TECHNICAL SUMMARY

digital

RSX-11

Contents

1
Introduction

The RSX-11 systems are sophisticated multiuser, multiprogramming operating systems that permit realtime activities to execute concurrently with less-time-critical activities. This Technical Summary introduces the features and capabilities of the RSX-11 family. Application programmers, computer analysts, system programmers, system managers, and operators can use this summary to become familiar with the components, operations, management, and services of the RSX-11 systems.

This document is a technical introduction to many aspects of the three RSX-11 operating systems: RSX-11M, RSX-11M-PLUS, and RSX-11S. In addition to descriptions of the operating systems themselves, this Technical Summary includes information on DIGITAL's PDP-11 processors, peripherals, languages, communications, and support services available with RSX-11 systems.

RSX-11M and RSX-11M-PLUS are multiuser, multiprogramming realtime operating systems that support multiple languages, easy-to-use interactive interfaces, and program development tools and utilities. The operating systems are designed for such time-critical applications as data collection, reduction, and analysis; computation; monitoring; controlling; and, on RSX-11M-PLUS, batch processing.

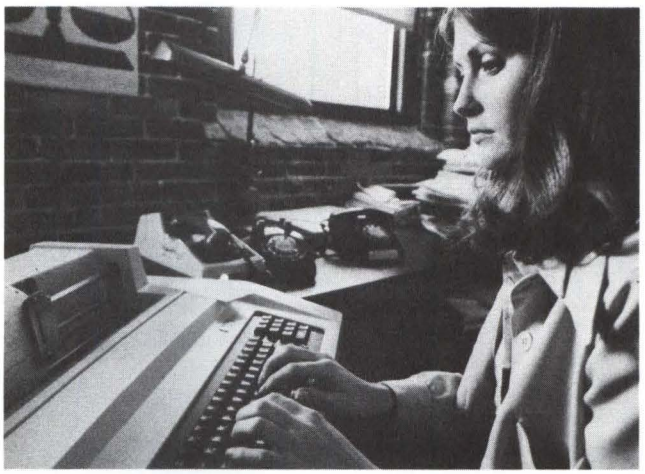
An extension of RSX-11M, RSX-11M-PLUS was specially developed to maximize performance on larger-memory PDP-11 systems. It supports more simultaneous tasks and terminals than RSX-11M. RSX-11M and RSX-11M-PLUS offer unsurpassed compatibility—RSX-11M programs can run with few or no changes on RSX-11M-PLUS.

RSX-11S, the smallest member of the RSX-11 family and a subset of RSX-11M, provides a dedicated, execute-only environment for monitoring and controlling many realtime processes concurrently. RSX-11S program development and system generation take place on a host RSX-11M, RSX-11M-PLUS, or VAX/VMS system. RSX-11S applications are developed using the extensive program development tools available on these larger operating systems, and all RSX-11S system resources are devoted to executing realtime applications.

All three RSX-11 systems are mature, proven operating systems that meet realtime demands efficiently and effectively. Designed for minimum size and overhead, they run on various members of DIGITAL's PDP-11 series of microcomputers and minicomputers. RSX-11 systems can be found in a wide variety of applications—from small, dedicated laboratory and industrial control systems to large multiuser information management systems.

This Technical Summary contains useful information for application programmers, system managers, and computer operators, and is especially appropriate for system programmers and computer system analysts. Nontechnical personnel familiar with computer industry terminology may also find the information useful.

The RSX-11 Technical Summary is designed to be read either sequentially or selectively. The reader might start with Section 1 and continue through the book, or first glance through the table of contents or the abstracts preceding each section to locate those topics of most interest. Many



of the systems' concepts and features are repeated throughout the book in various contexts.

Section 2, *The Systems*, presents a comprehensive overview of the RSX-11 systems' features. This section identifies the characteristics of RSX-11 systems and introduces the systems' features described in detail throughout the remainder of this document.

The *Operating Systems* (Section 3), *The Processors* (Section 8), and *The Peripherals* (Section 9) provide an indepth discussion of the systems' characteristics and capabilities. The concepts developed in the *Operating Systems* and the *Processors* sections are closely related; for example, the discussions on memory management and I/O processing in each section can be read together to gain full appreciation for the systems' design.

One of the most important features of RSX-11 systems is that programmers don't have to know assembly language to use the system effectively. Both the hardware and software contain many features that promote efficient and productive high-level language programming. In addition to the operating systems section, high-level language programmers should be interested in Sections 4, 6, 7, and 10—*Program Development*, *The Languages*, *Data Management Facilities*, and *Data Communications*.

The *System Management and Maintenance* section (Section 5) is especially useful for system managers and operators. It describes the many features of the RSX-11 systems that support system management and control.

Finally, Section 11, *Support Services*, presents an overview of the comprehensive services DIGITAL provides to its computer system users.

2
The Systems

RSX-11 operating systems provide the performance, reliability, and flexibility needed for a full range of realtime applications. These operating systems run on various members of the PDP-11 16-bit series of minicomputers and LSI-11-based microcomputers and can accommodate the most simple to the most complex application requirements.

The RSX-11 systems are multiuser, multiprogramming operating systems that permit realtime activities to execute concurrently with less-time-critical activities. This concurrent execution is accomplished through priority-based scheduling in which the assigned priority and the activities of a task (process) determine the level of service it receives. Realtime tasks receive service according to their priority and ability to execute, whereas such tasks as those used in program development have lower priority and thus receive service after tasks of higher priority execute.

RSX-11 systems are highly reliable. Built-in protection features ensure data integrity and increased system availability. Error detecting and logging and an online I/O exerciser verify system integrity. Many hardware and software features provide for rapid diagnosis and automatic recovery, should power, hardware, or software fail.

INTRODUCTION

The RSX-11M-PLUS, RSX-11M, and RSX-11S realtime operating systems run on DIGITAL's series of PDP-11 microcomputers and minicomputers. Because of their multiprogramming capabilities, RSX-11 systems permit realtime activities to execute concurrently with such less-time-critical activities as program development and data management.

In addition to the operating systems themselves, RSX-11 system software includes numerous utilities, a variety of optional high-level languages, and a choice of communications software.

RSX-11M and RSX-11M-PLUS are multiuser systems for both program development and application-system execution. RSX-11M/M-PLUS have program development tools and utilities and data management facilities with which users can easily develop and efficiently run many types of applications.

RSX-11S is a dedicated, execute-only system. Using the sophisticated development tools on an RSX-11M, RSX-11M-PLUS, or VAX/VMS system, programmers can develop, test, and debug applications and then downline load them to an RSX-11S system.

All RSX-11 systems are priority-scheduled and event-driven; the assigned priority and activities of the tasks determine the level of service they receive.

RSX-11 systems are mature, highly reliable systems. Built-in protection mechanisms in both hardware and software ensure data integrity and increase system availability. An online I/O exerciser and error-logging utility verify system integrity. Hardware and software features provide rapid diagnosis and automatic recovery, should the hardware, software, or power fail.

RSX-11 programmers can take advantage of RSX-11's program segmentation and overlaying capabilities to write programs that need more memory space than is available to them.

RSX-11M-PLUS users can effectively triple the memory directly addressable by tasks on PDP-11/44 and PDP-11/70 systems through the use of Supervisor-mode library routines and separate User-mode I- and D-space. Using this technique, programmers can use fewer (if any) overlays within tasks; this results in faster program execution and decreased program development and taskbuild time.

The family of PDP-11 16-bit minicomputers and LSI-11-based microcomputers is based on a common architecture. Because compatibility is inherent in the design, the RSX-11M and RSX-11S operating systems will run on all members of the PDP-11 family. The newer RSX-11M-PLUS runs on the PDP-11 processors that support 22-bit addressing—the MICRO/PDP-11, PDP-11/23-PLUS, PDP-11/24, PDP-11/44, and PDP-11/70.

Such compatibility provides an upward migration path for growing application needs and a downward migration path to accommodate distributed processing networks.

For example, it's generally possible to develop application programs on any PDP-11 and run them on any other PDP-11 minicomputer or microcomputer.

PDP-11 and LSI-11 processors offer a comprehensive instruction set, a Memory Management Unit that provides hardware memory protection through various operating modes, and an optional Floating-Point Unit. The PDP-11/44 and PDP-11/70 processors also feature fast cache memories as standard equipment. A cache memory is optional on the PDP-11/34A.

SYSTEM COMPONENTS

The major components of RSX-11 systems are:

- *Processors.* DIGITAL's PDP-11 series of minicomputers and LSI-11-based microcomputers offer a comprehensive instruction set, processor protection through various operating modes, an integral Memory Management Unit, and an optional Floating-Point Unit.
- *Peripherals.* These comprise a range of small- and large-capacity disk drives, magnetic-tape systems, hardcopy and video terminals, lineprinters, a card-reader, and several realtime and industrial control devices.
- *Operating systems.* The RSX-11 operating systems are designed specifically to support realtime applications and concurrent program development.
- *Languages.* These include the MACRO-11 assembly language and such optional, high-level languages as FORTRAN IV, FORTRAN-77, PDP-11 COBOL, COBOL-81, PDP-11 BASIC-PLUS-2, and CORAL 66.
- *Utilities and program development tools.* These include editors, linkers, librarians, debuggers, and other such tools to aid in program development.
- *Data management facilities.* These comprise a file system that provides device-independent access to RSX-11 peripherals, volume structuring, and protection; RMS Record Management Services for record management with sequential, random, and indexed record access; FMS-11, a forms management system; and DATATRIEVE-11, a data entry and query language designed specifically for online interactive access and manipulation of files.
- *Data communications services.* These include DECnet network software, Internet protocol emulators, and Packetnet and Ethernet capabilities.

These components combine to provide an easy-to-use, high-performance, flexible, reliable, realtime system.

THE HARDWARE

Hardware for the RSX-11 systems comprises the PDP-11 and LSI-11 processors and associated peripheral devices.

Processors

The PDP-11 minicomputer and LSI-11 microcomputer processors provide 16-bit addressing. A Memory Management Unit expands the physical addressing capability to 18 bits—and, on the MICRO/PDP-11, PDP-11/23-PLUS, PDP-11/24, PDP-11/44, and PDP-11/70 processors, to 22 bits. The Memory Management Unit lets tasks address up to 64 Kbytes of memory on RSX-11M and RSX-11S systems—and 128 Kbytes of memory on RSX-11M-PLUS systems—by mapping (assigning) virtual addresses to physical memory. RSX-11M-PLUS requires the 22-bit Memory

Management Unit. Memory management also provides memory protection in multiuser, multiprogramming systems that require memory protection and relocation facilities.

These processors feature the PDP-11 instruction set, an easy-to-learn yet comprehensive instruction set that is widely known and accepted in the industry; an optional Commercial Instruction Set (CIS) on the MICRO/PDP-11, PDP-11/23-PLUS, PDP-11/24, and PDP-11/44; and eight addressing modes. The processors' general registers can be used as accumulators, pointers, and index registers. Full-vectored interrupts eliminate high-overhead software, and CPU priority levels can be set under software control.

The processors provide the basis for protection and sharing in a multiuser, multiprogramming environment through access modes. Kernel, User, and Supervisor access modes constitute the privilege levels at which programs can run.

High-speed cache memory, which is standard on PDP-11/44 and PDP-11/70 processors and optional on PDP-11/34A processors, buffers data between the processor and main memory, reducing memory access time.

The optional Floating-Point Unit performs all floating-point arithmetic operations and converts data between integer and floating-point formats.

Refer to Section 8, "The Processors," for a more detailed description of the PDP-11's architecture and processors.

PDP-11/24

The smallest UNIBUS processor in the PDP-11 series—the entire CPU fits on a single board—the PDP-11/24 uses large-scale integration to provide performance and memory management capabilities comparable to larger PDP-11 systems. Four Mbytes of memory are supported with the extended 22-bit addressing option. This means the PDP-11/24 can handle more memory-resident tasks and more users, with faster response times. In addition to supporting DIGITAL's higher-capacity disks, the PDP-11/24 supports Winchester disks.

PDP-11/34A

The PDP-11/34A is a midrange minicomputer system offering powerful state-of-the-art features at economical prices. It is a microprogrammed processor with a compact CPU logic contained on two circuit boards for cost-efficient expansion with configuration flexibility. The PDP-11/34A supports up to 256 Kbytes of memory and an optional two-Kbyte cache memory for higher memory performance.

PDP-11/44

DIGITAL's PDP-11/44 processor offers capabilities and performance features previously unavailable on a computer in its price range. It provides such high-performance, large-machine features as a high-speed central processor, support for up to four Mbytes of main memory, optional Commercial Instruction Set, and a large eight-Kbyte high-speed cache memory. In addition, the PDP-11/44 was designed to meet rigorous reliability and maintainability standards.

PDP-11/70

The PDP-11/70 is the most powerful computer in the PDP-11 family. It provides both high system throughput in multiuser, multitasking environments that require large amounts of memory and fast response and computational power for high-speed realtime applications. It's the system-level processor that combines high throughput with expandability and reliability. The PDP-11/70 supports four Mbytes of memory, a fast two-Kbyte cache memory, and both UNIBUS and MASSBUS peripheral interfaces. The MASSBUS is a dedicated 32-bit bus optimized for high-speed data transfers.

LSI-11-Based Processors

The LSI-11 family of 16-bit microcomputers, which is based on the PDP-11 architecture, provides the same capabilities and uses many of the same peripherals as PDP-11s, yet the microprocessor functions are performed by N-channel MOS microprocessor chips that are available in both board and box configurations. The PDP-11/23, PDP-11/23-PLUS, and MICRO/PDP-11 include LSI-11 microprocessors. Peripherals are connected to these processors through the LSI bus.

PDP-11/23 — The PDP-11/23 microcomputer provides the power and the performance of the PDP-11/23-PLUS and PDP-11/24 at lower cost. Its features include memory management with dynamic relocation, segmentation and protection capabilities, up to 256 Kbytes of addressing, and an optional floating point unit.

PDP-11/23-PLUS — Extended addressing on the PDP-11/23-PLUS supports up to four Mbytes of parity memory. This means more tasks can be memory-resident, so, in many applications, more active users can be supported with better system response. In addition, the PDP-11/23-PLUS features an optional Commercial Instruction Set that, in particular, enhances performance of the COBOL-81 compiler.

