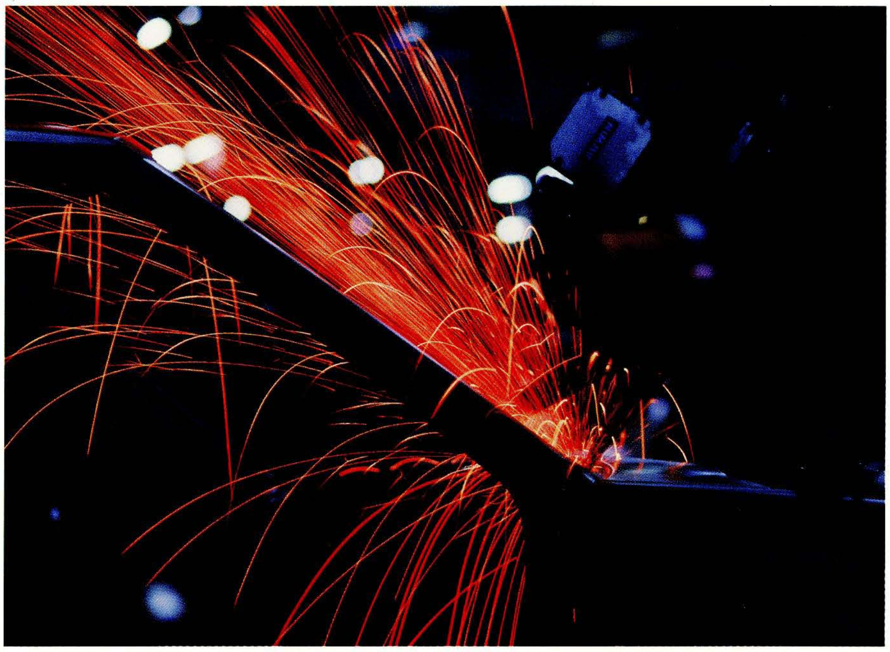
VAXELN Version 2.2

Realtime Software Toolkit for Dedicated VAX and MicroVAX Applications

digital



Features

- Full VAXELN Toolkit that runs on a VMS or MicroVMS host to produce tight execution code on a dedicated system.
- Includes an extended ISO Pascal compiler (EPascal) that is optimized for realtime code. Runtime libraries are also included for EPascal, C, and FORTRAN-77.
- Supports an Ada compiler and Ada runtime libraries, a C compiler, a FORTRAN compiler, and Rdb/ELN Relational Database Software. (All of these are options for the VAXELN Toolkit.)
- Provides a simple, small kernel to control sharing of the system's resources on the dedicated system.
- Supports local debugging on the target system's console terminal and remote debugging over an Ethernet from a user terminal on a VAX/VMS system.
- Includes a powerful system builder that combines program images, the kernel image, and runtime routines into a finished VAXELN system image.
- Supports multitasking and multiprogramming from the VAXELN kernel, the EPascal compiler, the C runtime library, and Ada.®
- Offers device drivers for a range of Q-bus and UNIBUS devices including disks for file storage.
- Provides transparent network communications among VAXELN and VMS nodes via Ethernet-based DECnet (DAP).

Description

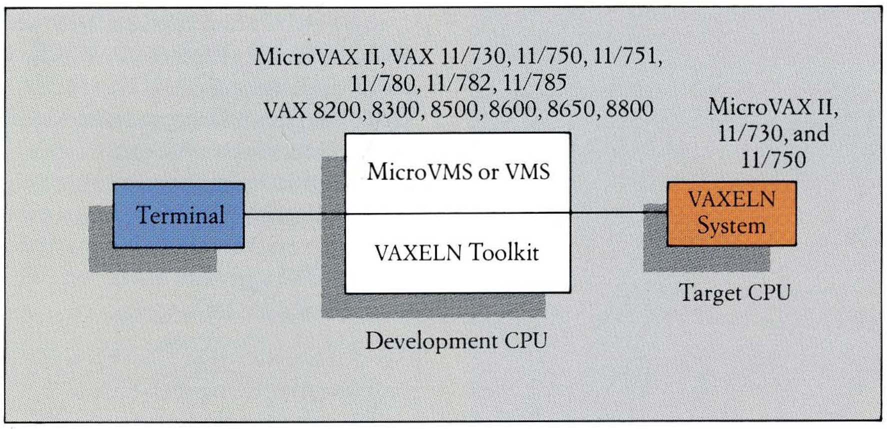
VAXELN Version 2.2 adds support for the VAX FORTRAN-77 runtime library to Digital's realtime software development toolkit. The VAXELN Toolkit runs on a VAX or MicroVAX system to generate tight execution code for a dedicated realtime processor. On the dedicated system, the VAXELN kernel controls the use of shared system resources.

