March 1987

MICA Software Business Plan

Preliminary Plan

At Exit from Phase 1

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY—DO NOT COPY
RESTRICTED DISTRIBUTION

Revision number: 0.2

Prepared By:

Catherine Richardson, *MICA Business Product Manager*
Terry Morris, *MICA/VMS Technical Product Manager*
Rockie Morgan, *MICA/ULTRIX Technical Product Manager*
Reid Brown, *PRISM Systems Product Manager*
Donna Meikle, *DECwest Finance*

DECwest Engineering, *Bellevue, WA*

Approved by:

David Cutler—DECwest Group Manager

Bill Demmer—Vice-President of Mid-Range Systems

Rich Butler—Mid-Range Systems Finance Manager

PRISM SYSTEMS PROJECT TEAM

PRISM Systems Manager:

David Cutler, *Group Manager*

Engineering:

Robert Short, *Crystal/Jewel Hardware Engineering Manager*
Tom Miller, *MICA Software Engineering Manager*
Don MacLaren, *PILLAR and C Engineering Manager*
John Balciunas, *DECwest Hardware Program Manager*
John Gilbert, *DECwest Software Program Manager*
Dave Ballenger, *MICA/ULTRIX Project Leader*
Robert Bismuth, *MICA Project Leader*
Jeff East, *MICA Project Leader>*
Darryl Havens, *PILLAR Project Leader*
Lou Perazzoli, *MICA Project Leader*
Chris Saether, *MICA Project Leader*
Benn Schreiber, *MICA Project Leader*
Ken Western, *DECwest Documentation Manager*
Jim Jackson, *MICA/VMS Documentation Project Leader*
Michael Tardiff, *MICA/ULTRIX Documentation Project Leader*
Craig Kosak, *MICA/ULTRIX Documentation Production Project Leader*
Lon Willoughby, *DECwest Financial Manager*

Product Management:

John Coombs, *Crystal/Jewel Product Manager*
Dick Angel, *PRISM Marketing Manager*
Susan Usilton, *Emerald Product Manager*

Manufacturing:

Jim Byrne, *Software Manufacturing Product Manager*
Mike Dolan, *Manufacturing Project Manager*
Tony Boglin, *Manufacturing Purchasing*
Bob Johnston, *Manufacturing Engineering Manager*

Customer Services:

Doug Hanzlik, *PRISM Customer Services Program Manager*
Mike Li, *CSSE Systems Engineer*
Glenn Sweeney, *CSSE Software Engineer*
, *CSSE Software Engineer*
Charlie Tharp, *Educational Services*

PRISM Marketing Task Force:

Aaron Holzer, *Computer Aided Engineering and Manufacturing*
Dick Loveland, *Office Information Systems*
Randy Levine, *Laboratory Data Products*
Russell Durkee, *Government Systems Group*
Lee Stommes, *Artificial Intelligence Marketing*

Sonja Israel, *Technology Interface Manager*
Steve Meidell, *Large Systems Marketing*
Dave Quimby, *Mid-Range Systems*
George Hayes, *ULTRIX Marketing*
Jim Mills, *DIGITAL Information Systems*

Preface

The MICA Software Business Plan describes the market, technological and financial impact of the MICA software for PRISM Systems. Other issues addressed include the MICA/VMS and MICA/ULTRIX user interfaces, VMS compatibility, ULTRIX compatibility, programming interfaces, packaging, forecasts, and layered/bundled products.

Associated Documents

1. The PRISM System BUSINESS PLAN describes the market, financial and corporate impacts of using the PRISM architecture to build systems running MICA.
2. The Crystal/Jewel and MICA Documentation Plan describes the hardware and software documentation efforts, respectively.
3. The PRISM System Market and Product Requirements describes general market needs for medium- to high-performance systems and the requirements for specific products to meet those market needs.
4. The PRISM Marketing Plan describes the marketing strategies and programs needed for successful introduction and continued sales of PRISM systems.
5. The MICA Working Design Document (WDD). This document is the software functional specification and it translates the Phase 0 software product requirements into a design document which describes the software functionality and implementation of MICA.
6. The Crystal and Jewel Functional Specification transales the Phase 0 hardware product requirements into a design document which describes functionality and implementation.
7. The Crystal/Jewel and MICA Project Plan is the integrated development plan specifying commitments and costs for all groups necessary to achieving project objectives.
8. The PRISM Sales Impact Document describes the goals, strategies, and sales requirements for the Crystal/Jewel and MICA products.
9. The Alternatives/Feasibility Study examines the project risks and techniques for minimizing those risks.
10. The PRISM Manufacturing System Impact Statement outlines the overall manufacturing strategies for Crystal/Jewel hardware and MICA products.
11. The PRISM Systems Customer Services Combined Requirements Document describes the goals, strategies, and maintenance requirements for both the hardware and software of the Crystal/Jewel and MICA products.
12. The FRS (First Revenue Ship) Language and Tools for MICA Business Plan covers the business rationale for the selection of FRS products for PRISM Systems. This plan is produced by SDT (Software Development Technologies).
13. The Emerald Phase 0 Plans includes the following: Market and Product Requirements, Business Plan, Alternatives and Feasibility Document, Manufacturing Impact Statement, Sales Impact Statement, and Customer Services Impact Statement. These plans are produced by the Emerald Development Group.

Change History

Date	Issue #	Description
Jan 1987	0.1	MICA Preliminary Software Phase 0 Business Plan, Rev. A
Mar 1986	0.2	MICA Preliminary Software Phase 0 Business Plan, Rev. B

Date and Time of Printing:

11-MAR-1987 15:30:35.85

1 EXECUTIVE SUMMARY

1.1 SYSTEM DEFINITION

PRISM is a new DIGITAL computer architecture designed for **high performance** coupled with a high level of **VAX compatibility**. The PRISM architecture is a careful combination of: an optimized instruction set, instruction-level parallelism, multiprocessing, VAX data types and addressing, and vector operations, all designed to exploit the current state-of-the-art in high-level-language compiler and operating system technology. PRISM is thus a synergism of hardware and software providing more computing power per dollar invested than any previous combination of hardware and software from DIGITAL.

One major reason why DIGITAL is implementing the PRISM architecture is to gain software functionality. After half a dozen major releases, an operating system or layered product becomes encumbered with complexity. Adding new functionality or supporting new hardware can become more difficult as the product matures. In addition, operating system and compiler technology has continued to improve over the years. The PRISM architecture and MICA operating system give DIGITAL the opportunity to take advantage of technical advances made since VAX came to market in 1977 and to build on VAX, VAX/VMS and ULTRIX-32 experience.

MICA is the operating system for PRISM systems. It is a high-performance, general-purpose operating system supporting two user interfaces (MICA/VMS and MICA/ULTRIX), Symmetrical Multiprocessing (SMP), parallelism and vector operations. In addition, MICA is a portable and object-oriented system written in the PRISM System Implementation Language; PILLAR. Users can access either the MICA/VMS *and/or* the MICA/ULTRIX interfaces simultaneously on the same system. MICA has a programming interface providing VAX/VMS compatibility, ULTRIX-32 compatibility and native mode calls and routines. These interfaces enable MICA to support VAX/VMS and ULTRIX-32 application and programming environments. MICA and MICA/ULTRIX support the proposed IEEE P1003 (POSIX) standards.

Crystal and Jewel are the high-end hardware implementations of the PRISM architecture. Emerald is the first midrange PRISM system implementation and is based on the MicroPRISM chip set being developed in Hudson. Crystal is an air-cooled, multiprocessor system, supporting one, two, three or four processors in a single configuration. Crystal processors are either scalar or, with the addition of a vector processing option, combined scalar and vector. A fully configured Crystal system would have four scalar processors, each with a vector processing option. Jewel is a uniprocessor Crystal system with limited expandability that has been optimized for price/performance. Jewel provides customers with a low cost entry to Crystal levels of performance. A vector processing option is also available for Jewel. Emerald is a DEC CMOS II mid-range implementation of the PRISM Architecture. Emerald supports one to four scalar processors or two scalar processors with two vector options. *See the Crystal/Jewel, and Emerald business plans for further details.*

This business plan addresses the MICA operating system (the MICA/VMS and MICA/ULTRIX user interfaces and the programming interfaces), DECnet, C and PILLAR.

1.2 TARGET MARKET AND APPLICATIONS

Based on the current hardware business plans, PRISM systems will be introduced initially into the technical price/performance market. This market is characterized by its criteria for price/performance and by it's dedicated or semi-dedicated use of the system for one job or class of jobs. This market is compute intensive and requires that systems "provide one-sixth to one-fourth the performance of a super-computer at one-tenth the cost or better" (Dataquest, Introduction to Mini-Supercomputer 1986). PRISM systems will be focused initially in the following compute-intensive, price/performance application areas: Engineering, Design Automation, Earth Resources, Education, Government, Imaging/Graphics, Scientific, Software Development, and Real-Time.

As additional layered products and third party software become available, systems can then be sold into the technical general purpose and commercial markets. Products and applications for the technical general purpose market will be available six months to two years after FRS and one to two years after FRS for the commercial market. To satisfy these long-term PRISM systems market goals, the MICA operating system is a general purpose operating system, as opposed to a technical special-purpose operating system.

Based on the work of the High End Task Force, Crystal and Jewel's market in 1990 will be 100% technical price/performance. By 1995, 67% of their market will be technical price/performance, 10% will be technical general purpose, and 23% will be commercial. In 1990, 18% of DIGITAL's high-end new systems sales ($) will be Crystals and Jewels; by 1995, 36% will be Crystals and Jewels.

Emerald %??

1.3 MICA OPERATING SYSTEM

The "best" features of both VAX/VMS and ULTRIX-32 have been merged to create a system unlike either in its internals, and yet, from a user or programming perspective, appears the same as the original. The MICA general purpose operating system includes:

- a VMS-like user interface (MICA/VMS)
- an ULTRIX-like user interface (MICA/ULTRIX, an AT&T license is required and included)
- a native system management interface
- a VMS compatibility programming interface [VAX/VMS Compatibility Library (VCL)]
- an ULTRIX programming interface (unencumbered)
- a native programming interface
- an executive and kernel

INCLUDE PICTURE OF SYSTEM HERE

MICA supports:

- VAX data types
- Shared file access with heterogeneous systems via domain group groups (*See Section 9.1.3.6, "domain group groups", for further details*)
- A subset of DECnet Phase V and TCP/IP
- ANSI magnetic tape, therefore, ANSI magnetic tapes are interchangeable between VAX/VMS, ULTRIX-32 and MICA
- ODS-2-extended file system (extended for ULTRIX-32 features and internationalization)
- PRISM architectural features, such as, SMP, vectorization and parallelism

The following is either under consideration for FRS or will be provided in future releases:

- DECWindows (XLIB, client software only) and License Management Facility (LMF) is highly desired and is currently under evaluation for FRS. DECwindows support will enable PRISM systems to communicate with VAX workstations.
- The XI architecture will be supported when it becomes available

- NFS* will not be supported at MICAs FRS, however, it is a goal to announce NFS with the PRISM program announcement

A MICA system can run **either or both user interfaces (MICA/VMS and/or MICA/ULTRIX) concurrently on the same system**. For example, One user can be logged into the MICA/ULTRIX interface, while another user can be logged into the MICA/VMS interface. This also means, that a user logged into MICA/VMS can run a MICA/ULTRIX application and a user logged into MICA/ULTRIX can run a MICA/VMS application. Applications or programs written in a higher level language are 100% source code compatible between VAX/VMS, ULTRIX-32 and MICA systems. There will be one version of a layered or bundled product which will run under either the MICA/VMS or MICA/ULTRIX user interface, this is accomplished by the common executive.

MICA provides an MICA/ULTRIX compatibility programming interface for compatibility with ULTRIX-32, DIGITAL's implementation of AT&T's UNIX™ operating system. The MICA/ULTRIX compatibility programming interface provides system services and library routines that are compatible with ULTRIX-32 and IEEE P1003.x (POSIX) interface definitions. Berkeley 4.3 and AT&T System V compatibility will be provided through alternate run-time libraries.

VAX/VMS and ULTRIX-32 compatibility is a major goal of MICA. To the *user* or *programmer*, MICA/VMS appears to be VAX/VMS and MICA/ULTRIX appears to be ULTRIX-32. A user usually can run their application or program with few or no changes after recompiling and relinking their program on a PRISM system. All the important VAX/VMS and ULTRIX-32 features that define VAX to our customers are supported by MICA.

See section 9 for further details on the MICA operating system.

1.3.1 MICA New Features

The ability of one system to support two user interfaces is a new and unique feature for DIGITAL. Within one system DIGITAL not only supports an industry standard operating system but also a proprietary operating system.

User programs can exploit parallelism and vector operations for increased performance using decomposing and vectorizing compilers. There will be, at a minimum, an automatic vectorizing and user-directed decomposing FORTRAN available for FRS. FORTRAN will also have the ability to do loop unrolling at FRS. Automatic decomposition and parallelism for FORTRAN, as well as for other compilers, is being evaluated by Software Development Technologies (SDT) and will be addressed at their Phase 1.

Multithreading is supported at FRS, this supports parallel decomposition within FORTRAN programs.

In support of vectorization, there is a vector version of the Math Library at FRS. VECTOR EMULATION MODE ???

Symmetric Multiprocessing (SMP) is an integral part of the MICA design and is supported at FRS.

Processor affinity (the ability to dedicate one processor to a specific task) is supported by MICA. In addition, an uneven mix of scalar and vector options is also supported (i.e., 4 scalar processors and 1 vector option in a system).

For disks there is software-implemented support for volume sets, shadowing, and striping.

---

* NFS is a trademark of Sun Microsystems

TM UNIX is a registered tradmark of AT&T

1.4 OTHER SOFTWARE PRODUCTS

FORTRAN, PASCAL, C, Bliss * and PILLAR are supported at MICA FRS. Other products/utilities that are available at FRS are; Debugger, TPU, Sort/Merge, PCA, DECnet, Math Library and LSE. Products that are being examined for FRS are: GKS, PHIGS, Rdb, CMS, MMS, DTM and Tools Integration. Ada will not be available until after MICA FRS, although it is a goal to announce it with the PRISM program. *(Refer to the SDT PRISM Products Phase 0 Business Plan for further details.)* Some AI products may be available at PRISM FRS, but no Phase 0s have been held for these products to date.

The goal is that all products needed to support the PRISM market will be ported/rewritten to the PRISM Architecture within two years after FRS. Based on current marketing plans, second wave products (products that will be available six months to two years after MICA FRS) will include products which will support the technical general purpose and commercial markets. Technical general purpose products *might* include: Ada, Rdb, AI products, communication products, and other compilers. Commercial products *might* include: OLTP, office products, and 4GL products. Products that support IBM connectivity will also be important for PRISM systems. SDT second wave products will be opening their Phase 0s shortly. The PRISM engineering groups will work closely with the market groups to insure that important third-party application packages are available at or soon after FRS. Availability of additional layered products and third-party software will be critical to the success of PRISM systems in the technical general purpose and commercial markets.

1.5 SCHEDULES

1.5.1 Phase Planner

Table 1 Phase Planner

Phase	Date of Exit
0 - Strategy and Requirements (PBU)	November 1985
(PRC)	May 1986
1 - Planning	June 1987
2 - Implementation	October 1988
3 - Qualification	March 1989- Jewel
	September 1989- Crystal
4 - Production & Support	TBD
5 - Retirement	TBD

2 GOALS/NON-GOALS, ASSUMPTIONS AND CRITICAL DEPENDENCIES

2.1 GOALS

2.1.0.1 Development Goals

The development goals for MICA are:

- Provide a **quality** operating system supporting DIGITAL's image in the marketplace.
- Provide a **robust, extensible,** and **maintainable** operating system.
- Provide a contemporary operating system to support **new functionality** for the PRISM Architecture, i.e., **parallel processing, symmetric multiprocessing and vectors**.
- Provide an operating system that supports **two user interfaces** (MICA/VMS and MICA/ULTRIX).

* Bliss will be made available if needed by customers, however, it will not be marketed/sold. It will also be available to DIGITAL Engineering to aid in the porting of products to the PRISM Architecture.

- Provide an operating system which supports **VAX/VMS and ULTRIX-32** applications and programs written in a high-level language.
- Meet current **schedules**.
- Provide a **high-performance and general purpose** operating system which supports the high-performance, compute intensive PRISM market.
- Provide software **family pricing** which complements PRISM systems and their market.

2.1.1 Manufacturing Goals

The manufacturing goals for MICA are:

TBS

2.1.2 Service Goals

The service goal for the PRISM products is to support the strategic goals set forth in the service organizations (customer satisfaction, revenue growth and profitability) and support the goals of the PRISM software. The PRISM and MICA service goals are:

- Provide an important **competitive advantage** for PRISM systems by being a full-service vendor and offering superior services to those offered by near-supercomputer vendors.
- Lead in **customer satisfaction** by providing the best service the industry has to offer in this market. Achieve a customer satisfaction rating that will support DIGITAL's image of a quality service vendor.
- **Integrate hardware and software product support** into our contracts and our service delivery mechanisms to achieve high customer satisfaction and cost-effective service.
- Ensure the service organizations **support the product at FRS** through effective training, availability of service tools, resource and capital equipment planning.