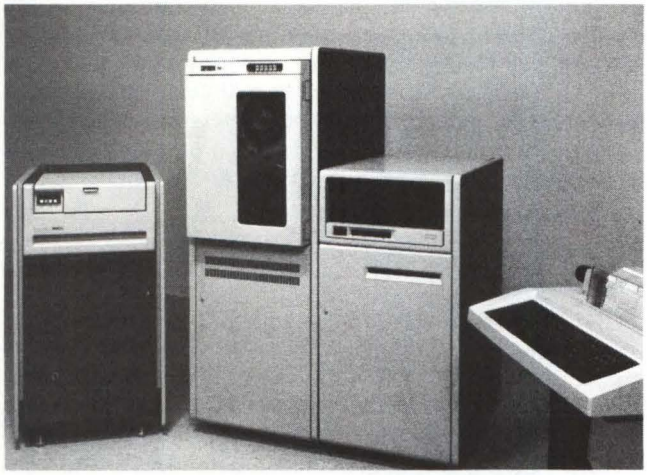
MICRO/PDP-11 — The PDP-11/23-PLUS CPU is also found in DIGITAL's lowest-priced entry-level microcomputer system, the MICRO/PDP-11. The MICRO/PDP-11 includes a CPU, several options, and mass storage—all in a single enclosure that fits on a tabletop, on a narrow floorstand, or in a rack mount. The system combines all the capabilities of the PDP-11/23-PLUS with a ten-Mbyte Winchester system disk and two 400-Kbyte diskettes.

Peripherals

RSX-11 supports a family of peripheral devices to suit the requirements of most applications. Disk drives and magnetic-tape drives can be used for data storage and retrieval and for software distribution. Video terminals, graphic video terminals, and hardcopy terminals are available for communicating with the system and performing special I/O operations. Unit record peripherals, such as lineprinters and cardreaders, provide reliable, high-speed input and output.

The peripherals are connected to the processor through the UNIBUS on larger PDP-11s and through the LSI bus on DIGITAL's microcomputers. In addition to the UNIBUS, the PDP-11/70 uses a MASSBUS peripheral interface to accommodate high-speed storage devices.

Many of DIGITAL's standard realtime interfaces for laboratory and industrial control, data acquisition, and communications can be used with RSX-11. Custom devices can also be connected to RSX-11 systems through user-written device handlers or the connect-to-interrupt executive directive.



THE SOFTWARE

System software comprises the operating system, utilities, data management facilities, language processors, and communications software. Through a question-and-answer system-generation procedure, users can select the features and capabilities they need for their particular application.

The Operating Systems

RSX-11M-PLUS, RSX-11M, and RSX-11S are the industry's leading multiuser realtime operating systems. Designed for minimum size and overhead, this family of operating systems covers a wide variety of hardware and application environments—from small dedicated laboratory and industrial control systems to large multiuser information management systems.

RSX-11M and RSX-11M-PLUS are multiuser, multiprogramming realtime operating systems that support multiple languages, easy-to-use interactive interfaces, and program development tools and utilities. The operating systems are designed for such time-critical applications as data collection, reduction, and analysis; computation; monitoring; controlling; and, on RSX-11M-PLUS, batch processing.

RSX-11S, the smallest member of the RSX-11 family, provides a dedicated, execute-only environment for monitoring and controlling many realtime processes concurrently. RSX-11S program development and system generation take place on a host RSX-11M, RSX-11M-PLUS, or VAX/VMS system. RSX-11S applications are developed using the extensive program development tools available on these larger operating systems, and all RSX-11S system resources are devoted to executing realtime applications.

The RSX-11 systems are mature, proven operating systems that meet realtime demands efficiently and effectively. RSX-11M and RSX-11M-PLUS feature:

- realtime multiprogramming
- multiuser task support
- event-driven priority scheduling
- 250 task priority levels
- dynamic memory allocation
- task checkpointing to disk
- automatic memory compaction
- system-controlled partitions
- Memory Management Unit support
- extensive file-processing, file-sharing, data protection, and data management facilities
- an indirect command file processor
- intertask communication and task sharing
- the easy-to-use DCL—DIGITAL Command Language—interface (compatible with that used on the VAX/VMS operating system)
- user-written CLI (command line interpreter) support
- a comprehensive selection of high-level languages
- system library routines
- clustered libraries support
- online I/O exerciser and error-logging utilities
- a comprehensive selection of program development tools and utilities
- utilities for system management and maintenance
- lineprinter spooling
- DECnet support
- Internet capabilities
- Ethernet support
- Packetnet system interfaces
- DECnet/SNA Gateway interface

In addition to these features, RSX-11M-PLUS offers users:

- overlapped disk seeks
- disk-seek optimization routines
- rotational disk latency optimization
- multipathed disks
- Shadow Recording support
- complete batch-spooling and transparent-spooling utilities
- virtual terminal support
- increased Dynamic Storage Region (pool)
- User-mode Instruction and Data (I- and D-) space support
- the ability to map shared libraries in Supervisor mode
- accounting

RSX-11S features:

- a memory-resident executive
- a dedicated multitask environment
- event-driven priority scheduling
- 250 software priority levels
- multiple task partitions
- system-controlled partitions

- support for memory-resident overlaid tasks
- online taskloading
- System Image Preservation (SIP) program
- Task Termination Notification program
- a set of RSX-11S system activity display programs
- the Basic MCR user interface, a subset of the RSX-11M MCR Monitor Console Routine
- DECnet support

Realtime tasks do not have to compete with tasks of lower priority for scheduling services. RSX-11 systems schedule CPU time and memory residency on a preemptive priority basis. Realtime tasks can be placed in the upper ranges of the 250 scheduling priorities. Although the scheduler does not alter their priority, it can adjust the priorities of tasks assigned one of the lower 250 priorities to allocate CPU cycles equitably in a timesharing-like fashion. Task priorities can be altered by the system manager or by an appropriately privileged user.

Intertask communication is provided through common event flags, shared data files, use of the send and receive directives, and shared regions. This means users can share data, access resident libraries, and send messages between tasks.

Both DCL, the easy-to-use standard DIGITAL Command Language, and the MCR Monitor Console Routine are available on RSX-11M and RSX-11M-PLUS. The DCL implemented on RSX-11 is subset-compatible with VAX/VMS DCL. Users with special requirements that neither DCL nor MCR can meet can easily implement their own command interpreter.

A subset of MCR called Basic MCR is the user interface to an RSX-11S system. Strict subset compatibility with MCR is provided.

Rather than repeatedly typing a commonly used sequence of DCL or MCR commands on an RSX-11M/M-PLUS system, users need only type the sequence once into an indirect command file. Indirect command files can be used to execute strings of frequently used sequences of DCL or MCR commands or to create new commands from the existing command set. The indirect command filename is specified in place of the command line normally typed. Many tasks also accept strings of commands in a file.

Many system utilities and tools help various levels of system users to interact with the system, develop applications interactively, and manage and control system resources. Through these utilities, privileged users can monitor and control system use, tune system performance, and supervise day-to-day operations. Nonprivileged users can interact easily with the system to maintain files and enter, validate, and extract data. Fast, compact, interactive editors, online debuggers, trace facilities, and dump analyzers make program development fast and easy.

Languages

RSX-11 systems support a broad range of programming languages.

FORTRAN IV and FORTRAN-77 provide powerful tools for rapid, effective development of engineering and scientific applications. PDP-11 COBOL and COBOL-81 are full-featured ANSI-compliant COBOL implementations for administrative and accounting applications. PDP-11 BASIC-PLUS-2 is a sophisticated, extended BASIC compiler that provides a highly productive development environment for both technical and commercial applications. CORAL 66, the standardized general purpose language prescribed by the British government for realtime and process control applications, is also supported. Machine-level programmers can use the MACRO-11 assembler language.

Data Management Facilities

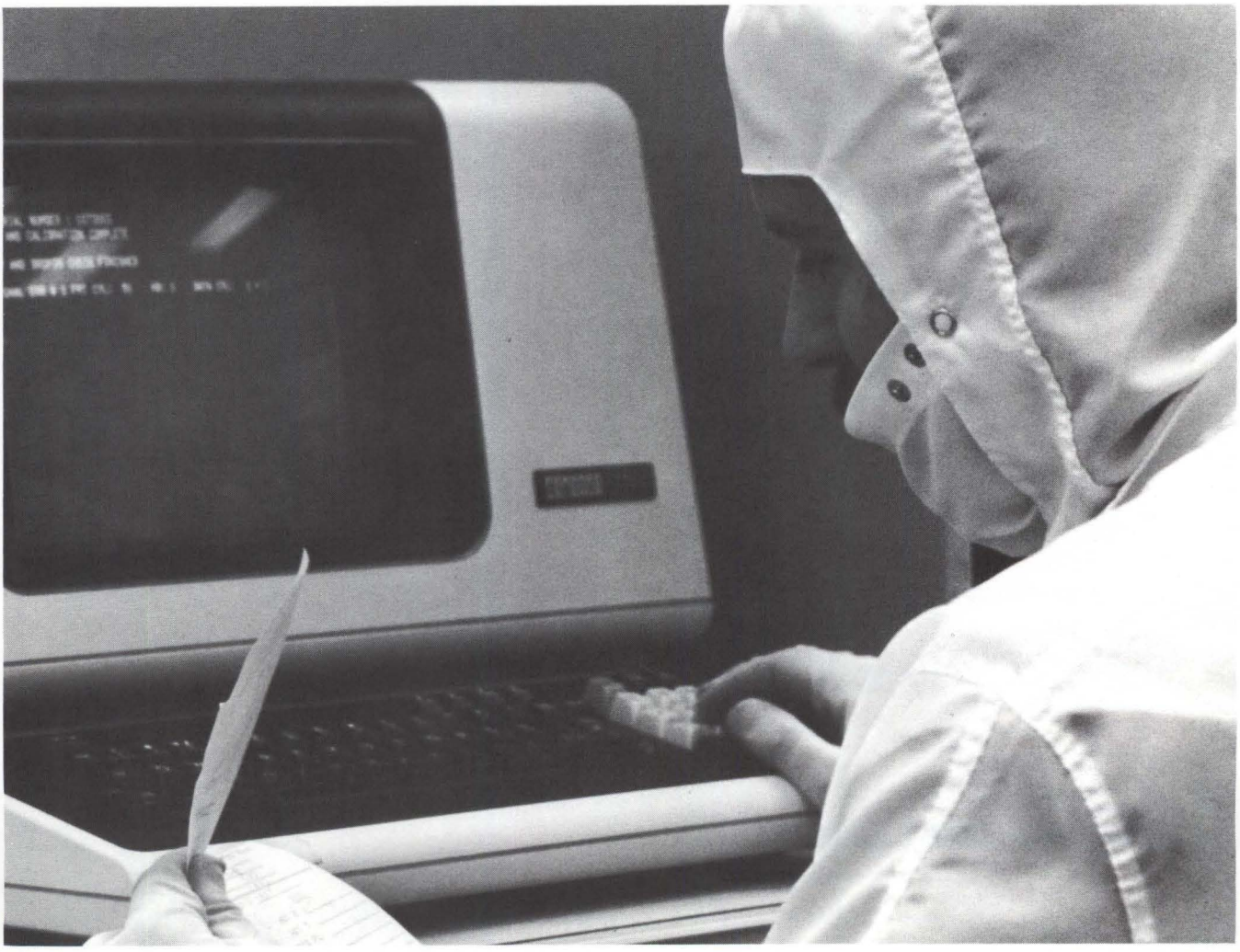
RSX-11 programmers have comprehensive record management facilities available to support databases created by high-level language programs. Programmers can choose the file organization and record access method appropriate to the application. The file organizations (sequential, relative, and multikey indexed sequential) and the record access methods are common to most RSX-11 programming languages.

With the optional DATATRIEVE inquiry and report system, users can find, display, print, update, and sort data. FMS-11 Forms Management System and SORT-11 are also available.

Data Communications

RSX-11 communications software provides extensive distributed data processing capabilities. Through DECnet distributed processing networks, RSX-11 systems can communicate with other DIGITAL systems. DIGITAL's protocol emulator software products, called Internets, open communications channels between RSX-11 systems and other manufacturers' systems. The DECnet/SNA Gateway allows users to implement applications involving cooperation between a DECnet network and an IBM SNA network. Packetnet switching support can significantly reduce the cost of moving data among computers at different locations. And Ethernet communications allows reliable high-speed, high-bandwidth communications among a variety of information processing equipment within a limited geographic area.

3
The Operating Systems



RSX-11 realtime operating systems provide a reliable, high-performance environment for rapid response to realtime demands as well as to such less-time-critical activities as program development.

RSX-11 systems are tailored to the exact requirements of individual applications. Through the system generation process, SYSGEN, users can include just the system features and capabilities they need for their particular application.

RSX-11 systems include such basic facilities as:

- taskbuilding
- priority-based scheduling
- multiprogramming
- dynamic memory allocation
- memory management
- checkpointing
- intertask communication
- power failure automatic restart

RSX-11 schedules processor time based on the priority assigned to a task. An optional roundrobin scheduler rotates jobs of equal priority in the queue.

RSX-11 uses its processor's memory management feature to map virtual memory addresses into direct physical addresses. If a task requires an area of memory to run, and if that area is occupied by another task of lower priority, the task of lower priority can be sent—checkpointed—to a disk. The memory is allocated dynamically.

RSX-11 systems provide intertask communication facilities for sharing data, synchronizing execution, and sending messages. The sharing of areas of memory and the use of shared resident libraries of software routines result in significant memory savings and increased performance.

INTRODUCTION

The RSX-11 family comprises three compatible realtime multiprogramming operating systems: RSX-11M, a compact, efficient operating system; RSX-11M-PLUS, a high-performance superset of RSX-11M that runs on all PDP-11s that have 22-bit addressing; and RSX-11S, a small execute-only operating system for dedicated application environments.

How the RSX Operating Systems Differ

RSX-11M and RSX-11M-PLUS systems include: an Executive; two command interfaces—DCL, the DIGITAL Command Language and MCR Monitor Console Routine (MCR); data management facilities, including File Control Services (FCS) and Record Management Services (RMS); and a complete set of utility programs. Under RSX-11M/M-PLUS, programs can be written in MACRO-11, FORTRAN IV, FORTRAN-77, PDP-11 COBOL, COBOL-81, PDP-11 BASIC-PLUS-2, and CORAL 66.

