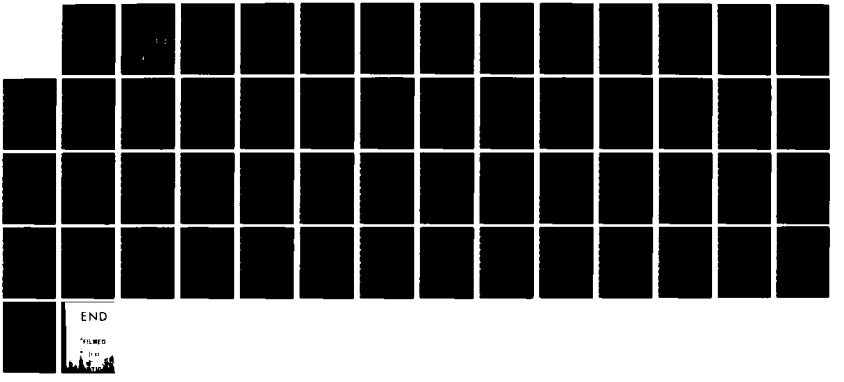
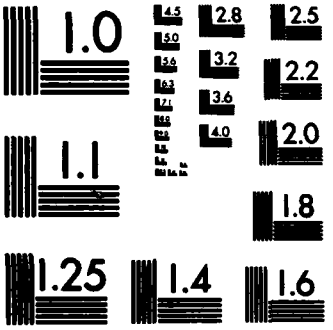
AD-A134 080 SYSTEM SPECIFICATION FOR ADA INTEGRATED ENVIRONMENT TYPE A AIE(1)(U) INTERMETRICS INC CAMBRIDGE MA 12 NOV 82 IR-676-2 F30602-80-C-0291 1/1

UNCLASSIFIED F/G 9/2 NL



END

FILMED



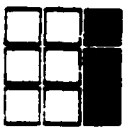
IR-676-2
SYSTEM SPECIFICATION
FOR
Ada INTEGRATED ENVIRONMENT
TYPE A
AIE(1)

12 NOVEMBER 1982

CONTRACT F30602-80-C-0291

PREPARED FOR: ROME AIR DEVELOPMENT CENTER
CONTRACTING DIVISION/PKRD
GRIFFISS AFB, N.Y. 13441

PREPARED BY:



INTERMETRICS, INC.
733 CONCORD AVE.
CAMBRIDGE, MA 02138

This document was produced under Contract F30602-80-C-0291 for the Rome Air Development Center. Mr. Donald Mark is the Program Engineer for the Air Force. Mr. Mike Ryer is the Project Manager for Intermetrics.

TABLE OF CONTENTS

INTERMETRICS INCORPORATED • 733 CONCORD AVENUE • CAMBRIDGE, MASSACHUSETTS 02138 • (617) 661-1840

TABLE OF CONTENTS (Cont'd.)

TABLE OF CONTENTS (Cont'd.)

TABLE OF CONTENTS (Cont'd.)

LIST OF FIGURES

INTERMETRICS INCORPORATED • 733 CONCORD AVENUE • CAMBRIDGE, MASSACHUSETTS 02138 • (617) 661-1840

1.0 SCOPE

This specification establishes the performance, design, development and test requirements for the Ada Integrated Environment (AIE), an integrated set of software tools designed to support the development and maintenance of software written in the Ada Programming Language.

INTERMETRICS INCORPORATED • 733 CONCORD AVENUE • CAMBRIDGE, MASSACHUSETTS 02138 • (617) 661-1840

LEFT BLANK INTENTIONALLY

2.0 APPLICABLE DOCUMENTS

2.1 Program Definition Documents

Requirements for Ada Programming Support Environments, "STONEMAN", February 1980, Dept. of Defense.

Revised Statement of Work, 15 March 1980.

Reference Manual for the Ada Programming Language, proposed standard document, U.S. Department of Defense, July 1982.

Formal Definition of the Ada Programming Language, Preliminary Version for Public Review, November 1980, U.S. Government as represented by the Director, Defense Advanced Research Projects Agency.

2.2 Inter Subsystem Specifications

Computer Program Development Specifications for Ada Integrated Environment (Type B5), March 1981:

Ada Compiler Phases, AIE(1).COMP(1).

KAPSE/Database, AIE(1).KAPSE(1).

MAPSE Command Processor, AIE(1).MCP(1).

MAPSE Generation and Support, AIE(1).MGS(1).

Program Integration Facilities, AIE(1).PIF(1).

MAPSE Debugging Facilities, AIE(1).DBUG(1).

MAPSE Text Editor, AIE(1).TXED(1).

Virtual Memory Methodology, AIE(1).VMM(2).

Technical Report (Interim).

2.3 Military Specifications and Standards

Data Item Description, DI-E-3101.

2.4 Miscellaneous Documents

Diana Reference Manual, G. Goos and Wm. Wulf, Institute Fuer Automatik II, Universitaet Karlsruhe and Computer Science Dept., Carnegie-Mellon University, March 1981.

IBM Virtual Machine Facility/370: System Programmer Guide, IBM, Inc.

OS/32 Programmer Reference Manual, Perkin-Elmer, Computer Systems Division, Oceanport, N.J., April 1979.

Intermetrics LG System Description, 31 August, 1980, IR-536.

LG User's Guide, December 1979, IR-427.

3.0 REQUIREMENTS

3.1 System Definition

The Ada Integrated Environment (AIE) is an integrated set of software (and hardware) tools designed to fulfill the requirements for a standard Ada Programming Support Environment (APSE) as outlined in the "STONEMAN" document. It consists of a Kernel Ada Programming Support Environment (KAPSE), which encapsulates machine dependencies, and a set of portable tools, referred to as the MAPSE (Minimal Ada Programming Support Environment) tools. While the set of MAPSE tools provide comprehensive support for the development of Ada software, they should be viewed as the basis for a larger set of tools that will, in the future, provide a richer environment. With this in mind, extensibility is a motivating factor throughout the AIE design.

3.1.1 General Description

The KAPSE portion of the AIE plays two major roles. One, it provides a standard interface between MAPSE tools and the host system(s) on which they will operate. The initial version specifies an IBM 4341 as the initial host with a rehosting to the PE 8/32. Secondly, the KAPSE supports a centralized database facility that will contain all user data and programs, serving as the primary medium for tool-to-tool communication and coordination.

The MAPSE tools being implemented for this initial AIE provide all the necessary facilities for the intended application - developing embedded computer software in Ada. This portion of the AIE consists of seven major subsystems:

COMPILER (COMP)
TEXT EDITOR (TXED)
DEBUGGER (DBUG)
MAPSE COMMAND PROCESSOR (MCP)
PROGRAM INTEGRATION FACILITY (PIF)
VIRTUAL MEMORY METHODOLOGY (VMM)
MAPSE GENERATION AND SUPPORT (MSG)

See Appendix A for a complete list of AIE configuration items.

3.1.1.1 Kernel Ada Programming Support Environment (KAPSE)

The Kernel Ada Programming Support Environment (KAPSE) provides a machine-independent portability interface as well as database communications, program control, and run-time support functions. The KAPSE facilitates portability by isolating dependencies and by providing functional interfaces and capabilities to software tools and users.

The KAPSE database facilities provide for the construction, organization, and partitioning of large configurations of inter-related program, data, and documentation elements. It records the nature and purpose of these elements and allows for access control and synchronization. Finally, these facilities maintain historical information about each object with sufficient indices to fully reconstruct source text from disk or archival storage.

Configuration reporting and management are not separable from the rest of the KAPSE database facilities, but are rather integral to the reporting and management of attributes and partitions. The KAPSE primitives particularly relevant to configuration management are described in AIE(1).KAPSE(1). There are, in addition, a prototype set of standard MAPSE tools that exemplify the use of KAPSE facilities for this application. These are given in AIE(1).KAPSE(1) (Section 3.3.4.3).

3.1.1.2 MAPSE Command Processor (MCP)

The MCP is the primary user interface to the AIE, coordinating the execution of MAPSE tools. User commands to the MCP may be issued interactively or collected into "scripts" for subsequent (batch mode) execution.

3.1.1.3 Program Integration Facilities (PIF)

PIF includes program library support tools and the linker. The program library is the means by which the AIE supports independent, modular program development. To facilitate separate compilation, PIF keeps track of dependencies among subprograms so that the appropriate source units are recompiled when a related unit is changed. The linker combines compilation units created by the compiler into a single executable program.

3.1.1.4 Ada Compiler (COMP)

The MAPSE Ada compiler processes the complete Ada Language as defined in the LRM and provides a range of optimization levels that are user selectable. To support rehostability and retargetability, the compilation process is broken down into three major parts: (1) the Front End, which performs syntactic and semantic analysis, creating a standard intermediate representation of a program (in DIANA); (2) the Middle Part, which produces a lower-level representation (BILL) that expresses the run-time model and performs Ada-specific optimizations; and (3) the Back End, which produces optimized, linkable target code.

3.1.1.5 Text Editor (TXED)

TXED is used to create and revise text files, such as Ada program source code, documentation, and user command scripts. It may be used in either interactive or batch mode.

INTERMETRICS INCORPORATED • 733 CONCORD AVENUE • CAMBRIDGE, MASSACHUSETTS 02138 • (617) 661-1840

3.1.1.6 Debugging Facilities (DBUG)

DBUG provides symbolic debugging support in either batch or interactive mode. The program to be tested is executed under DBUG control. At user-specified breakpoints, execution is suspended and a variety of program objects may be examined. In interactive usage, objects may be temporarily modified so that resumed execution can determine whether a potential change will correct a detected error. The design of DBUG includes facilities for expanding existing features to support advanced simulation/emulation techniques.

3.1.1.7 Virtual Memory Methodology (VMM)

VMM is a tool for creating and manipulating abstract data structures in a machine-independent manner. A virtual memory paging scheme makes the size of any data structure independent of the constraints of any particular hardware configuration.

3.1.1.8 MAPSE Generation and Support (MGS)

The MAPSE Generation and Support facility is a collection of programs, routines, and/or data that support the construction and maintenance of the AIE itself. These include: (1) the bootstrap facility; (2) the parser and lexer generators; and (3) routines to facilitate rehost of the AIE.

3.1.2 Mission

The overall objective of an AIE is to provide a standardized environment for Ada programmers so that their experience is portable from one hardware configuration to another. This environment must provide comprehensive support for program development and maintenance by providing basic tools as well as the capability for creating new tools as needed. The environment must encourage modular programming and must support the integration of independently developed modules. To provide adequate life cycle support, the AIE must maintain a history of program development that is accessible to both technical and managerial personnel. Software produced in this environment must be retargetable and rehostable to keep pace with hardware changes and to promote the reuse of existing software.

This prototype AIE aims to meet all the requirements stated above in terms of its basic design. The KAPSE will be first implemented on an IBM 4341 and then rehosted to a PE 8/32. Future AIE's will, it is expected, be rehosted in the same manner. The MAPSE provides the minimal set of tools required.

3.1.3 Threat

Not applicable.

3.1.4 System Diagrams

Figures 1 and 2 provide, respectively, a system overview and a high-level flow diagram for the AIE. Figure 3 shows interfaces between MAPSE tools and KAPSE database objects.

3.1.5 Interface Definition

The three major interfaces within the AIE are: KAPSE/host, MAPSE tool/KAPSE, and tool/tool.

3.1.5.1 KAPSE/Host Interface -- VM/SP and OS/32

The KAPSE is designed to be as host-independent as possible. This is accomplished by defining KAPSE/Host interface packages which together provide a uniform interface from one host to the next for the rest of the KAPSE. To rehost the AIE, only the bodies of these packages will need rewriting. Any embedded machine code is also restricted to the KAPSE/Host interface packages.