2.1.3 Training Goals

The training goals for MICA are:

TBS

2.2 NON-GOALS

- It is a non-goal to support the UNIX Computer Science/Research market. This market requires that the kernel/executive be written in C and the ability to make modifications to the kernel/executive. No provision are made in MICA to support kernel-level modifications.
- It is a non-goal to provide binary code (executable images) transportability between VAX/VMS, ULTRIX-32, UNIX and MICA systems.
- It is a non-goal to support *all* VAX/VMS system calls, as some VAX/VMS system calls are VAX architecture specific and have no corresponding or equivalent function on PRISM systems. (ROCKIE—SAME FOR ULTRIX?.????)
- It is a non-goal to track VAX/VMS Ver 5.*x* feature-per-feature. Not all features added to releases after Version 5 may be needed by MICA, each feature will need to be individually evaluated for inclusion in MICA. (ROCKIE—ULTRIX??)
- It is a non-goal to provide technical general purpose and commerical products for the FRS of MICA.

2.3 ASSUMPTIONS

2.4 CRITICAL DEPENDENCIES

Table 2 Critical Dependencies

		CRITICAL DEPENDENCIES
1.	SDT:	SDT will provide all Version 1 languages and products (other than PILLAR and C). These products are: FORTRAN, Bliss, Pascal, Debug, TPU, LSE, PCA, Sort/Merge, RTL, Math Library. The following are under consideration as SDT Version 1 products: GKS/PHIGS, CMS, MMS, and Tools Integration.
2.	Other Layered Product Groups:	Rdb, Ada and AI products are also desired as FRS products, but are not committed by Engineering. All software products that are needed to make PRISM systems successful in its target markets should be available within two years after MICA FRS.
3.	VAX/VMS Group:	The VAX/VMS group must allow for non-VAX/VMS (i.e., heterogeneous) cluster/domain group members. DECwest must have access to VAX/VMS sources, so that they may be ported (or rewritten) for MICA. License Management Facility (LMF) and its administrative systems must be available and working successfully on VAX/VMS systems before it can be implemented on PRISM systems. DECWindows (XLIB) must be written in a highly transportable language before it can be ported to and available for the FRS of MICA.
4.	ULTRIX-32 Group:	The following is needed from UEG: sources for ULTRIX-32, assistance in porting utilities, and licensing coordination for the encumbered portion of MICA/ULTRIX.
5.	Prototypes:	DECwest Software Engineering must have access to prototypes nine (or twelve???) months before FRS (before Customer FT???)
6.	Third Party Applications:	Third party applications for MICA will need to be available at or soon after FRS. Seed units will need to be available for third party vendors.
7.	Educational Services:	Internal and Customer Educational Service products are desired by the field test dates, with availability no later than FRS.
8.	Customer Support Center (CSC):	CSC support at FRS is critical to the success of MICA.
9.	Customer Services (CSSE) and Software Product Services (SPS):	Appropriate software product services for the PRISM market is critical to to the success of MICA.

3 TARGET MARKETS AND APPLICATIONS

3.1 TARGET MARKETS

"Since 1982, a number of new companies have formed with the common objective of building and selling computers that would occupy the 'performance gap' between high-end superminicomputers and supercomputers. Their machines are generally claimed to provide one-sixth to one-fourth the performance of a supercomputer at one-tenth the cost or better." [Source: Dataquest, Introduction to Mini-Supercomputers, 1986]

"12% of the sales that were lost in FY85 went to RISC/Vector vendors (e.g., Ridge, Pyramid, Elxsi, Convex and Alliant). This trend is expected to continue in FY86 [Source: Joel Berman, Long Range Sales Planning].

The 1986 VAX Positioning Study cites a dramatic rise in the importance of compute power to DIGITAL customers, particularly in the technical segment. This attribute was ranked 30th and 29th overall in 1984 and 1985 respectively; in the 1986 study, it rose to the 10th most important customer buying factor. In the technical segment, compute power was the 3rd most important attribute cited. It is clear that DIGITAL cannot ignore the need for higher performance systems, and the increasing competitive threat offered by the new, higher performance near-supercomputers.

The marketing strategy for PRISM systems is to *initially* focus on specific application niches in the technical market segment where new technologies (vector/array processing, symmetric multiprocessing and parallel processing) incorporated into the PRISM Architecture have already achieved market acceptance. By focusing on specific application niches, DIGITAL will minimize its exposure in introducing a new architecture. The marketing strategy for PRISM systems will become broader in scope after FRS. Availability of additional layered products and third party applications will enable PRISM systems to be sold into the technical general purpose and commercial markets.

The marketing strategies for the MICA operating system are to supply a high-performance, state-of-the-art, leadership product which supports the new technologies that have been designed into the PRISM Architecture, and to support the VAX/VMS and ULTRIX-32 application and programming environments.

It is a *non-goal* to support the following UNIX markets: Computer Science/Research, Telecom, and government (RFPs/direct sales) markets (although some limited opportunities could exist in these markets for MICA/ULTRIX). Rather, the *goal* is to support the UNIX technical and commercial application industry market. The customers is this market want an industry standard operating system so they can achieve some degree of vendor independence/flexibility, have application portability across different vendor systems, and so they can control training costs by having "one" operating system. In addition, this market does not need the ability to make kernel-level modifications to their system. These customers also want excellent service and support for their UNIX system so they often choose one of the top vendors in the market, i.e., IBM or DIGITAL. Examples of customers in this market are: Ford Engineering, General Motors, TRW, NBC, and Disney.

3.2 APPLICATIONS

It is a goal to support any third party application software package that will be needed by our marketplace. DECwest and the Emerald group will be actively working with the marketing groups in identifying which vendor packages will be needed to help market PRISM Systems, and in ensuring that those vendor packages are in place by or shortly after FRS. Current plans are to make seed units available to major applications vendors.

PRISM systems are particularly well-suited to compute intensive applications in the following areas:

Table 3 Third Party Application Software

Application	Subapplication (Example(s))
Electrical CAD	Circuit & Logic Simulation (SPICE, HILO), VLSI Layout & Routing (SCICARDS), Design Rule Checking
Mechanical CAD	Solids Modeling (EUCLID, ROMULUS), Finite Element Analysis (ANSYS, ABAQUS), Piping Design (ADLPIPE)
Earth Resources	Seismic Analysis and Mapping (BMAP, ZMAP), Reservoir Modeling (BETA II, THERM)
Scientific	Computational Chemistry,Fluid Dynamics (FLUENT, FIDAP), Nuclear Physics (PISCES), Monte Carlo Simulation (MCSIM)
Graphics	Animation, Image Processing (MAPSS = MAGIC)
Software Development	Cray Front-End Processing, 3rd Party Application Development, Artificial Intelligence
Government	Military Command & Control, Signal Processing, Cryptography, Weapons and Energy Research
Education	Computing Resource for University Departments in Engineering, Computer Science, Physics, and Chemistry

Technical general purpose and commercial applications will be addressed in a later revision of this business plan.

TERRY-IS THE ABOVE STILL UP TO DATE??

4 COMPETITIVE ANALYSIS

Based on the hardware business plans, the major competitors for PRISM systems are: Alliant, Convex, Cray, IBM and SCS.

PRISM Systems— PRISM systems will have many advantages over IBM and the mini-supercomputer vendors. Our advantages are:

- better or equal **price/performance**
- better **absolute performance**
- DIGITAL **networking**
- **VAX family compatibility**
- **PRISM family compatibility**
- a **state-of-the-art** operating system
- an operating system which supports **VAX/VMS and ULTRIX-32 application and programming compatibility**
- highly optimized and contemporary **compilers** (decomposing and vectorizing)

- DIGITAL **sales and support**
- an unique operating system which has **two user interfaces** that offer the best of the VAX/VMS and ULTRIX-32 (UNIX) worlds.
- DIGITAL will be offering an operating system that meets **industry standards** (IEEE P1003.x, POSIX) and is **proprietary** at the same time

Alliant—Alliant currently has two products on the market, the FX/1 and the FX/8. Net sales in 1985 were $4.406M, and as of September 1986 they were $18.443M. Fifty-four systems have been sold to twenty-nine customers as of October 1986. Alliant has agreements with fourteen companies for twenty-two third party application programs. Their future goals are to: expand sales in areas with high concentrations of technical industries and scientific applications (for 1987); expand OEM activities; and continue to produce enhanced compilers and additional tools. [Source: Mary Rodock, memo on Alliant Prospectus, December 1986]. The current market share in 1985 for Alliant systems in the $500K to $1M range was less than 1%, however, Alliant is definitely in the race for market share in the mini-supercomputer market.

In the Datamation annual mini/micro survey (based on 6,900 responses), 18.5% stated that if they were interested in multiprocessing or parallel processing they would consider an Alliant system. [November 1986] Alliant is currently targeting systems in the following market areas: computer-aided and design, circuit and semiconductor simulation, signal processing, geophysical analysis, aerodynamic simulation, molecular modeling and mathematical algorithm development. Apollo is an OEM for Alliant and Sun is a marketing partner. This enhances their marketability in the scientific and technical marketplace.

Concentrix, the Alliant operating systems, is an extended BSD 4.2 UNIX and supports the C and Bourne shells. TCP/IP, and IEEE 802.3 are supported. Their FORTRAN is a automatic decomposing, vectorizing, and VAX-compatible compiler. C and PASCAL are also offered on Concentrix and neither is a decomposing or vectorizing compiler. EMACS and vi are the supported editors. Alliant claims that their systems are fault tolerant; if a processor shuts down, it is removed from the processing loop and its task is allocated to other processors. Alliant has a support center with a toll free line and they guarantee a one hour response.

Alliant claims that the FX/series performance is achieved by combining several features that distinguish it from other computer systems: parallel processing, multiprocessing, and vector processing. The FX/Series allows up to eight computational processors to work on a single application simultaneously, and it has the ability to run most standard FORTRAN programs in parallel with little or no re-programming. Their FORTRAN automatically detects the potential for parallel and vector processing in a program, and informs the programmer on the methods of structuring code to better take advantage of parallel and vector features. Alliant systems can run multiple programs simultaneously on different processors. They have introduced a job scheduling feature for the FX/8 that gives the customer the ability to target an application for either multiprocessing or parallel processing. Under this new configuration, large computational jobs designed for parallel processing can be run on a group of processors, while smaller tasks can be assigned to the remaining processors.

Convex—Convex was the first company to introduce and ship standalone interactive systems in the emerging mini-supercomputer market. Revenue in 1985 was $13.5M, and the first half of 1986 it was $16.6M. They have sold one hundred and ten systems (October 1986 product announcement). Customers are now beginning to purchase their second and third machines from Convex. They have about 32 third party application packages available (as of first half of FY86). Convex is targeting their systems to: aeronautical engineering, seismic processing, computational chemistry, signal and image processing, computational fluid dynamics and structural analysis. In addition, Convex is currently expanding its market to include the astronomy, astrophysics, chemistry, and biology markets. Convex

had a 1% market share in systems priced from $500K to $1M in 1985; this is expected to continue to grow in the future. In the above mentioned Datamation survey, 20.8% of the respondents stated that they would consider a Convex system.

The Convex operating system is an extended BSD 4.2 UNIX; features include a hierarchical file management system and disk striping. They currently have a vectorizing, VAX-compatible FORTRAN compiler, C and CFT (Cray FORTRAN compiler). They have announced plans for Ada and a vectorizing C compiler. TCP/IP and HYPERchannel are supported and they are currently working with Sun to support NFS. Several "rumors" have been floating around about development on a shell which converts VAX/VMS DCL commands to Convex operating system commands (CONVUE, Convex-VAX User Environment). Electronic mail, text editors, a token ring network, debugger tools, file maintenance, and systems accounting are included in the operating system.

The Convex C-1 family is a 64-bit integrated scalar and vector processor which uses a Cray-like architecture. Their integral vector processor and ability to take advantage of vectorizing software techniques, which have been developed over the last ten years for the Cray, works to their advantage. Convex appears to be able to provide reasonable support for their products and they stand behind what they sell. Their small sales force sometimes has an advantage over DIGITAL in that they are able to: pull in high-level technical or business expertise from Convex for a sales call (i.e., the President of Convex may participate in a call); demonstrate a superb technical knowledge of their product (they have few products);and deliver products quickly after an order is placed (little internal red tape). Convex will often let a potential customer "borrow" a system on site for a period of time. The customer will then target their application to the Convex system (note: with the VAX sitting in the background handling the normal work load). This can be a very effective selling tool (modeled after IBM).

Cray

IBM—What can one say new about IBM, except to reinforce the idea that an IBM system can sell itself, simply because it is an IBM system. The IBM systems that will compete against PRISM systems are: the 3090 series; its follow-on, the Summit series; the 4381-14; its follow-on, the Olympia series; and the 9370 series. VM is viewed as our IBM operating system competition in the technical market, with MVS the IBM operating system competition as PRISM systems move into the and commercial market. The vector facility for the 3090 series firmly places IBM as a competitor against PRISM systems in the technical market. In many cases the departmental MIS manager has added the vector facility to placate the technical/scientific user.

VM is enjoying a rebirth in the mainframe market and has been extended to the minicomputer market with the introduction of the 9370. There have been rumors that IBM is working on a version of VM for high-end personal computers. If this does happen, VM may bridge a gap in mainframe-to-PC connectivity. IBM customers like the VM ability to run multiple operating systems, however, there can be a performance hit of 10 to 20% depending on the workload.

There have been few sales of IX/370 (IBM's version of UNIX) which runs under the VM operating system, however it is not in full commercial release. Today MVS or VM are the operating systems of choice in the mainframe market; UNIX has not penetrated the IBM technical or commercial markets.

Future trends for MVS and MVS/XA (and possibly VM) appear to be: new functionality might be offered separately; higher functionality in exchange for a higher price; redefinition of functionality in terms of machine performance; and higher prices. IBM is striving to make more of its profits from software than hardware. Graduated pricing (similar to our tier pricing) was announced for some software systems in October 1986.

A variety of layered products and compilers, as well as third party application products, are available for VM and MVS systems. In addition, IBM has the reputation for reliable products and a highly rated sales and support organization. Customer satisfaction is one of IBM's major competitive tools. Motherhood and apple pie (as well as Charlie Chaplan) can indeed sell systems.

IBM had a 79% market share in systems priced over $2M in 1985, 65% in system priced $1M-$2M, and 50% in systems priced from $500K-$1M. Their market share alone makes them the default standard in many situations. DIGITAL has currently attracted IBMs attention in the minicomputer market. There are rumors that IBM has "DECK DEC" teams currently in place, PRISM systems will have to deal with the attention IBM will direct towards DIGITAL in an effort to thwart PRISM sales.

SCS (Scientific Computer Systems)

5 TECHNOLOGICAL CONSIDERATIONS AND IMPLICATIONS

Vector processing, symmetric multiprocessing, and parallel processing are well enough understood to make them a science (rather than a black art), and therefore, are essential to attaining a competitive architecture. Because these technologies are well understood and are being implemented into the VAX Architecture (with the exception of parallel processing) it is felt that these are a minimal risk to the PRISM Architecture, the MICA operating system and to DIGITAL.

MICA supports vector processing, but the ability to vectorize code is dependent on the availability of vectorizing compilers. The goal for FORTRAN is to support vector processing through GEM, the SDT compiler common backend. Therefore, other compilers which utilize GEM may also support vectorization. Vectorization is not viewed as a major technological risk to PRISM systems as vectorization is a technology which is understood within DIGITAL and the industry.

Advanced development work has been done on decomposition for parallel processing by SDT, however, currently there are no DIGITAL decomposing compliers available. It is SDT's goal to support decomposition. Directed decomposition for FORTRAN is committed for FRS. Automatic decomposition is being examined; final commitments will be addressed by the layered product groups at their Phase 1. Because decompostion, especially automatic decompostition, is a technology which is just coming unto its own, there is some risk associated with it.

MICA is designed to support SMP. Because SMP and Multiprocessing are widely understood and are integral to the architecture, the hardware, and the software; these are viewed as a minimal risk to the system.

The users ability to run two user interfaces simultaneously on the same system is one of the unique features of the MICA operating system. Because VAX/VMS and ULTRIX-32 are well understood products it is viewed as a minimal risk to support this feature.

6 GROUP FIVE YEAR STRATEGY

MICA five year goals are:

- Supply a portable high-performance, general purpose operating system which will support the long-range market goals of the PRISM program
- Support the wide range of layered product and application software that will be needed to support the PRISM program as it expands into new markets
- Provide support and new functionality for future releases of MICA
- Support future versions of the IEEE P1003.x standard
- Support future PRISM hardware systems (including 64-bit PRISM systems)

- MORE???

7 QUANTITATIVE SUMMARY

7.1 FORECASTS

7.1.1 Packaging

Three software packaging options will be offered for PRISM systems:

1. MICA (which includes both the MICA/VMS and MICA/ULTRIX user interfaces)
2. MICA/VMS
3. MICA/ULTRIX

The MICA option has been priced to encourage customers to choose it as an option. A typical PRISM customer may have a mixture of different vendors hardware and operating systems on site, such as VAX/VMS, ULTRIX-32, UNIX (Alliant, Convex or others), and IBM (VM or MVS) systems. The next logical step is for a VMS-like and ULTRIX-like operating system to be together on one system. By pricing MICA attractively, customers will be encouraged to choose the MICA option (option #1). The seperate MICA/VMS license (option #2) and seperate MICA/ULTRIX license (option #3) are each priced at 85% of the MICA option. (The MLP for upgrading/adding a user interface is 15% of MICA.) Our forecast projects that 50% of the customers will purchase MICA, 30% will purchase MICA/VMS, and 20% will purchase MICA/ULTRIX. (Another way to look at it is, 80% of the customers (50% + 30%) will have MICA/VMS and 70% (50% + 20%) will have MICA/ULTRIX.)

