapped (i.e., addressable by the memory mapping registers), additional savings in responding to an interrupt are possible.

Precisely Timed Interrupts/ Delays

Precisely Timed Interrupts (PTIs) provide a mechanism for performing time-based scheduling, entirely within the framework of

an Ada priority pre-emptive scheduler. PTIs consist of a logical collection of virtual timers that interrupt on a recurrent, nondrifting basis. These virtual timers, which are associated with underlying hardware timers, can be used to provide the functionality of cyclic schedulers, but with greater robustness and flexibility. Precisely Timed Delays (PTDs) utilize the Ada DELAY statement and also operate within the framework of an Ada priority pre-emptive scheduler. This facility depends upon delaying the task per the semantics of Ada, but immediately readying the task for execution at the expiration of the DELAY. The PTD provides for time-dependent behavior, while the PTI provides for time-critical behavior in the Ada program since tasks responding to interrupts execute at a priority higher than all other tasks.

Fault-Tolerant Damage Control

The ALS/N design makes damage control facilities available to application developers so that the Ada program can be apprised of hardware or software failures and take action that the application deems appropriate. For example, degraded mode operation after casualty reconfiguration. The ALS/N Run-Time Environment handles a hardware of software fault by isolating the event and notifying the Ada program through a Damage Control Task. It is the intent of the ALS/N implementation to expand the initial damage control facilities to perform certain types of fault-tolerant recovery actions for the user when failures occur.

Multiprogramming

The majority of the technical effort for implementing multiprogramming involves getting more than one program in execution within a single machine. Implementing the communications mechanism between programs, be they on the same, or a distributed node, is described as part of distributed Ada. The basic features of multiprogramming include the ability to create, delete, suspend, and resume a program (and all of its associated tasks) with a single-tiered scheduler where program priority is added to individual task priority to provide an overall system priority.

Unfortunately, multi-programming and FIEs are mutually exclusive capabilities in the baseline machines because FIEs are always placed in phase common and therefore always mapped to function. However, for the EMR processors, FIEs can still be implemented provided they are mapped in one of the memory groups.

Distributed Ada

For some time, there have been discussions about the lack of a basic "mailbox" facility in Ada. The classic paradigm of mailboxes describes a nonblocking message passing between the server and client tasks, provided that sufficient buffer space is available. Such a facility can be provided using normal Ada semantics, but there is a relatively high overhead associated with such an implementation. A mailbox facility is the form of asynchronous message passing utilized, with some limitations and using normal Ada semantics, to provide a basic communication mechanism for both distributed Ada and multiprogramming.

While the asynchronous message passing paradigm is the unifying mechanism to address communications between Ada programs whether on the same processor or distributed onto multiple processors, only the EMR processors can be supported because of addressability and performance requirements. The user is required to instantiate a generic package within each Ada program desiring to communicate in such a manner and guarantee that there is a proper correspondence between parameter types (there can be no type checking between different Ada programs). In all other respects, the communications appear to be the calling of an entry in one Ada program from another Ada program.

Minimal Ada Programming Support Environment

The ALS/N FSED phase is composed of two distinct efforts: the Minimal Ada Programming Support Environment (MAPSE) and the associated ALS/N Run-Time Environment. The MAPSE supports the development and deployment of real-time application software for the Navy's standard embedded computers. The major functions provided are: Ada compilation and linking/exporting facilities to generate applications for the target computers, interactive symbolic debugging and performance measurement facilities for execution analysis, general-purpose text editor and formatter, configuration management identification tools and report generator, a command language processor, and the associated environment database to support these functions.

The ALS/N language processing tools provide a full production capability to translate Ada source code into executable images for the AN/UYK-43, AN/UYK-44, or PPPI AN/AYK-14 targets and includes the EMR versions of these machines. These tools consist of compilers, linkers, exporters, importers, listing tools, stub generators, and the Program Library Manager (PLM) as shown by figure (3). The language processing tools support the generation of machine code to run in either task (unprivileged) mode or executive (privileged) mode as part of the Executive Option capability. The Executive Option capability allows the application developer to logically include portions of any Ada program, such as a specialized I/O driver, in the ALS/N Run-Time Environment.