The KAPSE/Host interface packages, detailed in AIE(1).KAPSE(1) are as follows:

(KAPSE.SIMPCOMP) :

Package PHYS_BLOCK_IO
Package DEVICE_IO

(KAPSE.MULTPROG) :

Package PROGRAM_LOADING
Package KAPSE_TOOL_COMMUNICATION
Package KAPSE_KAPSE_COMMUNICATION

(KAPSE.RTS) :

(Most of it, except the "Language-Defined Packages" CPC)

The overall architecture provided by the KAPSE/Host interface packages, using whatever host facilities are appropriate, is a number of independently executing Ada programs running concurrently on the host machine. Each independent Ada program has its own run-time system, including an Ada task scheduler.

The KAPSE/Host interface packages implement (with help from the host) device drivers, as well as the loading, timesharing, memory management, and swapping of the independent programs. The KAPSE/Host interface packages also provide a low-level communication path between the KAPSE and each user program (package KAPSE_TOOL_COMMUNICATION), and between two KAPSEs on separate (virtual) machines (package KAPSE_KAPSE_COMMUNICATION).

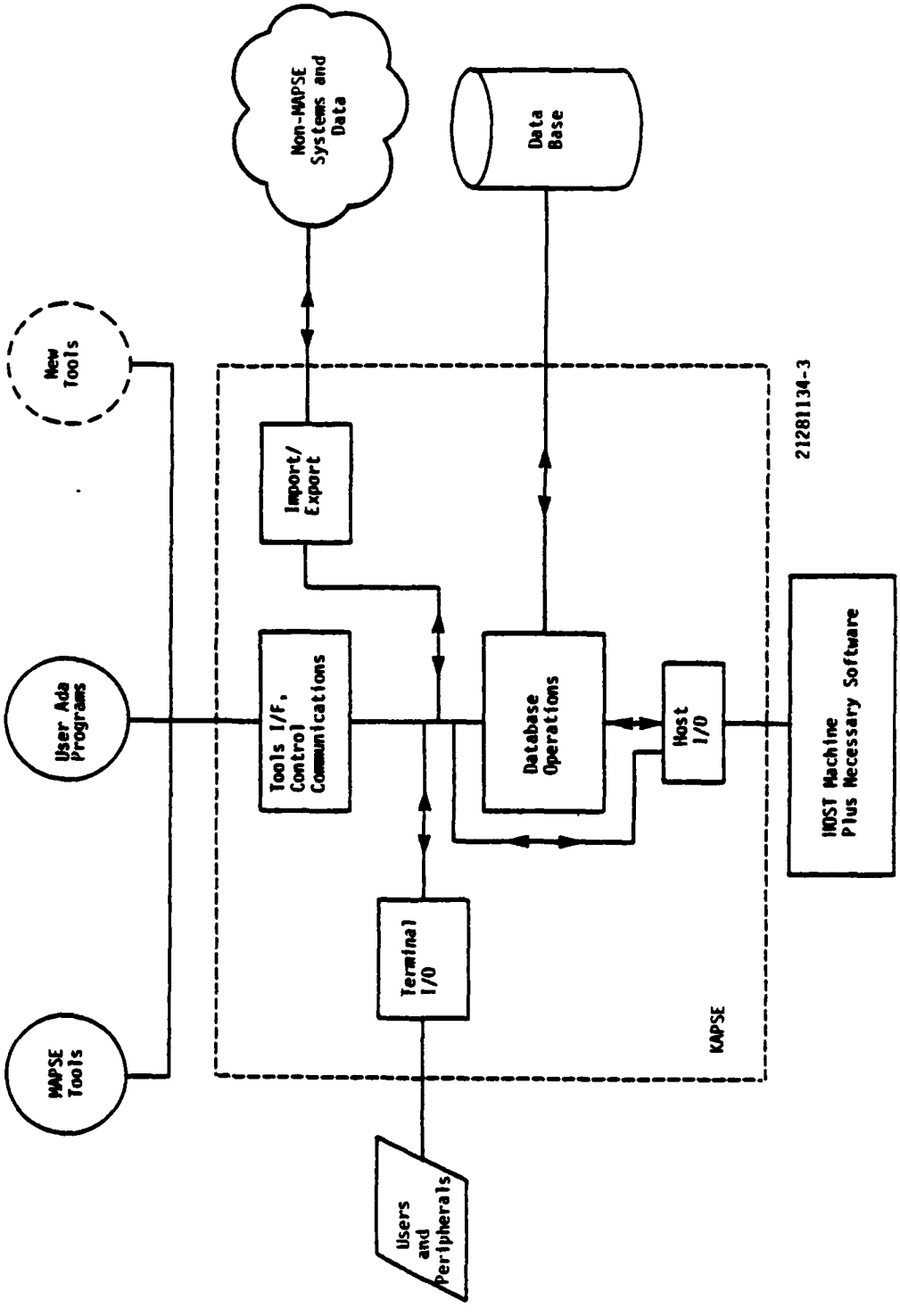


FIGURE 1: MAPSE OVERVIEW

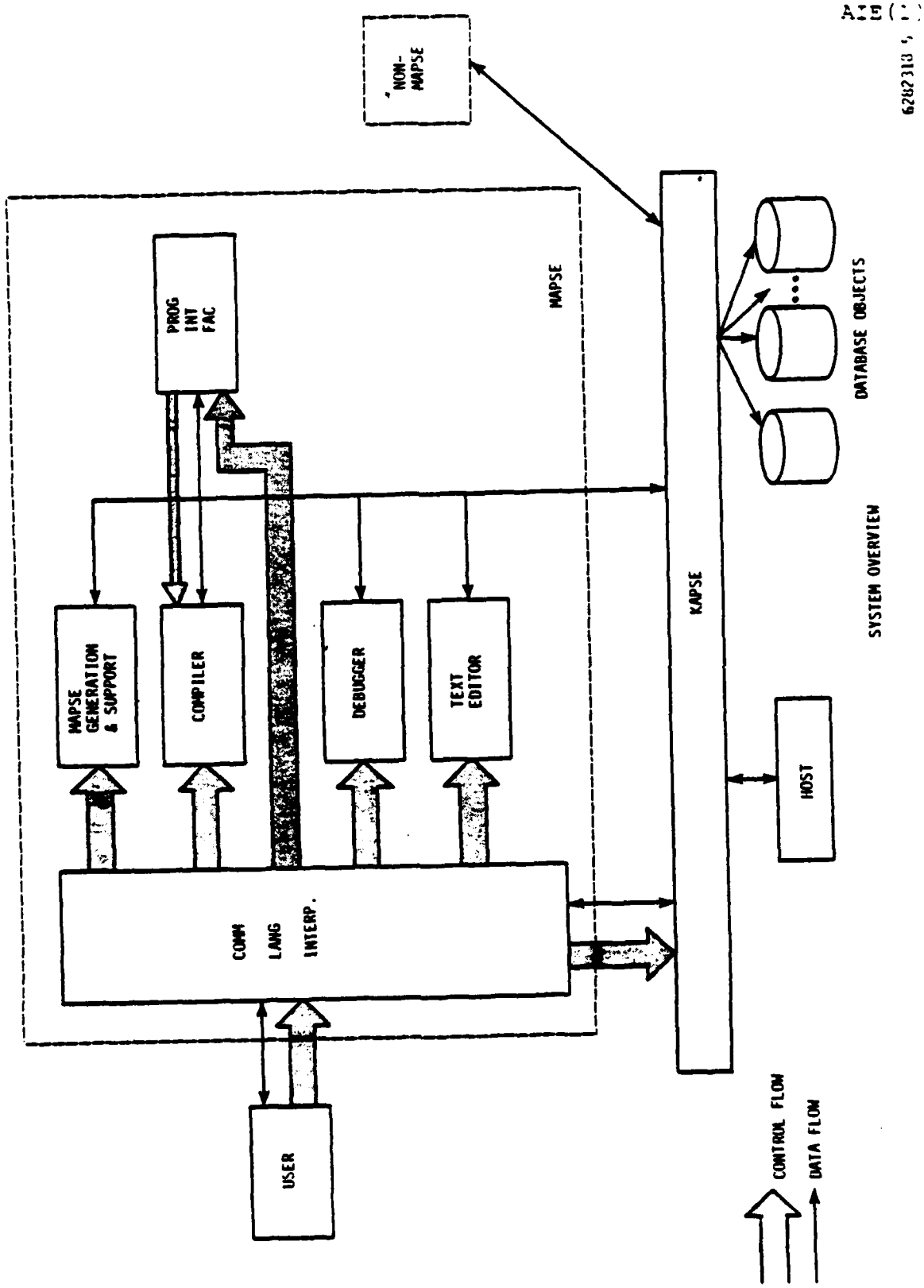


FIGURE 2: SYSTEM DIAGRAM

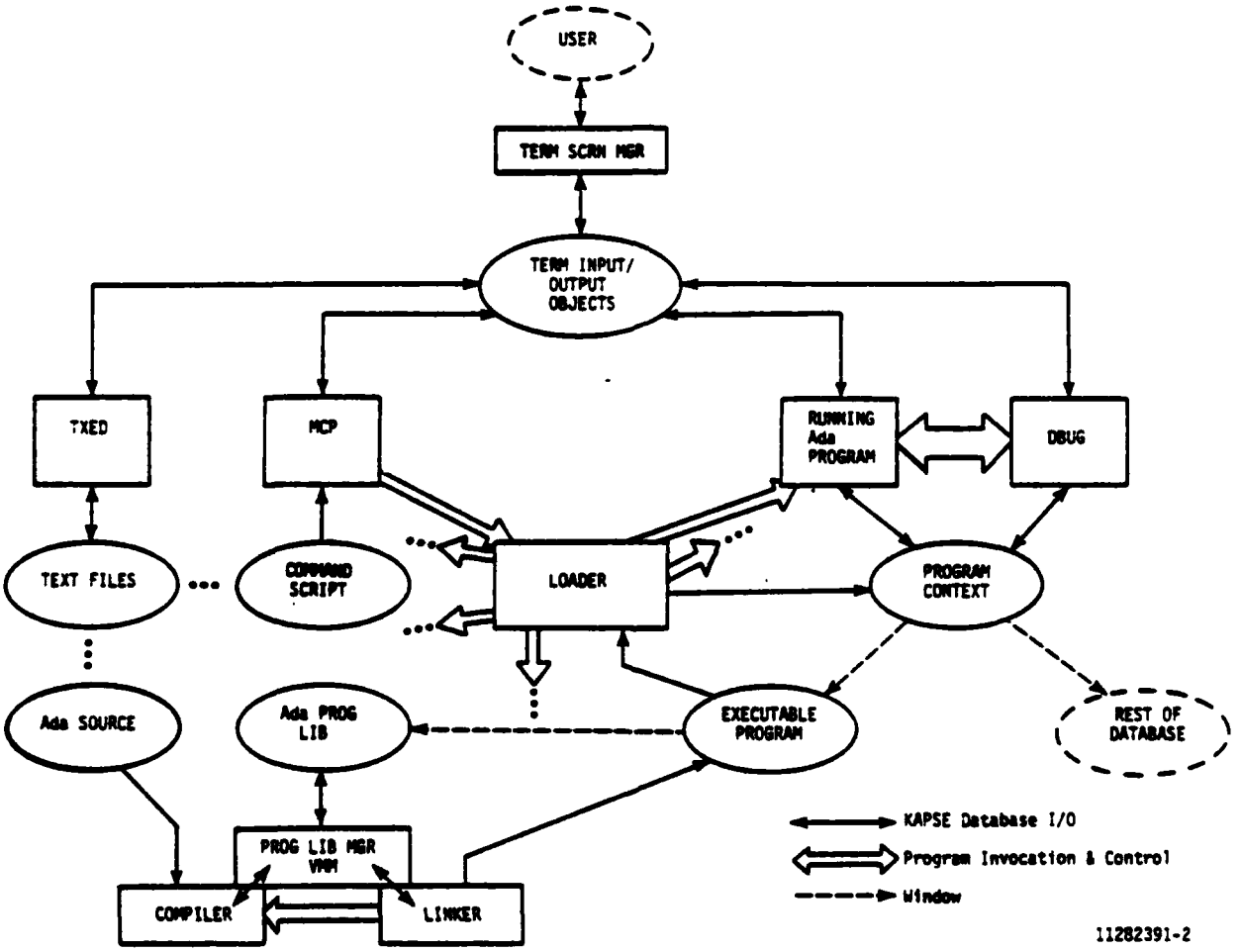


FIGURE 3: MAPSE TOOL/DATABASE INTERFACES

The KAPSE itself is an Ada program, using a specialized version of the Ada run-time system lacking the high-level IO packages, and supporting the connection of entries of tasks within the KAPSE/Host interface packages to real hardware interrupts.