Each of the above mentioned packaging options (MICA, MICA/VMS, MICA/ULTRIX) includes the following:

- DECnet and TCP/IP
- Two system implementation languages (C and PILLAR)
- TPU, Debug, Sort/Merge, Math Library
- System Management Interface
- Programming Interfaces (Native Mode Libraries, ULTRIX Compatibility Programming Libraries, and VMS Compatibility Programming Libraries)
- Base system (Executive and Kernel)

Software part numbers will be assigned for DECnet, TCP/IP, C, PILLAR, TPU, Debug, SDT RTLs, Math Library and Sort/Merge.. Revenue for these products are included in the system pricing, and revenue credit for these products will be arranged with the respective engineering groups. This will enable the responsible engineering groups to potentially show a profitable P&L statement, justify headcounts, and have a flexible release schedule which is not necessarily dependent on the operating system release schedule. (Flexible release schedules will be dependent of future distribution plans by the SDC, LMF and CDROM.) This will also enable engineering groups to coordinate product releases for VAX and PRISM versions. This method of including the "bundled product" Q-numbers in the operating system kit Q-number will also allow ease of ordering (simplicity) for customers.

In the future, when there is a greater breath of software products available for PRISM systems, these products may be able to be repackaged (possibly to allow for an application run-time system) and offered seperately from the operting system. In the meantime, this method allows for flexibility of schedules, revenue credit and ease of ordering. (For MICA Phase 1 revenue credit for these products have been estimate, however, they will change when the SDT products go through Phase 1 exit.)

% for DECnet, C , Pillar??

sources??

7.1.2 Unit Forecast

Table 4 Hardware Unit Forecast

System	FY89	FY90	FY91	FY92	FY93	FY94	FY95	TOTAL
Emerald*	120	1530	2580	4320	2670	NA	NA	11220
Emerald-box*	-	230	540	870	525	NA	NA	2165
Jewel**	-	250	626	833	793	673	248	3423
Crystal**	-	54	179	253	275	240	93	1094
Total	120	2064	3925	6276	4263	913	341	17902

* Unit forecast numbers for Emerald are from the Phase 0 (Revision 1) Business Plan

** Unit forecast for Crystal and Jewel are from the Phase 1 Business Plan

Assumptions:

- 10% of the hardware units are sold with no DIGITAL software license, customers will be purchasing UNIX licenses (i.e., BSD or System V)

Table 5 MICA Unit Forecast

System	FY89	FY90	FY91	FY92	FY93	FY94	FY95	TOTAL
Emerald	54	689	1161	1944	1201	NA	NA	5049
Emerald-box	-	103	243	392	236	NA	NA	974
Jewel	-	112	282	375	357	303	111	1540
Crystal	-	24	80	114	124	108	42	492
Total	54	929	1766	2824	1918	411	154	8056

Table 6 MICA/VMS Unit Forecast

System	FY89	FY90	FY91	FY92	FY93	FY94	FY95	TOTAL
Emerald	32	413	697	1166	721	NA	NA	3029
Emerald-box	-	62	146	235	142	NA	NA	585
Jewel	-	67	169	225	214	182	67	924
Crystal	-	14	48	69	74	65	25	295
Total	32	557	1060	1695	1151	246	92	4833

Table 7 MICA/ULTRIX Unit Forecast

System	FY89	FY90	FY91	FY92	FY93	FY94	FY95	TOTAL
Emerald	22	275	464	778	481	NA	NA	2020
Emerald-box	-	41	97	157	94	NA	NA	389
Jewel	-	45	113	150	143	121	44	616
Crystal	-	9	32	46	50	43	17	197
Total	22	372	706	1130	767	164	61	3222

7.1.3 MICA Software Licensing

A systems approach to pricing has been taken in calculating PRISM software licenses, as system level pricing (and cost of ownership) is important to our customer base. In addition, VAX/VMS and ULTRIX-32 prices have played a major role in the shaping of the software pricing strategies, as comparisons will be made by the DIGITAL customer base. The license price of MICA Version 1 and its full range of functionality will be compared to the license price of VAX/VMS Version 5.x (and/or ULTRIX-32) and its full range of functionality. Another major factor influencing pricing is the market in which these systems will be introduced; the technical price/performance market. Systems in this market need to priced for price/performance, as Convex and Alliant (or the new "kid" on the block) will drive the pricing in this market. Simplicity and consistency are the last major factors influencing software pricing. MICA supports up to 32 processors (SMP). If SMP (performance tiers) was the major factor in determining pricing then there could potentially be 32 pricing tiers per each processor-type, which would introduce a high level of complexity into a pricing strategy.

"Family tier pricing" is the method which is used to price PRISM software licenses. In addition, the software license is priced as a percentage of the Average System Value (ASV). The family pricing tiers for PRISM systems are:

- workstations
- micro's
- bounded systems
- expandable systems I (1-4 processors)
- expandable systems II (5-16 processors)
- expandable systems III (17-32 processors)

(The ASV is determined by calculating the overall ASV of the forecasted units in that processor family, which then allows pricing for the "design" center of that family.)

These tiers (and percentages of ASVs) allow the ability to charge for a range of performance and yet its structure is simple, consistent and straight forward. These "family tiers" convey a message of software-familiy-consistency to our customer base and enable the software licensing to be consistent with *system* pricing. By having the tier be a percentage of the ASV, the software pricing strategy matches the hardware pricing strategy, whether the strategy be price/performance or high profit margins; the end result (if managed correctly) is *system pricing*.

Following is the proposed MICA, MICA/VMS and MICA/ULTRIX license pricing:

Table 8 MICA Software Licensing Summary

CPU	VUPS(1 = 780)	Paid Up Price ($K)	1st Year Rental * ($K)	2nd Year Rental ** ($K)
Emerald tier-expandable 1	15-45	38.1	14.5	11.8
Emerald-Box tier-bounded	15?	7.2	2.7	2.2
Crystal tier-expandable 1	30-96	171.1	65.0	52.8
Jewel tier-bounded	30	64.5	24.5	19.9

* First year rental includes initial license fee and twelve monthly payments

** Second year rental includes only twelve monthly payments

Table 9 MICA/VMS or MICA/ULTRIX Software Licensing Summary (One User Interface)

CPU	VUPS(1 = 780)	Paid Up Price ($K)	1st Year Rental * ($K)	2nd Year Rental ** ($K)
Emerald tier-expanable 1	15-45	32.4	12.3	10.0
Emerald-Box tier-bounded	15?	6.1	2.3	1.9
Crystal tier-expanable 1	30-96	145.4	55.3	44.9
Jewel tier-bounded	30	54.8	20.8	16.9

* First year rental includes initial license fee and twelve monthly payments

** Second year rental includes only twelve monthly payments

Following is the methodology that was used in calculating the above licenses:

- An assigned percentage (see below) of the Average System Value (ASV) equals the base price for MICA. (The base price is a number from which paid up licenses, initial and monthly rental are calculated. The base price is a number which is only used internally for price calculations.)
- The initial rental fee and monthly rental payments are calculated from the base price. The initial fee is calculated at 10% of the base price, this is only paid the first year.
- First years rental represents the initial fee and 12 monthly payments.
- Second and subsequent years rental represents 12 monthly payments.
- The base price times 140% eguals the paid up license fee, for customers who do not want to rent software, but prefer to pay for the license at the time of the systems purchase. Because the paid-up license is more than the rental license, some customers may choose to rent versus buy. As the corporation gains experience in rental license, MICA licenses may change to take advantage of that knowledge.
- Pay back period for customers who have rental licenses is 25 months.
- MICA/VMS and MICA/ULTRIX base price, initial fee, monthly payment and paid up license is calculated at 85% of MICAs pricing.
- If processors are added to a system, an additional software licenses (upgrade) is not required, unless by adding a CPU the customer enters a new tier.

The Base price is calculated by using an assigned percentage of the ASV. *The PRISM operating system group should be contacted for exact pricing for new hardware systems, these figures should be used only as a reference.*

- Workstations—The base price is calculated using 4% of ASV

- Micro's—The base price is calculated using 6% of ASV
- Bounded Systems—The base price is calculated using 6% of ASV

  i.e., Emerald-Box and Jewel
- Expandable Systems (up to 4 processors)—The base price is calculated using 8% of ASV

  i.e., Crystal and Emerald
- Expandable Systems (5 to 16 processors)—The base price is calculated using 10% of ASV
- Expandable Systems (17 to 32 processors)—The base price is calculated using 12% of ASV

7.1.4 Pricing Comparisons (External and Internal)

THE FOLLOWING SECTION WILL BE DELETED IN THE PUBLISHED PLAN AND USED AS A BACKUP. VMS AND VAX PRICING IS SUCH A MOVING TARGET THAT THIS DATA CHANGES TOO FAST TO PUBLISH IT. ALSO APPLES TO APPLES COMPARISION IS DIFFICULT AS VAX AND VMS DON'T SEEM TO PLAY BY THE SAME RULES... **VAX/VMS and MICA Software Licenses**—The following table compares VAX/VMS and MICA licenses. (The VAX/VMS pricing is an estimate which has been provided by Alan Arsenault. These prices have not gone through pricing approval and therefore should not be considered final. They are used here for comparision and planning purposes only.)

If the paid up license is compared as a percentage of the ASV then MICAs licenses are compareable or below the VAX/VMS pricing estimates. Since MICA offers less functionality than VAX/VMS and ULTRIX-32 at FRS (MICA is not comparable to a Version 5 of VAX/VMS or a Version 2 or ULTRIX-32, MICA is a Version 1 product), pricing which is below or close to VAX/VMS or ULTRIX-32 is desireable.

Table 10 MICA and VAX/VMS* Software Pricing Comparison

CPU	VUPS(1 = 780)	Paid Up Price ($K)	Paid Up is x% of ASV	1st Year Rental ($K)	2nd Year Rental ($K)
Emerald-Box	15?	7.2	8%	2.7	2.2
Emerald	15-46	38.1	10%	14.5	11.8
Argonaut 1	12	81.0	12%	25.8	21.0
Argonaut 2	22	135.0	11%	42.9	34.9
Argonaut 4	43	256.0	10%	81.6	66.2
Jewel	30	64.5	9%	24.5	19.9
Crystal 1	30	171.1	17%	65.0	52.8
Crystal 2	56	171.1	12%	65.0	52.8
Crystal 3	76?	171.1	9%	65.0	52.8
Crystal 4	102	171.1	7%	65.0	52.8
Aridus 1	21	132.0	22%	42.0	34.8
Aridus 2	38	226.0	25%	72.0	58.4
Aquarius 1	30	182.0	17%	58.1	47.2
Aquarius 2	54	316.0	17%	100.9	82.0
Aquarius 4	100	619	19%	197.4	160.3

* VAX/VMS pricing is an esitmate from Alan Arensault, please contact him for exact pricing.

First year rental includes initial license fee and twelve monthly payments. Second year rental includes only twelve monthly payments.

FOLLOWING IS THE INFORMATION THAT WILL APPEAR IN THE PUBLISHED PLAN: **VAX/VMS and MICA Software License**—Based on VAX/VMS pricing estimates provided by Alan Arsenault and published Argonaut and Aquarius pricing, VAX/VMS paid up licenses range from 10% to 19% of the ASV. MICA's paid up license range from 7% to 17% of PRISMs ASV. Since MICA offers less functionality than VAX/VMS and ULTRIX-32 at FRS (MICA is not comparable to a Version 5 of VAX/VMS or a Version 2 or ULTRIX-32, MICA is a Version 1 product), pricing which is below VAX/VMS or ULTRIX-32 is desireable. Because MICA has been priced as part of the average system value, its price is governed by the same pricing strategies that drive hardware pricing, in this case price/performance. By using this pricing method the software can match the system/market pricing strategies, which is critical to systems level pricing.

Table 11 MICA and External Competitive Software Pricing Comparison

CPU	VUPS(1 = 780)	Paid Up Price ($K)	Paid Up is x% of ASV	1st Year Rental ($K)	2nd Year Rental ($K)

7.1.5 Software Revenue

Table 12 Software Revenue

System	FY89	FY90	FY91	FY92	FY93	FY94	FY95	TOTAL
Emerald								
MICA								
MICA/VMS								
MICA/ULTRIX								
. subtotal								
Jewel								
MICA								
MICA/VMS								
MICA/ULTRIX								
. subtotal								
Crystal								
MICA								
MICA/VMS								
MICA/ULTRIX								
. subtotal								
. subtotal MICA								

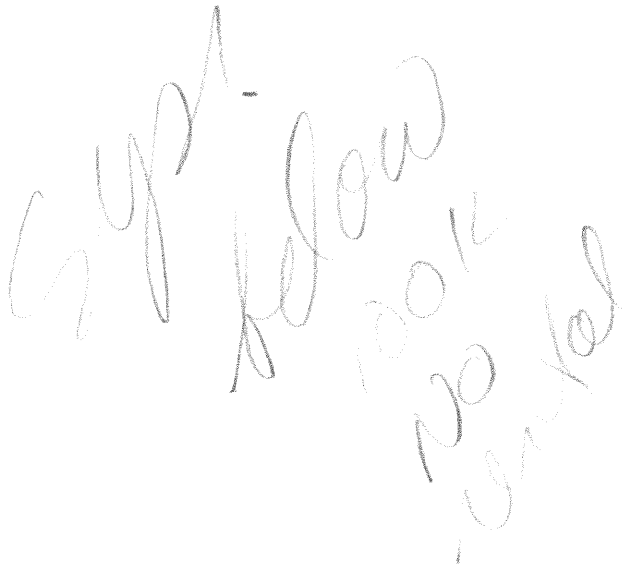

Table 12 (Cont.) Software Revenue

System	FY89	FY90	FY91	FY92	FY93	FY94	FY95	TOTAL
. subtotal MICA/VMS								
. subtotal MICA/ULTRIX								
TOTAL								

Assumptions:

- On the average, 50% of the customers will rent or purchase MICA, 30% will rent or purchase MICA/VMS, and 20% will rent or purchase MICA/ULTRIX
- The MICA and MICA/ULTRIX license includes an AT&T royalty
- Licenses include revenue for DECnet, TCP/IP, PILLAR, C, TPU, Debug, RTL and Sort/Merge.
- Price increase of ??% per year included

7.2 SOFTWARE BUDGETS

Table 13 Software Budgets

System	FY89	FY90	FY91	FY92	FY93	FY94	FY95	TOTAL

7.3 SOFTWARE FUNDING NEEDS

Table 14 Software Funding Needs

System	FY89	FY90	FY91	FY92	FY93	FY94	FY95	TOTAL

8 ISSUES, RISKS AND CONTINGENCIES

8.1 ISSUES

- **D_ and H_ Floating Point Data Types**

  Many customers over the years have used these data types. D_ has been very widely used since the first VAX11/780 supportted G_ instructions in hardware. H_ is not nearly as significant although some customers have used this data type. It is essential that PRISM systems provide a mechanism for this data to be converted to G_ or F_ transparently. The converted data must also return the *exact same answer* as if the data had not been coverted. In addition, we must provide the customer the opportunity to return the data to mass storage (disk) in the original form (D_ or H_ format).

  OK-WHO IS RESPONSIBLE-US OR THE COMPILER GROUPS-WHAT IS C AND PILLAR DOING???

- **VAXcluster/Domain group Participation at FRS**

  PRISM systems must be able to participate in VAXclusters/domain group . PRISM machines will initially be sold into accounts where single system compute power is needed. These will be situations where VAXclusters/domain group can not provide the raw compute power. Most, if not all, of the data will initially reside on VAX systems and will need to be moved. Additionally, users will want to "launch" jobs from VAX systems to run on PRISM systems. It is therefore essential that PRISM systems play very closely and with VAX systems.

  SHOULD WE ANSWER HERE-I AM CONFUSED...??

- **Layered Software Products**
- **Application Software Packages**
- **Software Pricing**
- **Automatic Decomposistion Support in Compilers**
- **NFS Support**
- **DECWindow Support**

8.2 RISKS

There several risks which are unique to MICA and need to be highlighted, they are:

- **VAX/VMS**

  DECwest expects the VAX/VMS group to allow PRISM systems to join in VAX/VMS clusters/domain group as heterogeneous elements.

  Although not a direct risk, it would aid the DECwest effort and lower our overall project risk if the VAX/VMS group would provide specifications (all the better if under ECO control) for such key VAX/VMS components as the On-Disk Structure (ODS-II), RMS, DCL, the lock manager, and cluster/domain group support and management. LMF and DECWindows (XLIB) are also critical to the operating system effort. TERRY IS THIS STILL TRUE????

- **ULTRIX-32**

  DECwest will need access to the ULTRIX-32 sources, so that utilities and shells can be ported to the MICA system. Assistance in porting the utilities and shells to MICA/ULTRIX is needed from UEG. ROCKIER-ANY MORE??

- **Vector Processor**

  DIGITAL has never delivered a vector processor before, nor has it ever developed software (compilers) to support vector processors.

- **New Operating System**

  Terry-Rockie risks here???

- **Layered Products**

  If the current list of FRS products were not available for MICA FRS, then PRISM systems would not be able to achieve their market or forecast goals.

- **Software Services Support**

  Software Services must be able to support customers at MICA FRS.