RSX-11M and RSX-11M-PLUS are disk-based multiuser systems. More than one user can interface with DCL or MCR services simultaneously.

The RSX-11M-PLUS operating system takes advantage of the expanded addressing capabilities of the MICRO/PDP-11, PDP-11/23-PLUS, PDP-11/24, PDP-11/44, and PDP-11/70 processors while retaining the superior reliability and the successful architecture of RSX-11M. RSX-11M-PLUS uses hardware features in these PDP-11 processors that are not available in the other members of the PDP-11 family. With the use of Supervisor-mode library routines and separate User-mode I- and D-space on PDP-11/44s and PDP-11/70s, an RSX-11M-PLUS task can address up to 196 Kbytes of memory. By using this additional memory, programmers can create tasks that use fewer memory and disk overlays. These programs can be developed, task-built, and executed faster than programs written using traditional memory-management techniques.

In addition, RSX-11M-PLUS supports multistream batch, accounting, dynamic dual-ported disks, additional memory management capabilities, more simultaneous tasks and terminals than RSX-11M, and other large-system features.

RSX-11M excels in performance on small- and medium-sized PDP-11s. It is designed to support factory automation, laboratory data acquisition and control, graphics, process monitoring, process control, communications, and other applications that demand immediate response. Its multiprogramming capabilities permit realtime activities to execute concurrently with such less-time-critical activities as program development, text editing, and data management. Up to 64 Kbytes can be addressed by one program.

RSX-11M and RSX-11M-PLUS offer unsurpassed software compatibility. All nonprivileged tasks that run on RSX-11M can run on RSX-11M-PLUS, without change or reassembly. Privileged tasks usually require little or no change.

RSX-11S, a subset of RSX-11M, is intended for use in an environment where a disk is not required or cannot exist, such as the floor of a manufacturing facility. RSX-11S provides a dedicated, execute-only environment for monitoring and controlling multiple, concurrent realtime processes. Response is very fast because all programs are memory-resident. RSX-11S runs on any of the PDP-11 processors—from the LSI-11-based PDP-11/23 to the system-level PDP-11/70. RSX-11S can support such storage devices as disks or tapes, although it doesn't use them as system devices.

RSX-11S requires a host RSX-11M, RSX-11M-PLUS, or VAX/VMS system for program development and system generation. This means that all RSX-11S system resources are devoted to realtime applications. Tasks can be written in MACRO-11 or another supported, high-level language and compiled and linked on the host system. They are then transported to the RSX-11S system for execution. The minimum RSX-11S system includes an Executive (with device drivers) and a special File Control Service (FCS) that does not support file-structured devices. The user can also add a subset of RSX-11M MCR services if the hardware configuration includes a terminal. If online task loading is desired, the user can include an Online Task Loader (OTL) utility. And to save a system image for subsequent rebooting, the user can include the System Image Preservation (SIP) utility.

Because RSX-11S is a memory-only system, it does not support a file system, nonresident tasks, task checkpointing, or program development. It does, however, support data storage on all devices supported by RSX-11M. Its purpose is to provide a runtime environment for the execution of tasks on a small system. RSX-11S tasks can address up to 64 Kbytes of memory.

RSX-11 System Configurations

RSX-11M-PLUS runs on the MICRO/PDP-11, PDP-11/23-PLUS, PDP-11/24, PDP-11/44, and PDP-11/70 processors. The minimum system supported has 256 Kbytes of memory, a console terminal, and a disk. It supports up to four Mbytes of memory.

RSX-11M runs on any of the PDP-11 UNIBUS minicomputer processors and the LSI-11-based PDP-11/23 and PDP-11/23-PLUS microprocessors. The minimum system requires a console terminal and a disk. Without a Memory Management Unit, the system can support between 32 and 56 Kbytes of memory. With memory management, memory can range from 48 Kbytes to four Mbytes, depending on processor capabilities. At least 56 Kbytes of memory is required for system generation and/or for concurrent application execution and program development.

The minimum configuration for an RSX-11S system is a PDP-11 processor (including the LSI-11) with at least 16 Kbytes of memory—on systems without a Memory Management Unit—and 56 Kbytes of memory—on systems with memory management—and one load device. At least 32 Kbytes are required for online taskloading or the execution of tasks written in FORTRAN-77. Up to 56 Kbytes of memory are supported on systems that do not include the hardware Memory Management Unit. Up to four Mbytes of memory is supported on systems with a Memory Management Unit, depending on processor capabilities.

SYSTEM GENERATION

The RSX-11 systems are designed to use system resources efficiently. To meet the needs of realtime applications and still obtain maximum performance for a given operating environment, RSX-11 provides users with a system generation procedure to control system features and capabilities.

The system generation utility, called SYSGEN, lets users define their specific hardware configuration and select software options.* Through an online, interactive question-and-answer session, users can develop a system tailored to their exact application requirements. The answers to many of SYSGEN's questions are saved in a file. Later, the saved answer file can be used to modify the system, if its features need to be changed, or to generate another RSX-11 system with the same capabilities.

* RSX-11M-PLUS does not support SYSGEN on PDP-11/23-PLUS systems or on systems with RL02 disk drives only. However, pregenerated RSX-11M-PLUS systems are available for these systems.

Once the SYSGEN process is complete, there are programs to facilitate the testing and design phase of the system. These consist of command files that exercise both the software and hardware and verify it as a working system with the configuration that was requested during SYSGEN. The system generation process is described in more detail in Section 5.

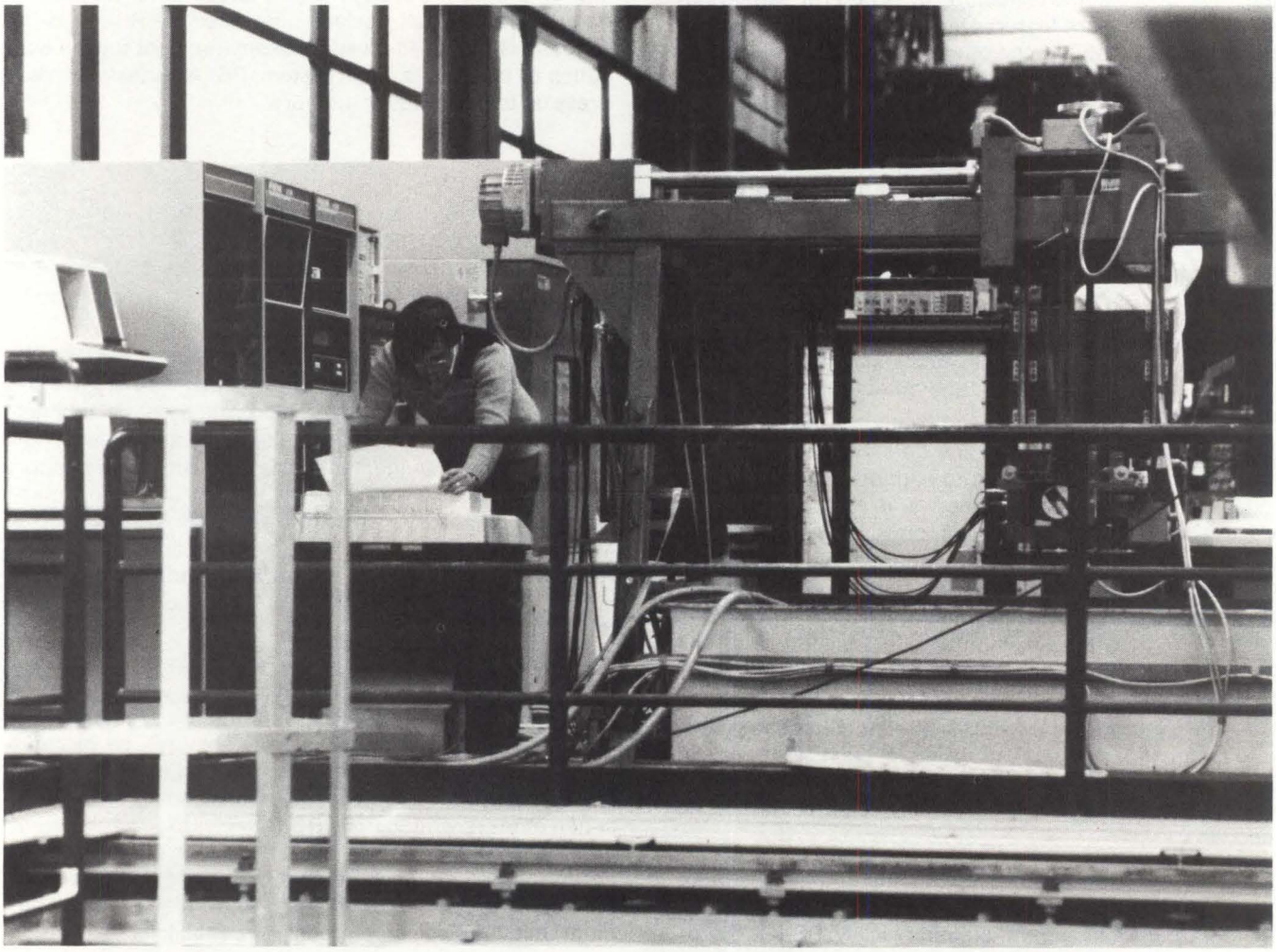
PROGRAMS AND TASKS

The basic unit of work that RSX-11 facilities service is called a task (process). A task consists of a program written in a source language, such as MACRO or FORTRAN, that has been assembled or compiled into an object format and then built into a task image by the link utility called the Task Builder.

The Task Builder

The Task Builder is a multiple-purpose tool. It allows users to create a loadable entity (called a task image), define and structure a shared area of memory (called a resident common), and arrange sharable routines (called resident libraries) to reside in memory.

In addition to the normal linkage functions of combining object modules or creating overlays, the Task Builder sets



up the basic task attributes that determine the task's resource requirements and relationships to other tasks in the system. The significant task attributes that affect a task's operation in a realtime multiprogramming environment are:

- *partition*—the section of memory where the task will reside when it executes
- *priority*—the task's relationship to other tasks in competing for system resources
- *checkpointability*—whether or not a task can be swapped out of memory when it is not executing in order to make room for a task of higher priority that is ready to run

Consistent with RSX-11's flexibility goals, all these attributes can be changed when the task is installed or when it executes.

To build a task image, the Task Builder accepts, as basic input, the output of a language processor in the form of an object module or multiple object modules. The user directs the Task Builder to generate a file of executable code (the task image), a file of memory allocation information (a map), and/or a special file of symbol definitions used in constructing the task (the symbol definition file).

The task image, residing on a disk, is in a format suitable to be loaded into memory and executed. In addition, the Task Builder can generate a cross-reference listing showing all the global references.

In creating a task image, the Task Builder's primary functions are linking, address binding, and building system data structures. Linking involves resolving global references and program-section references for all object modules. Address binding is the assigning of virtual address space within the task. Building system data structures involves the creation of elements that the system needs in order to load the task image into memory and to execute it. To resolve global symbols that are not defined in any of the input object modules, the Task Builder searches whatever object libraries are specified and, as a default condition, searches the system object library.

Task Installation

Once a task is built, it can be installed in the system and executed. Task installation simply registers a task's attributes with the system. When a task is installed (using a command-language or Virtual Monitor Console Routine command), the system records a number of task parameters in a system-resident list. Called the System Task Directory (STD), this listing is a directory of all installed tasks in the system by priority.

An installed task need not be resident in memory nor in competition for system resources. The system considers the installed task to be dormant until it is put in active competition for system resources by an operator, user, or another active task in the system.

Task installation is the basis for efficient task operation. An installed task that is dormant uses no memory; yet, when the task is needed to service a realtime event, for example, it can be introduced into the system quickly because its basic parameters are already in the STD.

When an installed task is activated, the system places the task in priority-based competition with other tasks for the resources it needs to execute. Processor time is allocated to it only when it is resident in memory.

Tasks can share code and data among themselves through the common partition facility. A common partition is made accessible to the system and to tasks by installing the common partition and the tasks that will use it.

Each task is either privileged or nonprivileged. A privileged task has special device and memory access rights that a nonprivileged task does not have. For example, privileged tasks can directly access the I/O page and can call parts of the Executive as subroutines. A nonprivileged task cannot.

SYSTEM PROTECTION

Under RSX-11, a number of terminals can operate concurrently. Each user terminal operates independently of others in the system so that each can run a different task and each can run more than one task. In a system that supports multiuser protection, a user must log onto the terminal before issuing commands.

To provide system security and authorize privileges in a multiuser environment, RSX-11 includes the protection scheme based on User Identification Codes (UICs). System management assigns each user a UIC that consists of two numbers: a group number and a member number. The UIC also determines whether the user is privileged or nonprivileged. Privileged users have access to every part of the operating system. Nonprivileged users can use most of the operating system, but they cannot change it.

When a file, program, or intertask communication facility is created, the system places the filename in a User File Directory (UFD) and stores the creator's UIC in the file header to indicate the owner.

The file's UIC determines which groups of users or programs have controlled access to the file. Every file also has a protection code that specifies what the users may do to the file when they access it. Either the file's owner or a privileged user sets and can change the file's protection code.

To access a file, a user must know the UFD in which it is listed. This knowledge, however, is not sufficient to guarantee access. The user must satisfy conditions specified in a protection code associated with the file to be accessed. In addition, the volume that contains the file must be mounted before the file is accessed.