The communication path from the user Ada programs to the KAPSE is analogous to a "system call" of SVC. Except for this communication path, the KAPSE/Host interface packages entirely isolate the user Ada programs from one another, from the KAPSE, and from the host (using hardware protection where possible).

In summary, the KAPSE/Host interface packages insulate the KAPSE from the idiosyncrasies of the host system facilities. The rest of the KAPSE, in turn, implements the high-level KAPSE/Tool interfaces in terms of these low-level host-independent interfaces. In addition, the KAPSE/Host interface packages prevent the application programs running under the KAPSE from accessing the host facilities directly, thus ensuring that the KAPSE Database is not contaminated.

3.1.5.1.1 IBM VM/SP

This overall logical architecture is implemented on top of the VM/SP system using multiple virtual machines (see IBM VM/SP documentation), each with its own KAPSE program. A particular user is allowed to DIAL into one of the running virtual machines, or to IPL his own (or his project's) if it is not already running. After IPL, the KAPSE begins running and spawns multiple LOGIN programs within its virtual machine to handle the terminal associated with the IPL, and each terminal which connects later via DIAL.

After connecting via IPL or DIAL, the user must LOGIN by providing a user name and password. If accepted, LOGIN then invokes the command processor identified in the user's initial program context. The additional programs initiated by the command processor will share the same virtual machine with the KAPSE and those of other simultaneous users of the virtual machine. The multiple programs within a single virtual machine are managed by the KAPSE/Host interface package.

To provide access to each other's databases, multiple KAPSEs on the same physical machine may communicate via one of the KAPSE/Host interface packages. This communication between virtual machines is implemented using the Virtual Machine Communication Facility (VMCF) or the Inter-User Communication Vehicle (IUCV), both of which are high-band-width memory-to-memory data paths provided by the VM/SP Control Program.

3.1.5.1.2 Perkin-Elmer OS/32

The same overall logical architecture is implemented differently on top of the OS/32 system by placing each independently executing Ada program in its own OS/32 task. User programs execute in a mode whereby the only OS/32 SVC 6 system calls they can perform are inter-task communication. They are not permitted to directly stop, start, or otherwise interfere with other tasks (NOCON mode -- see OS/32 Programmer's Manual).

The KAPSE runs in its own OS/32 task without NOCON mode, allowing the KAPSE/Host interface packages access to all OS/32 system calls. They initiate all physical I/O, including terminal and disk, and thereby can optimize physical disk access and provide a central buffer cache. All OS/32 tasks communicate using the standard OS/32 inter-task communication primitives, a memory-to-memory queue-based data path (see OS/32 Programmer's Manual).

3.1.5.2 MAPSE Tools/KAPSE

MAPSE tools interface with the KAPSE by instantiating the appropriate package specifications. KAPSE tool interfaces fall into four categories:

(a) Program Invocation and Control. These packages are used by tools that must explicitly invoke other tools. These are: the compiler (COMP), the linker (PIF), the command processor (MCP), and the debugger (DBUG).

(b) File I/O. These packages are used by all tools to perform input/output.

(c) Data Base Management. These packages are used by tools that directly manipulate objects in the KAPSE database. These are: the program integration facilities (PIF), virtual memory methodology (VMM), and the command processor (MCP).

(d) Utilities. These packages provide standard operating-system facilities such as a system-maintained clock and calendar.

3.1.5.3 Interfaces Among MAPSE Tools

The KAPSE controls the execution of all MAPSE components and provides the primitives whereby they transmit data back and forth and communicate with the user. Data interfaces among components rely on the use of the intermediate languages BILL and DIANA, and other VMM data structures, as well as object code for the host system. Major data interfaces are:

Interface	Between
BILL	COMP.MID/COMP.BE
DIANA	COMP.FE/COMP.MID COMP.MID/COMP.BE COMP.BE/PIF COMP.DBUG
object modules	COMP.BE/PIF

3.1.6 Government Furnished Property List ("4341")

(a) **Processor:** 4341 L01 (i.e., Group I), 4Mb memory;

(b) **Console:** 3278A02 Display, 4632 Keyboard (optional);

(c) **Line Printer:** 3203 Printer, 1416 Inter Chain;

(d) **DASD (Disks):** 3375 A01, B01, 2 of each (600 Mb each), with 3880 DASD Controller;

(e) **Tape:** 3410 Tape with 3411 Controller;

(f) **Communication:** 3705 Communication Controller, with 1551 Channel Adapter, with 9712 half-duplex line (8 lines).

(g) All necessary defining and operating manuals

3.1.7 Operational and Organizational Concepts

The AIE shall be installed on an IBM 4341 (see 3.1.6 above) and on a PE 8/32 computer, both at Rome Air Development Center.

3.2 Characteristics

3.2.1 Performance Characteristics

The AIE shall provide a minimal programming support environment to aid programmers and managers in the development and maintenance of Ada computer programs. It shall be designed to run on at least o host computers; viz the IBM 4341 and the PE 8/32. For the IBM ost, the AIE shall be compatible with the CP-370/VM operating system; for the Perkin/Elmer, the AIE shall be compatible with the OS/32 operating system. The AIE shall have facilities to:

(1) compile, link, load, execute, and debug programs written in the full Ada programming language on the specified host machines. Debugging shall be accomplished in terms of Ada statements, symbols, names, etc.;

(2) build and develop (and maintain) Ada programs by linking (and maintaining) collections of separate Ada compilations within appropriate program libraries. Such programs may be general Ada user programs, MAPSE tools, KAPSE functions, new APSE tools, and embedded computer software. Embedded computer software may be intended to execute on the host or intended for execution on a target machine, but developed in the host environment;

(3) process a user command language to activate and otherwise control (job control) all of the MAPSE tools and functions (See 3.1) in batch, remote-batch, and on-line modes, allowing such control to be exercised by both users and executing Ada programs;

(4) accept and manipulate (edit) user-supplied text representing Ada source programs, program documentation, and other general text. User interfaces shall be supported by batch, keyboard interactive, and high-band-width graphics terminals. The standard Ada character set is specified;

(5) establish and maintain (that is, enter, retrieve, manipulate) a central data base as a respository of source text, derived results (from compilation and execution), and other documentation;

(6) provide an ability to organize and view the database as a hierarchical or relational set of objects. Each object shall belong to a category and possess distinguishing characteristics by which, in combination, it can be uniquely referenced and used. In addition, a history shall be appended to each object so that it can be re-established if necessary;

(7) provide mechanisms for establishing a variety of management disciplines and protocols in conducting a project to develop Ada software. Mechanisms shall be based on controlling access to database objects (including tools) by a system of allocated and enforced permissions;

(8) establish and maintain configuration control by associating objects into controlled configurations and arbitrary partitions, based on their distinguishing attributes;

(9) support simultaneous use of the system by more than one user, either in cooperation or acting independently (that is, in a non-interfering manner);

(10) coexist with non-AIE users when operating on the IBM 370 under the VM/SP operating system or on the PE 8/32. In such cases, MAPSE data, tools, and functions shall be fully protected from interference. A MAPSE/non-MAPSE interface shall be supported via an import/export function.

(11) design tools and supporting functions so that the AIE can be maintained on itself.

3.2.1.1 Speed and Size Characteristics

The AIE's requirements in terms of speed and memory utilization are detailed below for the GFE equipment given in Section 3.1.6. Estimates of performance parameters appear in the Type C-5 specifications. This level A specification describes only those size and speed characteristics which are significant to the AIE user.

(1) The compiler/linker combination shall process any Ada program in no more than 512 K bytes of main memory. Up to an additional 512 K bytes, if available, will be utilized to speed up the compilation of large programs.

(2) The minimum, always resident, portion of the Ada runtime system shall not exceed 15 K bytes.

(3) The tasking support routines bound into Ada programs which use tasking shall not exceed 60 K bytes.

(4) The permanently resident portion of the KAPSE shall not exceed 256 K bytes.

(5) The loader residue (if any) shall not exceed 1 K bytes.

(6) The debug support task linked into programs for debugging shall not exceed 64 K bytes.

(7) Exclusive of edit buffers, the running editor shall not require more than 256 K bytes.

The parameters listed below are based on a baseline configuration, consisting of the IBM 4341 equipped as specified in the AIE contract, and with four users logged on, one running the Ada compiler and the others running the command processor (interactively) or the debugger or editor.

(1) Elapsed time for compiling and linking a single 1000 line Ada program, exclusive of listing, optimization, and debug options shall not exceed four minutes.

(2) CPU time for the above shall not exceed one minute.

(3) Average response time (carriage return until first response character) for MCL commands shall not exceed the following:

- Program invocations 1 second
- Simple DB queries 1 second
- Multi-file DB queries 5 seconds
- Logon/Logoff 10 seconds

(4) Average single-character echo time shall not exceed 100 milliseconds.

(5) The average time to start printing a line from an edit buffer after a print command is entered shall not exceed 1/2 second.

(6) The average time to start printing a simple Ada scalar variable after a completed request by name is entered to the debugger while at a breakpoint shall not exceed 1/2 second.

3.2.1.2 Operational Characteristics

The AIE can be used interactively or in batch mode. In both modes, the MCP processes user commands and activates the appropriate tool(s) via the KAPSE. In interactive mode, the MCP issues a prompt character to the user when it is ready for input. The user then requests a job to be done (i.e., execution of some program). The job can be executed in the foreground, in which case the MCP will issue a prompt only when the job is complete. The job can also be executed in the background, in which case the MCP issues a prompt and continues to process user commands, interrupting the interaction when the background job is completed. The database management function of the KAPSE coordinates concurrent usage of shared data.

Batch mode usage is an extension of interactive use, accomplished by creating a script of MCP commands to be executed in sequence with no user prompting. Batch jobs can be submitted with a specified time for execution.