- **Prototypes**

  In order to do finish, debug and test the operating system Engineering group must have access to prototypes nine months (or 12 ???) before FRS (BEFORE CUSTOMER FT???)

8.3 CONTINGENCIES

ROCKIE AND TERRY WE GOT ANY OF THESE???

9 PRISM SYSTEM SOFTWARE

9.1 MICA OVERVIEW

MICA is a contemporary high-performance proprietary operating system for present and future (32- and 64-bit) PRISM systems. MICA combines many features required by todays high-performance systems such as instruction and program parallelism, vector operations, and Symmetrical Multiprocessing (SMP). In addition, MICA supports the concurrent execution of two (2) user interfaces; MICA/VMS and MICA/ULTRIX.

A new language, PILLAR (PRISM Implementation Language), has been developed at DECwest as part of the PRISM project. It is a high-level language specifically designed for systems programming on 32- and 64-bit DIGITAL systems. The PRISM operating system, MICA, and most of the associated software components will be written using PILLAR. In addition, PILLAR, not MACRO, will be distributed with the PRISM operating system as the language of choice for systems programming.

MICA is composed of four distinct layers of functionality:

- **Kernel**

  The kernel is the software layer positioned nearest to the actual hardware. The kernel implements all thread dispatching, exception handling, fork processing, and interprocessor synchronization to make Symmetrical Multiprocessing transparent to "higher" level layers of software.

- **Executive**

  The executive, the next layer of MICA, provides support for: process-, memory- and signal-management; system services; networking, clusters and domain groups; I/O devices; the file system; and system security. The executive supports the following critical ULTRIX features: symbolic links, access and modification date, sparse files, special files, EXEC and FORK system services, and Byte Stream I/O. The executive code of the system is responsible for most of the *"policy"* implemented by MICA.

- **Programming Interface**

  The third layer is the programming interface which is comprised of the native mode programming interface, User File Services (UFS), the ULTRIX-32 compatibility programming interface, and the VAX/VMS compatibility programming interface.

The native mode programming interface includes support for PILLAR, native system services and library routines. The ULTRIX-32 compatibility programming interface is unencumbered and includes support for "C" (the MICA/ULTRIX implementation language), ULTRIX-32 system services and library routines, and POSIX (IEEE 1003.1 only) compatibility. The VAX/VMS compatibility programming interface provides support for VAX/VMS system services, library routines, and Record Management Services (RMS).

- **User Interface**

  The fourth layer, or user interface layer, includes the system management interface, an encumbered (AT&T licensed required and included) ULTRIX-32 user interface consisting of a shell and associated utilities and commands, *and/or* a VAX/VMS user interface which includes a command line interpreter (DCL), associated utilities and commands.

  NOTE: Both user interfaces (MICA/VMS and MICA/ULTRIX) are optional, however, one or the other is required by the operating system.

MICA supports many new or improved contemporary operating system features such as:

- **Symmetrical Multiprocessing (SMP)**

  MICA will support true symmetric multiprocessing for up to 32 processors. Each processor is be able to execute system and/or user code as well as perform independent I/O functions. There will be no master-slave relationships between processors, the *kernel* layer of the operating system will be responsible for scheduling of all processors.

- **Two User/Programming Interfaces**

  MICA supports concurrent execution of a VAX/VMS interface (*MICA/VMS*) and an ULTRIX-32 interface (*MICA/ULTRIX*). The respective combined user and programming interfaces together create a VAX/VMS- or ULTRIX-like environment for users and application programs. This provides an application "environment" for thousands of applications written for either VAX/VMS or ULTRIX-32. These applications may then coexist concurrently on the same PRISM system. *NOTE: Application programs must be compiled and linked on a PRISM system to execute on a PRISM system.*

- **Hardware Vector Processing**

  MICA includes support for hardware vector options which provide increased performance for floating point operations. In addition, MICA supports an uneven mix of vector and scalar processing units.

- **Multi-threading**

  Multi-threading is the ability to schedule multiple executable entities, called threads, across multiple processors. The threads execute independently of one another, in parallel on separate processors; hence the term multithread. Programs that take advantage of multithreading may benefit with higher performance and lower overall execution time.

- **Extendibility and Maintainability**

  MICA is written in a high language specifically designed for systems implementation, PILLAR. Implementation of the operating system in a language such as PILLAR provides advantages over assembly and macro languages. These benefits include easier incorporation of new features and correction of deficiencies.

- **Portability**

Systems programming must avoid dependence on the specifics of the hardware architecture. MICA will be written in PILLAR specifically to avoid such architectural dependencies. Most application programs should be completely portable between VAX, PRISM and future architectures. In fact, the operating system itself, MICA, will be portable to future hardware architectures.

- **Disk Striping**

  MICA supports disk striping for increased I/O performance. Disk striping spreads one I/O request across multiple disk drives. A disk request is broken into multiple reads or writes to a *disk stripe set* in which portions of a file are read or written in parallel to the individual disks. Thus large disk requests are spread over multiple spindles in parallel rather than serially. Disk striping helps to increase the amount of data that may be moved between disk and memory and therefore increasing overall performance of the system.

- **Automatic Code Sharing**

  All images are automatically and transparently shared among all users which reduces memory requirements and the time it takes the processor to "activate" or start up a program image.

- **ODS2 File Structure**

  MICA features a new ODS2 file structure that incorporates features from the ULTRIX-32 file system (Berkeley File System), the VAX/VMS file system (ODS2 version 1), as well as features required for internationalization. This new file structure supports ULTRIX-32 features such as file reference counts and mount points, eight-bit character set file names required for internationalization, and sparse files.

- **Class Scheduling**

  MICA provides the ability to apportion available processor time between *classes* of users in a prescribed manner. The system manager, through the system management interface, defines the "scheduling classes" and sets the target processor time utilization for each of the defined classes. Each user of the system is assigned to one of these "scheduling classes". The system attempts to give each "class" its allotted time. Target utilization is expressed as a fraction of available processor time.

- **Native System Management Interface**

  There is one system management interface for MICA. This interface is used by MICA/VMS and MICA/ULTRIX as its system management interface. System management tasks are broken into groups of related functions. The system management interface supports a menu-driven front-end for video display terminals and command-line support for hardcopy terminals.

- Parallelism

  The system supports the following types of parallelism: I/O (disk striping), instruction (multiple instruction functional units), and thread execution (execution of multiple threads from the same address space).

Following is a representation of the MICA operating system:

9.2 MICA KERNEL

The kernel implements all thread dispatching, exception handling, fork processing, and interprocessor synchronization that make symmetrical multiprocessing (SMP) transparent to "higher" level layers of software.

With few exceptions, the kernel is not responsible for the implementation of operating system "policy". This is left to the higher levels of software in the system. For those cases where it is essential for policy to be located in the kernel, external controls are provided so that higher levels of the operating system software can influence, if not directly control, the actions of the kernel.

There is a formal interface between the executive software and the kernel and a well defined set of rules must be obeyed. The kernel software is a mixture of PILLAR and assembly language.

9.3 MICA EXECUTIVE

The MICA executive resides between the kernel and the user of the system. The executive contains the software for supplying services users need to perform functions that require action by the system. For example, an I/O request issued by the user or user program is actually carried out by executive level code. The executive is a collection of these functions as well as some administrative functions such as the collection of job and process accounting data.

The executive has several main functions to perform. These functions include memory management; process management; system service support; network, cluster, and domain group support; I/O and file system support; error logging; security; and account and performance statistic support.

9.3.1 Memory Management

The memory management subsystem is responsible for the migration of portions of memory, called pages, between disk and physical memory, the management of process working sets, and the mapping of process virtual addresses to physical addresses. Due in part to the page size on MICA (8K pages versus 512 byte pages for VAX/VMS) process swapping is not implemented. To avoid swapping, processes are trimmed to a minimum size but stay memory resident.

9.3.2 Process Management

Process management involves the creation, deletion, and control of jobs, processes and threads in the MICA executive. A job is a single logged in session containing at least one process. A process is an address space and at least one thread of execution. A thread is the entity that is scheduled for execution on a processor.

A job has an associated Job Information Block (JIB) which contains a collection of quotas, restrictions, and authorization information for a user of the system. All processes in the system must be associated with a job. The JIB maintains a running count of processes in the job structure and is deleted when the last process in the job exits.

When a process is created, its creator may specify a parent process or that it should have no parent. As processes are created and assigned parent processes, a process tree is formed. MICA/VMS and MICA/ULTRIX user interfaces both use process trees, with the semantics of these trees being different. In MICA/VMS, when a process exits, all of its descendant processes are forced to exit; while in MICA/ULTRIX, when a parent process exits, no direct action is taken on its children, the processes continue execution without a parent process.

A process will contain one or more threads of execution. Each thread is scheduled independently of one another and, therefore, multiple threads from the same process may be scheduled to run in parallel on separate processors. Additionally, compiler manipulation may request explicitly that

threads be executed in parallel as a *thread set*. No guarantee is made by the system to execute all threads of a thread set simultaneously on separate processors.

9.3.3 System Services

MICA contains native system services for manipulation of the various subsystems of the executive. System services are routines and, therefore, conform to the PRISM calling standard. Any system service may be called from any PRISM language, hence, from either user interface (MICA/VMS or MICA/ULTRIX). Examples of services include: object services, logical name services, timer services, and process control services.

9.3.4 Networking, Clusters and Domain groups

MICA will implement DECnet Network Architecture (DNA) Phase IV with end node routing at FRS. MICA's DECnet will be completely compatible with other DIGITAL DECnet implementations and will fully participate as an end node in Local (LAN) and Wide Area Networks (WAN) at FRS. MICA will use the 802.3 Ethernet protocol for both DECnet and TCP/IP. MICA will supply Berkeley BSD 4.2/4.3 sockets for TCP/IP. DNA Phase V is currently being evaluated at this time by the DECwest engineering staff for inclusion in future releases of the operating system.

Domain groups will provide the mechanism for PRISM systems to connect to VAXclusters. Domain groups are defined as groups of logically connected individual systems or domains. A domain may be thought of as a single system, which may be a single or multiple CPU system (e.g. a PRISM system), or a distributed system of particular internal architecture (e.g. a VAXcluster). Examples of domains include: a VAXcluster, a single PRISM system, a LAT terminal server, a print server, an ULTRIX-32 system, or a PC.

The domain group structure will allow the building of hybrid computing environments into a well integrated extended VAX/VMS computing environment. This extended computing environment may be composed of VAXclusters and PRISM systems, as well as other heterogeneous systems and/or domains. Domain groups are considered to be the next generation of "clusters" and will allow VAX/VMS and PRISM systems to efficiently and transparently share data files as well as services such as batch and print queues.

Supported domain group interconnects will be the NI and CI. This means a PRISM system will have the ability to attach to the Ethernet or connect directly into the Star Coupler as a member of a VAX/VMS domain group.

9.3.5 I/O Architecture and File System Support

The I/O architecture of MICA is implemented through the use of "function processors". These function processors are executive level code that replace VAX/VMS device drivers, pseudo drivers, ancillary control processors (ACPs), and extended QIO processors (XQPs). MICA has additional function processors to implement such features as disk striping and software volume shadowing. Thus, function processors are the executors of I/O requests. The I/O architecture, through the use of function processors, is designed to provide a single efficient I/O interface to the system.

File system services, or UFS (User File Services) on MICA, is the facility which implements transparent file and device access. Unlike VAX/VMS, all file access is done from user mode. UFS implements the features of both VAX/VMS-RMS and ULTRIX-32 as well as access and record semantics. UFS is completely compatible with VAX/VMS-RMS and ULTRIX-32 file system from high level languages such as FORTRAN, C, PASCAL.

For every VAX/VMS-RMS system service there exists an identical MICA UFS system procedure which completely duplicates the functionality of VAX/VMS-RMS. An additional set of MICA UFS procedures are provided to allow for the management of the ULTRIX-32 file descriptors and provide the ULTRIX-32 file system interface. Thus, UFS extends VAX/VMS-RMS functionality to include ULTRIX-32 byte stream access, file and byte range locking, file information, and file sharing. MICA UFS implements the use of UNIX *named pipes*, symbolic links, device files and mount points, and VAX/VMS *mailboxes*.

UFS implements all of the VAX/VMS-RMS records structures (sequential, relative and multikey index sequential) with the limitation that only PROLOGUE 3 file types will be supported at FRS. The MICA File Description Language (FDL) will be identical to VAX/VMS.

UFS supports shared and exclusive access for all record organizations (sequential, relative, ISAM and byte stream). Except for byte steam files, shared access is implemented transparently for all record organizations. Access to UFS is via the Remote Procedure Call (RPC) mechanism.

UFS supports the ULTRIX-32 system service *fork()*. *Fork()* is the UNIX service for new process creation. The new process created is as an identical copy of the original with the same resources (open files, etc.), rights, and privileges. Only files which are opened for shared access remain open in the child process after the fork(). All other files remain open in the parent but not in the child.

MICA utilizes a new version of the ODS2 file system. This new file system combines features of both VAX/VMS ODS2-1 (On Disk Structure 2, version 1), the Berkeley file system (ULTRIX-32), and extensions for internationalization. Each interface, MICA/VMS or MICA/ULTRIX, utilizes its own native file system transparently from and in concert with the other interface.

9.3.6 VAX Data Type Support

VAX data types, with the exception of D_, and H_floating, are supported by the PRISM hardware. These include: byte (8-bits), word (16-bits), longword (32-bits), quadword (64-bits), F_floating (32-bits), and G_floating (64-bits). F_floating point values range from 0.29*10**-38 to 1.7*10**38 with precision approximately 7 decimal digits. G_floating point values range from 0.56*10**-308 to 0.9*10**308 with precision approximately 15 decimal digits.

Explicit software conversion routines for D_floating data will be supplied with MICA. H_floating data will be emulated in software. FORTRAN is adding a new keyword to the OPEN statement that specifies the external representation in the file for any double precision values (DOUBLE PRECISION, REAL*8, COMPLEX*16). This allows double-precision data in the file to be automatically converted converted "on-the-fly" as it is read into or written from the FORTRAN program. This method provides the most direct mechanism for conversion of customer data from D_ to G_ floating point values.

9.3.7 Error Logging

MICA logs hardware and software errors as well as system events. Errors and events are logged to a system file the same as in VAX/VMS. Error logging is implemented with a "function processor" which supports reads and writes to the error file. The error logger function processor is similar in nature to the I/O subsystem function processor (i.e., the executor of error logging requests). A filtering capability is provided so that a reader (devices, user processes, system routines, etc.) can request to receive only certain kinds of errors.

The error logger is responsible for allocation of buffers for callers who write error log entries, writing a sequence number into the buffer, checking to see who is queued to receive the error, filtered on type, and recording the error in the error log file.

Processes are allowed to read error log entries selectively as they occur. Selection of specific errors occurs via "filtering" of corresponding fields in the error log buffer. The information that may be used as filtering criteria are:

- Error entry type

- Severity code
- Device class
- Device type
- Full device name
- CPU identification
- IOP identification

MICA records, as a minimum, the same information on errors and events as is currently done on VAX/VMS. In some cases, because of hardware differences (e.g., I/O processors), it logs additional information on the same error types.

9.3.8 Security

The initial MICA release is intended to provide the means of achieving a C2 class rating as defined in *"DOD Trusted Computer System Evaluation Criteria, DOD 5200.28-STD"*.

Security in MICA includes the prevention of:

- Unauthorized access to objects within the system
- Unauthorized use of privileged system functionality
- Unauthorized access to the system

MICA also includes some capabilities for notification of attempted accesses (successful or unsuccessful) to the system or objects/functions within the system.

The security scheme includes:

- Authenticating users before access to the system is granted
- The use of effective user and effective group values to establish an authenticated user as an individual and as a member of a primary group
- The use of mode and Access Control Lists (ACLs) to selectively allow access to objects within the system
- The use of privileges to override ACL object protection
- The use of privileges to perform functions within the system whose use needs to be controlled
- The ability to monitor access to the system and objects via security audit logging and alarms
- Client access rights available to protected subsystem servers

The security design of MICA provides for compatibility with both POSIX and VAX/VMS while the complete security mechanism is a superset of both POSIX and VAX/VMS. The features included under system access security are:

- System level password
- Primary and secondary passwords
- Password expiration
- Minimum password length
- Password generation
- Intrusion detection and prevention

- Automatic login
- Captive accounts
- Command Line Interpreter (CLI) restrictions
- Secure server (terminal logins)
- Access time restrictions
- Account duration
- Last login messages
- DECnet object accounts
- Interactive login limits

NOTE: DECwest is not developing a PRISM Security Kernel. If this effort is undertaken it will be developed by another engineering group within DIGITAL.

9.4 MICA PROGRAMMING INTERFACE

MICAs programming interface consists of the MICA native programming libraries, MICA/ULTRIX compatibility programming interface, and the MICA/VMS compatibility programming interface. The native programming libraries are collections of routines that support PRISM and MICA specific features. Some examples of these features and associated routines include: object service routines, process control services, and memory management services.

The MICA/ULTRIX and MICA/VMS compatibility programming interfaces provide compatibility and emulation support for the various VAX/VMS and ULTRIX-32 routines and system services required to run VAX/VMS and/or ULTRIX-32 programs and applications. In most cases programs will port directly from VAX/VMS and/or ULTRIX-32 with only recompilation and relinking required.

For exceptions, see Section 9.4.2, MICA/ULTRIX Compatibility Programming Interface; and Section 9.4.3, MICA/VMS Compatibility Programming Interface.