Ada compilers for the ALS/N process Ada source program units by checking for correct Ada syntax/semantics and translating source code into linkable object code. These compilers support the full Ada Programming Language, MIL-STD-1815A, and their implementations include size representations, representation specifications, interrupt entry addresses, and low-level I/O support. In addition, implementation-defined PRAGMAs are recognized and processed, which provide applications with real-time capabilities. The ALS/N compilers share common baseline components for checking correctness of Ada code and for managing Ada program libraries, which contributes to a consistent and user-friendly interface to all ALS/N language processing tools.

The ALS/N linkers resolve external and internal references in the linkable object code to produce linked object code. Object code may be linked into phases, which are analogous to the target machine's virtual memory, and then into programs, which are analogous to the target machine's physical memory. The ALS/N linkers allow Ada programs to take full advantage of all physical memory resources in a target machine. The linkers also support the incremental linking of phases within Ada programs, thus reducing the overhead of rebuilding an application. As well as resolving external references, the linkers bind all required Run-Time Library support to the Ada program. Run-Time Library support may be selectively bound so that only the support that is actually needed by the application is included in an executable image.

Linked object code is processed by the ALS/N exporters to bind the Run-Time Executive with the application and perform a final layout of the object code into target memory. The exporters also implement a flexible mechanism allowing an application developer to take advantage of target machine resources (e.g., assignment of physical memory, I/O channels, etc.) for the application. The exported executable image is an Ada program that needs no further processing to run on a target machine or under a simulator. The exporters will produce a bootstrap image on tape or disk, which is acceptable to the target machine bootstrap loader; or a Target System Format file, which is acceptable to the MTASS simulators or Programmer Oriented AN/Link (PORTAL) products.

The ALS/N importers provide the capability of implementing Ada package and procedure bodies in assembly code. The importer will format the assembly implementation in a manner acceptable to the linker. It is a user responsibility to ensure that hand-written assembly code conforms to ALS/N Generated Code Conventions prior to importation. Additionally, a foreign code importer for CMS-2 is available to interface with the Ada program. This tool supports the reuse of existing Navy application code.

Human readable listings may be produced by invoking the listing tools. Compiler listings showing the translation of Ada code to object code, linker listings showing the layout of code into phases or programs, and exporter listings showing the final binding of code to machine locations are among the listings that can be produced. Finally, the stub generators will aid Ada developers by accepting compiled Ada package or procedure specifications and automatically producing stubbed-out Ada package or procedure bodies corresponding to the specifications.

The integrated set of tools and associated support within the Programming Support Environment (PSE) consists of:

- Command Language Processor - the user need only know one command language to access all the tools and move around in the hierarchical file system of the environment database.

- Configuration Management Tools - the user has powerful tools for version and source maintenance, software problem report tracking, documentation updating, and a report generator for displaying this information.

- Document Processing Tools - the user has powerful tools for general-purpose text editing (both line and screen) and formatting (including macro expansion).

- Environment Database Manager - the associated hierarchical file system, consisting of directories, files, attributes, and associations; in support of the tools outlined above.

Run-Time Environment

The Run-Time Environment (RTE) functions support the Navy's real-time embedded mission critical requirements and have been designed utilizing state-of-the-art computing techniques to deliver a reliable and efficient capability. The RTE developments have been performed in parallel with the associated MAPSE developments. The RTE consists of a standard executive and configurable library that can be tailored to meet application requirements, stand-alone and host/target interactive debuggers, performance measurement tools for execution analysis, and dynamic program loaders. The RTE can be separated into the Run-Time Operating System and Run-Time Application Support for the AN/UYK-43, AN/UYK-44 or PPPI AN/AYK-14 and includes the EMR versions of these machines.

Run-Time Operating System

The Run-Time Operating System (RTOS) provides the Run-Time Executive, the Navy's standard executive for the Ada programming language, which can be tailored to each Navy application. The RTOS also provides the Run-Time Library, which is configurable and extendable for the Ada program. RTOS capabilities include multiprocessing, multiprogramming, distributed Ada, and provisions to have user-supplied executive code (i.e., Executive Option user routines) for specific application needs. Note that the Executive Option capability is the basis of the Run-Time Operating System implementation.