Figure 4 is a diagram illustrating an interactive user's view of the AIE. Circled numbers indicate significant actions or operational interfaces and are defined as follows:

1. User logs on and enters commands or text.

2. MCP is activated automatically when user logs on.

3. MCP can activate any MAPSE tool or function through the KAPSE program control and loader routines.

4. Desired Ada program is loaded for execution.

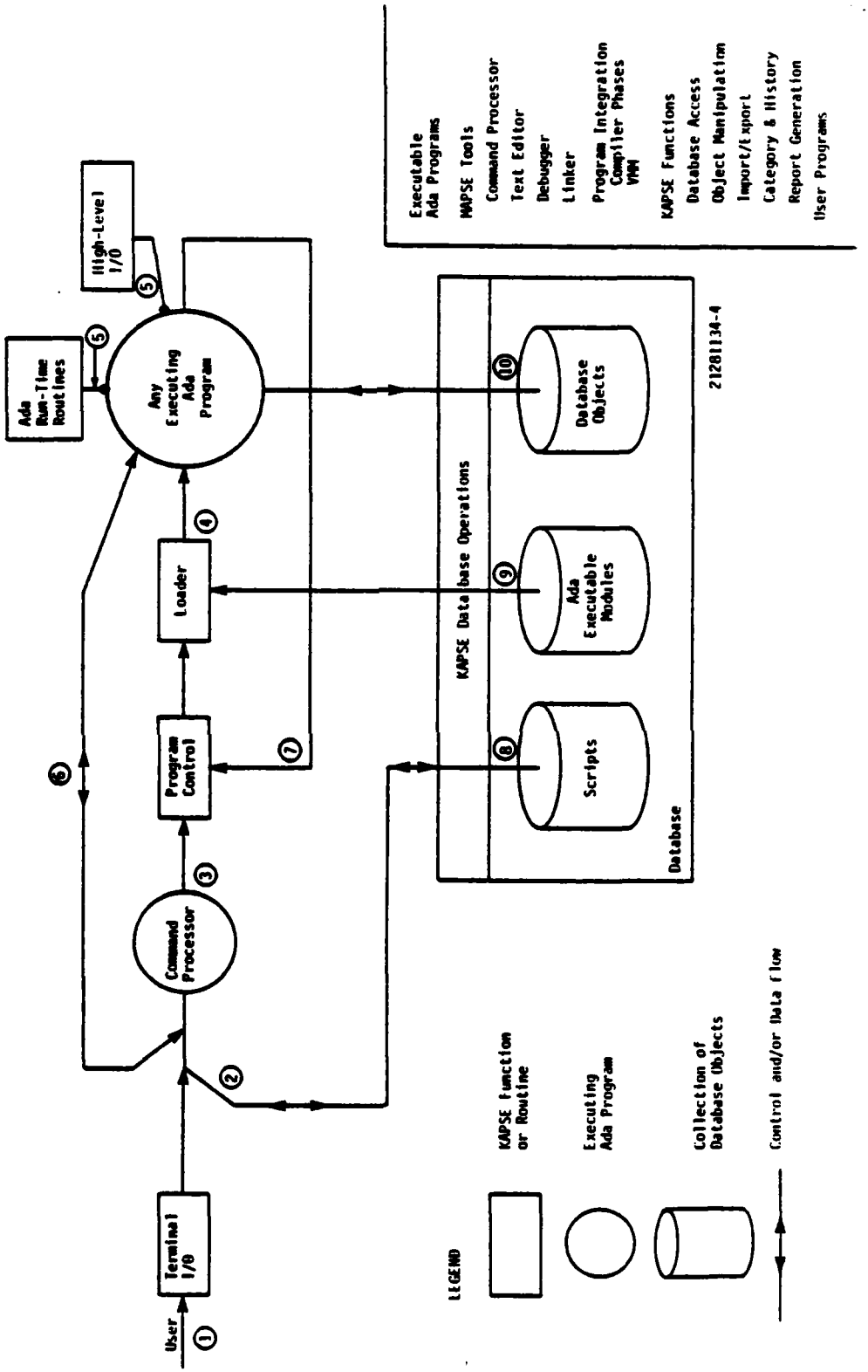


FIGURE 4: USER VIEW OF THE AIE (See Accompanying Scenario)

5. Execution is supported by Ada run-time and high level I/O routines within KAPSE.

6. User can interface to executing Ada program.

7. Executing Ada program can activate any MAPSE tool or function (including MCP) through the KAPSE program control and loader routines.

8. Stored scripts within the database (created via the editor), can affect the MCP or any executing Ada program in the same manner as the user.

9. Stored Ada executable modules within an Ada program library in the database (created via the linker) are accessed by KAPSE database operations and loaded for execution by the KAPSE loader.

10. General database objects are stored within, and retrieved from the database by executing Ada programs using the KAPSE database operations.

3.2.1.3 Design Goals

1. Usability - the AIE shall be easy to use for novice as well as experienced users. Simple things shall be simple to do without full knowledge or exercise of all options and capabilities.

2. Applicability - the AIE shall be applicable to the development and maintenance of Ada programs, including the MAPSE itself, as well as embedded computer software for DOD projects: large and small, of varied duration, and subject to a variety of management requirements and disciplines. While providing specific tools, the MAPSE shall not dictate the way in which they are used. The system shall provide support only; the programmers will program and the manager will manage.

3. Portability - the system shall be portable, in that it can be rehosted and retargeted to other host computers with minimum effort. To support this requirement, host dependencies shall be isolated so that they are easily identified. In addition, maximum use shall be made of the Ada programming language in implementing the MAPSE.

4. Functionality - basic facilities provided by the MAPSE will be the same for all hosts.

5. Modularity - the MAPSE shall be constructed in a modular fashion so that functional areas (tools) are lightly coupled and can easily be replaced, modified, or extended.

6. Open-Endedness - the MAPSE shall be an open-ended system so that new tools and facilities can be added within the context of the original design. The MAPSE shall support and be the foundation for the development and implementation of a full APSE and AIE.

7. Integrated Design - an integrated approach shall be taken to the design of the MAPSE, making maximum use of uniform interfaces and reusable portions.

8. Reliability - the reliability of the MAPSE shall be ensured by the use of dependable, disciplined design and development methodologies; it shall be designed for testability (with built-in test techniques), and full qualification shall be undertaken (PQT, FQT) prior to release for use. The system shall contain features to protect itself from user and system errors. Built-in aids shall be provided to detect anomalies in operation and to correct such anomalies with a minimum of effort. These aids shall be compatible with the use of Ada and modular construction techniques.

3.2.2 Physical Characteristics

Physical characteristics shall be as specified for equipment designated in Section 3.1.6. No particular security criteria shall be applicable to the MAPSE.

3.2.3 Reliability

Reliability of hardware (computer) facilities shall be as specified for equipment designated in Section 3.1.6. Reliability of the AIE is discussed in Section 3.2.1.3(8).

3.2.4 Maintainability

3.2.4.1 Equipments and Vendor Supplied Support Software

Maintainability shall be as prescribed in the specification listed in 3.1.6.

3.2.4.2 AIE Software Systems

The AIE shall be maintained on both hosts by contractor personnel in accordance with a Maintenance Manual for at least 12 months following delivery and acceptance of the MAPSE by the Air Force. Since the AIE is composed of clearly delineated host independent and host dependent components, maintenance will have to

occur on both the IBM AIE and the PE AIE. All host independent components will be maintained on the IBM AIE. Host dependent components will be maintained on their respective hosts. If a host dependent problem is corrected on one host, the other host will be examined to determine if a related problem exists. After the 12 month maintenance period, maintenance procedures shall be specified in sufficient detail to permit maintenance of the system by other than the designing contractor. Maintenance shall consist of the corrections of operational and documentation errors experienced during test or field use, and necessary modifications to meet specification changes. (See 3.2.1.3(8)).

3.2.5 Availability

Following acceptance by the Air Force, the AIE shall be in an operable and maintainable state on both the IBM 4341 and PE 8/32 computers provided by RADC. Use, schedule and commitment of the system shall be by Air Force/RADC direction.

3.2.6 System Effectiveness Models

Not applicable.

3.2.7 Environment Conditions

Not applicable.

3.2.8 Nuclear Control Requirements

Not applicable.

3.2.9 Transportability

The AIE shall be transportable with minimum effort from the IBM 370 to the PE 8/32. In addition, the MAPSE shall be designed for transportability to any other host computer (or network of computers) that can reasonably be anticipated at the time of design. Such computers shall include all appropriate 32-bit (or greater) mainframes of general register or stack architecture, but need not include state-of-the-art machines, (e.g., pipeline, array and parallel processing architectures). The design of the AIE shall facilitate transportability by isolating all machine dependencies within the KAPSE, by implementing the AIE (to the greatest degree possible) in the Ada programming language, and by guaranteeing constant user functionality regardless of host.

3.3 Design and Construction

The AIE shall be designed and constructed in accordance with the methods put forth in AIE(1).CPDP(1).

3.3.1 Materials, Processes and Parts

Not applicable.

3.3.2 Electromagnetic Radiation

Not applicable.

3.3.3 Nameplates and Product Marking

Not applicable.

3.3.4 Workmanship

3.3.4.1 Hardware

Not applicable.

3.3.4.2 Software

In accordance with AIE(1).CPDP(1).

3.3.5 Interchangeability

3.3.5.1 Hardware

Not applicable.

3.3.5.2 Software

AIE software shall be modularly designed with well-defined interfaces and utilizing Ada's abstract data type facilities to a maximum degree. Specifically with respect to the Ada Compiler, the Back End (or code generating portion) shall be replaceable, meeting a well defined DIANA intermediate language interface specification [AIE(1).COMP(1).DIANA(1)].

3.3.6 Safety

Not applicable.

3.3.7 Human Performance/Human Engineering

3.3.7.1 Hardware

Not applicable.

3.3.7.2 MAPSE Tools

A uniform interface between the user and MAPSE tools is provided by the MCP command language. The KAPSE assumes responsibility for concurrent use of the database to support interactive use.

User documentation will be augmented by an on-line "help facility". Individual tools are designed so that a novice user can work with a simple subset of commands while providing more sophisticated capabilities for the experienced user. Maintenance of development history within the KAPSE database and program library support the recreation of current status in case of system failure.

3.3.8 Computer Programming

In accordance with the SOW and CPDP, the AIE will be implemented primarily in Ada. For efficiency purposes, assembly language will be used in two instances in implementing the KAPSE: (1) in the storage management facilities of the run-time system; and (2) in the interfaces preceding and following rendezvous.

3.4 Documentation

The following scheme provides a unique identification for AIE documents within the AIE hierarchy. The identification model is:

[Host.]System(rev).Subsystem or Doc Type(rev).CPCI(rev)

implemented as follows:

Identifier	Document
AIE(i)	System Spec. Level A
AIE(i).Subsystem(j)	B5 or CPDS
[Host.]AIE(i).Subsystem(j).CPCI(k)	C5 or CPPS
AIE(i).Subsystem(j).CPCI(k).TPROC(m)	Test Procedures
AIE(i).TPLAN(j)	Test Plan
AIE(i).CPDP(j)	CPDP
Host.AIE(i).USER(j)	User Manual

Identifier	Document
Host.AIE(i).MAINT(j)	Maintenance Manual
AIE(i).RR(j)	Rehost/Retarget Manual

The acceptable identifiers for Subsystem and CPCI may be found in the Configuration Components Chart in Appendix A.

Figure 5 illustrates the AIE documentation hierarchy.

3.5 Logistics

3.5.1 Maintenance

3.5.1.1 Hardware

Not applicable.

3.5.1.2 Software

Maintenance of the AIE system shall be accomplished in accordance with the Maintenance Manual (see 3.4)for each host.

3.5.2 Supply

Not applicable.

3.5.3 Facilities and Facility Equipment

Facilities and facilities equipment shall be as specified in Section 3.1.6.

3.6 Personnel and Training

3.6.1 Operations Training

A one week training course in the use and operation of the AIE shall be conducted at RADC for up to 20 personnel following acceptance of the system by the Air Force.

3.6.2 Maintenance Training

A two week training course in the maintenance, rehosting and retargeting of the AIE shall be conducted at RADC for up to ten Ada programming personnel.