9.4.1 Native Programming Libraries

MICA native programming libraries provide support for MICA specific features. MICA is a new operating system and as such is *internally* much different than VAX/VMS or ULTRIX-32. Many contemporary features have been built into the system that are foreign to other DIGITAL operating systems. New service routines are required to support the operating system, the implementation language (PILLAR), and utilities that make up the system. The native programming libraries provide this support.

9.4.2 MICA/ULTRIX Compatibility Programming Interface

MICA provides an MICA/ULTRIX compatibility programming interface for compatibility with ULTRIX-32, DIGITAL's implementation of AT&T's UNIX operating system. MICA/ULTRIX will permit customers to run existing ULTRIX-32 compatible applications and develop new ULTRIX applications on PRISM systems.

The MICA/ULTRIX Compatibility Programming Interface provides system services and library routines that are compatible with ULTRIX-32 and IEEE P1003.*x* (POSIX) interface definitions. Incompatibilities between ULTRIX-32 and POSIX will be resolved in favor of POSIX. Berkeley 4.3 and AT&T System V compatibility will be provided through alternate run-time libraries.

The MICA/ULTRIX Compatibility Programming Interface will be written in PILLAR. All MICA systems will ship with the MICA/ULTRIX Compatibility Programming Interface enabling MICA users to run MICA/ULTRIX applications from either MICA/VMS or MICA/ULTRIX User Interfaces. The MICA/ULTRIX Programming Interface is unencumbered by AT&T copyrights or licenses.

MICA/ULTRIX goals provide for an unencumbered programming interface which provides functionality compatible with:

- UNIX Programmer's Manual sections 2 and 3
- ULTRIX-32 Compatibility
- VAX-C RTL functionality

The MICA executive provides support for MICA/ULTRIX processes, Files-11, User File System, terminal driver, signals, inter-process communications, networking, security and password/group databases. Each of these topics are discussed in the following sections in greater detail.

9.4.2.1 Processes

The MICA/ULTRIX programming interface provides the facilities to execute and to control the execution of MICA/ULTRIX programs. Under MICA/ULTRIX, a user executes a program in an environment called a process. A process is created by the MICA/ULTRIX system call *fork()*. The newly created process is a copy of the creating process. A program is run in a process context using the *execve()* system call. The ULTRIX-32 definition of a process and the mechanics of running a program within a process are significantly different from VAX/VMS. MICA will therefore provide the facilities to implement both ULTRIX-32 and VAX/VMS process environments.

The MICA process identification is the process object identification, a 64-bit quantity. MICA will translate this ID into a 32-bit quantity suitable for ULTRIX-32 compatibility. MICA will provide support for ULTRIX compatible process groups.

The MICA/ULTRIX *system()* call invokes a utility or program using the AT&T Bourn shell (*sh*) and may not work if the customer does not have the MICA/ULTRIX User Interface.

9.4.2.2 Files-11

The MICA/ULTRIX Programming Interface provides a UNIX-like view of the new ODS2 file system. MICA/ULTRIX software will see the same hierarchy of directories and sub-directories as ULTRIX-32. The MICA/ULTRIX Programming Interface provides standard ULTRIX-32 system calls and library routines for creation, modification, deletion and other access of files.

9.4.2.3 User File Services

ODS2 will be changed to incorporate file reference counts; eight-bit character set file names; file read, write and modify dates; and provide support for MICA/ULTRIX special files, named pipes and symbolic links, and sparse files.

9.4.2.4 Terminal Driver Support

The MICA terminal driver provides the functionality required to support both 4.3 BSD *tty* and System V *termio* functionality. The MICA/ULTRIX Programming Interface *ioctl()* provides these interfaces.

The MICA terminal driver provides support for process groups and C shell job control.

9.4.2.5 Signal Support Support

MICA provides procedure based condition handling. A MICA/ULTRIX condition handler is provided as a last chance condition handler which catches conditions and converts them into an ULTRIX-32 compatible signal.

9.4.2.6 Inter-Process Communication

MICA provides support for inter-process signaling (or raising a condition) in another process. This support provides the basis for POSIX compatible *kill()* and ULTRIX-32 compatible *killpg()* functions.

Socket support will be provided for Berkeley 4.3 BSD compatibility. Message queues, semaphores and shared memory will be provided for System V compatibility.

9.4.2.7 Networking

The MICA executive will provide support for DECnet, TCP/IP and other future (OSI) networking protocols. Network services access will be provided to MICA/ULTRIX applications through the 4.3 BSD sockets compatibility support. It is a goal to provide the Sun Microsystems Network File System (NFS) at FRS.

9.4.2.8 Security

MICA provides support for the POSIX concepts of user and group IDs, superuser, the POSIX semantics for granting file access.

9.4.3 MICA/VMS Compatibility Programming Interface

MICA provides the functionality of VAX/VMS within the limits of the PRISM architecture. The VAX/VMS compatibility programming interface maps the system services of VAX/VMS into the functionality provided by MICA. The purpose of the compatibility programming interface is to allow easy porting of VAX/VMS programs to PRISM and to allow easy maintenance of programs that run under both the VAX and PRISM architectures. The MICA/VMS Compatibility Programming Interface libraries provide support for VAX/VMS program and application compatibility.

There are however, a few hardware and software constraints imposed which limit compatibility to less than 100%. Hardware constraints are no supervisor or executive mode and memory protection granularity larger than 512 bytes (8K bytes on PRISM). Software constraints are different condition frames and descriptor formats. Programs or applications that utilize these VAX or VAX/VMS specific features will need to be modified before they will execute on MICA/VMS.

The following are goals of the VAX/VMS compatibility programming interface:

- Provide efficient mapping of VAX/VMS system services to MICA services
- Require no source code changes for most ported applications
- Require minor source changes for most of the remaining ported applications
- Require major source changes for only a very small number of ported applications, if any
- Allow mixing of VAX/VMS services with unrelated MICA services
- Be compatible with VAX/VMS release V 5.0
- Be updated to track VAX/VMS system service additions
- <emphasis(VMS System Service Emulation\bold)

The VAX/VMS compatibility library (VCL) contains routines for the following VAX/VMS facilities:

- LIB$, STR$, OTS$, CRF$ general purpose routines
- SMG$ (Screen Management routines)
- MTH$ and OTS$ math routines
- OTS$ language support routines
- Support routines for the VAX FORTRAN and PASCAL Languages

All routines from these are facilities provided *except*:

- Routines that give the user access to VMS specific features that cannot be reasonably simulated on MICA
- Support for obsolete capabilities such as the SCR$ routines and obsolete system services

- **Math Library**

  MICA provides a VAX/VMS compatible math library for support of existing programs and applications. In addition, the math library includes the following enhancements to the VAX/VMS RTL:

  - Vector support for math routines
  - Support for engineering and scientific subroutine libraries
  - Complex instruction sequences to support compiler-generated code

NOTE: Since the VAX/VMS compatibility programming libraries are part of the MICA Programming Interface, these libraries will be available for use by MICA/ULTRIX users and programs as well as MICA/VMS users and programs.)

9.4.3.1 Jobs and Processes

The MICA/VMS process structure is very similar to the process structure of VAX/VMS. Both VAX/VMS and MICA/VMS have "jobs", "processes", and "subprocesses". Jobs are classified as interactive, detached, network, or batch, just as in VAX/VMS. The process structure provides all the facilities needed to execute programs. Processes on MICA/VMS differ in that MICA/VMSs processes have one or more threads of execution. These differences will not be "seen" by normal system users and programs. Programs ported from a VAX/VMS system will behave no differently on MICA/VMS.

9.4.3.2 RMS

RMS functionality is implemented in MICA/VMS via VAX/VMS-RMS compatibility calls to the MICA User File Services (UFS). For every VAX/VMS-RMS system service there exists an identical MICA UFS system procedure which duplicates the functionality of VAX/VMS-RMS completely. Incompatibilities exist only where VAX/VMS programs that are being ported contain explicit references to VAX/VMS-RMS data structures. Programs that contain these incompatibilities will need to be modified. However, these incompatibilities are completely invisible from high level languages such as FORTRAN, PASCAL, C, and so on.

All of the VAX/VMS-RMS records structures (sequential, relative and multikey index sequential) are supported. PROLOGUE 1 and 2 file types are older VAX/VMS-RMS file types that are obsolete and therefore will not be supported. Only PROLOGUE 3 file types will be supported by MICA/VMS RMS. In addition, MICA/VMS RMS functionality has been extended to include ULTRIX-32 style byte stream access, file and byte range locking, and eight-bit international character set file names.

UFS supports shared and exclusive access for all record organizations, sequential, relative, ISAM and byte stream. For all record organizations, except byte steam files, shared access is transparently implemented.

9.4.3.3 Terminal Driver Support

Since driver support is really a function of the MICA executive, the terminal interface capabilities are a superset of the capabilities provided by the VAX/VMS and ULTRIX-32 terminal drivers. Capabilities are broken into three (3) categories: common (i.e. common to both ULTRIX-32 and VAX/VMS), ULTRIX-32 specific, and VAX/VMS specific. Common capabilities are: control character processing, type-ahead buffering, modem control and monitoring, and output formatting. VAX/VMS specific capabilities are: line editing, input character validation, disconnectable terminals, and associated mailboxes. Features formerly available only to ULTRIX-32 users are now also available to MICA/VMS users. These features include changeable special keys and support for process groups (used to control access to terminals).

Support for terminals which do not support the DEC Multinational Character Set (MCS) is also provided. This includes support for character sets such as National Replacement Character (NRC) and for "foreign" character sets.

9.4.3.4 Networking

Networking is a MICA executive function and therefore all networking features such as DECnet, domain groups, and TCP/IP that are available to MICA are also available to each of the user and programming interfaces.

Support for applications that are ported to MICA/VMS from VAX/VMS are through calls to the compatibility library routines. Support for developing applications that use the network will be through the native programming interface.

9.4.3.5 Security

The security scheme of MICA/VMS will provide a high degree of compatibility with VAX/VMS. Although MICA, and therefore MICA/VMS, will not use the VAX/VMS SOGW (System, Owner, Group, and World) format, SOGW format from DCL commands are converted into ACL (Access Control List) format when assigned internally. In addition, SOGW information stored in ACL format is recognized and converted to SOGW format for display purposes when compatibility with VAX/VMS commands must be maintained.

Access Control Lists (ACLs) provide the most flexible means of system protection. ACLs in MICA/VMS contain an ordered set of Access Control Entries (ACEs). These ACEs may be used to grant or deny access to a service or to generate a security audit/alarm. Access is granted or denied via identifiers. Identifiers are issued as a system management task and are unique to that system user.

9.5 MICA USER INTERFACES

MICA provides three (3) distinct user interfaces; a Native user interface which is used solely for system management of MICA; MICA/ULTRIX, an encumbered (AT&T Licensed) shell and set of utilities that provides an ULTRIX-like interface; and MICA/VMS, which uses DCL VAX/VMS-like utilities and commands to provide a VAX/VMS-like interface.

9.5.1 Native System Management Interface

There is one system management interface for MICA no matter which (or both) user interface is chosen (MICA/VMS or MICA/ULTRIX). System management tasks will be broken into groups of related functions. The system management functions may be dependent on whether the "system" is executing MICA/VMS only, MICA/ULTRIX only, or both interfaces concurrently. The Native System Management Interface will support a menu-driven front-end for video display terminals and command-line support for hardcopy terminals. *Please note that SMI (System Management Interface) is only accessible through the MICA/ULTRIX or MICA/VMS user interfaces, not as a standalone interface.*

The Native System Management Interface is broken into the following groups:

- **System Group**

  The System Group contains those functions that pertain to the management of resources on the system as a whole. It is subdivided into system configuration, functions that are used to "tailor" a system to a site's specific needs; system operations, functions that are required to maintain a running system on a day to day basis; accounting/auditing, functions that are used to monitor system resource usage; and software management, functions used to add and update software (system and layered product software) on the system.

- **User Group**

  The User Group contains functions that pertain to an individual user's access and rights to resources on the system. These functions allow the system manager access to all relevant user parameters such as identifiers, passwords, class assignments, adding/deleting users, maintaining currently authorized users, and proxy access assignments.

- **Device/Media Group**

  The Device/Media Group contains functions needed to manage and monitor devices on the system and the media the devices use. Some of these functions include media initialization; device allocation, mounting, and dismounting; error reporting; diagnostics; crash analysis; backup; and disk usage reporting.

- **Network Maintenance Group**

  The Network Maintenance Group contains functions related to running and maintaining the network. These functions include create and maintain the network; restrict user access via the network; network diagnostics; network startup and shutdown; network error reporting; and proxy access.

9.5.2 MICA/ULTRIX User Interface

The MICA/ULTRIX User Interface is derived from AT&T's UNIX utilities and shells and will include an AT&T license. The MICA/ULTRIX User Interface is implemented in coordination with the ULTRIX-32 Engineering Group (UEG).

9.5.2.1 MICA/ULTRIX Shells

The MICA/ULTRIX User Interface provides both standard ULTRIX shells (the C shell and the Bourne shell) to serve as the command language interface into MICA.

9.5.2.2 Ported Tools and Utilities

See Appendix C for a list of ported utilities and commands.

9.5.3 MICA/VMS User Interface

MICA/VMS's user interface provides the commands and utilities available to the user. MICA/VMS features a VAX/VMS compatible DCL. Command files from VAX/VMS systems, with few exceptions, will run without modification. Utilities available to the user are, with few exceptions, exactly the same as VAX/VMS. In addition to DCL, MICA/VMS features VAX/VMS compatible languages (FORTRAN, PASCAL, BLISS and C), file system (RMS), library routines, system services, and utilities.

9.5.3.1 MICA/VMS Command Interface

Command Languages. DIGITAL Command Language (DCL) is implemented on MICA as part of the MICA/VMS user interface. The DCL user interface looks, feels, and acts the same as the VAX/VMS version of DCL. It has the same features such as internal, external and foreign commands, flow control (i.e., if-then-else), and lexical functions. However, some of the lexical functions that allow retrieval of VAX/VMS-specific information are either not implemented or not completely compatible, as they have no equivalent function in the PRISM architecture.

9.5.3.2 MICA/VMS Utilities and Commands

See Appendix D for a list of MICA/VMS utilities and commands.

9.5.3.3 Programming Tools

MICA/VMS features many of the same programming tools that are available on VAX/VMS such as TPU (Text Processing Utility); LSE (Language Sensitive Editor); Debug; and PCA (Performance and Coverage Analyzer). These tools have at least the same functionality as those available on VAX/VMS. In the case of LSE and debug, functionality is added to support PILLAR.

9.5.3.4 System Management

Management of the MICA/VMS User Interface will be administered through the Native System Management Interface. Some functions may be similar in nature to VAX/VMS but only by pure accident!!!

9.6 SYSTEM IMPLEMENTATION LANGUAGES

9.6.1 PILLAR

PILLAR is the PRISM System Implementation Language. It has been developed expressly for this project for implementation of the PRISM operation system (MICA), its user and programming interfaces (MICA/VMS and MICA/ULTRIX) and their associated utilities.

PILLAR is a high-level systems programming language for use on 32- and 64- bit DIGITAL systems. PILLAR has been developed as a fundamental part of the PRISM project to implement the PRISM executive and PRISM software components that operate above the executive (compilers, the linker, run-time libraries, and such).

Although PILLAR is a high-level language, it has many features to support systems programming, such as typecasting and built-in functions that generate specific PRISM instructions. However, PILLAR emphasizes the general features of high-level programming: modules, data type declarations, structured programming, and the use of procedures.

When compared with low-level systems programming languages, PILLAR has the following advantages:

- Software is more portable because hardware dependencies are isolated in declarations and small routines. Accidental hardware dependencies are avoided.

- Code is easier to read and maintain.
- Program development is faster because the compiler detects more errors, and the language provides explicit help in difficult areas, such as exception handling.
- PILLAR yields the fastest object code for PRISM's architecture and compilers.

The PRISM architecture was designed to be an optimal target architecture for compilers; it is not aimed at machine-level programming. PILLAR has been designed to match both the properties of PRISM and the capabilities of today's advanced compiler technology. Thus, PILLAR programmers will be able to work well above machine level and still fully exploit PRISM's performance advantages.

9.6.2 'C'

'C' is an extended implementation of the C programming language originally developed at Bell Laboratories. 'C' is the implementation language of the MICA/ULTRIX User Interface and is developed for the PRISM project based on the VAX C language front-end and uses the PILLAR code generator ('back-end'). As a PRISM language, 'C' is integrated into the <common> Programming Interfaces. All <common>, MICA/VMS and MICA/ULTRIX system services are thus available to programs written in 'C'. 'C' programs can invoke, as functions, modules written in other PRISM languages.

'C' has been designed to match both the properties of PRISM and the capabilities of today's advanced compiler technology and will fully exploit PRISM's performance advantages.

'C' is designed to be portable between VAX, PRISM, and future 64-bit architectures. 'C' adheres to the proposed ANSI X3J11 standards and is backward compatible with the ULTRIX-32 Portable 'C' Compiler ('pcc'). 'C' represents a more current definition and implementation of the language than is described in the initial guiding document for C, The C Programming Language by Kernighan and Richie. Some incompatibilities among implementations do, however, exist. In general, many programs written in C for other compilers can be successfully recompiled under 'C'. This provides a high degree of portability for existing 'C' programs to PRISM and future 64-bit architectures.