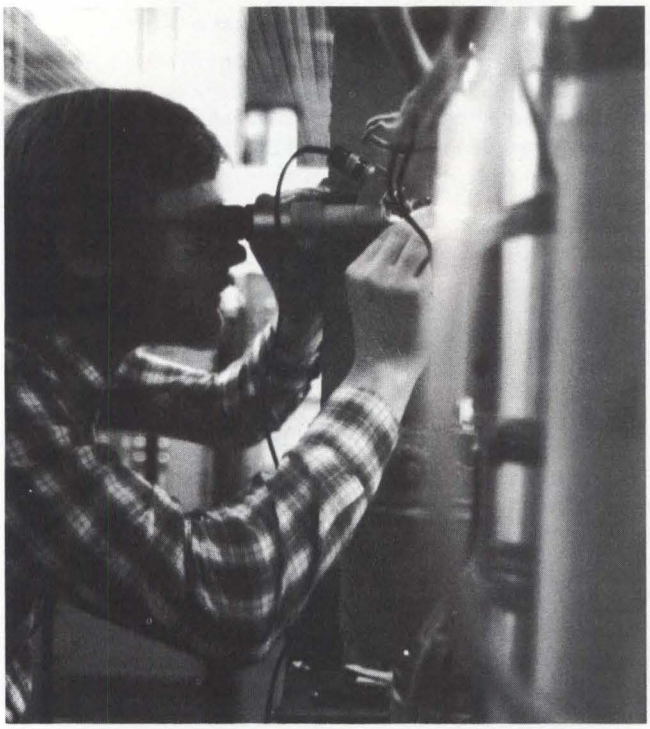
RSX-11M/M-PLUS allows four types of access:

- *Read.* The user or the user's task may read, copy, print, or type the file, and, if it is a task, run it.
- *Write.* The user or the user's task may add new data to the file by writing to space already allocated to the file.
- *Extend.* The user or the user's task may change the amount of disk space allocated to the file.
- *Delete.* The user or the user's task may delete the file.

The four user groups are defined by their UICs:

- *system*—every user or program whose UIC group number is a system-privileged group number (group number less than ten)
- *owner*—the user whose UIC is the same as the UIC assigned to the file

- *group*—every user whose UIC group number is the same as that assigned the file
- *world*—all other users



USER TASK ENVIRONMENT

The user program environment is the task—the entity the operating system schedules for execution. The following paragraphs describe how RSX-11 handles task execution.

Multiprogramming

Multiprogramming, the concurrent execution of two or more tasks residing in memory, is possible because task execution almost always involves more than just CPU (Central Processor Unit) usage. A realtime task that initiates a process and then waits for the process to complete might not need CPU time while it is waiting. Therefore, while one task waits for an event to complete, the Executive gives control of the CPU to another task. This happens so rapidly that many individual users seem to have control of the CPU at the same time. This is often referred to as apparent concurrency.

In the RSX-11 family, tasks are multiprogrammed by logically dividing available memory into a number of named partitions. These partitions are set up at system generation using the Virtual Monitor Console Routine. For maximum flexibility, privileged users can use DCL or MCR commands to establish and eliminate partitions whenever the need arises.

A partition is a contiguous area of memory with:

- a name
- a defined size
- a fixed base address
- a defined type

The relationship between a task and the partition in which it runs depends on whether the partition is system- or user-controlled, and whether the system is mapped or unmapped. In general, RSX-11M and RSX-11S systems can have two kinds of partitions: system-controlled and user-controlled. Because RSX-11M-PLUS systems are larger, mapped systems, they support system-controlled partitions only.

User-controlled partitions are used when the programmer wants to handle the allocation of memory during the execution of a task. System-controlled partitions are intended for the execution of tasks in which the user wishes the system to handle the allocation of memory.

Space in a system-controlled partition is dynamically allocated by the Executive to contain as many tasks as will fit simultaneously in the partition. This allocation can involve shuffling resident tasks to arrange available space into a contiguous block large enough to contain a requested task. Only mapped systems support system-controlled partitions. Mapped and unmapped systems are discussed below.

A user-controlled partition is allocated to only one task at a time. The user has complete control over system activity in this type of partition. In RSX-11M systems, a user-controlled partition can be subdivided into as many as seven nonoverlapping subpartitions. The subpartitions occupy the identical physical memory occupied by the main partition. Tasks built to execute in the subpartitions can execute in parallel. Tasks cannot, however, be resident in a main partition and its subpartitions simultaneously. If a main partition is occupied, the subpartitions cannot be. All subpartitions can have tasks residing in them; therefore, up to seven tasks can execute in parallel within a preempted, user-controlled main partition.

The goal of subpartitioning is to reclaim large memory areas in an unmapped system. When a large task that requires a main partition is no longer active, or when it can be preempted (checkpointed), subpartitioning allows the partition space to be used for a number of smaller tasks.

Dynamic Memory Allocation

Dynamic memory allocation is an extension of the RSX-11 multiprogrammed partition structure. Dynamic memory allocation enables the system to respond rapidly to changing requirements for system resources.

RSX-11 lets users load and execute more than one task in a system-controlled partition. If a loaded task does not fill the entire partition, another task can be loaded into the space above or below it, as long as the remaining contiguous physical space is large enough to contain it.

The Executive keeps a record of used areas and determines free space when needed. Tasks are brought in from disk according to their priority and are loaded into the first available memory area in the partition. The Executive continues to load tasks as long as sufficient contiguous physical memory is available in the partition. When a task terminates and is sent back to disk, the memory it occupied becomes available again.

Normally, a task cannot be loaded into a system-controlled partition unless sufficient contiguous space

between other tasks loaded in the partition is available. When a task terminates, it can leave a space that alone is insufficient to hold another task. If combined with other unused areas, however, this space could be large enough to contain the task.

In a system-controlled partition, the noncontiguous free spaces are combined by an automatic memory-compaction feature that shuffles or moves tasks to ensure sufficient space is available to load another task. Dynamic memory allocation is always present with RSX-11M-PLUS, although it can be turned off. It is optional on RSX-11M and RSX-11S.

Mapped and Unmapped Systems

The 16-bit architecture of the PDP-11 dictates that a task can directly address only 64 Kbytes of memory. A special hardware device, a Memory Management Unit, makes it possible to use more than 64 Kbytes of memory. The Memory Management Unit associates addresses used in tasks (virtual addresses) with actual locations in memory (physical addresses). The concepts of virtual and physical addresses are discussed later in this section.

A PDP-11 system that includes a Memory Management Unit is referred to as a mapped system. Systems without the unit are called unmapped. RSX-11M and RSX-11S systems can be mapped or unmapped. Mapped RSX-11M and RSX-11S systems can have both system-controlled and user-controlled partitions. Unmapped systems can have only user-controlled partitions.

Because all the PDP-11s supported by RSX-11M-PLUS include memory management hardware, RSX-11M-PLUS systems are always mapped systems and support only system-controlled partitions.

Mapped and unmapped systems install tasks into a partition differently. In unmapped systems, a task is linked to be installed and run in a partition with a specific base address. The task cannot run at a base address different from that specified in the Task Builder command.

In mapped systems, a task can be installed into any partition large enough to contain it. That's because the Memory Management Unit maps the virtual addresses of a task to the actual physical addresses in which the task resides.

The memory management hardware provides automatic memory protection on RSX-11 systems. Because each task has an absolute address range in which to execute, the memory area assigned to a task is protected from other tasks executing in the system. A task can reference and alter memory only within the specific task area it owns.

Scheduling

Because the RSX-11 family is designed specifically for realtime environments, event-driven scheduling is used to determine a task's eligibility to execute. The basis of event-driven task scheduling is the software priority assigned to each active task. A task's default priority is set when the task is built; it can either be changed dynamically from within a task or altered interactively via a command.

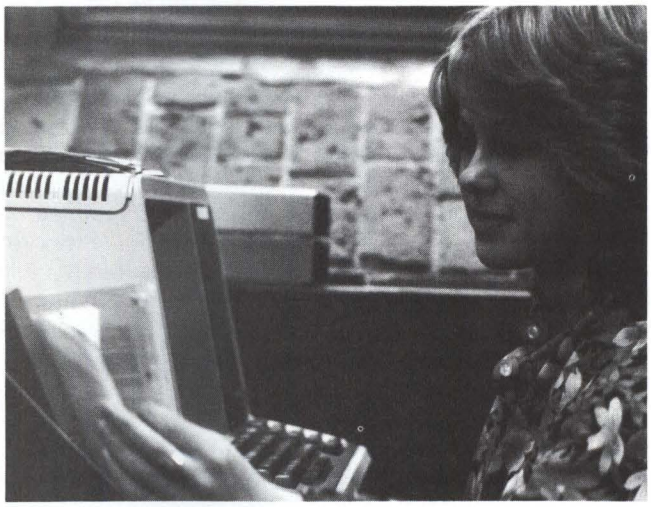
Tasks are run at a software priority level ranging from 1 (low) to 250 (high). The Executive grants central processor resources to the task of highest priority that can be executed. That task retains control of the central processor until the task gets blocked. Blocking occurs when a task issues a system directive that implicitly or explicitly suspends its execution. For example, a task can issue a directive that indicates it wants to wait until an I/O operation is complete before continuing execution.

Blocking is one type of significant event. A significant event is declared whenever there is a change in system status. Whenever there is a significant event, the Executive reviews the eligibility of tasks to execute.

Event Flags — A task can recognize that a significant event has occurred by means of event flags. In requesting a system operation (such as an I/O transfer), a task can associate an event flag with the completion of the operation. When the significant event occurs, the Executive sets the specified flag, then interrupts the executing task and searches for a task capable of executing. The task of highest priority that has all the resources it needs to run and that can make use of those resources will be the task that gains control of the CPU.

Event flags can be used to coordinate task execution for intertask communication. There are 96 event flags: 1 through 32 are local to the task, 33 through 64 are common to all tasks, and 65 through 96 are group-global. Although group-global event flags can be used in any application, they must be used by tasks containing the group code specified when the group-global event flags were created. A task can set, clear, test, and wait for any event flag or combination of event flags to achieve efficient synchronization between itself and other tasks in the system.

For example, upon completion of I/O requests, the Executive normally sets a requester-indicated event flag and declares a significant event. If a requesting task instructs the system that it cannot run until an event flag is set (signaling task I/O completion), other eligible tasks of lower priority may run.



Roundrobin Scheduling — Although event-driven scheduling is the primary RSX-11 task-scheduling mechanism, it is not the only mechanism available. As an option during system generation, RSX-11 systems allow the user to supplement event-driven task scheduling with round-robin scheduling for some or all tasks.

Roundrobin, or first-in-first-out (FIFO), scheduling is available for a priority range specified during system generation. If there is more than one task active at a given priority level, and if roundrobin scheduling was included in SYSGEN, the tasks are rotated in the priority queue. A form of timesharing, roundrobin scheduling gives tasks of similar priority an equal share of CPU time.

Task Checkpointing — With RSX-11M/M-PLUS, effective multiprogramming is achieved when many tasks, residing in memory simultaneously, spend some of their residency waiting for I/O completion, waiting for synchronization with other tasks, or being unable in some way to continue execution—while one other task utilizes the central processor's resources.

This multiprogramming scheme normally applies only to tasks resident in memory. Once a task is in memory, the Executive allows it to run until a significant event occurs, even if its memory space is required for the execution of a nonresident task of higher priority. However, if it is desirable to free memory for the execution of a task of higher priority, a task can be declared checkpointable when it is taskbuilt, installed, or running.

A checkpointable task can be swapped out of memory when a task of higher priority requests the partition in which it is active. With checkpointing, more tasks can be brought into memory to run, thus increasing system throughput.

In RSX-11M/M-PLUS systems, task priority normally determines which tasks can checkpoint other tasks. A checkpointable task currently active in a partition, but of a lower priority than another task requesting the partition, can be preempted and rolled out to a disk. The task is saved on the disk exactly as it was when interrupted. When memory is available, the task of lower priority can be restored to active execution at the point where it had been interrupted.

In addition, a memory-resident task that is waiting for terminal input is always checkpointable, regardless of its priority.

Stop-Bit Synchronization — RSX-11 also provides a stop-bit synchronization feature that allows tasks to be stopped until an event occurs. When a task is stopped, it is blocked from further execution, its priority for memory allocation effectively drops to zero, and it can be checkpointed by any other task in the system. If checkpointed, the task remains out of memory until its stop-bit is cleared. At this time the task becomes unstopped, and its normal priority for memory allocation is restored. This feature is particularly useful when a long waiting period is anticipated.

System Directives

When a task requests the Executive to perform an operation, it is called a system directive. Programmers use these directives to control both the execution and interaction of tasks. The MACRO-11 programmer usually issues directives in the form of macros defined in the system macro library. The FORTRAN programmer issues system directives in the form of calls to subroutines contained in the system object module library.

System directives affect the way the Executive shares system resources among concurrently active tasks, directly or indirectly. The following list groups the directives by function into nine categories.

- *Task Execution Control Directives* deal principally with starting and stopping tasks. Installed tasks can be either dormant or active. The active state has three substates: ready-to-run, blocked, and stopped. Each of the Execution Control Directives (except Extend Task) results in a change of the task's state (unless the task is already in the state being requested).
- *Task Status Control Directives* alter the checkpointable attribute of a task and/or change the running priority of an active task.
- *Informational Directives* provide the issuing task with system information and parameters, such as the time of day, the task parameters, the console-switch settings, and partition or region parameters.
- *Event-Associated Directives* provide a means in the system for intertask and intratask synchronization and signaling.
- *Trap-Associated Directives* provide the user with trap facilities that allow transfer of control (software interrupts) to the executing tasks.
- *I/O- and Intertask Communications-Related Directives* allow tasks to access I/O devices at the driver interface level or interrupt level, to communicate with other tasks in the system, and to retrieve the command line used to start the task.
- *Memory Management Directives* allow a task to manipulate its virtual and logical address space and to set up and dynamically control the window-to-region mapping assignments. These directives also provide the means by which tasks can share and pass references to data and routines. Memory management concepts are discussed later in this section.
- *Parent/Offspring Tasking Directives* permit tasks to start other tasks, connect to tasks in order to receive status information, stop for terminal I/O, and unstop other tasks. A more detailed description of parent/offspring tasking appears later in this section.
- *RSX-11M-PLUS Directives*. In addition to the directives listed above, RSX-11M-PLUS includes directives that support virtual terminals, Supervisor-mode library routines, variable-length send/receive data buffers, and parity error and exit AST routines.

System Traps

System traps, also called software interrupts, are another means of governing task execution. While significant events can have a systemwide scope, traps are local to a task. Traps interrupt the sequence of instruction execution in the task and cause control to be transferred to a prespecified point in the program. In this way, traps provide the ability to service certain conditions without continuously testing for their existence.

To use the system trap facility, a task must contain a trap service routine. This routine is automatically entered when the trap occurs, using the task's normal priority and privilege. If a service routine is not supplied, the action taken by the Executive is dependent upon the type of trap.

There are two types of traps: Synchronous System Traps (SSTs) and Asynchronous System Traps (ASTs).

SSTs provide a means of servicing fault conditions within a task, such as memory protection violation and floating-point-unit exceptions. These conditions, which are internal to a task, occur synchronously, with respect to task execution. In these cases, if an SST service routine is not included in the task, the task's execution is aborted.