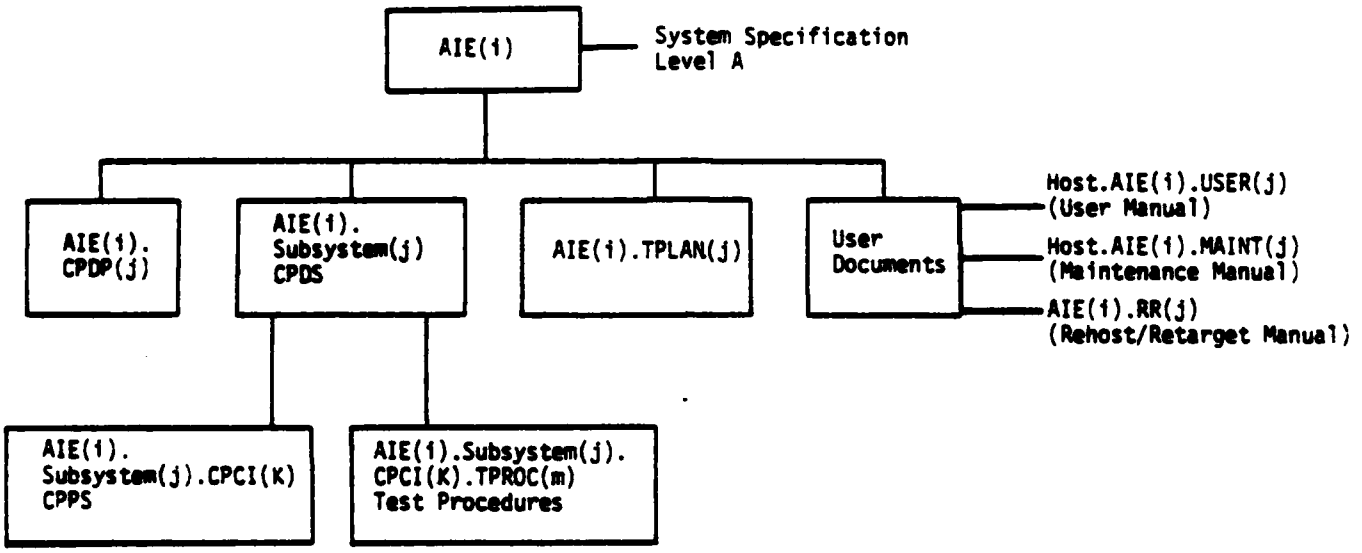


FIGURE 5: AIE DOCUMENTATION HIERARCHY (See Appendix A for Subsystem and CPCI Identification)

3.7 Functional Area Characteristics

The MAPSE is subdivided into the functional areas listed in 3.1.1; they shall exhibit the characteristics described below.

3.7.1 KAPSE

The KAPSE shall include the operations necessary to manipulate, organize, and control access to a central database. The database shall be the repository of all data objects and their attributes including source text, the various forms of DIANA, executable modules, intermediate results of program execution, and any other data defined and associated with MAPSE tools or operations. Figure 6 is a KAPSE overview.

More specifically, the KAPSE/Database shall:

1. provide all the interfaces between the MAPSE and any software or hardware systems outside the MAPSE;

2. encapsulate all hardware and host dependencies, including interfaces to vendor-supplied support software;

3. provide the run-time routines to effect an Ada virtual machine;

4. provide keyboard/terminal interfaces to users and/or other equipments;

5. include database operations to assure exclusive access to and control of the database including management and summary reporting of: configurations, partitions, attributes, archiving and protection;

6. provide standard data and program control interfaces among all defined MAPSE tools and the database, and supply uniform interface definitions for future tools;

7. facilitate invocation of any tool from any other tool, including the Command Processor;

8. include a loader to load the host memory with executable units from an Ada program library;

9. provide high-level Input/Output capabilities similar to those available with FORTRAN;

10. provide the necessary capabilities to effect system recovery/reconfiguration after experiencing a malfunction; within either the host operating system or the KAPSE itself. (Note: the KAPSE may be reloaded if necessary without stopping the host.)

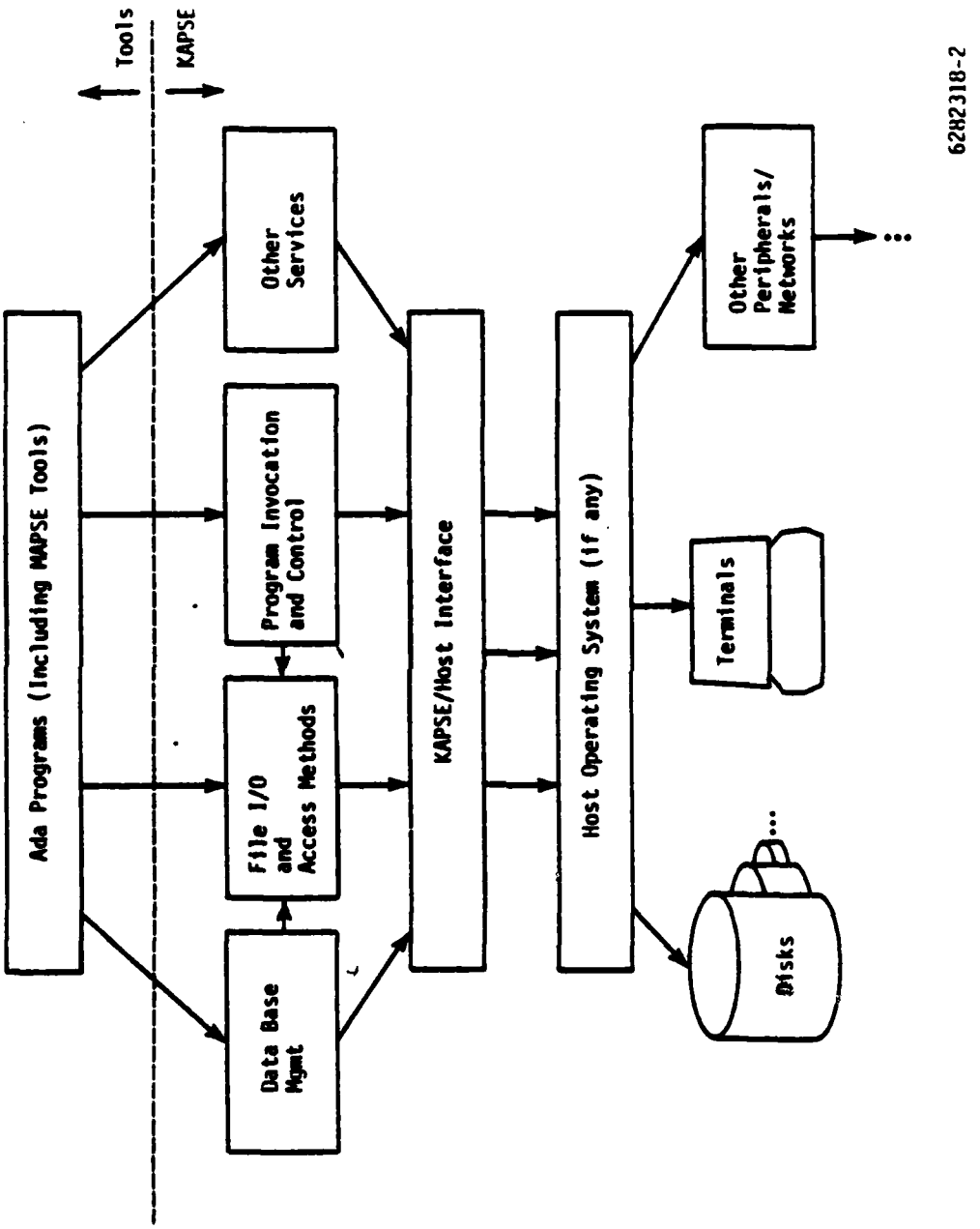


FIGURE 6: KAPSE/DATABASE OVERVIEW

3.7.2 MCP

The MAPSE Command Processor, MCP, shall be the primary interface between the user and the MAPSE tools and facilities. The MCP shall be activated automatically when a user logs into the AIE, or when requested by an executing Ada program. Figure 7 is an overview of the MCP.

More specifically, the MCP shall implement a MAPSE Command Language (MCL) with which the user can perform the following functions:

1. invoke and control the execution of an arbitrary Ada program;

2. receive data from a program, create and manipulate variables and query the state of a program;

3. request help on a per-program basis;

4. connect arbitrary sequences of MCL commands for execution in foreground or background;

5. redirect the standard input or output of MCL sequences from/to arbitrary files.

3.7.3 PIF

Program Integration Facilities, PIF, shall control all access to, and manipulation of, the Ada libraries, including support needed for compilation and linkage editing. Since the libraries are proper objects of the database, access must be gained through the KAPSE database operations (See 3.1.1.1, 3.7.1). Figure 8 is an overview of PIF. More specifically, PIF:

1. minimize the requirement for recompilations by analyzing the interdependencies of the unit(s) presented for compilation with previously compiled (and stored) library units, and recompile only when necessary;

2. provide a tool that performs an analysis to determine which library units must be recompiled to maintain a consistent compilation set;

3. support a program build by selecting a consistent set of library units for input to the linker. Only those units actually used shall be included;

4. provide a "stub generation" facility by which missing bodies of Ada subprograms, package and task specifications may be automatically generated as valid Ada source code and included in a consistent set of library units. Such source code shall be the minimum necessary to permit program execution;