As a VMS layered product, the VAXELN Toolkit gives you access to the rich software development resources of the VMS environment. The Toolkit itself provides optimized tools to help you create an execute-only application that runs under the VAXELN kernel on the dedicated system. VAXELN applications work in situations – such as factory automation, dedicated CAD/CAM workstations, and large-scale data collection/reduction systems – in which individual processors have dedicated or predetermined functions. These applications can be either stand-alone units or part of an Ethernet Local Area Network (LAN) with VAX/VMS nodes or any other node using DECnet service and protocols.

The VAXELN Toolkit Helps You Generate Efficient Runtime Applications

The VAXELN Toolkit gives you the development resources you need to write efficient realtime code. It includes the extended EPascal compiler, libraries with the VAXELN kernel and system services, a system build utility, and a symbolic debugger. An extended VAX C runtime library and VAX FORTRAN-77 routines ship with the VAXELN kit. You also have the option of ordering Toolkit options such as the VAXELN File Service, Network Service, high-performance device drivers, extended VAX Ada runtime routines, template device drivers, the Rdb/ELN relational database, and compilers for VAX C and FORTRAN.

With VAXELN, you work in the familiar VAX/VMS development environment to create your realtime application. (Figure 1 shows the general elements in the application development cycle.) Use VMS language-sensitive editors to help you create the source code. Write your application in the Toolkit's EPascal, with calls to VAX C, VAX FORTRAN-77, and, optionally, VAX Ada.® Link the compiled code to the VAXELN Toolkit runtime library using the standard VAX/VMS linker. For easier debugging either locally or remotely across an Ethernet LAN, use the VAXELN debugger. The VAXELN debugger is based on the VAX debugger and lets you debug both modules and entire applications. When you're ready, combine the necessary images with the VAXELN system builder to create an efficient, low-overhead runtime application.



MicroVAX II, VAX 11/730, 11/750, 11/751, 11/780, 11/782, 11/785
VAX 8200, 8300, 8500, 8600, 8650, 8800
MicroVAX II, 11/730, and 11/750
Terminal
MicroVMS or VMS
VAXELN Toolkit
VAXELN System
Target CPU
Development CPU

Figure 1: *VAXELN Application Development Elements*

EPascal Provides an Extended ISO Pascal Compiler

EPascal is a highly optimized native mode Pascal compiler that builds on the ISO standard to provide realtime extensions. These extensions let you work in the flexible Pascal language – instead of assembly language – to write even device drivers and other performance-critical parts of your code. The EPascal program also lets you include routines written in VAX C, FORTRAN, and MACRO-11.

The EPascal runtime library features support for special I/O operations, standard Pascal routines such as SIN, and certain procedures used in system programming. This support is provided in both object-library and shareable-image forms.

Powerful language extensions simplify and speed application development. For example, you can use a generalized mechanism, flexible types, to declare data with parametric extents. You can predeclare fixed-length strings, varying length strings, and a series of uninterpreted data types. And you can declare procedure and function parameters with conformant extents.

Another EPascal compiler feature effectively extends Pascal-type checking to separately compiled modules to let you create and compile source code in separate modules. As each module is compiled, declarations or cross-references involving it and the modules already in the program library are automatically checked for consistency. In addition, full data-type checking is maintained on the predeclared kernel data types DEVICE, EVENT, MESSAGE, NAME, PORT, PROCESS, and SEMAPHORE.