ASTs commonly occur as the result of a significant event—thus asynchronously with a task's execution. A task does not have direct or complete control over the timing of an AST's occurrence. ASTs are for information purposes, such as for signifying an I/O completion that a task needs to know about immediately.

If an AST service routine is not provided, a trap does not occur, and task execution is not interrupted.

MEMORY MANAGEMENT

Without using advanced programming techniques along with the memory management hardware available on some PDP-11s, an RSX-11 task cannot explicitly address more than 64 Kbytes of memory. The 16-bit word size of the PDP-11 imposes this restriction on a task's addressing capability. Two programming techniques can overcome this addressing limitation:

- overlaying segments of a task with either disk-resident or memory-resident code
- mapping to different regions of memory outside the physical limits of the current task space

Overlaying

Users can reduce the memory and/or virtual address space requirements of a task by using overlay structures created with the Overlay Description Language. RSX-11 supports two kinds of overlaid segments: those that reside on disk and those that reside permanently in memory.

To create an overlay, a task must first be divided into segments: a single root segment—which is always in memory, and any number of overlay segments—which are loaded into memory as required.

The overlaid segments either reside on disk and share virtual address space and physical memory with one another (disk-resident overlays), or they reside in memory and share only virtual address space with one another (memory-resident overlays).

Disk-resident overlays save space, but they introduce overhead activity because the segments must be loaded into memory each time they are needed but not present in memory. Memory-resident overlays save time. They are loaded from disk only the first time they are referenced. Thereafter, they remain in memory and are referenced by remapping. This, of course, takes up memory space.

The Task Builder enables the user to build overlaid tasks and call these overlays from disk. The use of overlaid tasks and disk-based data can significantly expand the achievable peak load of an RSX-11 system and still maintain acceptable response time.

Several large classes of tasks can be handled effectively by an overlay structure. For example, a task that moves sequentially through a set of modules is well suited to an overlay structure. So, too, is a task that selects one of a set of modules according to the value of an item of input data.

Memory Management Directives

Data are disk-based, not only because of limited memory space, but also because transmission of numerous instructions or large amounts of data between tasks is only practical via disk. However, an overlaid task or a task that needs to access or transfer large amounts of data incurs a considerable amount of disk activity over and above that caused by the task's function.

Task execution could obviously be faster if all or a greater portion of a task were resident in memory at runtime. RSX-11M/M-PLUS includes a group of memory management directives that provide a task with this capability. These directives overcome the limited addressing restriction by allowing the task to dynamically change the physical locations that are referred to by a given range of virtual addresses. With these directives, a task can increase its execution speed by reducing its disk I/O requirements at the cost of requiring more physical memory.

The Memory Management Unit is the PDP-11 hardware that translates (relocates) a task's 16-bit virtual address into either an 18-bit or 22-bit (depending on processor) physical address. The hardware consists of an adder, a number of registers that perform the actual address translation, and an overall internal system protection scheme.

The basic function of memory management is to perform memory relocation and provide extended memory addressing capability for systems with greater than 64 Kbytes of physical memory. In order to perform this basic function, the memory management system utilizes a series of Active Page Registers (APRs). The APRs are a set of hardware relocation registers that permits several users' programs, each starting at virtual address 0, to reside simultaneously in physical memory.

Address Space

The concepts of virtual address space, logical address space, and physical address space provide a basis for understanding the functions performed by memory management directives.

- A task's *virtual address space* corresponds to the 64-Kbyte address range imposed by the PDP-11's 16-bit word length. The task can divide its virtual address space into segments called virtual address windows.
- A task's *logical address space* is the total amount of physical memory to which the task has access rights. The task can divide its logical address space into various areas called regions. Each region occupies a contiguous block of memory.
- A task's *physical address space* is the actual physical memory in which the task resides and executes.

RSX-11M/M-PLUS supplies memory management directives to assign or map a range of virtual addresses (a window) to different logical areas (regions); this enables the user to extend a task's logical address space beyond 64 Kbytes. Without these directives, a task's virtual address space and logical address space would directly correspond; thus, a single virtual address would always point to the same logical location.

Windows

To manipulate mapping virtual addresses to logical areas, the user must first divide the task's 64 Kbytes of virtual

address space into segments. These segments are called virtual address windows.

When a task uses memory management directives, the Executive views the relationship between the task's virtual and logical address space in terms of windows and regions. Unless a virtual address is part of an existing address window, references to that address will cause an illegal address trap to occur. Similarly, a window can be mapped only to an area that is either all or part of an existing region of the task's logical address space.

Once a task has defined the necessary windows and regions, it can issue memory management directives to perform such operations as:

- mapping a window to all or part of a region
- unmapping a window from one region in order to map it to another region
- unmapping a window from one part of a region in order to map it to another part of the same region

Regions

A region is a portion of physical memory to which a task has access. The current window-to-region mapping context determines the part of a task's logical address space that the task can access at one time. A task's logical address space can consist of various types of regions:

- *task region*—a contiguous block of memory in which the task runs
- *static common region*—an area defined at SYSGEN time or at runtime, such as a global common area
- *dynamic region*—a region created dynamically at runtime by issuing memory management directives
- *sharable regions*—on RSX-11M-PLUS only, a read-only portion of multiuser tasks

Address mapping not only extends a task's logical address space beyond 64 Kbytes, but also allows the space to extend to regions that have not been linked at taskbuilding time. One result is an increased potential for task interaction by means of shared regions. For example, a task can accommodate large amounts of data. Any number of tasks can then access that data by mapping to the region. Another result is the ability of tasks to use a greater number of common routines. Thus, tasks can map to required routines at runtime, rather than link to them at taskbuilding time.

A region becomes part of a task's logical address space by being attached. A task can map only a region that is part of the task's logical address space. There are three ways to attach a task to a region:

- all tasks are automatically attached to regions that are linked to them at taskbuilding time
- a task can issue a directive to attach itself to a named static common region or a named dynamic region
- a task can request the Executive to attach another task to any region within the logical address space of the requesting task

Attaching identifies a task as a user of a region and prevents the system from deleting a region until all user tasks have been detached from it. A task cannot indiscriminately attach to any region because each region has a protection mask to prevent unauthorized use.

Memory management directives are capable of performing a number of separate actions. The complexity of the directives requires that there be a special means of communication between the user task and the Executive. This is achieved through user data structures that allow the task to specify which directive options it wants the Executive to perform and permit the Executive to provide the task with details about the outcome of the action it is requesting.

RSX-11M-PLUS Memory Management Directives

RSX-11M-PLUS uses the extra set of memory mapping registers available on the PDP-11/44 and PDP-11/70. The Supervisor-mode registers can be mapped by RSX-11M-PLUS to map read-only libraries of user code, the RSX-11 file control services (FCS), or other application code. This capability gives an RSX-11M-PLUS task a full 64 Kbytes of normal address space plus an extra 64 Kbytes of read-only routines. Using this technique, with optimal library sizes, a user task can now directly utilize twice as much memory as it could have with RSX-11M. Frequently, increased task performance is gained by putting routines into an overlaid Supervisor-mode library rather than overlaying the task.

RSX-11M-PLUS also supports the separate I- and D-space hardware that is available on the PDP-11/44 and PDP-11/70. This means a user task can address up to 128 Kbytes of memory—64 Kbytes of instructions and 64 Kbytes of data. Thus, when used with Supervisor-mode libraries, I- and D-space tasks can address up to 196 Kbytes of memory—which can both greatly improve system performance and simplify program development.

Memory Maps

In a mapped system, the user does not need to know where a task resides in physical memory, because the map associates the task addresses with available physical memory. Memory management directives are used with the memory management hardware to perform address mapping at a level that is visible to and controlled by the user.

The memory allocation information (map) produced by the Task Builder shows users how program sections are arranged in task memory (their starting virtual addresses and extents on mapped systems and their physical addresses and extents on unmapped systems), what attributes are in a program section, any undefined symbols, and the optional cross-reference listing of global symbols. During debugging, users can use the starting addresses, combined with the relative location counter values (provided by the assembler), as offsets to access locations within the memory-resident task.

RSX-11M-PLUS PERFORMANCE ENHANCEMENTS

RSX-11M-PLUS contains many performance and throughput enhancements. Memory is better utilized by separating pure (read-only) and impure (read/write) sections of tasks. When the first copy of a multiuser task is executed, one copy of the pure and impure sections is loaded. Successive users receive only a copy of the impure code, and all users share the pure section, thereby using memory more efficiently.

On RSX-11M-PLUS, shared libraries, pure sections of multiuser tasks, and common data areas can be checkpointed to disk when not being used. These tasks are automatically brought back into memory when an executing task references them. Shared areas can also be shuffled to compact memory and to remove space between tasks.

RSX-11M-PLUS users can effectively triple the memory directly addressable by tasks on PDP-11/44 and PDP-11/70 systems through the use of Supervisor-mode library routines and separate User-mode Instructions and Data (I- and D-) space. Because this increased address space means programs can be written that require fewer (if any) overlays, faster program execution and decreased program development and taskbuild time result.

User-Mode I- and D-Space

A conventional RSX-11M-PLUS task operating in User mode (as most user tasks do) can contain 64 Kbytes of virtual address space and can access approximately 64 Kbytes of physical memory. However, a task using both User-mode I- and D-space APRs can contain 128 Kbytes of virtual address space and can access approximately 128 Kbytes of physical memory. RSX-11M-PLUS supports the separate I- and D-space hardware available on PDP-11/44s and PDP-11/70s.

A conventional task running in an I- and D-space system uses both sets of APRs. However, the relocation addresses in both I-space and D-space APRs are identical. Also, the task windows refer to I-space APRs in a task that does not use D-space.

An I- and D-space task can use separately both I- and D-space APRs; that is, APRs used in this way are not overmapped. Because of this, the task can use eight D-space APRs to access and use data and eight I-space APRs to access and execute instructions. Using 16 APRs allows the I- and D-space task to access a total of 128 Kbytes of physical memory at one time.

Supervisor-Mode Libraries

Supervisor-mode libraries are sharable resident libraries of routines that are used only in Supervisor mode. A task switches to Supervisor mode when it calls a routine within the Supervisor-mode library. By using this technique alone, a task's virtual address space can be doubled to 128 Kbytes.

Supervisor mode is one of three possible modes of operation (User and Kernel are the other two) in which PDP-11/44 and PDP-11/70 processors can operate. Each mode has associated with it 16 Active Page Registers (APRs): eight I-space and eight D-space APRs.

Normally, a task has an address space of 64 Kbytes, by using the eight User-mode APRs. When a conventional RSX-11M-PLUS task links to a Supervisor-mode library and calls a routine in the library, the Executive copies the User-mode I-space APRs into the Supervisor-mode D-space APRs and maps the Supervisor-mode library with I-space APRs. Therefore, while in Supervisor mode and within the library, the task can access 64 Kbytes of its own space with the D-space APRs and 64 Kbytes of library routines with I-space APRs.

In an I- and D-space task, mapping is done differently. When an I- and D-space task links to a Supervisor-mode library, the Executive copies the User-mode D-space APRs into the Supervisor D-space APRs. Therefore, the Supervisor-mode routines can access User-mode data space and access Supervisor-mode instruction space with Supervisor-mode I-space APRs.

On systems with I- and D-space hardware, RSX-11M-PLUS also uses separate hardware memory mapping for the Executive code and Executive pool. Pool is the memory available to applications for dynamic data structures. The separation of data and the use of extra memory management registers allow RSX-11M-PLUS to devote a full 40 Kbytes to system pool (minus the size of the system I/O data structures). This, in addition to the amount of system pool that has been freed by the creation of a secondary pool area, means that RSX-11M-PLUS often has double the amount of pool in contrast to comparable RSX-11M systems. So RSX-11M-PLUS can support about twice as many terminals and active tasks as RSX-11M systems.

PARENT/OFFSPRING TASKING

The RSX-11M and RSX-11M-PLUS operating systems also support parent/offspring tasking. A parent task is one that starts or connects to another task, called an offspring task. This type of tasking allows simpler, more straightforward multitasking synchronization schemes and has many realtime applications.

Another major use for parent/offspring tasking is batch processing; task relationships and parameters can be set up online to control the processing of one or more batch jobs offline.

Starting (activating) offspring tasks is called spawning. Spawning also includes the ability to establish task communications. A parent task can be notified when an offspring task exits and can receive status information from the offspring task. Status returned from an offspring task to a parent task indicates successful completion of the offspring task or identifies specific error conditions.

Parent/offspring tasking also supports chaining. An offspring task can pass its parent connection to another task—thus making the new task the offspring of the original parent.

A parent task can connect to more than one offspring task. An offspring task can, in turn, be connected to more than one parent task.

Under RSX-11M-PLUS, offspring tasks can perform I/O operations with virtual terminals, just as they can perform I/O with physical terminals. A virtual terminal is not a hardware device; it is implemented in software with data structures created by the Executive.

Virtual terminals are not interactive. A user does not issue requests or receive prompts; there is no immediate interaction between the user, the terminal, and the running task. Virtual terminals are used in batch processing and other offline processing environments to provide terminal I/O support for offspring tasks that normally would require user or operator intervention. For example, if a spawned task prompts for (requests) terminal input (for instance, if the spawned task is PIP), then the user needs a method of supplying the input, so it will look like it came from a terminal. A virtual terminal can be used to supply the required input. Thus, the spawned task code is the same whether the task is spawned or initiated from a terminal.

INTERTASK COMMUNICATION

RSX-11 provides intertask communication facilities for synchronizing execution, for sending directives, and for sharing common data.

Tasks are frequently required to pass information among themselves in order to perform their functions. For instance, in a typical process control situation there may be tasks performing such functions as measuring variables (for example, temperature), controlling the process, and preparing status reports. One task may be measuring the temperature while a second task uses the temperature measurement by comparing it with a desired temperature. And a third task may be compiling a process log for inspection by the operator. One task must pass information to other tasks, in order for them to carry out their functions as required.

Intertask communications can use:

- common and group-global event flags
- shared data files
- send/receive directives (variable length on RSX-11M-PLUS)
- shared regions
- virtual terminals on RSX-11M-PLUS

Common and Group-Global Event Flags

Common and group-global event flags are fast and simple. Each one conveys one bit of information suitable for communication in time-critical situations.