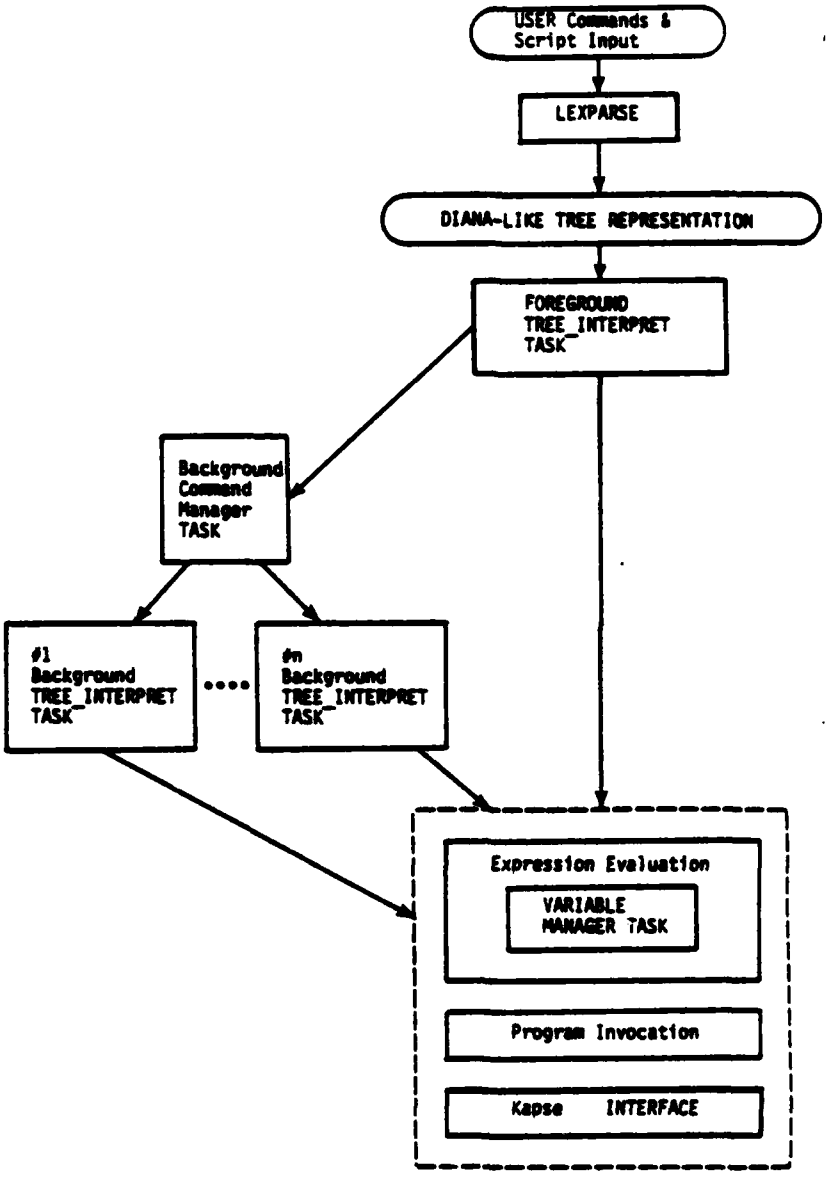


FIGURE 7: MCP OVERVIEW

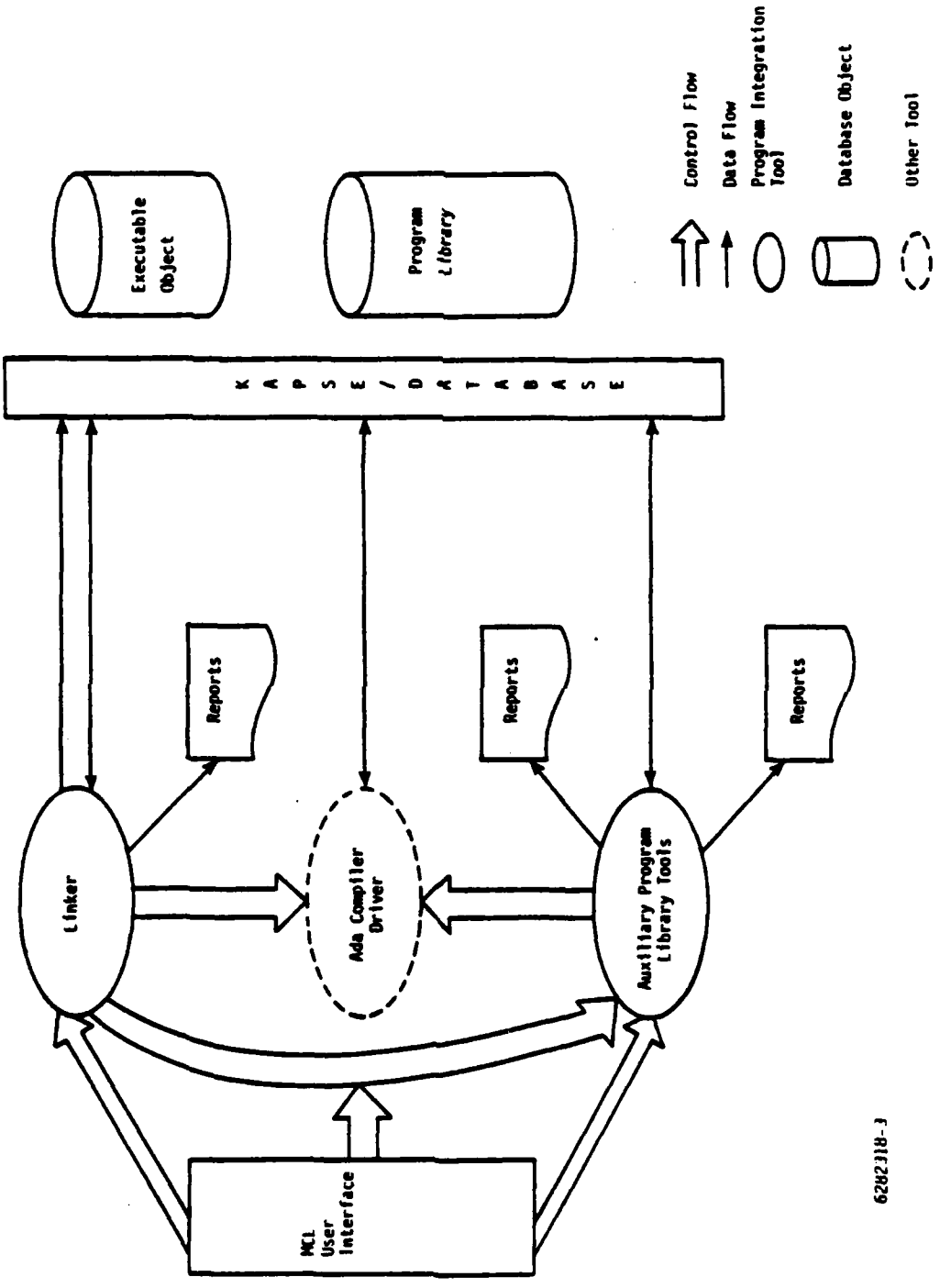


FIGURE 8: PIF OVERVIEW

5. provide a linker that shall accept linkable Diana program library units and produce near-executable memory images for input to the KAPSE loader (see 3.1.1.1).

3.7.4 COMP

The Ada compiler, COMP, shall translate Ada language source text input into linkable modules. The intermediate languages DIANA and BILL will serve as interfaces within the compiler. Figure 9 is a compiler phases overview. More specifically, COMP shall:

1. process the complete Ada language as specified in the Ada Language Reference Manual;

2. compile Ada source code statements at a minimum rate of 1000 statements per minute, computed by dividing CPU time by the total number of statements in the compilation unit;

3. be designed in terms of relatively independent processing phases. The information interface between phases shall consist of the various forms of the DIANA intermediate language. The phases shall comprise a target machine-independent set and a target machine-dependent set. Retargetting shall affect only the middle and back end, and machine dependencies shall be parameterized where possible;

4. perform error detection and reporting, including classification of errors by severity;

5. perform optimizations for timing and/or space, without changing the functional meaning of a program by the user. Available optimizations are: constant propagation; redundant constraint check elimination; constant folding; elimination of unreachable code; movement of loop invariant code; redundant computation elimination; reduction of multiplications to additions within a loop; algebraic simplification of expressions and statements; as well as peephole optimization. Level of optimization is user-selectable. When no optimization is specified, the compiler will perform most optimizations at a reduced rate (it will not do invariant code movement, reduction of multiplications, or peephole).

6. identify any necessary limitation on capacities (such as symbols and nesting level).

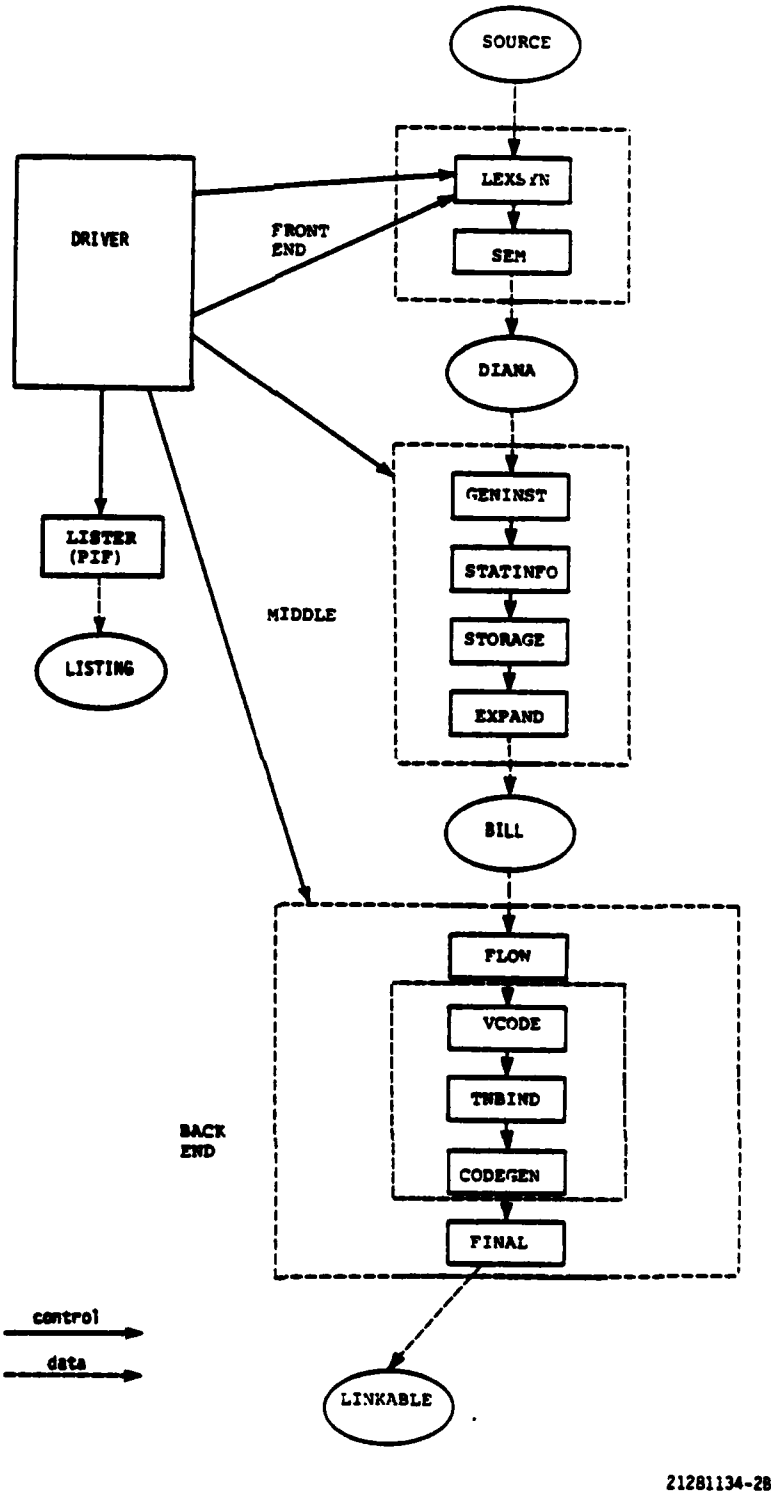


FIGURE 9: COMP OVERVIEW

3.7.5 TXED

The Text Editor, TXED, shall provide a basic editing facility suitable for editing general text (characters). Figure 10 is a TXED Overview. More specifically the TXED shall:

1. edit all text files including but not limited to Ada source programs;

2. provide sufficient commands to find, alter, insert, delete, input, output, move, copy, and substitute text;

3. be operable in interactive or batch modes;

4. accept scripts or files of editing commands saved in the database as control input. Furthermore, the long forms of all commands shall be sufficiently self-explanatory so that scripts are readable by humans as well as the editor;

5. support both standard hard-copy and CRT-type terminals effectively;

6. be able to invoke MAPSE tools from within the editing session. The editor can pass the text it is editing to MAPSE tools as input data, and receive output data from tools into the edited tool;

7. be simple for new users to learn yet satisfying in its capabilities and its interface for more experienced personnel.

3.7.6 DBUG

The Debugger, DBUG, shall be capable of controlling the execution, examination, and modification of an Ada program. DBUG shall process a control language that permits a user to specify a variety of actions to be taken at execution control points (breakpoints). (See 3.1.1.6). Such points shall be based on Ada statements or labels, task activations, subprograms calls, returns, and exceptions.

When a program in execution is suspended at a breakpoint, a user shall be able to display, dump, or modify selected data values in either machine or scalar type representation, trace the flow of program control, as well as display the formal names and values of subprogram arguments.

The design shall also provide interfaces for functional simulation; that is, future APSE tools may be added to advance a 'pseudo-clock' (in order to keep track of problem time), activate Ada tasking functions based on that time or other multi-programming conditions, and interface to the KAPSE for other support, such as environment model updating.