Run-Time Executive

The Run-Time Executive (RTExec) runs in the executive state (privileged and protected) and controls all of the hardware resources of the AN/UYK-43, the AN/UYK-44, and the PPPI AN/AYK-14. The RTExec responds to either Run-Time Library requests for an executive service or requests from the user-supplied Executive Option routines. These Executive Service

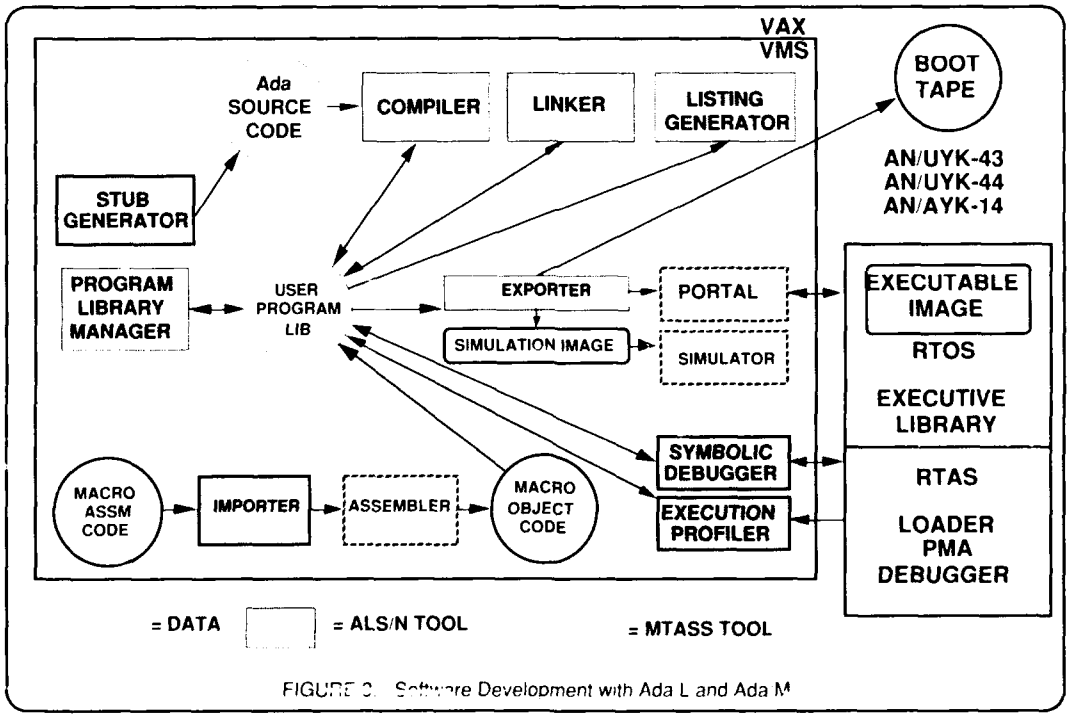


FIGURE 2. Software Development with Ada L and Ada M

Requests (ESRs) or direct requests from Executive Option routines are presented to the RTExec through the RTEXEC_GATEWAY by the ESR Interrupt, with a corresponding ESR code. When the requested action is completed, control is returned via the normal interrupt return mechanism of the target computer (i.e., reloading the saved general purpose registers, the program status registers, and the program counter).

Run-Time Library

The Run-Time Library (RTLib) runs in the task state (unprivileged and unprotected) and responds to all Ada program requests, except for Executive Option user requests. The RTLib either acts upon the request itself or directs the request to the RTExec through the RTEXEC_GATEWAY. Note that user-supplied Executive Option routines have the capability to either utilize the features of Ada normally through the RTLib or go directly to the RTExec and RTLib interfaces when required by the application. The Executive Option capability is the foundation on which all of the Run-Time Application Support functionality is implemented.

Run-Time Application Support

The Run-Time Application Support (RTAS) augments the Run-Time Operating System with capabilities to enhance the development and deployment of application programs as shown by figure (4). The RTAS consists of a Run-Time Loader (RTLoad), Run-Time Debugger (RTDebug) that cooperates with the Embedded Target Debugger, and Run-Time Performance Measurement Aids (RTAids) that operates in conjunction with the Embedded Target Profiler.