Cooperating tasks can communicate using common event flags and group-global event flags. Common event flags are common to all tasks; they can be set or cleared as a result of a task's operation. There are 32 common event flags, of which eight are reserved for use by the system. A task can read, set, clear, or test for common event flags to be set.

The group-global event flags can be used in any application where common event flags can be used; however, a group of event flags can only be used by tasks running under UICs containing the group code specified when the group-global event flags were created. A task can set, read, clear, or test for group-global event flags.

Shared Data Files

Very large amounts of data can be shared using shared data files. Synchronization at the sending and receiving tasks may be required. The access to the data is at disk file access speeds.

Send/Receive Directives

The send/receive directives can send and receive messages of moderate size at moderate speed. Synchronization is required at both the sending and receiving tasks.

Using the send/receive directives, a task can transfer a buffer load of data to another task. The data are transferred in a 13-word block via the Dynamic Storage Region (pool) in the Executive. When sent, the message is queued in the pool with other messages for the receiver task. If the receiver task is not running, its queue of messages is maintained until it runs and issues receive directives. In addition, RSX-11M-PLUS supports variable-length send/receive directives that can handle up to 255 bytes.

Shared Regions

A shared region is a block of data or code that resides in memory and can be used by any number of tasks. A shared region can contain data for use by several tasks. It can be an area where one task writes data for use by another task. Or a shared region can contain routines for use by several tasks.

Shared regions are a primary resource for conserving memory. In addition, they can increase productivity by allowing programmers and tasks to share data and code.

There are two kinds of shared regions.

- *Resident commons*—which provide a way for two or more tasks to share their data
- *Resident libraries*—which provide a way for two or more tasks to share a single copy of commonly used subroutines

The term "resident" denotes a shared region that is built and installed into the system separately from the task that links to it. That is, programmers use the Task Builder to build a shared region much as they would build a task. Switches are used to designate the kind of shared region (a library or a common) to be built.

RSX-11M-PLUS handles shared regions dynamically. It automatically loads them into partitions and checkpoints them when they are no longer referenced. In addition, RSX-11M-PLUS has two other types of commons: shared segments of multiuser tasks and Supervisor-mode libraries.

Multiuser Tasks — With the Task Builder, RSX-11M-PLUS users can build multiuser tasks. In a multiuser task, one portion of its code is protected as read-only and the other portion is protected as read/write.

Once the multiuser task is installed in memory, when any other task requests the multiuser task to run, the system duplicates only the read/write, or impure, portion of the task. The read-only, or pure, portion is sharable but hardware protected.

Supervisor-Mode Libraries — Supervisor-mode libraries, discussed above as a means of increasing a task's virtual address space, are resident libraries of routines that are used only in Supervisor mode. They are available only on RSX-11M-PLUS systems running on PDP-11/44s and PDP-11/70s.

I/O PROCESSING

The I/O processing system consists of several highly modular, interdependent components. The modular construction enables programmers to choose the programming interface and processing method appropriate for their needs without incurring runtime space or performance overhead for features not used. To maximize data throughput and minimize interrupt response time, the I/O-request-processing software also takes advantage of the hardware's ability to overlap I/O transfers with computation, to switch contexts rapidly, and to generate interrupts on multiple priority levels.

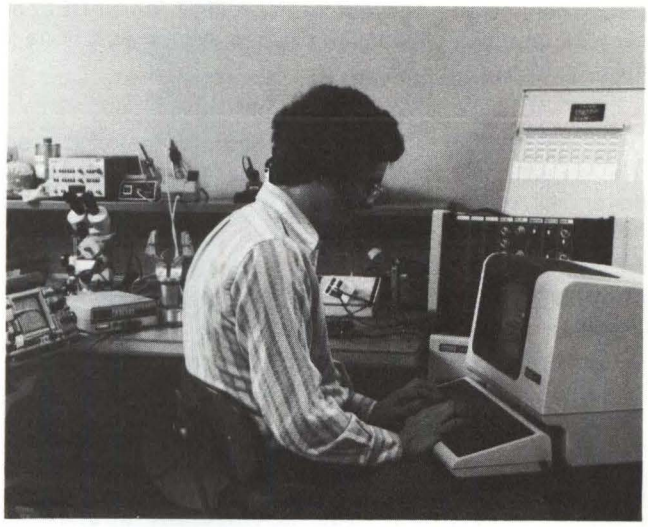
To achieve performance and space goals, the RSX-11 I/O system attempts to centralize common functions, eliminating the inclusion of repetitive code in every driver in the system. To do this, RSX-11 data structures are used to drive the centralized routines. This, in turn, reduces the size of the I/O drivers.

The RSX-11 system is structured to allow entry at any level. The top level contains the File Control Services (FCS) or the optional Record Management Services (RMS). Both provide device-independent access to devices in the system.

The lowest level of task I/O is the QIO directive. Any task can issue a QIO directive. The directive allows direct control over devices that are connected to a system and that have an I/O driver.

Custom devices can be connected to the RSX-11 system through special user-written device handlers or through Connect-to-Interrupt-Vector system directives.

With the Connect-to-Interrupt-Vector system directive, a task can process hardware interrupts through a specified vector. Through a subroutine included in the task's space, the task connects to the interrupt vector and can process interrupts directly from I/O devices.



Programming Interfaces

The I/O programming tools are the File Control Services (FCS) and the Record Management Services (RMS) for general purpose file and record processing and the Queue I/O system services for direct I/O processing.

FCS and RMS (discussed in Section 7) provide device-independent access to file-structured I/O devices. The most general purpose type of access enables programs to process logical records. FCS and RMS automatically provide record blocking and unblocking.

RMS users can also choose to perform their own record blocking on file-structured volumes, such as disk and magnetic tape, either to control buffer allocation or to optimize special record processing. To block their own records, users address blocks with a virtual block number (the number of the block relative to the file being processed) for volume-independent processing.

The I/O system services provide both device-independent and device-dependent programming. Users perform their own record blocking on file-structured and non-file-structured devices. Virtual block addressing is used on Files-11 disk or ANSI magnetic-tape volumes. In addition, users with sufficient privilege can perform I/O operations using either logical or physical block addressing for defining their own file structures and accessing methods on disk and magnetic-tape volumes.

Ancillary Control Processors

FCS, RMS, and the I/O system services use the same I/O control processors. Called ancillary control processors (ACPs), they are used for processing file-structured I/O requests. An ACP provides file structuring and volume access control for a particular type of device. Typical ACP functions include creating a directory entry or file, accessing or deaccessing a file, modifying file attributes, and deleting a directory entry or file header. There are three

kinds of ACPs provided in the system: Files-11 disk, ANSI magnetic tape, and network communications link.

The FCS, RMS, and I/O system services programming interfaces are the same regardless of the ACP involved; but, since ACPs are particular to a device type, they do not have to be present in the system if the device is not present. There is one network ACP process for all DECnet network communications links in the system and none if the system is not in a network.

Device Drivers

Once the ACP sets up the information for file-structured I/O requests, a request can be passed on to a device driver. All non-file-structured I/O requests are passed directly to a device driver.

A driver:

- defines the peripheral device for the rest of the RSX-11 operating systems
- defines the driver for the operating system procedure that maps and loads the driver and its device database into system virtual memory
- initializes the device (and/or its controller) at system startup time and after recovery from a power failure
- translates software requests for I/O operations into device-specific commands
- activates the device
- responds to hardware interrupts generated by the device
- reports device errors
- returns data and status from the device to software

Device drivers work in conjunction with the RSX-11 operating systems. The operating system performs all I/O processing that is unaffected by the particular specifications of the target device processing (that is, device-independent). When details of an I/O operation need to be translated into terms recognizable by a specific type of device, the operating system transfers control to a device driver that performs device-dependent processing.

The RSX-11 operating systems contain device drivers for a number of standard DIGITAL-supported devices. These include LSI-bus, UNIBUS, and MASSBUS devices. Also, the user can write additional drivers for nonstandard devices. The RSX-11M and RSX-11M-PLUS *Guide to Writing an I/O Driver* manuals provide detailed information on writing, loading, and debugging drivers.

I/O Request Processing

All I/O requests are generated by a Queue I/O (QIO) Request system service. If a program requests RMS procedures, RMS issues the Queue I/O Request system service on the program's behalf. Queue I/O Request processing is rapid because the system can:

- use each device as efficiently as possible by minimizing the code that must be executed to initiate requests and post request completion
- use each disk controller as productively as possible by overlapping seeks with I/O transfers (for RSX-11M-PLUS only)

The processor's many interrupt priority levels speed interrupt response because they enable the software to have the minimum amount of code executing at high-priority levels by using low-priority levels for code-handling request verification and completion notification. In addition, device drivers take advantage of the processor's ability to overlap execution with I/O by enabling processes to execute between the initiation of a request and its completion. User tasks can queue requests to a driver at any time, and the driver immediately initiates the next request in its queue upon receiving an I/O completion interrupt.

On RSX-11M-PLUS, the driver initiates the request and returns to the Executive when it is called. Disk seeks do not require the controller, once they are initiated. If a disk driver receives a seek request while the controller is busy with an I/O transfer request on another disk unit, the driver queues the request. When it has finished the current transfer the controller will initiate the seek request before any pending I/O transfers.

RSX-11M-PLUS employs optimization algorithms when choosing among I/O requests of equal priority. I/O Queue Optimization increases the throughput of the disk subsystem. For maximum flexibility, RSX-11M-PLUS provides three methods of I/O Queue Optimization. Queue Optimization is discussed in detail in Section 5.

In some cases, I/O Queue Optimization results in smoother disk operation by reducing erratic head movement.

Devices

The purpose of devices of any type is to handle I/O from tasks. RSX-11M/M-PLUS offers users a number of ways to identify the physical devices that handle I/O:

- mnemonics
- pseudo device names
- logical unit numbers (LUNs)
- logical device name assignments

Mnemonics — In I/O operations, the operating systems and its users deal with device drivers. Each device driver has a unique identifier consisting of a two-letter mnemonic. LP, for example, refers to the lineprinter driver, and DB is the driver for an RP04/05/06 disk.

Individual devices are identified by the two-letter mnemonic of their driver and an octal unit number, terminated by a colon. On systems with two lineprinters, they are distinguished as LP0: and LP1:, for example.

Pseudo Device Names — A pseudo device name is a device unit name that does not correspond to a real device until it has been redirected. These pseudo devices are forwarding addresses used by the operating system to locate real devices needed by tasks. The main function of pseudo devices is to establish relationships between devices that are tied in to the system by the system generation process. Since tasks can refer to pseudo devices instead of specific physical devices, pseudo devices allow tasks a great deal of freedom in performing I/O.

An example of how pseudo devices are useful is the commonly used pseudo device, LB:. As part of SYSGEN, LB: is defined as the device on which the system resides. In a disk-based system, most system task-image files remain on a disk until they are needed. System task images must

be installed from LB: for the system to work, but LB: need not be the same device each time. For instance, an installation with two systems, one with system-image files on DB0: (DB is the mnemonic for an RP04/05/06 disk pack), and one with system-image files on DB1:. can redirect LB: to DB0: or DB1:, depending on which system is running. Users can confidently refer in their tasks to LB: as the device from which system task images are installed, without wondering from day to day which system is running.

Logical Unit Numbers — Each task includes Logical Unit Numbers, or LUNs, which establish a relationship between the I/O done by the task and physical device units. This relationship can be different for each task.

LUNs can be assigned by the programmer at taskbuild time, at the time the task is installed, or by the task itself at runtime. Because the system provides default LUN assignments, it is not always necessary to assign a LUN to a task. Furthermore, LUNs can be changed by a DCL or MCR command for any installed, inactive, nonfixed task.

Logical Device Names — A LUN is simply a name used to represent the relationship between a physical device and a logical device name. Logical device names provide a means by which tasks can maintain device independence. Logical device names have the same syntax as other device names. The logical device name can be the same as a standard RSX-11 device or pseudo device, such as LP0: or LB1:, or it can use two letters with an arbitrary meaning, such as AZ:.

There are three types of logical device assignments.

- *Local assignments*, which can be made by any user, apply to tasks initiated from the terminal used to make the assignment. Local assignments override both other categories of assignments. Different users can assign the same local logical name to different devices.
- *Login assignments* are controlled by privileged users through ACNT, the Account File Maintenance Program, or through ASSIGN/LOGIN, a privileged command. Login assignments are reestablished each time the user logs in. They override the next category—global assignments—and are only used in systems with multiuser protection. Login assignments remain in effect until the user logs off or until a privileged user deassigns them.
- *Global assignments*, which can be made only by privileged users, apply to all tasks running in the system.

RSX-11S SYSTEM COMPONENTS

RSX-11S requires an RSX-11M, RSX-11M-PLUS, or VAX/VMS system for system generation and program development. An RSX-11S system is generated by a process very similar to the RSX-11M system generation process, which is described in detail in Section 5. The maximum hardware configuration is the same as that of an RSX-11M system.

Since it is based on RSX-11M, RSX-11S enjoys most of the inherent features and generation capability of that system. For example, RSX-11S automatically supports all of the peripheral devices that RSX-11M supports, as well as such hardware features as Floating-Point Units, parity memory, and memory management. All are selectable at system generation. When included in an RSX-11S system, these features, of course, take up additional memory.

However, because RSX-11S is memory-resident, it does not support a file system, nonresident tasks, checkpointing, disk-resident overlaid tasks (memory-resident overlaid tasks are supported), or program development. Its main purpose is to provide a runtime environment for executing tasks.

The basic software building blocks of an RSX-11S system are:

- a subset of the features available in the RSX-11M Executive
- a special File Control Services (FCS) that contains no support for directory devices

The minimum software system is one with only an Executive. The smallest Executive that can be generated requires five Kbytes of memory. However, the Executive must be augmented with other software for it to do useful work.

The following services, omitted from the minimum five-Kbyte Executive, can be generated into an RSX-11S system and take up additional memory:

- address checking
- Asynchronous System Traps (required for FORTRAN)
- I/O rundown
- external MCR functions (user-written functions)
- install, request, and remove-on-exit support
- SEND, RECEIVE, GET TASK PARAMETERS, GET SENSE SWITCHES, and GET PARTITION PARAMETERS directives
- parity memory support
- network support

In addition, the following programs can be included in an RSX-11S system:

- all RSX-11M I/O device drivers, except the console-logger device driver
- Basic MCR
- Online Task Loader
- Task Termination Notification program
- System activity display programs
- SETTIM subroutine used to set the system's internal time
- System Image Preservation program

Device Drivers

If the PDP-11 system includes I/O devices, the RSX-11S software system must include the Executive and the appropriate I/O device drivers. All RSX-11M I/O device drivers, except the console-logger device driver, are supported.