9.7 SUPPORTED DEVICES

MICA will support the following devices:

- DEBNT (AIE, NI, LAT, TK70) *This boards name may change but operating system support for a board that supports the TK50/70 is required for Emerald*
- KDB50 (RA disks, MSCP)
- DMB32 (terminals, printers)
- HSB50 (RA disks, TA tapes)
- BCA (BI to CI adapter for cluster support)

9.8 DIAGNOSTICS

MICA incorporates a strategy for providing a common interface to error logging analysis, on-line diagnostics, and standalone diagnostics. Common menu-driven user interfaces and error-report formats are used. A minimal version of MICA is provided as the basis for off-line diagnostics. In general, the tools are designed to do as much on-line error detection and isolation as possible.

The goal for MICA installation and system management is that it will be simpler and easier than VAX/VMS or ULTRIX-32.

No provision is made for compatibility with existing ULTRIX system installation or system management procedures.

9.9 SOFTWARE PACKAGING

9.10 CAVEATS

One of major reasons for DIGITAL to implement the PRISM architecture is to gain software functionality. In order to gain that functionality 100% compatibility with VAX/VMS and ULTRIX-32 cannot be possible. However, it is a goal to maintain compatibility whenever possible.

Following are some of caveats regarding VAX/VMS and ULTRIX-32 compatibility.

- Command procedures that are dependent on VAX/VMS data structures may need to be edited in order to run on MICA/VMS user interface. As an example, lexical functions in a command procedure that return VAX system specific job information may need to be changed.
- Not all VAX/VMS System Services can be emulated perfectly due to differences in the two architectures.
- MICA device drivers are not compatible with VAX/VMS or ULTRIX-32 device drivers.
- MICA/ULTRIX adheres to the IEEE P1003.1 (POSIX) system interface standards and may conform to this standard before ULTRIX-32. This may cause 'two ULTRIX' confusion for PRISM customers.
- User application and systems software that presumes knowledge of the system kernel, system management functions and files (such as */etc/passwd* and that attempts to circumvent the spirit of UNIX portability by using such knowledge will not run on MICA/ULTRIX without significant code rewrites.

VAX/VMS and ULTRIX-32 application and programming compatibility are major goals of the MICA systems. All tradeoffs which will hinder compatibility will be carefully weighed before decisions are made. The migration of applications and programs from VAX/VMS and ULTRIX-32 to MICA systems will be similar to to initial migration of RSX-11M to VAX/VMS.

10 DOCUMENTATION

Because of the compatibility that will exist between MICA/VMS and VAX/VMS and because the current ULTRIX-32 or other UNIX documentation sets do not provide documentation which is as comprehensive as the VAX/VMS set, the MICA documentation set is largely based on the Version 5.0 VAX/VMS documentation set. The same documentation set structure and functional organization of information by subkit and volume is also used in order to maintain a high degree of familiarity for the user. An additional subkit is added to the basic documentation set structure to document the MICA/ULTRIX user interface. This subkit is structured similar to the MICA/VMS subkit.

The MICA documentation set includes the following subkits: MICA/VMS User Interface, MICA/ULTRIX User Interface, MICA System Management Interface, and MICA Programming Interface. The MICA documentation set does not include a condensed documentation set, which would be similar to the Version 5 of VAX/VMS set. However, a condensed set of documentation for MICA will be evaluated for future releases.

The definition strategy for the MICA/VMS and MICA/ULTRIX subkits includes reviewing the VAX/VMS Version 5.0 and ULTRIX-32 documentation sets to determine which manuals will be revised to produce the corresponding PRISM operating system manuals, and which new manuals will be written from scratch.

The MICA manuals are written using the VAX DOCUMENT production system. DOCUMENT was selected in order to produce manuals in the same format as the VAX/VMS documentation set and to provide access to the existing VAX/VMS DOCUMENT source files. The strategy is to use the source text of the existing Version 5.0 VAX/VMS documentation set in all cases where the functionality (from the user standpoint) is unchanged between VAX/VMS and MICA.

For the MICA/ULTRIX User Interface documentation and the the ULTRIX-32 Compatibility Programming Interface documentation, the strategy is to convert the source text of the existing ULTRIX-32 documentation to DOCUMENT format. The writers will translate the ULTRIX-32 source files for the reference manual descriptions to DOCUMENT format as they revise them and new manuals will be written in DOCUMENT.

The online documentation strategy for MICA is to track the corporate strategy in this area and to produce a documentation set that could, if product requirements dictate, be successfully accessed and displayed via the proposed VMS-based workstation software reader product (formerly named VIOLA) being developed by the Corporate User Publications Advanced Development Group.

In addition, the DECwest documentation group provides the text for a traditional online HELP library system. This library includes help information on all DCL commands, interactive utilities, MICA Run-Time Library routines, and system services. It also includes information of the differences between MICA/VMS and VAX/VMS.

Facilities for producing printed and online documentation have been a traditional part of UNIX systems, so support for these facilities is integral to many system components. Thus, the source files for the ULTRIX-32 reference descriptions are coded using the *man* macros and the *ditroff* text formatter. The same source files are supplied online and accessed using the *man* command and *nroff* text formatter. Along with a few other sorting and referencing tools, these elements form the standard UNIX system online help facility.

SDT is responsible for documenting all languages, other than PILLAR and C, and for documenting all other SDT layered products. In addtion, SDT will be providing writers for certain manuals in the MICA documentation set, i.e., Sort/Merge, TPU, DSR, Debugger, and the RTL manuals.

In addtion to the hardcopy MICA documentation set, electronic masters are provided to software manufacturing in nonrevisable Interpress or PostScript format to meet the needs of the DIGITAL Demand Printing Program, assuming this program is in place prior to PRISM FRS. A revisible version of electronic masters may also be needed for translation purposes.

11 FIRST REVENUE SHIP (FRS) SOFTWARE PRODUCTS

11.1 DIGITAL SOFTWARE PRODUCTS

Products that have been committed for MICA FRS are:

- FORTRAN
- PASCAL
- C *
- Bliss **
- DECnet *
- PILLAR *
- Debugger
- TPU
- LSE
- PCA
- Sort/Merge

* Done at DECwest

** Bliss will be made available if needed by customers, however, it will not be marketed/sold. It will also be available to DIGITAL Engineering to aid in the porting of products to the PRISM Architecture.

- RTL (Some RTLs will be done by SDT and some will be done by DECwest)
- Math Library

The following products are being examined for MICA FRS:

- GKS
- PHIGS
- AI Products
- Rdb
- Tools Integration
- CMS, MMS, DTM

Ada will not be available until after MICA FRS, however, it is a goal to announce it as part of the PRISM program.

The goal is for all products which are needed to support the PRISM market will be ported/rewritten to the PRISM Architecture within two years after MICA FRS. The long term goal is to have a well rounded product set, comparable to the VAX/VMS product set.

12 SERVICES

The availability of a quality service program at FRS for PRISM systems will be a key factor in ensuring that customers are successful in their transition to this new architecure and to the success of the PRISM program. This section highlights the deliverables and strategies in support of the service goals.

12.1 STANDARD SERVICE OFFERINGS

MICA software service offerings will be the same or similar to the software service offerings available for other DIGITAL software products in the FY89 timeframe. CSSE Product Management, with input and assistance from DECwest CSSE and Engineering, has the responsibility for determining what software service products will be offered, its pricing and marketing strategies, and in insuring that the field is ready to deliver services at FRS.

See the CSSE ??? Document for futher details.

TALK TO BARB-CSC SUPPORT? DESC OF BASIC SERVICE SEEMS INCOMPLETE

Table 15 Standard Software Service Offerings

Service Offering	Features
H-Kit *	Media and documentation, Newsletter/Dispatch document
BASIC Service	Access to DIGITAL Software Information Database (DSIN) Immediate access to DIGITAL service representative User Inquiry on product use Assistance on problem identification and resolution Automatic problem escalation and problem management Document on DIGITAL services delivery
Subscription Service	Newsletter/Dispatch document Updates to media and documentation
OPTIONS:	
Extended Service	Single account management for all service needs Priority response/continuous problem management Individualized service representative Installation/Revision management services Remote performance monitoring
Additional Telephone Support Center Contact	A remote site contact (or greater than 3 contacts at a single site) can be added to the BASIC service
DECstart *	On-site orientation to PRISM system operation
DECstart-Plus *	On-site technical assistance and orientation for the startup a PRISM system

* These offerings will be priced by the Software Services (SWS) ogranization

The availability of the above offerings is dependent on significant changes to the packaging currently being planned by the service organization.

12.2 PRISM UNIQUE SERVICE OFFERINGS

There is a need for some unique service offerings for the PRISM program. These unique services are required because: DIGITAL is introducing a new architecture, the technical price/performance market may require services that are different from the standard DIGITAL offerings, and because some systems will be priced at levels above most other DIGITAL systems ($1M to $3M range). Futher studies of potential PRISM customers will be required in order to define specific service deliverables. However, following is a list of customer needs which are being evaluated for service possibilities:

- **Application support—** The DIGITAL service community is evaluating the possibility of supporting third party applications.
- **Migration/coexist services—** VAX customers may need assistance in migrating from (or coexisting with) VAX systems.
- **Optimal performance/networking configuration consulting—** Customers may need consulting on performance tuning and/or network configuration.
- **Scientific computing services—** Customers may need a service which assists them in optimizing their application code, especially optimizations concerning the directed decomposing FORTRAN compiler.
- **Application run-time services—** These services could provide a cost-effective service for environments predominately using their system as an application engine. BARB-WHAT....?

12.3 WARRANTY SERVICES

MICA will include at least a one year warranty of BASIC service with every processor sold.

BARB, more specific here, consulting??? etc, etc.....????

12.4 FIELD READINESS

CSSE is responsible for insuring that the service community is prepared for the PRISM program. Some areas that CSSE will need to cover in order to prepare the field for the products are: training, FRS support teams, problem escalation management, and capital equipment needs. See the CSSE Business Plan for further details.correct name?????

13 TRAINING

14 SOFTWARE MANUFACTURING

Adherence to the current software introduction processes is being planned. The documentation is delivered to software manufacturing, from the development group, in camera ready form. A DECwest typesetter produces the camera ready copy of the final text for all manuals in the MICA documentation set and its artwork is produced by the DECwest production group illustrators and pasted into the camera ready copy during final production. Binary code will be delivered to software manufacturing on the exact media types and in the precise format that will be distributed to customers. Todays plans call for the distribution media to be TK70, although CDROM is being considered. Both the documentation and binary media will be packaged in accordance with packaging standard 073. The goal is to have this product look very similar to VAX/VMS.

In conjunction with this planning, an examination into innovative manufacturing alternatives is being conducted to determine benefits and risks. Two of these alternatives are:

- The electronic tranfer of documentation and binary code from the development group(s) to the software manufacturing site(s)

- The use of CDROM as a distribution media type for both documentation and binary code

The above alternatives will be addressed fully by Phase 2.

Initial software manufacturing locations will be determined, in part, by:

- The ability of software manufacturing to meet the startup goals of Engineering and other manufacturing groups
- Proximity to the software engineering group(s) developing products for PRISM systems
- Proximity to POM (Point of Manufacture) consolidation sites
- Cost to manufacture the software product(s)

See the Software Manufacturing Business Plan for further details.

APPENDIX A

OVERVIEW OF MICA/VMS SYSTEM COMPONENTS

APPENDIX B

GLOSSARY

APPENDIX D

MICA/VMS UTILITIES AND COMMANDS

Utilities. The following is a list of MICA/VMS utilities and commands:

Command	Description
Accounting	Accounting utility - collects, records and reports accounting data
Analyze/crash	Analyze system crash dumps
Analyze/disk	Analyze disk structure and reports errors and inconsistencies
Analyze/error_log	Reports contents of error log file
Analyze/image	Analyzes contents of an executable image file
Analyze/media	Records and reports location of bad blocks
Analyze/object	Analyzes contents of object files
Analyze/process_dump	Analyzes process dump file
Analyze/rms	Analyzes and inspects contents of RMS file
Analyze/system	Analyzes running system
Autogen	Automated system modification and tuning tool
Authorize	Controls access to system
Backup	File backup and restore utility {on-line,stand-alone}
CDU	Creates, deletes, or changes command tables
Convert	Copies files changing file organization types
Convert/reclaim	Reclaims empty space in indexed files
Copy	File copy utility
Create {file}	Creates sequential disk files
Create/directory	Creates new directories or subdirectories
Create/fdl	Creates RMS file from FDL file specifications
Delete	File deletion utility
Diff	Compares the contents of two files
Directory	Lists files in a directory
Diskquota	Controls disk volume usage
Dismount	Releases a disk or magnetic tape volume
Dump	Displays the contents of files in various ways
Edit {file	Invokes TPU interactive editor
Edit/acl	Editor for creating or modifying access control lists
Edit/fdl	Editor for creating RMS File Definition Language specifications
Help	Utility to display help messages
Init	Formats and writes labels on mass storage volumes
Install	Makes images know to system and to elevate image privileges
Job Controller	Controls batch and print queues
LMF	Controls use of licensed products
Librarian	Creates and modifies library files
Link	Links program object modules into program images

Command	Description
Mail	Electronic mail utility
Message	Compiles message definition files
Monitor	Displays or records system usage
Mount	Makes disk or tape volumes available
Opcom	Operator communication facility
Print	Queues files for printing
Product Installation	Procedure for installing products {layered, system}
Rename	Changes names of files
Reply	Operator to user communication facility
Request	System resource request facility
Runoff	Text format utility
Search	Searches specified files for string
Set-Show	Sets or shows process characteristics
Shutdown	System shutdown procedure
Submit	Queues batch jobs
Symbionts	Controls print and batch queues
Sysboot	Conversational bootstrap program
Sysgen	Modifies system parameters
Type	Displays files on terminal
UETP	System test suite

APPENDIX C

MICA/ULTRIX PORTED COMMANDS AND UTILITIES

The MICA/ULTRIX User Interface provides the following tools and utilities for compatibility with the proposed IEEE 1003.2 (POSIX) standard.

ar
at
awk
basename
batch
cancel
cat
chmod
cmp
comm
cp
cpio
cron
crontab
cut
date
dd
diff
dircmp
dirname
echo
ed
egrep
env
expr
false
fgrep
find
grep
id
join
kill
line
ln
logname
lp

lpstat
ls
mesg
mkdir
mknod
mv
newgrp
nl
paste
passwd
pr
pwd
rm
sed
sh
sleep
sort
stty
su
tail
tar
tee
test
time
touch
tr
true
tsort
tty
umask
uniq
wc
xargs

The MICA/ULTRIX User Interface provides the following tools and utilities for compatibility with ULTRIX-32.

addbib
apply
apropos
bc
cal
calendar
catman
ccat
checknr
chfn
chgrp
chown
chsh
clear
col
colcrt
colrm
compact
dc
deroff
df
diction
diff3
diffmk
du
edit
ex
expand
explain
file
finger
fold
graph
groups
head
indxbib
install
iostat
ipcrm
ipcs
last
lastcomm
learn
leave
lock
login
look
lookbib

m4
makekey
man
more
mount
mt
nm
nroff
od
page
pagesize
plot
print
printenv
ps
pstat
red
refer
renice
reset
rev
rmdir
roffbib
script
size
soelim
sortbib
spell
spellin
spellout
spline
split
strings
style
sum
sync
tabs
tbl
tset
ul
umount
uncompact
unexpand
uptime
users
vc
vi
vmstat
w
wall
whatis

whereis
which
who
whoami
write
yes

The MICA/ULTRIX User Interface provides the following compiler and programming tools for compatibility with ULTRIX-32.

ar
cc
eyacc
lex
lint
ld
make
sccs
yacc

The following utilities make up the Source Code Control System (SCCS) and are included in MICA/ULTRIX.

admin
cdc
comb
delta
get
prs
rmdel
sccs
sccsdiff
unget
val
what

The MICA/ULTRIX User Interface includes the following utilities related to the 'C' programming language.

Utility	Description
cb	C program beautifier
cflow	generate C flow graph
cxref	generate C program cross reference
indent	indent and format C program source
mkstr	create an error message file by massaging C source
xstr	extract strings from C programs to implement shared strings

The MICA/ULTRIX User Interface provides the following ULTRIX-32 compatible printer interface. These utilities are modified to support the MICA system print spoolers.

Utility	Description
lpc	line printer control program
lpd	line printer daemon
lpq	spool queue examination program
lpr	off line print
lprm	remove jobs from the line printer spooling queue
lprsetup	automated line printer spooler set up
cancel	cancel print job
lp	submit print job
lpstat	show print queue status

The MICA/ULTRIX User Interface provides following ULTRIX-32 compatible mail utilities.

Utility	Description
biff	be notified if mail arrives and who it is from
comsat	biff server
from	identifies sender of mail
mail	send and receive mail
newaliases	rebuild the data base for the mail aliases file
prmail	print out mail in the post office
rmail	handle remote mail received via uucp
sendmail	send mail over the internet

The MICA/ULTRIX User Interface provides the following ULTRIX-32 compatible networking utilities.