Run-Time Loader

Dynamic, static, and overlay loading is supported by the Run-Time Loader. In conjunction with the RTOS, the RTLoad provides applications with the capability to efficiently manage the resources of the target computer, support of run-time assignment of physical memory locations to load modules, and includes a mechanism for dynamic reconfiguration of an application to accommodate changes in the computing environment.

Run-Time Debugger

Access to all hardware resources of the AN/UYK-43, AN/UYK-44 and PPPI AN/AYK-14 not restricted to the RTOS is provided by the Run-Time Debugger. Operating completely within the target computer and requiring only a terminal device, the RTDebug equips the application developer with a powerful interactive tool for Ada program checkout. Facilities provided within the RTDebug include: setting and clearing breakpoints, examining and modifying registers or memory, blocking and unblocking tasks, and tracing call stacks. In addition, the Run-Time Debugger supports the host/target relationship with the Embedded Target Debugger.

Embedded Target Debugger

Symbolic access to all code and data elements of an executing Ada program is available through the Embedded Target Debugger. This user-oriented symbolic debugging tool is designed to support efficient

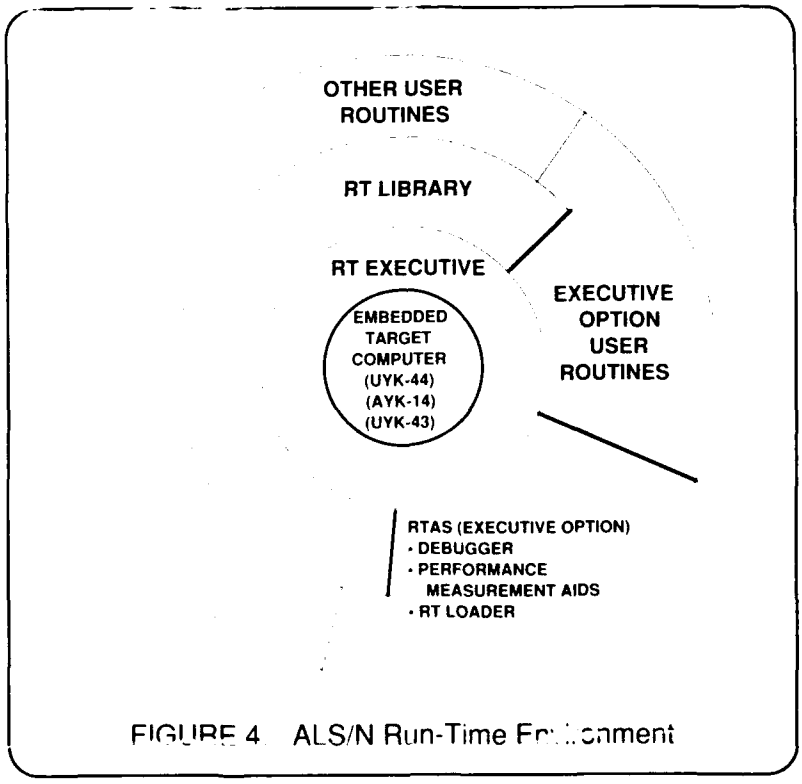


FIGURE 4 ALS/N Run-Time Environment

application development in a laboratory environment. Reference to source code and data through the symbols defined in the Ada program isolates users from the details of the Ada language implementation, thus allowing the developer to more productively debug applications. Furthermore, resources of the host and target are utilized in a hardware testbed to ensure an efficient and reliable operation of the Embedded Target Debugger.

Run-Time Performance Measurement Aids

Detailed run-time performance analysis of application programs is performed through use of the Run-Time Performance Measurement Aids and Embedded Target Profiler. The RTAids works in concert with the RTOS to collect data characterizing the performance of an executing program. Collected data is then processed at a later time by the Embedded Target Profiler producing histograms or "profiles" of the application's performance characteristics.