Most drivers use an average of three Kbytes; for example, the lineprinter driver needs less, the terminal driver more.

Basic MCR

If operator communication is required, Basic MCR can be included in a system. It takes up an additional six Kbytes of memory. Basic MCR provides commands to control and modify the execution of tasks installed in an RSX-11S sys-

tem. It is a subset of the RSX-11M MCR Monitor Console Routine. Basic MCR and MCR are strictly subset compatible.

Online Task Loader

The Online Task Loader (OTL) can be included in an RSX-11S system, if online loading of tasks is desired.

Tasks are created on a host RSX-11M, RSX-11M-PLUS, or VAX/VMS system, transferred to the load medium using the File Exchange Utility (FLX), and then loaded into a running RSX-11S system using OTL. OTL expects the load-device medium to be written in either DOS or RT-11 format, depending on the load device.

OTL reads a task image from the load-device medium, verifies the task-image format, and checks for error conditions. It then creates a Task Control Block (TCB) for the task, places the TCB in the System Task Directory (STD), and fixes the task in memory, making the task known to the system. For system-controlled partitions, OTL allocates a subpartition PCB (Partition Control Block) for the task.

The minimum size for OTL is seven Kbytes. In seven Kbytes, however, OTL supports only one load device. Online taskloading requires a 32-Kbyte system, since approximately 21 Kbytes will be required for system software (five Kbytes for the Executive, six Kbytes for Basic MCR, three Kbytes for one device driver, and seven Kbytes for the OTL).

Task Termination Notification Program

The Task Termination Notification program (TKTN) outputs abnormal task termination and device status information.

If a task aborts, TKTN prints on the task's initiating terminal the notification and appropriate error message. TKTN also displays the contents of the task's registers at the time the task aborted. If a task aborts with I/O requests outstanding, the error message will include the phrase "and with pending I/O requests."

All task-abort messages are printed or displayed at the initiating terminal. All other TKTN message are printed on the console terminal. TKTN requires an additional two Kbytes of memory.

System Activity Display Programs

System activity display programs are a set of privileged tasks that display information concerning task activity in an RSX-11S system. Users can invoke and continually run these tasks on DIGITAL's VT05-B, VT52, and VT100 video terminals. The display format is similar to that of the RSX-11M/M-PLUS RMDEMO display, which is described in Section 5. These programs require approximately 12 Kbytes of memory.

The programs display the following information:

- current date and time
- the currently active task
- all tasks, loaded drivers, and common blocks currently in memory, displayed to show their individual memory requirements and locations relative to other tasks
- the number of active tasks currently in memory and the total amount of memory occupied by each task
- the current amount of available system dynamic memory (pool), including the largest available block, the number of fragments, and the worst case of pool space since bringing up the display programs (not displayed on a VT05-B)
- a graphic display of partition information

SETTIM

SETTIM, a FORTRAN-callable subroutine, is used to set the system's internal time. It is supplied to allow a running program to set the time in a configuration that does not include a console terminal or Basic MCR. The module is included in a user task by linking with the file SETTIM.OBJ.

System Image Preservation

The System Image Preservation (SIP) program is an online utility task that can save the image of a running system on a load device medium in bootstrap format. The saved system can subsequently be restored by bootstrapping it from the load device medium. The minimum size for SIP is three Kbytes. In three Kbytes, it supports only one load device.

File Control Services

The standard RSX-11M File Control Services (FCS) record I/O package contains a large amount of code to support file-structured devices. Since RSX-11S contains no file support, this code is unnecessary. The special version of FCS provided with RSX-11S is the standard FCS without the file support code. This provides a significant size reduction.

The RSX-11S FCS subset does not support:

- file-structured devices (Files-11 disk and ANSI magnetic tape)
- block I/O operations
- random access record I/O
- sequenced records

In general, the subset FCS handles I/O in the same way as the RSX-11M FCS handles I/O for record I/O operations.

Virtual MCR

The Virtual Monitor Console Routine (VMR), an RSX-11M system program, allows the complete interactive configuration of an RSX-11S system on an RSX-11M, RSX-11M-PLUS, or VAX/VMS system. Users can define partitions, load and unload device drivers, install and fix tasks, and establish other system parameters. Users can then transfer the entire system to an appropriate load medium for bootstrapping on the target RSX-11S system.

Both RSX-11M-PLUS and VAX/VMS use the RSX-11M VMR for generating an RSX-11S system.

POWER FAILURE AUTOMATIC RESTART

Power failure automatic restart is the ability of a system to smooth out intermittent short-term power fluctuations with no apparent loss of service and without losing data, while maintaining logical consistency within the system itself and the application tasks.

When power begins to fail, the processor traps to the Executive, which saves all register contents. When power is restored, the Executive again receives control and restores the previously preserved state of the system.

The Executive then informs any tasks that have requested power failure restart notifications through the Asynchronous System Trap mechanism that a power failure has occurred. These tasks can then, if required, make the restorations of state they deem necessary. The Executive calls all device drivers that were active at the time the power failure occurred at their powerfail entry point. Drivers have the option of always being called on power recovery or of being called only when the driver has outstanding I/O. These drivers can then, if required, make those restorations of state (for example, repeating I/O requests) that they deem necessary. This approach is quite efficient because the repeating of I/O is placed nearest the source most likely to contain instructions on how to make the restoration.

4
Program Development Tools

RSX-11 users usually interact with applications via an online terminal. To aid in the development of interactive and realtime applications, RSX-11 users can:

- write, compile, taskbuild, and test programs interactively, taking advantage of the source code, object code, and macro and source libraries
- design applications that require a high degree of data sharing, intertask communication, and file manipulation

Users can directly control the operation of RSX-11 through the operating system's command languages. Text editing facilities provide for fast and easy data entry and modification.

INTRODUCTION

RSX-11M and RSX-11M-PLUS provide a complete program-development environment. In addition to the MACRO assembly language, they offer the optional high-level languages commonly needed in applications: FORTRAN IV, FORTRAN-77, PDP-11 COBOL, COBOL-81, PDP-11 BASIC-PLUS-2, and CORAL 66.

RSX-11M and RSX-11M-PLUS provide the software tools users need to develop programs quickly and efficiently. These tools include:

- a choice of comprehensive command languages
- a HELP facility
- indirect command files
- a choice of editors
- a choice of language processors
- debugging aids
- system libraries
- a wealth of utilities such as:
  - — a manipulation utility to copy, spool, and transfer files between volumes
  - — file-spooling utilities to handle the printing of files on a lineprinter
  - — programming utilities that allow programmers to work with library files, to examine file contents, and to print a listing of the contents of a file
  - — program maintenance utilities to modify, patch, and compare files

RSX-11 systems provide programmers with tools that support highly modular program and applications development. By taking advantage of these tools, programmers can build applications quickly and can easily modify and extend them later.

SYSTEM COMMAND LANGUAGES

RSX-11M and RSX-11M-PLUS systems can have one or more command line interpreters (CLIs) installed. All RSX-11M/M-PLUS systems include MCR, the Monitor Console Routine—the basic command line interpreter on RSX-11. Many systems also include DCL—the DIGITAL Command Language, and some systems support other CLIs as well. Both MCR and DCL are interactive, comprehensive, and flexible command languages. Both include commands to invoke most system tasks and utilities and to set and display certain system characteristics. In general, MCR commands name tasks—PIP, for example, a utility used to manipulate files (such as copying and typing them), while DCL commands specify actions directly, as in the COPY and TYPE commands.

MCR is the fundamental command line interpreter for the RSX-11M/M-PLUS operating systems. It provides the most direct interface with both the RSX-11M and RSX-11M-PLUS operating systems.

In general, MCR commands must be entered in exact syntax. MCR commands follow no set syntax rules, however. Most MCR commands are terse abbreviations or mnemonics.

DCL is an optional user-oriented CLI included in most systems with many users. Commands in DCL are English words that follow well-defined syntax rules. Full commands are self-documenting. DCL is designed for consistency and ease of use.

DCL is based on the command languages used on a number of DIGITAL's operating systems. In particular, RSX-11M/M-PLUS DCL is designed for compatibility with VAX/VMS DCL.

DCL on RSX-11M and RSX-11M-PLUS systems is a CLI task that translates DCL commands into MCR commands for execution by the system. The DCL SET DEBUG command displays on a terminal the MCR translation for any DCL command.

Depending on the way they use the system and on the nature of the system itself, users may find it more convenient to use DCL or MCR, or both. All nonprivileged system functions are available directly from DCL, but some privileged functions are not. All program-development facilities and all common utility functions are available from DCL.

To issue an MCR command, the user types a command string consisting of a command name and any required parameter. A parameter can be a task name, the name of a file, or a device specification. A DCL command line consists of the command name, which is a verb describing the action the system is to take, with optional qualifiers and parameters to further define the action of the command.

Some DCL commands require parameters or arguments as part of the command line. If the user fails to supply a required command element, DCL supplies a prompt with one or two words indicating the general nature of the required element. DCL will also supply help information when a question mark is typed in response to a prompt.

As a simple illustration of how MCR commands differ from DCL's, both of the following command lines will print a copy of a file named TECSUM.TXT on the user's terminal. The first example is from an MCR terminal; the second is DCL.

MCR>PIP TI:=TECSUM.TXT

DCL>TYPE TECSUM.TXT

The system displays MCR or DCL and the angle bracket on the terminal as an explicit prompt for a command. To explain the MCR command: *PIP,* the Peripheral Interchange Program, is a file manipulation utility. *TI:* is the input pseudo device that stands for the user's own terminal, and the equals sign is required as part of the syntax. The DCL command, on the other hand, is self-explanatory.

Tables 4-1 and 4-2 list the basic DCL and MCR commands. To save input time, most commands can be abbreviated; users need only type those characters needed to distinguish the command or qualifier from all others.

Because RSX-11M and RSX-11M-PLUS are designed to be tailored to the needs of each installation, not every feature of DCL and MCR listed here is available on every system. Some commands depend on layered products that may not be available at a particular installation. Many features, particularly those on RSX-11M systems, are system-generation options that may not have been selected at the time the system was generated.

Table 4-1
DCL Command Summary*

General Session Information and Control

ASSIGN	Associates a logical name with a physical device, pseudo device, or other logical device.
ASSIGN/ REDIRECT	Redirects output from one physical device to another. (P)
ASSIGN/ TASK	Reassigns an installed task's Logical Unit Numbers (LUNs) from one physical device to another. (P)
BROADCAST	Displays a specified message at one or more terminals.
DEASSIGN	Disassociates logical names from physical device names, pseudo device names, or logical device names assigned by ASSIGN.
HELLO	Grants access to a multiuser protection system and establishes a user's privileges on the system. Synonym of LOGIN.
HELP	Displays information about the system, MCR, DCL, and most utilities.
LOGIN	Grants access to a multiuser protection system and establishes a user's privileges on the system. Synonym of HELLO.
LOGOUT	Logs the user off a multiuser protection system. It also aborts any active nonprivileged tasks running from the terminal, as well as dismounts any private devices allocated from the terminal.
MCR	Enters an MCR command from a DCL terminal without leaving DCL.
REQUEST	Sends a message to the operator's console.
SET	Defines or alters systemwide default characteristics. These include:
SET [DAY]TIME	Sets the system data and time. (P).
SET DEFAULT	Establishes user's default device or UFD, or both.
SET DEVICE	Establishes certain device characteristics. (P)
SET LIBRARY/ DIRECTORY	Establishes RSX-11M-PLUS directory where the system utilities and other nonprivileged system tasks are kept. RSX-11M-PLUS only. (P)
SET [NO] PARTITION	Creates or eliminates a partition. (P)
SET PRIORITY	Alters the priority of an active task. (P)
SET PROTECTION	Establishes the protection status of files.
SET SYSTEM	Establishes certain characteristics of the system. (P)
SET TERMINAL	Sets various attributes of a terminal.
SHOW	Displays systemwide or terminal characteristics and other system information.

* Certain aspects of the commands followed by the letter P in parentheses are available only to privileged users.

Volume and Device Resource Control

ALLOCATE	Obtains exclusive ownership of device and enables the user to assign a logical name to the device.
BACKUP	Backs up and restores Files-11 volumes.
DEALLOCATE	Frees a private device for use by others. Counteracts ALLOCATE.
DISMOUNT	Marks the volume mounted on the specified device to be logically offline and disconnected from the system.
INITIALIZE	Produces a volume in Files-11 format.
INITIALIZE/ UPDATE	Invokes the HOME utility to alter values in the Volume Home Block without affecting the other data in the volume.
MOUNT	Declares a volume to be logically known to the system, online, and available for use. Users can mount disk volumes, DECtapes, or ANSI magtapes.

File Manipulation

APPEND	Appends records from one or more sequential files, or all records from within an indexed or sequential file, to an existing sequential file.
CONVERT	Invokes the RMSCNV utility, which moves records from one file to another.
COPY	Copies the contents of a file or files. Unless specified otherwise, COPY preserves the file organization of the input file.
CREATE	Creates a new file from data subsequently entered in the input stream (user at terminal or batch stream).
CREATE/ DIRECTORY	Creates a User File Directory on a FILES-11 volume and enters its name into the volume's Master File Directory (MFD).
DELETE	Deletes one or more files from a mass storage disk volume and releases the storage space the files occupy.
DIFFERENCES	Compares two ASCII (text) files line by line and produces a listing of the differences between the two files, in any.
DIRECTORY	Displays information (size, protection, ownership, etc.) on a given file or set of files.
EDIT	Invokes an editor. EDI is the default editor.
EDIT/EDT	Invokes EDT, the DIGITAL standard editor.
EDIT/SLP	Invokes the Source Language Input Program (SLP), a program maintenance editor.
PRINT	Queues files for printing on a lineprinter or other spooled output device.
PURGE	Deletes all but the latest version of a given file or files and releases the storage space the deleted files occupy.
RENAME	Changes the name, type, or version of one or more existing files.
SORT	Invokes the SORT-11 utility. Creates a file by rearranging the records in a given file based on the contents of key fields within the records.
TYPE	Prints selected files on the user's terminal.