VAXELN Kernel Controls Tight Execution Code

The VAXELN kernel supports both multitasking and multiprogramming. Its multitasking capability lets you declare parts of a program to be scheduled for concurrent execution with the main program. Its multiprogramming feature enables you to schedule entire programs, including multitasking programs, for concurrent execution on the same CPU.

The kernel uses VAX memory management to map user processes in the P0 and P1 regions of a 4-Gbyte virtual address space. The user stack is independently mapped into the P1 (control) region for each process in the system. A VAXELN process running on a single processor can be as large as the amount of physical memory on that dedicated processor.

In general, VAXELN systems are independent jobs that exchange information by message transmissions. Jobs can be created dynamically to execute a specific program included with the system builder.

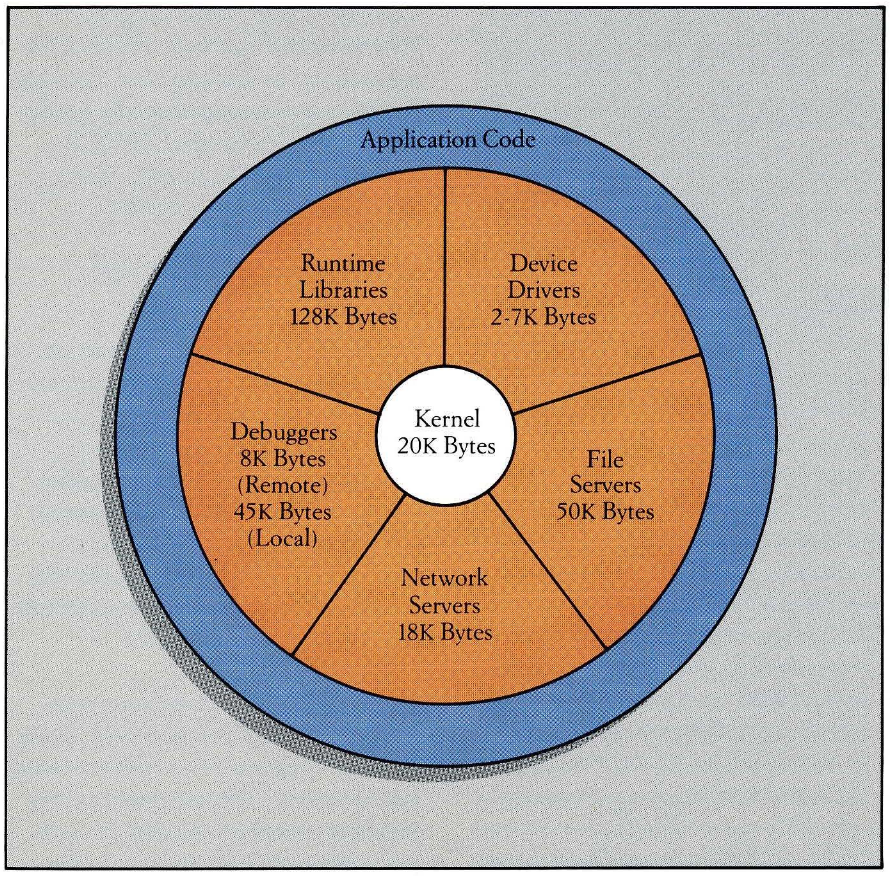

Application Code
Runtime Libraries 128K Bytes
Device Drivers 2-7K Bytes
Kernel 20K Bytes
Debuggers 8K Bytes (Remote) 45K Bytes (Local)
File Servers 50K Bytes
Network Servers 18K Bytes

Figure 2: *VAXELN Application Components*

Jobs running on the same processor are scheduled on a preemptive priority basis. Jobs can be assigned up to 32 priority levels. Within a job, 16 independent priority levels are available for preemptive process scheduling. You can stop then resume process rescheduling within a job.