Conclusions

The ALS/N FSED program represents a multi-million dollar government investment managed by the Naval Sea Systems Command (NAVSEA) PMS-412 in a Minimal Ada Program Support Environment and ALS/N Run-Time Environment to implement Ada and meet the goals set for it by the Department of Navy. The ALS/N is designed to reduce Software Life Cycle Maintenance (SLCM) costs and to support Mission Critical Computer Resources. The requirements set for the ALS/N insure that these goals are realized and include factors for performance that will reflect the challenge of the 1990s.

The competitive ALS/N FSED Contract for Ada/L, PSE tools, and integration of Ada/M into the ALS/N was awarded to the Control Data /TRW/SYSCON team. The Ada/M FSED was developed as a directed task to SofTech, Inc. and continued development is being performed by Control Data. The SLCM agent for ALS/N FSED is Fleet Combat Direction System Support Activity, San Diego (FCDSSA, SD) and Science Applications International Corp., Comsystems is providing SLCM Testing & Evaluation under a competitively awarded omnibus contract. FCDSSA, SD is responsible for distributing releases to all ALS/N users, collecting Software Problem/ Change Reports as they are submitted and providing corrections (and updates) to the ALS/N users. It is the goal of NAVSEA PMS-412 to make the ALS/N FSED products available to all Navy application developers at no charge.

The ALS/N project is the current phase of Navy Ada development for the AN/UYK-43, AN/UYK-44, or Pre-Planned Product Improvement (PPPI) AN/AYK-14 and includes the EMR versions of these machines. This phase began in September 1985 and will be completed by December 1990. (Note that enhancements to the ALS/N in support of the PPPI for the AN/UYK-43 and AN/UYK-44, started in September 1989 and will be finished by December 1992.) Its evolution has been:

- The Army's ALS -- target independent portions have been retained as appropriate to the ALS/N; rights to products are co-owned by the Government and SofTech; commercial work has been retrofitted into the VAX/VMS host and target; the Government continues to improve the ALS/N CAB.

- The Ada/M(44) prototype -- retarget to AN/UYK-44 with near production quality compiler was developed; prototype for ALS/N run-time capabilities targeted to embedded systems (e.g., memory management, precisely timed delays, I/O subsystem) was developed; the prototype run-time was written in the Ada language itself and was one of the fastest Ada run-time operating systems available.

- The ALS/N FSED -- production-quality compilers for AN/UYK-43, AN/UYK-44, and PPPI AN/AYK-14; all issues raised by Ada/M(44) prototype development (i.e., compiler code quality, RTOS algorithms, linker/exporter tools, etc.), are resolved; PSE tools and Embedded Target Debuggers/Profilers are provided.

Based on the sizing and timing test results gathered by the ALS/N FSED contractor team and the IV&V contractor, all ALS/N System Specification performance requirements are achievable. Additionally, an independent assessment of the ALS/N has reported performance information comparable to the ALS/N test results and has provided many valuable insights from a user perspective to improve the ALS/N CAB.

Ada/L, Ada/M, and PSE subsystems.

Major enhancements to the ALS/N will be made to support PPPI of the standard embedded computers, which are being upgraded to meet emergent Navy requirements. The improvements are the Enhanced Processor (EP) AN/UYK-44, the High Performance Processor (HPP) AN/UYK-43, and the Very High Speed Integrated Circuit (VHSIC) AN/AYK-14. The EP and HPP instruction set architecture permits direct addressing of all memory in task state and provides for flexible protection mechanisms. These features make the EP and HPP machines better targets for execution of Ada programs within the ALS/N implementation. The EP and HPP improvements will also provide for faster execution than the current machines (5-20 MIPS versus 1-4 MIPS). The VHSIC improvement provides faster processors for the AN/AYK-14 (5-10 MIPS versus 1-2 MIPS) and can be configured as two independent processors with 1 MByte of onboard memory.

The ALS/N has been specified and designed for portability. The first scheduled rehosts will be to the Extended Memory Reach (EMR) AN/UYK-43, under SHARE/43, and VAX, under Portable Operating System for UNIX (POSIX). The rehost to the EMR AN/UYK-43, running SHARE/43, will support Navy users throughout the extended life of their applications by using the EMR AN/UYK-43 as both their host (for development) and target computer. The rehost to VAX, running POSIX, will allow integration of ALS/N products with other Government and commercial software development environments.