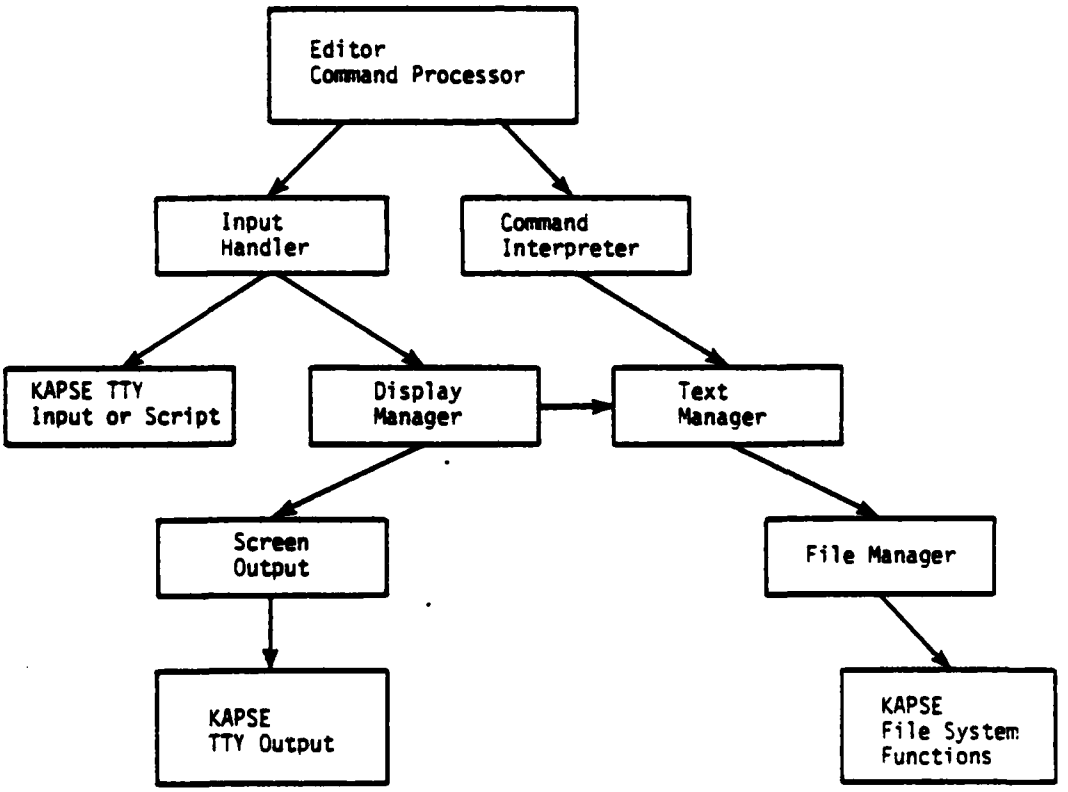


FIGURE 10: TXED OVERVIEW

DBUG may be used either interactively or, via MCP command scripts, in batch mode.

The design of the system shall be compatible with a future APSE-resident debug tool (using identical user commands) that permits control and debugging of embedded computer software executing on a target machine. Figure 11 is a Debugger Overview.

3.7.7 VMM

Virtual Memory Methodology, VMM, is a system to aid in the definition and access of database objects. This system shall include the capability to define arbitrary data structures and an access facility to automatically address virtual memory far in excess of the real storage available. Figure 12 is a VMM overview.

3.7.8 MGS

MAPSE Generation and Support Facilities, MGS, shall provide the tools to develop the AIE on the IBM 4341 or PE 8/32 hosts and to aid in its maintenance on these hosts. Figure 13 shows the use of two such tools. More specifically, MGS shall:

1. provide a bootstrap compiler so that the MAPSE Ada Compiler can be written in Ada itself. (This is necessary because no Ada compiler currently exists for either the IBM 4341 or the PE 8/32). The bootstrap compiler is not a deliverable item.

2. provide "secondary" MAPSE tools whose primary functions are to build, maintain, and extend the existing tool set. Included at this level are the lexer and parser generators,initial program loaders, code-table generators, and object module converters.

3.8 Precedence

The AIE design is based on the following precedence of requirements:

(1) to provide a supportive, machine-independent programming environment in which embedded computer software written in Ada can be developed, tested, and maintained;

(2) to provide a rehostable, retargetable KAPSE;

(3) to provide ease of maintenance and life-cycle support both for the AIE and for software produced within it;

(4) to lay the foundation for a richer environment by easy installation of new tools.

See Section 3.4 and Figure 5 for the precedence of AIE documentation.