Broad System Builder Combines Images

The VAXELN system builder does the majority of the application build work for you. It combines program images, the VAXELN kernel image, and runtime routines into one image of the finished VAXELN system. The resulting image keeps overhead to a minimum by including only those images necessary for that particular application.

The system builder accepts program images from a variety of sources, including programs written in EPascal, subprograms written in other VAX languages such as Ada,® FORTRAN-77, C, and MACRO-11. It also collects the necessary images from the VAXELN kernel itself and from the VAXELN Toolkit. Figure 2 illustrates the possible components in a VAXELN application and includes the minimum memory requirements for each.

VAXELN Debugger for Remote Or Local Interactive Debugging

You can use the VAXELN Debugger for interactive debugging either locally at the target system's console or remotely over an Ethernet connection. Either way, you can evaluate expressions in a Pascal-like syntax, execute VAX/VMS-like debugger commands, and debug kernel-mode code. You can also define new debugger commands and variables as necessary.

If you have the optional DECnet-VAX license and Ethernet hardware, you can debug the application from a terminal on another VAX/VMS node. This remote debugger supports symbolic debugging – that is, manipulating variables and other items by their declared names. It can also display the status of all VAXELN processes and jobs in the LAN.

VAX C and FORTRAN-77 Runtime Libraries Arrive With the Kit

The VAXELN product ships with an extended VAX C library and a library of VAX FORTRAN-77 routines. An extended VAX Ada® runtime library is also available as an option. These libraries let you include existing routines in your VAXELN application and give you the choice of development languages.

Extended VAX C Runtime Library. VAXELN supports the VAX C language with an extended C runtime library. The library contains over 130 routines that implement the standard VAX C runtime environment. Realtime extensions to the standard library let you work in this high-level language to write performance-critical code such as device drivers. The VAXELN debugger fully supports the VAX C language and the runtime library.

VAX FORTRAN-77 Runtime Library. The VAX FORTRAN-77 runtime library lets you include standard FORTRAN-77 modules in your application system image. The VAXELN debugger supports both these modules and the runtime library.

VAXELN Options Tailor the Environment to Your Needs

The optional software available for the VAXELN Toolkit lets you select exactly the components you need. In addition to the optional VAX Ada runtime routines, choose from the VAXELN File Service, the VAXELN Network Service, and the Rdb/ELN relational database software.

VAXELN also gives you a choice of the following devices that are supported on the dedicated realtime processor.

- RQDX1, RQDX2, or RQDX3 disk controller with the following drives as appropriate to meet hardware configuration requirements: RX50 (2 by 400 Kbytes) diskette drive; RD51 (11 Mbytes), RD52 (31 Mbytes), or RD53 (71 Mbytes) Winchester disk drive. VAXELN supports up to four of either type of drive.
- KDA50 Q-bus disk adapter for up to 4 Digital Storage Architecture (DSA) disks: RA80, RA81, or RA60.
- RQC25 52-Mbyte fixed/removable disk subsystem and RC25 add-on drives.
- DEQNA Ethernet-to-Q22 adapter
- DZV11 or the DZQ11 4-line asynchronous multiplexer
- DHV11 8-line asynchronous multiplexer
- Console terminal
- LPV11 lineprinter controller
- DLV11 asynchronous serial controller
- DRV11 parallel line interface
- ADV11-C, AXV11-C, and KWV11-C realtime devices
- TK50 streaming tape
- DMF-32 multifunction board (LP32 lineprinter and asynchronous line support)
- RA80/RL02 combined disk drives (VAX-11/730) or dual RL02 disk drives
- UDA50 UNIBUS Disk Adapter and up to 4 DSA disks: RA80, RA81, RA60
- RUC25 52-Mbyte fixed/removable disk subsystem and RC25 add-on disk drives
- TU81 UNIBUS tape drive
- DEUNA UNIBUS-to-Ethernet adapter
- TU58 cartridge tape

VAXELN Ada Routines Add Ada to Your Language Choices

The optional VAXELN Ada® routines enable you to include Ada modules in your application. If you have VAX Ada – recognized as among the best Ada compilers – you can also write your realtime application in Ada. The VAX Ada compiler lets you defer the choice of target system until you link the application, so you can easily build applications that run under either VAXELN or VMS. VAXELN Ada also uses the VAX Ada program library to make it easier to develop applications for both systems. The VAXELN Ada routines are fully integrated into the VAXELN execution environment, giving you access to system services, device utility routines, and distributed application features.