In summary, the ALS/N is a Navy user-driven system designed and implemented to support real-time embedded mission critical applications. The flexible ALS/N Run-Time Environment allows tailoring by diverse mission critical computer systems thus saving costs in the development phase of the life cycle. The portable Minimal Ada Programming Support Environment is available to reduce costs in the maintenance phase of the software life cycle. The requirements for the ALS/N provided by the Navy Design Review Group have taken into account future needs of the Navy's embedded Ada applications. NAVSEA and Control Data Corporation, the prime contractor for ALS/N, have developed the technology for a program generation and run-time environment that will support Ada in real-time embedded mission critical systems through the year 2000.

References

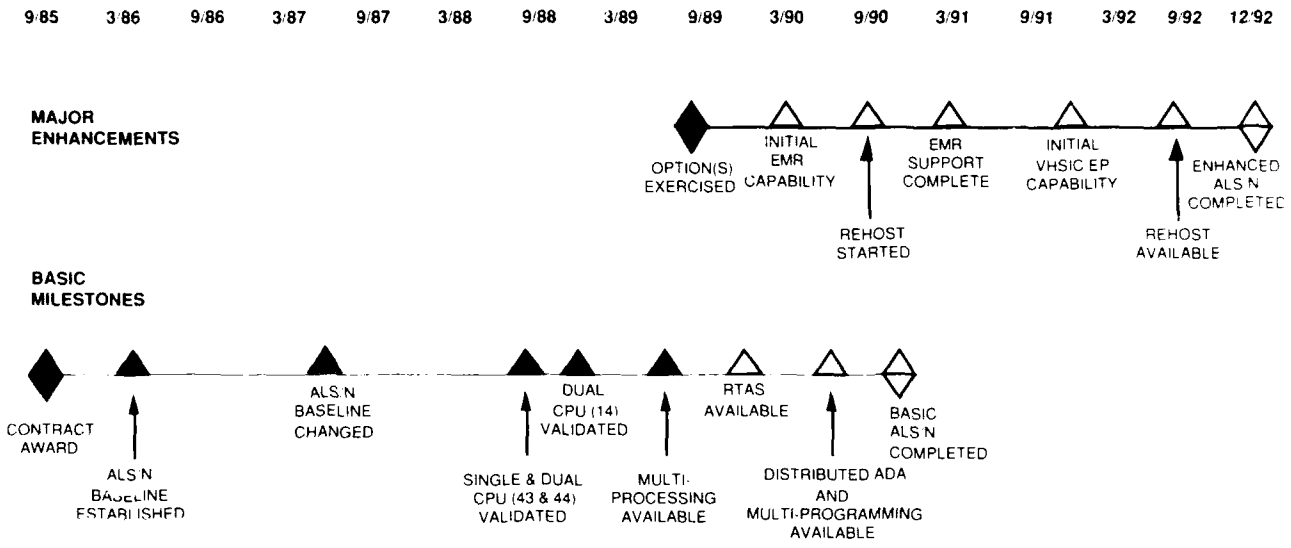
[1815A] Reference Manual for the Ada Programming Language, ANSI/MIL-STD-1815A, Department of Defense, January, 1983.

[SPEC88] Ada Language System/Navy System Specification, NAVSEA, 29 July 1988.

[2167A] Defense System Software Development, DOD-STD-2167A, Department of Defense, 29 February 1988.

[2168] Defense System Software Quality Program, DOD-STD-2168, Department of Defense, 29 April 1988.

This paper was presented at the eigth annual conference on Ada Technology, March 1990, in Atlanta Georgia and is reprinted here with the permission of the author, William L. Wilder of Naval Sea Systems Command, PMS-412.



Overall ALS/N FSED Milestones

For more information, please write or call:

Naval Sea Systems Command PMS 412
Department of the Navy
Washington, DC 20362-5101

OR

Control Data Government Systems Resource Center
8140 26th Avenue South
Minneapolis, MN 55425
(612) 853-5000

Control Data offices are located in principal cities throughout the world.

Specifications subject to change without notice.