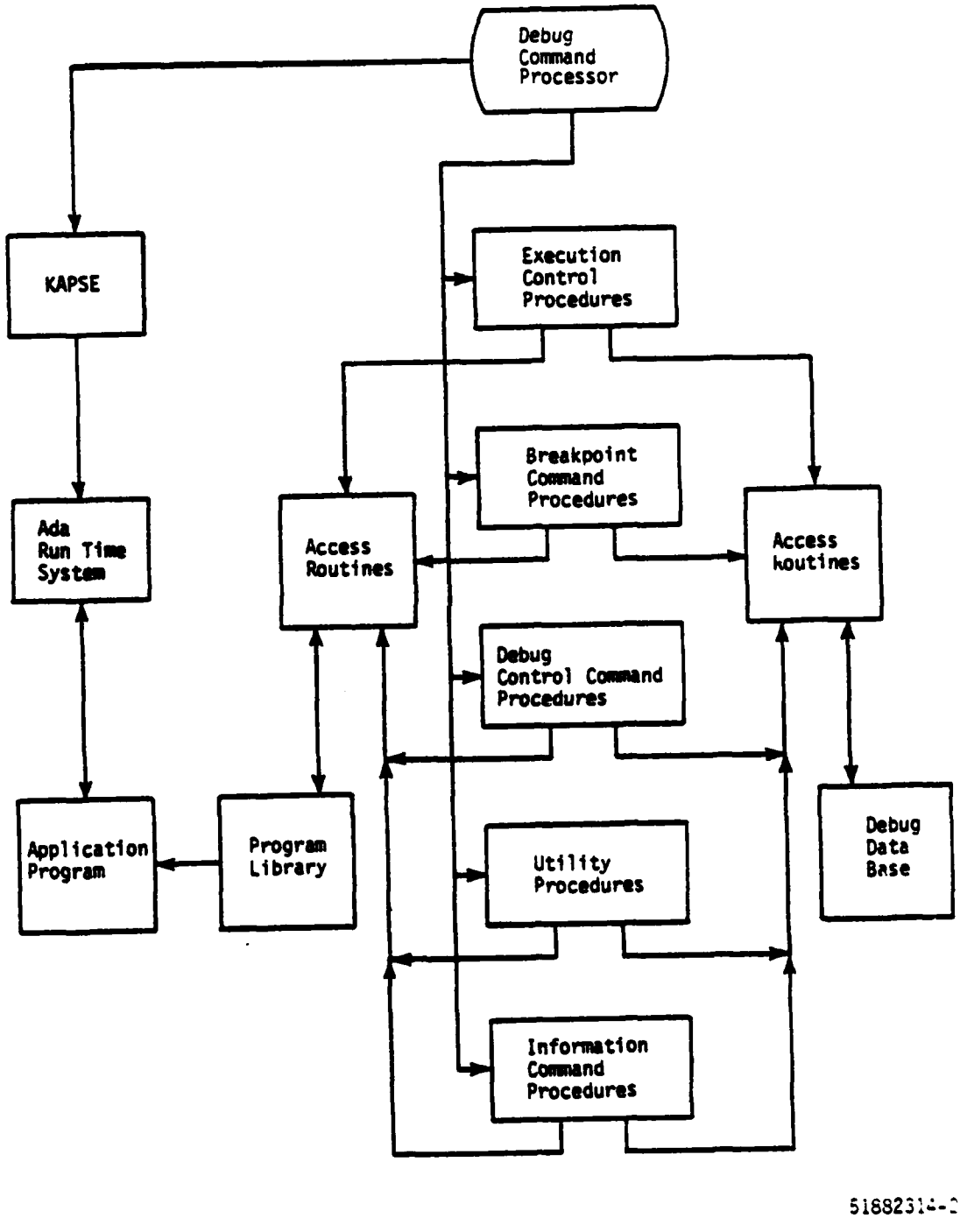


FIGURE 11: DBUG OVERVIEW

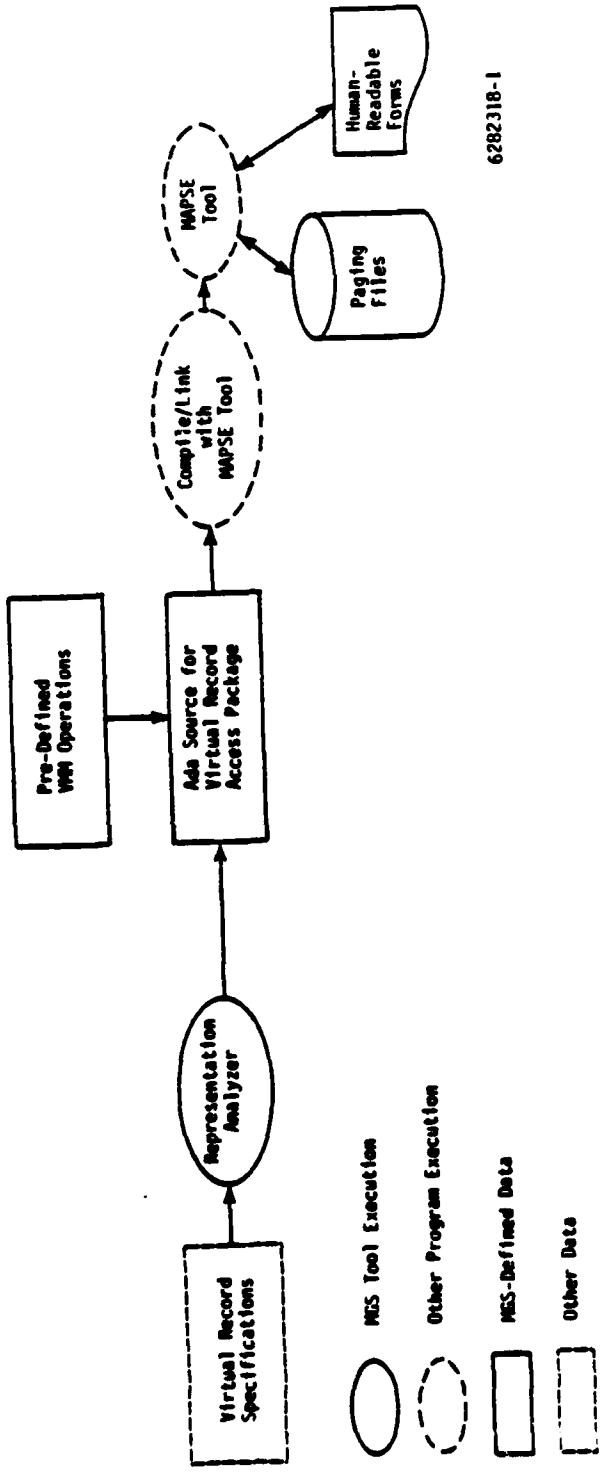


FIGURE 12: VMM OVERVIEW

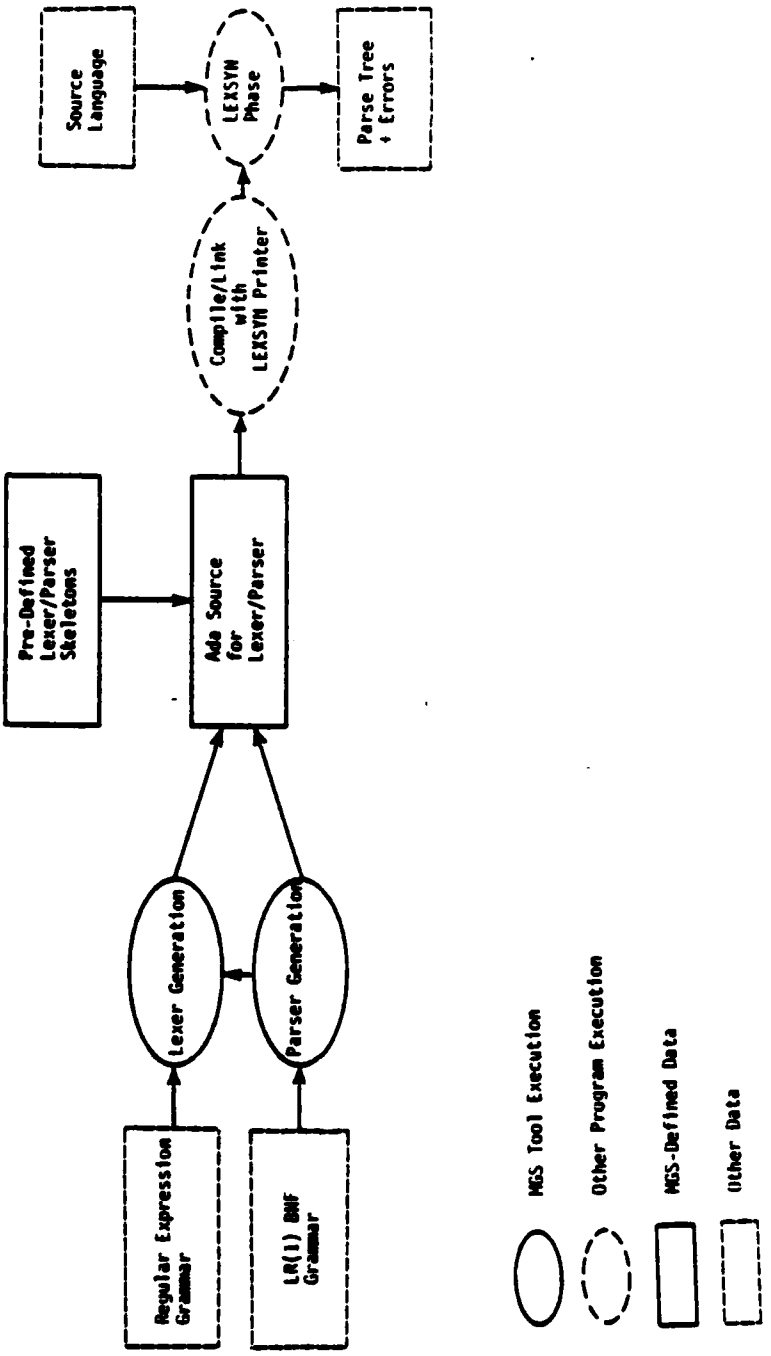


FIGURE 13: MGS TOOLS. Parser/Lexer Generator

4.0 QUALITY ASSURANCE PROVISIONS

4.1 General

The Quality Assurance (QA) program shall assure compliance of the AIE with the software requirements of the contract as required by MIL-S-52779A. The scope and responsibilities of the QA program are presented in the Computer Program Development Plan (CPDP). In addition, the planned developmental and formal testing of the AIE is presented in the CPDP and the Test Plan [AIE(1).TPLAN(1)]. This section discusses the system level testing and verification.

Tests and evaluations shall be conducted to verify that the design and performance of the MAPSE shall meet or exceed the requirements specified in Section 3 of this document. A program of testing and validation augmented by analysis shall be conducted to verify compliance with the requirements of this specification and subordinate specifications. MAPSE test objectives shall be implemented in accordance with the CPDP, the Test Plans and Test Procedures for each defined CPCI.

4.1.1 Responsibility for Tests

4.1.1.1 MAPSE Test Plans and Procedures

Except as the contract otherwise specifies, Intermetrics shall be responsible for the performance of all analysis, tests and demonstrations specified herein. All testing shall be witnessed by the Government.

4.1.1.2 Ada Compiler Validation

The Government shall be responsible, with contractor support, for the validation of the IBM 370 and PE 8/32 Ada compilers. The government shall employ the DoD Ada Compiler Validation Facility (ACVF) for this purpose. Acceptance of the compilers shall be contingent on certification by the Ada Certification Control Board.

4.1.1.3 Independent Testing

The Government shall engage an independent IV&V contractor to conduct independent testing of the AIE software. The design contractor shall support this effort.

4.1.2 Special Tests and Examinations

Not applicable.

4.2 Quality Conformance Verification

Formal Qualification verification shall be conducted (in accordance with the CPDP) to validate that the design and performance of the system satisfy the requirements of Section 3 of this document.

4.2.1 Software Quality Assurance

The quality assurance provisions defined in this section provide requirements for the performance verification of the MAPSE computer programs. These requirements shall assure that the performance and design requirements defined in Section 3 are met. These verifications shall be implemented in accordance with the Government-approved CPDP, Test Plans and Test Procedures.

4.2.1.1 Preliminary Qualification Testing

4.2.1.1.1 Component Level Qualification

The objective of component testing is to validate functional units. A component is defined as an Ada subprogram, package, task or subunit, and/or any computer program element that has a logical function. Each component shall compile successfully and shall further be validated by visual inspection of the code. Input/Output tests shall be utilized, as appropriate, to verify proper control flow, data results, etc.

4.2.1.1.2 Integration Level Qualification

The objective of integration testing is to test related groups or strings of components that constitute defined CPCI's. KAPSE and/or vendor supplied operating system functions may be necessary to successfully complete integration level tests. These tests shall verify inter-component communications and control, and parameter passage.

4.2.1.1.3 System Level Qualification

System level tests shall be conducted on groups of CPCI's processing simulated or real operational data. This level of testing requires an almost totally integrated computer program interface with the environment. Such tests shall demonstrate typical scenarios in the development and/or management of Ada software.

4.2.2 Formal Qualification Testing

Formal qualification testing shall be conducted on each CPCI in order to verify that all requirements of the design specifications are met. Successful completion of such tests, plus those listed below, shall constitute acceptance of the MAPSE by the Government.

4.2.2.1 Demonstrate Rehostability and Retargetability

The AIE system shall be demonstrated to be rehostable and retargetable from the IBM 4341 to the PE 8/32 computers at RADC.

4.2.2.2 Demonstrate Compilability

The IBM 4341 and PE 8/32 Ada compilers that are developed as part of the MAPSE shall be demonstrated to compile a non-trivial Ada program with at least 50 executable Ada source statements in at most 256 K bytes of memory. Compilation speed shall be not less than 1000 Ada statements per minute (see 3.2.1.1).

LEFT BLANK INTENTIONALLY

5.0 PREPARATION FOR DELIVERY

The AIE software system (and specified operable subset, viz. Ada compiler plus KAPSE) shall be delivered to the Government in the form of magnetic tapes and listings. The particular formats shall be specified by RADC and defined within DI-E-30145.

LEFT BLANK INTENTIONALLY

6.0 NOTES

None

APPENDIX A: CONFIGURATION COMPONENTS CHART

* indicates two CPCI's - one for the IBM 4341 and one for the PE 8/32.

Level	Subsystem (ID)	CPCIname (ID)	CPCname (ID)	
AIE				
APSE				
MAPSE				
	COMPILER (COMP)			
		FRONT END (FE)		
			DRIVER	(A)
			LEXSYN	(B)
			SEM	(C)
			UTILITIES	(D)
		*MIDDLE PART (MID)		
			GENINST	(A)
			STATINFO	(B)
			STORAGE	(C)
			EXPAND	(D)
			UTILITIES	(E)
		*BACK END (BE)		
			FLOW	(A)
			VCODE	(B)
			TNBIND	(C)
			CODEGEN	(D)
			FINAL	(E)
			UTILITIES	(F)
		DIANA (DIANA)		
		LOW-LEVEL INTERMEDIATE LANG (BILL)		

* indicates two CPCI's - one for the IBM 4341 and one for the PE 8/32.

Level	Subsystem (ID)	CPCIname (ID)	CPCname (ID)	
	TEXT EDITOR (TXED)			
		TEXT EDITOR (TXED)		
			COMMAND PROC	(A)
			INPUT HANDLER	(B)
			COMM INTERP	(C)
			KAPSE TTY INPUT	(D)
			DISPLAY MGR	(E)
			TEXT MGR	(F)
			SCREEN OUTPUT	(G)
			KAPSE TTY OUTPUT	(H)
	DEBUGGER (DBUG)			
		*DEBUGGER (DBUG)		
			COMM PROC	(A)
			EXEC CTRL PROCED	(B)
			BRKPT COMM PROCED	(C)
			UTILITY PROCED	(D)
			INFO COMM PROCED	(E)
			PROG LIB ACC PROC	(F)
			DEBUG DATABASE	(G)
	MAPSE COMMAND PROCESSOR (MCP)			
		MAPSE COMMAND PROCESSOR (MCP)		
			DRIVER	(A)
			LEXPARSE	(B)
			ALLOC INTERP	(C)
			TREE INTERP	(D)
			BACKGRD MGR	(E)
			EXPR PROC	(F)
			SCRIPT PROCESSING	(G)
			PROGRAM INVOCATION	(H)
			VARIABLE	(I)
			ERROR	(J)

* indicates two CPCI's - one for the IBM 4341 and one for the PE 8/32.

Level	Subsystem (ID)	CPCIname (ID)	CPCname (ID)
	PROGRAM INTEGRATION FACILITY (PIF)		
		*LINKER (LINK)	
			PROGRAM COMPLETENESS (A)
			INTEGRATION (B)
			CONTEXT CREATION (C)
			OBJECT MODULE FORMAT (D)
		PROGRAM LIBRARY TOOLS (PLTOOLS)	
			LISTER (A)
			PROG LIB MANAGER (B)
			PROG LIB STAT REPTR (C)
			LINK MAP LISTER (D)
			SOURCE EXTRACTOR (E)
			CHANGE ANALYSER (F)
			MAP (G)
			RECOMPILER (H)
			PREAMBLE GENER (I)
			BODY GENER (J)
	VIRTUAL MEMORY METHODOLOGY (VMM)		
		VIRTUAL MEMORY METHODOLOGY (VMM)	
			REP ANALYZER (A)
			ABSTRACT DATA TYPES (B)
			VRN IO (C)
			BASIC OPERATIONS (D)
	MAPSE GENERATION & SUPPORT (MGS)		
		COMPILER GENERATOR TOOLS (COMPGEN)	
			GENERATORS (A)
			LEXER/PARSER SKEL (B)
			BONSAI PREPROCESSOR (C)
			TABLE BUILDER (D)
		*INITIAL PROG LOAD GEN (IPLGEN)	
			IPLGEN (A)
		*OBJECT MODULE CONVERSION (OBCON)	
			OBJ MOD CONV (A)

* indicates two CPCI's - one for the IBM 4341 and one for the PE 8/32.

Level	Subsystem (ID)	CPCIname (ID)	CPCname (ID)
KAPSE			
	KAPSE (KAPSE)		
		*RUN-TIME SYSTEM (RTS)	
		UNIT EXEC SUPP	(A)
		STORAGE MGT	(B)
		TASKING SUPP	(C)
		EXCEP HANDLING	(D)
		PREDEFINED PKGS	(E)
		TYPE SUPP	(F)
		*SIMPLE & COMPOSITE OPS (SIMCOMP)	
		BLOCK IO	(A)
		DEVICE IO	(B)
		ACCESS METHODS & DATA CLUMPS	(C)
		SIMPLE OBJECTS	(D)
		COMPOSITE OBJECTS	(E)
		ACCESS CONTROL & CATEGORIES (ACCECAT)	
		WINDOW OBJECT	(A)
		CATEGORY & USER-DEFINED ATTRS	(B)
		ACCESS CONTROL	(C)
		*MULTIPLE PROGRAM INVOCATION, CONTROL, & COMMUNICATION (MULTPROG)	
		PROGRAM LOADING	(A)
		LOW-LEVEL KAPSE/PROG COMM	(B)
		PROG INVOC & CTRL	(C)
		KAPSE/KAPSE COMM	(D)
		TERM SCREEN MGR	(E)
		LOGIN/LOGOUT & USER CONTEXT	(F)
		MAIL	(G)
		HISTORY & ARCHIVING (HISTARCH)	
		HISTORY	(A)
		BACKUP RECOV	(B)
		CONFIG MGT	(C)

INTERMETRICS INCORPORATED • 733 CONCORD AVENUE • CAMBRIDGE, MASSACHUSETTS 02138 • (617) 661-1840