Optional Services for a Variety of Realtime Applications

VAXELN File Service. If your application includes disks, you can include the VAXELN File Service to provide the file support you need. The VAXELN File Service supports I/O operations from EPascal programs to file-storage devices, such as disks, and from other DECnet nodes (with Network Service). The VAXELN File Service uses the VAX/VMS on-disk structure (ODS-2) internal file format.

VAXELN Network Service. When you develop a distributed realtime application, choose the optional VAXELN Network Service. These services provide transparent network communication between VAXELN nodes in a LAN and between VAXELN nodes and VAX/VMS nodes. With Network Services, you can downline load applications across an Ethernet network.

Optional Rdb/ELN Provides Relational Database Support

Rdb/ELN enables you to build your realtime application on a relational database. As a member of Digital's relational products family, Rdb/ELN supports the Digital Standard Relational Interface (DRSI). This support means that applications you developed with other DRSI-based products will run under Rdb/ELN, and applications you create with Rdb/ELN will run under other DRSI products.

Rdb/ELN provides full relational database functionality with excellent performance. It also assures data integrity. Rdb/ELN simplifies application development and increases application flexibility by combining the relational data model with remote database access and the capability for handling application-defined datatypes.

Digital Means Service

The sophisticated work that goes into developing realtime applications requires equally sophisticated support and service. Digital has the worldwide resources required to provide this level of support. These resources take shape with trained hardware and software service representatives available worldwide 24 hours a day, 7 days a week and include a range of service offerings such as hardware and software installation, onsite consulting, and scheduled maintenance.

Digital can help you develop your own resources through our extensive training programs. Choose from options ranging from classroom courses available in our 25 worldwide training facilities to audio/visual courses that you can take at your own site within your own schedule.

Digital means service at every level, everywhere in the world.

VAXELN Toolkit Requirements

The host development system for the VAXELN Toolkit should have at least 1 Mbyte of physical memory, 2 Mbytes of virtual page file quota per user, and a 250-page minimum working set per user. Digital recommends the following systems as host for the Toolkit.

- MicroVAX II
- VAX-11/730, 750, 780, 782, or 785
- VAX 8200, 8300, 8500, 8600, 8650, or 8800

The target dedicated system that will run the VAXELN application should have at least 1 Mbyte of physical memory and a load device. Digital recommends the following processors as the realtime target system.

- MicroVAX II
- VAX-11/730 or 750

Digital recommends the following load devices.

- Ethernet adapters for downline loading
- Files-11 disk, RX50, RA60, RC25, RL02
- TU58 cartridge tape
- TK50 for the MicroVAX II
- ROM (MRV11-D)

Optional Software

- VAX C
- VAX FORTRAN
- VAX Rdb/ELN
- VAX Ada/ELN

Ordering Information for Available VAXELN Products

Order Number	Description
Q*375-UZ	Single-user License for Development System
Q*376-DZ	Single-user License for Target System
Q*387-90-DZ	Multiple Copies of the Single-user License for Target Systems
Q*375-H3, H5, HM	Distribution and Documentation Options
Q*375-HZ	Software Revision Right to Copy Option for the Development System
Q*376-HZ	Software Revision Right to Copy Option for the Target System
Q*372-GZ	Documentation Only Option
Q*375-I3, I5	Installation Service Option
Q*375-93, 95	Digital Support Service Options
Q*375-83, 85	Basic Service
Q*375-33, 35	Self-Maintenance Service
Q*375-E5	Source License and Sources Distribution

*The specific order number is determined by the system for which you are ordering the product.

digital

102707474