Utility	Description
arp	address resolution display and control
ftp	file transfer program
ftpd	DARPA Internet File Transfer Protocol server
gettable	get NIC format host tables from a host
hostid	set or print identifier of current host system
hostname	set or print name of current host system
htable	convert NIC standard format host tables
ifconfig	configure network interface parameters
implog	IMP log interpreter
implogd	IMP logger process
inetd	internet service daemon
netstat	show network status
netsetup	set up local area network files
netx	TCP/IP net exerciser
rcp	remote file copy
rdate	network date client
rexecd	remote execution server
rlogin	remote login
rlogind	remote login server
route	manually manipulate the routing tables
routed	network routing daemon
rsh	remote shell
rshd	remote shell server
ruptime	show host status of local machines
rwho	who's logged in on local machines
rwhod	system status server
setifaddr	set network interface address
talk	talk to another user
talkd	inter-terminal communications server
telnet	user interface to the TELNET protocol
telnetd	DARPA TELNET protocol server
trigger	trigger a target node to request a down-line load
trpt	transliterate protocol trace
cu	connect to a remote system
tip	connect to a remote system
rmail	handle remote mail received via UUCP
uuclean	UUCP spool directory clean-up
uucompact	UUCP administrative utilities
uucp	unix to unix copy
uucpsetup	set up UUCP files
uudecode	decode a binary file for transmission via mail
uuencode	encode a binary file for transmission via mail
uulog	show log of UUCP activity

Utility	Description
uumkspool	UUCP administrative utilities
uumonitor	monitor the UUCP system
uuname	list known UUCP systems
uurespool	UUCP administrative utilities
uusend	send a file to a remote host
uustat	UUCP status inquiry and job control
uux	unix to unix command execution

The MICA/ULTRIX User Interface provides the following ULTRIX-32 system management tools and utilities.

Utility	Description
ac	login accounting
accton	system accounting
adduser	add user accounts
addgroup	add group IDs
edquota	edit user quotas
eli	error logger initializer
pac	printer/plotter accounting information
quot	summarize file system ownership
quota	display disc usage and limits
quotacheck	file system quota consistency checker
quotaon	turn file system quotas on
removeuser	remove user accounts
repquota	summarize quotas for a file system
sa	system accounting
shutdown	close down the system at a given time
swapon	specify additional device for paging and swapping
syslog	log systems messages
uerf	ULTRIX error report formatter
vipw	edit the password file

The following tools and utilities are not provided the MICA/ULTRIX User Interface. Equivalent functionality is provided by the MICA System Management Interface.

To Be Determined

The following tools and utilities are not provided the MICA/ULTRIX User Interface. The software listed below is either obsolete or unsupported by ULTRIX-32.

adb
analyze
arcv
arff
as
bad144
badsect
ccr
chpt
clri
cmx
config
crash
dcheck
diskpart
doconfig
drtest
dskx
dumpfs
efl
error
f77
fed
flcopy
fp
fpr
fsck
fsplit
fsx
gcore
getty
halt
icheck
init
kgmon
lisp
listz
lorder
lpx
lxref
mdtar
memx
mkfs
mklost+found
mkproto
mtx
ncheck
newfs

opser
pdx
pi
pix
pmerge
pti
px
pxp
pxref
radisk
ratfor
rc
rdt
reboot
rmt
rxformat
savecore
setld
shmx
sizer
sticky
syscript
sysline
tc
tk
tp
trman
troff
tunefs
units
update
vgrind
vlp
vtfontinfo
vtroff
vwidth

+---------------+
! d i g i t a l ! I n t e r o f f i c e M e m o r a n d u m
+---------------+

To: List

Date: 30 May 1985
From: Richard Grove
Dept: Technical Languages
DTN : 381-2082 Loc.: ZKO2-3/N30
Enet: Orphan::Grove
File: RBG066 Rev.: 1

Subj: DECwest Trip Report: Technical Summary

CAUTION

The NONVAX program discussed here is a major strategic undertaking; all information about the program is COMPANY CONFIDENTIAL.

Please limit the distribution of this report.

Many of the topics mentioned here are still under discussion and all will be subject to a wide-ranging formal corporate review. This report is a snapshot taken in mid-May.

1 GENERAL

This memo summarizes my visit to DECwest Engineering May 13-16. The purpose of my visit was to meet with Dave Cutler and Don MacLaren and to begin work on software architecture issues for the NONVAX family of high-performance machines. The meeting was very successful. There is a substantial consensus on technical issues and approaches. We outlined the software architecture documents that need to be produced and we started work on some of the technical content. We agreed on an informal process for getting more people involved in the technical work.

The new architecture project has no official name or codeword, so I have used the name NONVAX in this report.

The hardware and software products that are being designed can be briefly described as "a non-VAX architecture that will be an upward extension of the VAX/VMS family". The initial product will be oriented toward the scientific/technical computational market. The operating system will have a compatible file system, network, and user interface with VMS. NONVAX systems will also be able to be clustered with VAX/VMS systems.

The system should provide higher-level languages that are compatible with VAX. Key languages for the FCS product are FORTRAN, and perhaps C or Pascal. BLISS will be needed for porting VMS layered products and a new systems implementation language will be used for writing most of the NONVAX operating system. We expect that there will be strong demand for the other VMS languages, and that all will be ported soon after FCS.

In general, Dave expects that DECwest will produce:

1. A NONVAX hardware emulator and the first NONVAX product
2. Operating system
3. File system and RMS
4. DECnet
5. Linker and some utilities
6. System implementation language, most likely based on the ELN Pascal language and compiler

There was agreement that SpitBrook would produce:

1. FORTRAN compiler
2. BLISS compiler to be used in porting various VMS layered products to NONVAX. While the operating system will not use BLISS, it will be an important implementation language for many products.
3. RTL
4. DEBUG
5. Additional compilers and layered products for future releases

2 SCHEDULE

The project has no schedule yet, but the following dates and timescales are being discussed:

- On the order of 3 years to FCS, 2.5 years to Field Test
- July 1985 - publish NONVAX architecture SRM, first draft
- Summer 1985 - review/revise/approve SRM
- Fall 1985 - begin detailed design of operating system. This effort is expected to proceed like the original VMS design. The first 6 months will be devoted to design of the operating system and documenting that design in a Working Design Document.
- March 1986 - First version of Implementation Language cross-compiler available to operating system group.

3 HARDWARE ARCHITECTURE

The NONVAX hardware architecture team was meeting at DECwest and Don MacLaren and I spent one day talking with them about various language issues. The hardware architecture team consists of:

- Dave Cutler
- Dileep Bhandarkar (VAX architecture, LTN)
- Wayne Cardoza (VMS)
- Dave Orbits (SAFE, MRO)
- Rich Witek (Chips, HLO)

The hardware architecture team is making good progress. They expect to produce a complete draft of the SRM by late June or early July. The draft SRM will be widely distributed and reviewed within DEC, just as the original VAX SRM.

Attachment 2 is the architecture team's statement of goals and constraints.

3.1 Architecture Characteristics

This section enumerates a number of key characteristics of the hardware architecture that is being defined and comments on some of the software implications.

- NONVAX has a simple instruction set architecture that is intended to facilitate high-speed implementations and pipelining.

  Each instruction is 32 bits long and there are a small number of instruction formats.

- NONVAX supports VAX datatypes, memory layouts, and byte addressability. These are the key attributes required in order to produce high-fidelity VAX-compatible compilers for NONVAX. The floating point format is the same scrambled VAX representation, and bytes and bits within a longword are numbered and addressed like a VAX.

- NONVAX has 64 registers, each is 64 bits wide. The machine is a Load/Store machine. That is, only Load and Store instructions can reference memory operands. All arithmetic and logical operations are performed on values contained in the registers. In short, it looks a good deal like a CDC 6600.

  The current NONVAX design is strongly influenced by the SAFE design that has been done in Marlboro. The NONVAX is much more like SAFE than RISC machines such as TITAN or HR32.

- The major departure from VAX datatypes is that NONVAX addresses are 48 bits. The standard semiconductor metrics (i.e. one address bit every 2 years) indicate that VAX will run out of physical and virtual address bits sometime in the 1990's. The architecture team is convinced that NONVAX must provide more address bits.

  The 48-bit address space is "flat". That is there are no visible segments, and address arithmetic propagates carries through 48 bits (across page and segment boundaries).

  The design of the page tables and the operating system is intended to support very large sparse address spaces at a very low cost in physical memory for segment and page tables. For example, it is deemed reasonable to allocate 32Mb for a task stack and to isolate that stack by a 32Mb guard region of unallocated memory.

  Larger addresses will be the most painful problem in providing VAX-compatible higher level languages. Data structures that contain pointers will have to allow a larger field, and so the structure layouts cannot be identical between VAX and NONVAX. In languages that have real pointer types (e.g. Ada, Pascal, PL/I), most of this can be hidden by the compiler. FORTRAN programmers that manipulate pointers using the %LOC function and 32-bit integers won't be so fortunate.

- The page size will be much larger than VAX, probably at least 16Kb. There is also talk of having the architecture specify a minimum (16Kb) and maximum (512Kb) page size rather than a single fixed page size. In order to port images between different implementations, the software would have to make the smallest unit of virtual allocation and protection be the maximum page size (512Kb).

  The larger page size is desirable because it will provide more untranslated address bits to be used by direct-map caches, and there will be less overhead for managing pages in the operating system. The larger page size is undesirable because the allocation granularity is larger and there will be more breakage when allocating small chunks that have different protection attributes.

- The NONVAX hardware does not support unaligned data access. That is, INTEGER*4 quantities must be aligned on a longword boundary, REAL*8 on a quadword boundary, etc. The NONVAX operating system will handle alignment faults and quietly fix up non-aligned references, much as MicroVAX emulates string and decimal instructions. However, this emulation will incur a substantial performance penalty; on the order of 100 times slower than a properly aligned reference. Because the performance hit is so large, we expect that compilers will provide an optional "natural boundary" alignment when laying

out records and data structures. The compilers will also provide the VAX "pack to the nearest byte" layout. Both VAX and NONVAX compilers should provide both layouts, and should provide a reporting option to identify misaligned items.

- NONVAX is a "base register" machine. Since all instructions are 32 bits long and addresses are 48 bits long, you can't include a full address in a single instruction. Thus unlike the PDP-10, PDP-11, and VAX you must use base-displacement addressing to reference memory operands. As another consequence, the calling conventions define the format of a linkage section and when you call a procedure you must provide a pointer to the called procedure's linkage section as an argument.

  Load/store instructions have a signed 14-bit displacement field, so the displacement range is +-8K bytes from the base address. Jump-class instructions (JSR, conditional and unconditional branches) have a 20-bit displacement in longword units which provides +-2Mb displacements. Compilers won't need Jump/Branch resolution since 2Mb of code in a single object module seems like plenty (at least for the first release).

- There are Load/Store instructions to load bytes, words, longwords, and quadwords. For short quantities, you can sign-extend or zero-extend as you load the value into a 64-bit register. All integer and logical operations operate on 64-bit register values. There are convert instructions to convert a 64-bit value to a byte/word/longword and check for overflow.

- The instruction set is much simpler than the VAX. This means that the following classes of VAX instructions are NOT provided:

  - There are no string instructions.

  - There are no packed decimal instructions.

  - There are no variable bitfield instructions. Field extraction and insertion are performed by loading a quadword into a register and using SHIFT, ROTATE, AND, and OR instructions. Accessing a bit within a packed bit array will involve separating the bit index into a byte offset and bit within byte, fetching the byte, and then shifting and masking to isolate the bit.

  - In the current proposal, the only floating type with full hardware support is G-floating. There are CVTFG and CVTGF instructions so you can do an F-floating VAX operation by converting both F operands to G, performing a G operation specifying chopping, and converting the result back to F to get the correctly rounded VAX F-float

result. Note that you must perform these converts after EVERY F operation in order to get the same results that VAX does. If you keep temps in G format, you get floating results like the C language where all intermediate calculations are done in double precision.

There has been an active debate about providing more support for F-floating, a full set of F-floating operations. It does not seem desirable to have a machine where single precision is slower than double precision.

- There is no hardware support for D-floating or H-floating.

  There is concern about whether customers will accept a NONVAX that does not efficiently support D-floating; there was a lot of resistance to the G-only MicroVAX I.

  Initial estimates suggest that NONVAX software can probably perform H-floating operations in times that are similar to the VAX warm-microcode approach on machines like VENUS and Nautilus.

We expect that the software will provide a CISRTL (Complex Instruction Set RTL) of highly tuned low-overhead subroutines to perform operations equivalent to many of the missing VAX instructions (strings, decimal, D-float, H-float, etc). The architecture team expects that these "micro code in macro code" routines should provide performance on string operations that is comparable to what VAX microcode delivers.

o NONVAX does not have exception enables (e.g. integer overflow) in the Processor Status word. Rather, enables and rounding mode are encoded directly into the instructions. Thus there are two integer add instructions: ADD (no overflow signaled) and ADDV (add and check for overflow), and similarly for other instructions.

  All arithmetic exceptions are faults; the PC is backed up to the faulting instruction and no result is stored.

o NONVAX does not have condition codes. The compare instructions produce a boolean result (0 or 1) in a general register, and you use a branch on low bit instruction to actually branch on the result of the comparison. This improves branching performance because the compiler can schedule the comparison so that the boolean result is available when the branch is executed. There is also a set of (integer) compare against 0 and jump instructions.

o NONVAX provides Execute-only protection on pages. Thus code generators cannot put literals in the code section.

- NONVAX provides a BPT instruction, but there is no T-bit. In order to execute an instruction that has been replaced by a breakpoint, the debugger will have to decode and interpret the instruction. The simple fixed-format instruction set should make this a reasonable approach.

- The hardware is designed to support symmetric multi-processing and the operating system will support multiple threads of execution within a single process (address space). The Run-Time Library will have to be fully reentrant; AST-reentrancy is not sufficient. All compilers should (at least optionally) generate fully reentrant code and calling sequences.

4 SOFTWARE ARCHITECTURE

Don MacLaren and I enumerated a number of specifications that will define the software architecture. These cover familiar topics from VAX such as data types, calling conventions, run-time environment structure, object language, debug symbol table, etc.

We expect that the software architecture process for NONVAX will be much like the VAXS/VAXL process. Initial proposals for new designs will be prepared by small working groups and reviewed by a larger group representing many languages and products. Stable specifications will be updated by an ECO process. We plan to collect all specifications for the software architecture in a multi-volume notebook set. An outline for the notebook set is included as Attachment 1 of this trip report.

We worked on calling conventions in detail and came up with Rev 0 of the calling conventions. We will be presenting this proposal to groups in DECwest and SpitBrook and then writing up a first draft.

We discussed goals and constraints for VAX compatibility. Our general goal is that user programs written in higher-level languages such as Pascal or FORTRAN should run without change on the NONVAX and produce similar results. In some cases it may be necessary to make changes in programs. For example, a FORTRAN program that manipulates addresses in INTEGER*4 variables won't work on a machine with 48-bit pointers. It should be possible to modify such programs so that the modified source can be compiled to work correctly on both machines.

NONVAX compilers will use the same set of default types that VAX compilers do. That is, in FORTRAN or Pascal, "integer" will mean a 32-bit signed longword, "real" will mean 32-bit F-floating, and "double" will mean 64-bit floating (G-floating or D-floating depending on the compiler's /G_FLOATING switch). Since INTEGER*8 will be an important type on NONVAX, we expect that we will also have to provide INTEGER*8 in VAX compilers.

We expect that VAX and NONVAX will coexist (in clusters and networks) for a relatively long time, so it is very important to do a good job on compatibility issues.

5 CALLING STANDARD

The NONVAX calling standard must preserve a number of key attributes of the VAX calling standard, including:

- Mixed language programming.
- Use of procedure calls as the primary interface to all services and subsystems.
- NONVAX must preserve the "programmer's view" of procedure calls as seen on VAX from the perspective of a higher level language.
- NONVAX will provide by value, by reference, and by descriptor mechanisms for argument transmission and the caller will have the same degree of control as on VAX.
- NONVAX will provide a condition handling mechanism that is functionally equivalent to VAX.

The NONVAX calling standard will add a number of new attributes:

- There will be greater emphasis on performance of procedure calls.
- NONVAX will provide "lighweight" procedures whose performance is similar to JSB linkage on VAX. Lightweight procedures will be invoked using the standard calling sequence, so this is purely an optimization performed by the compiler based on the complexity and requirements of the called procedure.
- The standard NONVAX procedure call will pass the first 4 arguments in registers; procedures with 4 or fewer arguments won't use an argument list in memory.
- All descriptor formats must be modified to accomodate 48-bit addresses. Some improvements in descriptor design are planned. For example, it seems desirable to provide only non-contiguous array descriptors; this will improve communication between FORTRAN and Pascal or PL/I.
- All parametric procedures will be passed as bound procedure values rather than as entry point addresses.
- The NONVAX calling standard must make effective use of the much larger number of registers. Registers R0-R15 will be scratch registers and need not be preserved. Interprocedural

analysis (in a mixed language environment) will tailor linkages to reduce call overhead.

I will be distributing a draft of the NONVAX calling standard for review and comment.

6 DEVELOPING THE SOFTWARE ARCHITECTURE

Don MacLaren will be visiting SpitBrook June 11-14 to continue work on the software architecture. I will serve as host and will organize a number of technical meetings. The primary topics for discussion are:

1. NONVAX software architecture

   A general discussion of the overall software architecture, the specifications that need to be written, and architectural process.

2. NONVAX calling standard

   Discussion and review of the first draft of the NONVAX calling standard.

3. System Implementation Language (SIL) requirements

   As noted above, current thinking is that the NONVAX systems implementation language will be based on the ELN Pascal compiler and language. The operating system group has a goal of writing "almost all" of the operating system in a higher-level language. They estimate that there will be 10K to 20K of very low level kernel code written in Macro, and all the rest in the SIL.

   All of the RTL, including the math library, will be written in the SIL. The SIL must provide facilities for exploiting all features of the hardware architecture.

   The SIL effort has a number of constraints, with schedule being the most pressing. With less than a year to develop an initial version, we can't plan to design and implement the ultimate SIL.

   The goal of this session is to have an open but disciplined brainstorming session on the requirements for a new systems implementation language. All participants must agree to avoid language chauvinism and religious fanaticism and focus on requirements.

   We assume there will be two primary implementation languages: BLISS-64 and a new NONVAX implementation language derived from ELN Pascal. The focus of this discussion will be requirements for the new language; we may also discuss BLISS

extensions to support VAX-NONVAX portability.

4. DEBUG and PCA

   Symbol table requirements and design for DEBUG and PCA.

5. RTL and multitasking

   A goal of the NONVAX software architecture is to provide support for multiple threads of execution within a single process (address space). This will provide facilities like tasking in Ada and VAXELN.

[end of RBG066.RNO]

SOFTWARE ARCHICTECTURE SPECIFICATIONS RELATED TO LANGUAGES

Don MacLaren -- 20-May-1985

This is an initial outline of specifications needed in this area with commentary on the purpose of some of them. It's based on notes from a discussion with Rich Grove, but I have added some additional material.

We have divided the specs into three "books", with the first book being the one of current interest. There is overlap between the material included here and what one finds in operating system documentation.

NOTE: This is a preliminary outline of a set of working documents. Nothing is approved; everything is subject to change.

1 THE EXECUTION ENVIRONMENT

This book, together with the hardware architecture book, describes the runtime environment in which compiled programs execute -- excluding operating system characteristics that have little effect on compiled code. Because the instruction set contains no high-level instructions such as the VAX call, some of the material here would fall in the VAX hardware manual.

1.1 Data Types

This section describes all data types recognized in the hardware architecture. Recognition doesn't imply comprehensive support. However it does mean that treatment of the data type will be covered in the calling standard and that the debugger will understand the type.

Alignment and structure mapping are covered.

1.2 Stack Structure And Register Conventions

Covers stack frames, LP, SP, FP, exception handling and unwinding.

1.3 The Calling Standard

1. Linkage conventions including linkage section related to calling.
2. Argument lists: registers vs. memory, optional, variable-length, etc.

3. Standard and Variant Argument Passing Conventions By Data Type.

4. By-value arguments -- a true thorn bush for those that don't fit in a register.

5. Descriptors.

6. Returning Function Values.

1.4 Tasking

1.5 AST's

In particular, an explanation of the best ways to make code AST reentrant or non-AST-interruptible.

1.6 Status Codes And Error Messages

This to include a more contemporary fao capability.

1.7 CIS Specifications

Specifications for standard complex instruction sequences. This covers sequences for things like MOVC, but it is not restricted to analouges of the VAX instructions.

How such sequences fit into the system must also be explained.

1.8 Miscellaneous Issues

1. One-time initilization in programs.

2. Same thing in tasks?

3. Effect of hardware exception treatment when reflected to the level of a typical language.

1.9 Compatibility

Significant differences from VAX/VMS and how to deal with them.

2 COMPILER INTERFACES

This book describes various substantial structures that are the compilers' interfaces to other system tools. The title was chosen in desperation.

The architectural specifications in this book are open; they will be made available to users.

1. The Object Language.

2. Debug Symbol Table.

3. CDD interface.

4. Librarian interface.

5. Diagnostics Output Records. This is what the language-sensitive editors read to help a user correct his source program.

6. Definition Modules. One of these modules contains definitions of data structures and procedure entries. The form is acceptable to all the compilers.

7. Information Output Records. These give additional information about a compilation, its usage of symbols, etc.

Items 6 and 7 are new, at least relative to our VMS environment. Their inclusion is aimed at supporting improved programming environments.

3 INTERNAL COMPILER ARCHITECTURE

This book describes the common structure of compilers that use the common back end. It is the basis for the development of the initial set of product compilers.

This is not likely to be a public document.

To: Distribution
Date: 28 May 1985

From: Dave Cutler
Dileep Bhandarkar
Wayne Cardoza
Dave Orbits
Rich Witek

Subject: Architecture Review

A strategic effort has begun within the company to define a new architecture for the 1990s that will complement our current VAX/VMS offering.

To this end, a small group has been chartered by engineering management to define and draft an architectural standard for this new family of machines. The intent is to run the architectural process as we did for the VAX, and to solicit and encourage various people to contribute. The architecture standard will be widely reviewed.

Below is the initial set of goals and constraints the architectural group has set for the new architecture. We would like you to review these goals and provide feedback in the form of a written response.

It is important to undertake this effort, and have the proper goalset.

In the future, you will also be asked to review the architecture document. We welcome your input and comments.

Assumptions

The following major assumptions have been made for this architectural effort:

1. There will be one, and only one, architecture across all implementations.

2. The architecture is not restricted to be a RISC architecture. It is a non-VAX architecture that will be an upward extension of the VAX/VMS family.

3. The aspects of VAX that make it hard to build fast machines quickly and easily will be changed.

4. This is a "from scratch" effort. The VAX System Reference Manual will be used as the starting point.

5. This is a strategic effort within the company. There is a high-level of commitment to implement this architecture and provide the necessary resources.

6. An implementation may use microcode. (That is, there no constraint against the use of microcode.)

7. This design work is similar to the the VAX architectural effort; we are not going to produce anything stopgap.

8. An architectural document will be produced as soon as possible. It will be reviewed by a larger group and then by the corporation.

9. It is expected that it will take around three months to define the architecture.

10. It is assumed that an operating system environment will be constructed that has a compatible file system, network, and user interface with VMS. It is also assumed that the system can be clustered with VMS.

11. It is assumed that ULTRIX will be ported to this architecture.

Goals

The major goals are:

1. To minimize the software investment over time.

2. To allow VMS layered products to be moved to the new architecture.

3. To allow faster, more cost effective machines to be built.

4. To allow shorter development cycles.

5. To allow for parallelism in instruction execution.

6. To allow for Symmetric Multiprocessing (SMP).

7. To have a "pipelineable" orientation.

8. To be "subsettable."

9. To be the new corporate architecture for 1990s.

10. To fix anticipated deficiencies and limitations in VAX, e.g., limited physical and virtual address size.

11. To support the Digital I/O interconnect strategy.

12. To provide floating point accuracy greater than or equal to VAX.

13. To ensure that high-level language programs produce the same results as they would on a VAX.

Non-Goals

Non-goals are:

1. To include VAX-compatibility mode.
2. To support Unibus/Qbus/Massbus peripherals.
3. To translate VAX macrocode transparently and efficiently.

Constraints

The architecture is constrained by the following:

1. It must support VAX-compatible data types.
2. It must support VAX-compatible memory addressing (byte addressability).
3. It must be compatible with the way VAX handles interlocked I/O queues.
4. It must be able to execute the same executable image on every implementation.
5. It must provide at least twice the performance/cost of a particular VAX implementation, using the same technology.

Posted: Tue 28-May-1985 15:03 PDST
To: @ART

PRISM

ARCHITECTURE

DECwest Engineering

Rev:2 / January 1986 #1

PRISM:

Parallel
Reduced
Instruction
Set
Machine

DECwest Engineering

Rev:2 / January 1986 #2

Is PRISM "RISC"?

- RISC = Reduced Instruction Set Computer
  - IBM 801
  - Berkeley
  - Titan, Safe, Cascade...
- Small Instruction Set
- Instructions in Hardware
- "RISC" compared to VAX?

But...
PRISM isn't RISC!

PRISM is:

- Parallel Architecture
- Vectors
- Some "RISC" Concepts
- Some "CISC" Concepts
- Designed for PERFORMANCE

PRISM ARCHITECTURAL GOALS

- High (Absolute) Performance
- 2:1 (or better) Cost/Performance
- VAX Compatibility
- VAX Extension Architecture for 1990's

DECwest Engineering

Rev:2 / January 1986 #5

Why Not Build A Faster VAX?

- Complex (microcoded) instruction set
- Variable length instructions
- Many addressing modes (good & bad)
- 512 byte page size
- Autoincrement & Autodecrement
- Condition codes force synchronization
- Lots of unused functionality

DECwest Engineering

Rev:2 / January 1986 #6

PRISM
ARCHITECTURE OVERVIEW

- 32-bit architecture
- 32-bit virtual address space
- 45-bit physical address space
- VAX compatible memory addressing
- VAX compatible data types
- Scalar and vector processing
- Symmetric multiprocessing

DECwest Engineering

Rev:2 / January 1986 #7

SCALAR PROCESSING

- 64 32-bit registers
- 8-, 16-, 32-bit integers/logicals
- 32- and 64-bit F_ and G_Floating
- Parallel instruction execution
- Comprehensive, yet simple, instruction set
- Load/Store memory referencing

VECTOR PROCESSING

- 16 vector registers
- 64 elements per vector register
- 64-bits per vector element
- 32-bit integer/logicals
- 32- and 64-bit F_ and G_Floating
- Single instruction processes entire vector register
- Similar to Cray-2 vector functionality

DECwest Engineering

Rev:2 / January 1986 #9

MEMORY MANAGEMENT

- 32-bit virtual address space
- Basis for relocation, protection and paging
- Execute protection for proprietary code
- 8KB page size for added TB efficiency

DECwest Engineering

Rev:2 / January 1986 #10

PRISM ADVANTAGES

- Fixed length instructions
- Lots of registers
- Parallel execution; out-of-order completion
- No (synchronous) condition codes
- No compound instructions
- No microcode required
- Large pages

CRYSTAL

HARDWARE SUMMARY

DECwest Engineering

Rev:2 / January 1986 #1

Crystal Processor

Basics:

- PRISM Architecture
- Air-Cooled, ECL Multiprocessor (1-4)
- Scalar with vector option(s)

Performance:

- 3X equivalent VAX per scalar unit
- 100+ MFLOPS (Peak) per vector option
- Memory Size = 128 to 512 MBytes
- I/O Bandwidth > 50 MBytes/sec

Transfer Costs:

	Proc	Memory	MLP	TC	Markup
Entry Kernel	2	128 MB	$ 1465K	$ 169K	8.7X
Max Kernel	4	512 MB	$ 3733K	$ 451K	8.3X

DECwest Engineering

Rev:2 / January 1986 #2

Crystal Scalar Unit

Performance through both fast clock cycle and parallelism

- High speed ECL gate arrays - 15nS cycle
- Retire an instruction each cycle
- Four independent function units
- Fully pipelined multiply and add
- Separate instruction and data caches
- Data cache is writeback

DECwest Engineering

Rev:2 / January 1986 #3

Crystal Vector Option

Architecture and fast clock cycle provide very high throughput

- Sixteen multiported vector registers
- Four autonomous function units
- Fully pipelined multiply and add
- 132 Mflops peak performance
- 2 board addition to Scalar processor

DECwest Engineering

Rev:2 / January 1986 #4

Crystal Memory System

Parallelism used to achieve very high performance

- 1-2+ Gigabytes/Second
- MultiLevel cache hierarchy
- Main memory is 32 way interleaved
- Memory size 128 to 512 MBytes

DECwest Engineering

Rev:2 / January 1986 #5

Crystal I/O System

Multiple Channels and Independent Processors used to achieve high bandwidth

- VAXBI allows standard DEC devices
- I/O Processors off-load main CPU
- VAX as IOP allows BCA
- Memory on IOP maximizes BI bandwidth
- Eight VAXBIs provide 64Mbytes/sec

DECwest Engineering

Rev:2 / January 1986 #6

PRISM/VMS

Software Summary

DECwest Engineering

Rev:2 / January 1986 #1

PRISM/VMS GOALS

- Quality
- Robustness, Extendability, and Maintainability
- New Functionality
- VMS Compatibility
- Schedule
- Performance

DECwest Engineering

Rev:2 / January 1986 #2

SOFTWARE SUMMARY

- Very similar to VAX/VMS
- ULTRIX
- Cluster Support
- Symmetrical MP
- Vector Support
- Multitasking

DECwest Engineering
Rev:2 / January 1986 #3

SOFTWARE SUMMARY - CONT.

- PILLAR Systems Implementation Language

- Layered Languages:
  - Vectorizing FORTRAN
  - BLISS
  - Pascal
  - C

VMS COMPATIBILITY

At user interfaces and at VAX/VMS interfaces:

- System services via compatibility layer
- Disk and Magtape structures
- DCL and utilities
- DECnet and remote terminal support
- Clusters
- Languages, RTL, and debugger
- Layered Products

DECwest Engineering

Rev:2 / January 1986 #5

PRISM COMPETITION

SUMMARY

DECwest Engineering

Rev:2 / January 1986 #1

Competition

- IBM
- Technology Leaders
- Others

High Technology Companies

- Convex (C-1)
- Alliant (FX/1 and FX/8)
- Scientific Computer Systems (SCS)
- Elxsi (System 6400)
- Floating Point Systems (FPS-164 and FPS-264)
- Market Share for These & Other High Technology Companies
  - . 4% FY85 ($625K-1.6M Priceband)
  - . 2% FY85 ($1.6M + Priceband)

DECwest Engineering

Rev:2 / January 1986 #3

Product Comparison Summary

	Crystal	IBM	Convex	Alliant	Amdahl	Cray
Model #	1-4 CPUs	3090 200&400	C-1	FX/8	5890 300&600	2
# Proc	1-4	2,4	5	1-8	2,4	4
VUPs	30-100	21-38	10	3.5-26	30-54	120*
Cost per VUP	$20K	$183K-$188K	$52K	$39K	$170K-$172K	$147K*
MFLOPS	100+ -200+	100 -200	60	94		1600
System Price	$571K-$2.15M	$3.9M-$7.2M	$515K	$270K-$1.0M	$5.1M-$9.3M	$17.6M
VP	Yes	Yes**	Yes	Yes	No	Yes

* Estimated

** Attached Vector Processor

Please note: Crystal systems (which FRS in FY89) are compared to currently shipping competitive systems.

DECwest Engineering

Rev:2 / January 1986 #4

Alliant FX/8

- Price/Performance
  - . 3.5-26 VUPs - $39K per VUP (1-8 Processors)
  - . 94 MFLOPS
  - . $270K-1.0M Systems
- 1-8 Processors
- Vector Processing
- Compiler Technology (Decomposing, Vectorizing)
- Full Fortran Support for Vector Hardware Parallelism

DECwest Engineering

Rev:2 / January 1986 #5

Crystal and Alliant Product Comparisons

	Crystal	Alliant
# of Processors	1-4	1-8*
Processor Bits	32	32/64
Cycle Time	15ns	170ns
Cache	64-256K	64-128K
System Memory	64-512MB	8-64MB
Memory Speed	800MB	188MB
I/O Architecture	IOP	IOP
Vector Processor	Yes	Yes

*1-8 Computational and 12 Interactive Processors

DECwest Engineering

Rev:2 / January 1986 #6

Convex C-1

- Price/Performance
  - . 10 VUPs - $52K per VUP
  - . 60 MFLOPS
  - . Entry System $515K
- 5 Processors
- Vector Processing

Crystal and Convex Product Comparisons

	Crystal	Convex
# of Processors	1-4	5*
Processor Bits	32	32/64
Cycle Time	15ns	50ns
Cache	64-256K	64K
System Memory	64-512MB	4-128MB
Memory Speed	800MB	80MB
I/O Architecture	IOP	IOP
I/O Bandwidth	64-100MB	80MB
Vector Processor	Yes	Yes

* With True Parallelism

DECwest Engineering

Rev:2 / January 1986 #8

Cray-1 and 2

- Cray-1

  . Entry System $8.8M
  . 250 MFLOPS
  . 64 Processor Bits
  . 12.5ns Cycle Time
  . 24 Channels

- Cray-2

  . 4 Processors
  . Entry System $17.6M
  . 1600 MFLOPS
  . 64 Processor Bits
  . 4.1ns Cycle Time
  . 40 Channels

Crystal and IBM Product Comparisons

	Crystal	IBM 3090
# of Processors	1-4	2,4
Processor Bits	32	32
Cycle Time	15ns	18.5ns
Cache	64-256K	64-256K
System Memory	64-512MB	64-256MB
I/O Architecture	IOP	Channels
I/O Bandwidth	64-100MB	96-288MB
Vector Processor	Yes	Yes

DECwest Engineering

Rev:2 / January 1986 #10

MAKING A SUCCESS

OF PRISM

DECwest Engineering

January 16, 1986 #1

PRODUCT POSITIONING

- MARKET
  - High performance
  - Scientific computation
  - Engineering
  - Research

- With AQUARIUS
  - Crystal = high end scientific, computational
  - Aquarius = high end commercial, MIS

- With ARGONAUT
  - Dual processor entry above Argonaut
  - Argonaut = mid-range VAX processor

CONCERNS

- Software schedules tight
- Too much (?) to do now
- FY89 will be here tomorrow
- Software will be gating item

HOW DO WE SUCCEED? - CONT.

- Deliver minimal, but key layered software:
  - Best Vectorizing FORTRAN (period)
  - Supporting tools (LSE, MMS, CMS, PCS, etc.)
  - Sell performance - use the iron!

HOW DO WE SUCCEED?

- Keep our focus narrow:
  - Scientific computing
  - Member of VAX family
  - Top Fortune companies

HOW DO WE SUCCEED? - CONT.

- Catch up with VAX over several years
  - Schedule layered product introductions
  - Complementary offerings (product families)