UNLOCK	Permits access to a file that was improperly closed because a task aborted or stopped execution while the file was open.

Program Development and Execution Control

ABORT	Forces an orderly end to a running task or to the action of a specific command.
BASIC/BP2	Compiles given BASIC-PLUS-2 language source modules, producing an object module.
CANCEL	Eliminates entries from the clock queue.
COBOL	Compiles given COBOL language source modules, producing an object module.
CONTINUE	Resumes execution of a previously suspended task.
DEBUG	Forces an RSX-11M-PLUS task to trap to a debugger. RSX-11M-PLUS only.
FIX	Causes an installed task or region to be loaded and locked into memory. (P)
FORTRAN	Invokes the FORTRAN compiler to compile one or more source program. FORTRAN IV is default.
FORTRAN/F77	Invokes the FORTRAN-77 compiler to compile one or more source program.
INSTALL	Includes a task in the System Task Directory, thus making it known to the system. (P)
LIBRARY	Creates and maintains user-written library files. Its functions include:
LIBRARY/ COMPRESS	Physically deletes from a library modules that have been logically deleted with LIBRARY/DELETE.
LIBRARY/ CREATE	Creates a library and optionally inserts modules into it.
LIBRARY/ DELETE	Deletes object modules from the library.
LIBRARY/ EXTRACT	Reads modules from a library and writes them to a specified output file.
LIBRARY/ INSERT	Inserts modules from one or more files into a library.
LIBRARY/ LIST	Lists on the terminal or output file names of modules in a library.
LIBRARY/ REMOVE	Removes global symbols (entry points) from a library.
LIBRARY/ REPLACE	Replaces a module in a library with new module of same name and deletes old module.
LINK	Invokes the Task Builder, which links object modules and routines from user and system libraries to form an executable task.
MACRO	Assembles one or more MACRO-11 source files into a single relocatable object module suitable for processing by the Task Builder.
REMOVE	Counteracts INSTALL. Takes the task name out of the System Task Directory. (P)
RUN	Initiates the execution of a task by installing, running, and removing upon execution a task from a task image file stored in a user's UFD, a system UFD, or a library UFD; and running immediately or on a schedule a task previously installed by a privileged user.
SET GROUPFLAGS	Creates and deletes group global event flags.
START	Resumes execution of a task stopped by a STOP$S directive.
START/UNBLOCK	Continues execution of a task blocked by the STOP/BLOCK command.
STOP/BLOCK	Blocks an installed running task. The task no longer executes or competes for memory.
UNFIX	Frees a fixed task or region from memory. (P)

Batch and Queue Operations

ASSIGN/ QUEUE	Assigns queues to print or batch processors. (P)
DEASSIGN/ QUEUE	Counteracts ASSIGN/QUEUE. (P)
DELETE/ PROCESSOR	Deletes print processors, output despoolers, or batch processors from the QMG subsystem by processor name or device name. (P)
DELETE/ QUEUE	Deletes a queue in the Queue Manager (QMG) subsystem by name. (P)
HOLD/ENTRY	Holds a QMG job in its queue by entry number.
HOLD/JOB	Holds a QMG job in its queue by queuename and jobname.
INITIALIZE/ PROCESSOR	Creates, names, and starts an output despooler, input spooler, cardreader processor, or batch processor. (P).
INITIALIZE/ QUEUE	Creates, names, and starts a queue in the QMG subsystem. (P)
RELEASE/ENTRY	Releases by entry number a print or batch job that has been held in its queue.
RELEASE/JOB	Releases by queuename and jobname a print or batch job that has been held in its queue.
SET QUEUE/ENTRY	Modifies by entry number some attributes of print or batch jobs once they are in a queue.
SET QUEUE/JOB	Modifies by job name some attributes of print or batch jobs once they are in a queue.
SHOW PROCESSOR	Displays information about the batch processors, printers, and about other output devices under control of the Queue Manager.
SHOW QUEUE	Displays information about print jobs and batch jobs.
START/ PROCESSOR	Starts a batch processor, card reader processor, printer or other output processor. (P)
START/ QUEUE	Starts a QUEUE (P).
START/ QUEUE/ MANAGER	Starts the Queue Manager. (P)

STOP/ABORT	Stops the current job on a lineprinter immediately.
STOP/ PROCESSOR	Stops a batch processor, card reader processor, printer or other output processor. (P)
STOP/ QUEUE	Stops the queue; no jobs are taken off. (P)
STOP/ QUEUE/ MANAGER	Stops at the end of the current active job. (P)

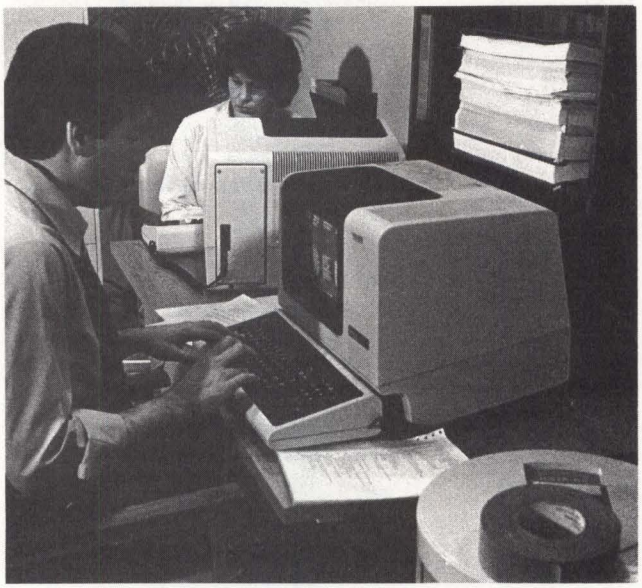


Table 4-2
MCR Commands*

System Initialization and Maintenance Commands

ACS	Allocate Checkpoint Space. Allocates or discontinues the use of a checkpoint file on disk. (P)
BOOT	Bootstraps a new system into memory and transfers control to it. (P)
BRK	Breakpoint to Executive Debugging Tool. Passes control to the Executive Debugging Tool (XDT). Valid only for systems with generated XDT support. (P)
BROADCAST	Broadcasts a message to one terminal or a set of terminals.
CLI	Work with a command line interpreter other than MCR.
DCL	Issues a DCL command from a terminal set to MCR.
OPEN	Open register. Displays on the terminal the contents of a memory location for examination or modification. (P)
REDIRECT	Redirects all I/O requests from one physical device to another. (P)
SAVE	Saves the image of memory in the file from which the system was booted. (P)
SSM	System Service Message. Inserts text into error log reports. (P)
SWR	Switch Register. Displays the current value of, or sets or clears a bit, in the switch register. RSX-11M-PLUS only. (P)

General Session Information and Control

ACTIVE	Displays on the terminal the names of the active tasks requested.
ALLOCATE	Allocates a device to a user (establishes the device as a user's private device).
ASN	Assign. Defines or deletes a logical device assignment. Lists current assignments on user's terminal.
ATL	Active Task List. Displays on the terminal names and status information for active tasks.
BYE	Logs the user off the system. It also aborts any active nonprivileged tasks running from the terminal, as well as dismounts any private devices allocated from the terminal.
CBD	Common Block Directory. Displays on the terminal names and status information for entries in the Common Block Directory (on RSX-11M-PLUS only).
CLQUEUE	Clockqueue. Displays information about tasks in the clock queue.
DEALLOCATE	Deallocates a private device. Complement of ALLOCATE.
DEVICES	Displays on the entering terminal the list of peripheral devices recognized by the system.
HELLO	Grants access to a multiuser protection system and establishes a user's privileges on the system. Synonym of LOGIN.
HELP	Displays the contents of a Help file.
LOGIN	Grants access to a multiuser protection system and establishes a user's privileges on the system. Synonym of HELLO.
LUNS	Logical Unit Numbers. Displays on the entering terminal the list of static LUN assignments for a specific task.
PARTITIONS	Displays on the terminal a list of the partition definitions.
SET	Alters or displays system, device, or task characteristics.
TAL	Task List—ATL format. Displays on the terminal the names and status of all tasks in the system.
TASKLIST	Displays on the entering terminal a description of each installed task.
TIME	Enters the time and/or date into the system; display the time and date on the entering terminal.

Volume and Device Resource Control

BRU	Invokes utility that backs up and restores Files-11 volumes.
DMOUNT	Dismounts a volume by marking it for dismount and releasing its control blocks. Complement of MOUNT.

* Certain aspects of the commands followed by the letter P in parentheses are available only to privileged users.

INITVOLUME	Initializes a disk, DECtape, or magnetic tape to produce a Files-11 volume.
HOME	Invokes the HOME utility to alter values in the Volume Home Block without affecting the other data in the volume.
LOAD	Reads a nonresident device driver into memory (P).
UNLOAD	Removes a loadable device driver from memory. (P)
MOUNT	Mounts a volume online for access by the disk or magtape file system.
UFD	Creates a User File Directory (UFD) in a specific volume's Master File Directory (MFD).

File Manipulation

CNV	Convert. Invokes the RMSCNV utility, which moves records from one file to another.
CMP	Compares two text files line by line and produces a listing of the differences between the two files.
SRT	Invokes the SORT-11 utility. Creates a file by rearranging the records in a given file based on the contents of key fields within the records.
PIP	Invokes the Peripheral Interchange Program, a file transfer program that provides ans for copying, concatenating, spooling, renaming, listing, deleting, and unlocking files.
PRINT	Queues files for printing on a lineprinter or other spooled output device.

Program Development and Execution Control

ABORT	Terminates execution of a running task.
ALTER	Alters the priority of a task. (P)
BASIC	Compiles given BASIC-PLUS-2 language source modules, producing an object module.
BLOCK	Blocks execution of a task.
UNBLK	Unblocks a previously blocked task. Continues execution of a task previously blocked with the BLOCK command.
CANCEL	Cancels time-based initiation requests for a task (no effect on current execution).
CBL	Compiles given COBOL language source modules, producing an object module.
DEBUG	Forces an RSX-11M-PLUS task to trap to a debugger. RSX-11M-PLUS only.
FIX	Fixes a task in memory (task becomes memory-resident). (P)
FLAGS	Creates, displays, or eliminates group-global event flags.
FORTRAN	Invokes the FORTRAN compiler to compile one or more source program.
INSTALL	Installs a task in the system. (P)
LBR	Creates and maintains user-written libraries.
MACRO	Assembles one or more MACRO-11 source files into a single relocatable object module suitable for processing by the Task Builder.
REASSIGN	Changes LUN assignments. (P)
REMOVE	Removes a task from the system (complement of INSTALL). (P)
RESUME	Resumes execution of a suspended task.
RUN	Initiates the execution of a task. The task can run immediately, after a time delay, or in synchronization with the system clock. Periodic rescheduling is optional.
TKB	Explicitly invokes the Task Builder, which links object modules and routines from user and system libraries to form an executable task.
UNFIX	Makes a memory-resident task nonresident. Complement of FIX. (P)
UNSTOP	Resumes execution of a stopped task.

Batch and Queue Operations

QUEUE/ DELETE	Deletes a queue in the Queue Manager (QMG) subsystem by name. (P)
QUEUE/HOLD	Holds a QMG job in its queue by entry number.
QUEUE/LIST	Displays information about print jobs and batch jobs.
QUEUE/ LIST:DEVICE	Displays information about the batch processors, printers, and about other output devices under control of the Queue Manager.
QUEUE/ MODIFY	Modifies some attributes of print or batch jobs once they are in a queue.
QUEUE/ RELEASE	Releases by entry number a print or batch job that has been held in its queue.

User-Written CLIs

An RSX-11 installation may have special command-language requirements that neither MCR nor DCL can meet. So RSX-11 supports the capability of easily implementing a custom command language interpreter that is specific to an application. A CLI is an RSX-11 task, and, like any other, it can be written in any RSX-11-supported programming language. Users can write CLIs without knowledge of operating system internals. Nor is privileged "system" code required. RSX-11 systems can support multiple CLIs, with the added convenience of each terminal being preset for the desired CLI.

Indirect Command Files

To eliminate the need for typing frequently repeated sequences of commands, users can create an indirect command file—a text file that contains complete command lines or a series of commands. When the user enters the name of the indirect command file, the system processes the command lines in the file just as if they were being typed successively at the terminal.

Indirect command files also allow system queries, string substitutions, multilevel indirect command files of up to

four levels, special symbol definitions, and an extensive number of directives. Among other things, the directives allow users to:

- define labels
- define and assign values to three types of symbols—logical, numeric, and string
- create and access data files
- control the logical flow within a command file
- perform logical tests of internal and system states
- invoke subroutines
- determine if an invoked task exited successfully
- do arithmetic
- control time-based and parallel task execution

There are two types of indirect command files: indirect task command files and indirect DCL/MCR command files. An indirect task command file is a sequential file containing a list of task-specific commands. Rather than typing commonly used sequences of commands, the sequence can be typed once and stored in a file. The indirect task command file is specified in place of the command line normally submitted to the task.

An indirect DCL/MCR command file contains a list of DCL or MCR commands. RSX-11 has an indirect file processor for interpreting commands in a file. An indirect DCL/MCR command file can contain both normal DCL/MCR commands and special commands (known as directives) that allow the user to program the execution of the indirect command file.

Batch Jobs

In addition to executing indirect command files at a terminal, an RSX-11M-PLUS user can submit batch jobs. Batch jobs are similar to indirect command files, except that the user does not have to be present or even logged onto the system when the batch job is run.

Batch jobs execute under the control of the Queue Manager program. Because a batch job doesn't require the user to be present, jobs that take a long time to run or otherwise tie up system facilities can be run when there are fewer demands on the system—at night or on weekends, for example. Users can submit a batch job and continue using their terminals for other work.

To create a batch job, a user enters batch-specific commands, CLI commands, and data into a file. The information contained in the batch job must duplicate a complete interactive terminal session, including logging itself into the system, controlling operations, and logging itself off. Using the SUBMIT command, the batch job is then submitted to the Queue Manager for batch processing.

Tasks called batch processors pass the commands in batch jobs to the operating system. Commands to be passed by the batch processor are the same as normal system commands, except that they are preceded by a dollar sign ($). (The only exceptions are for LOGIN/HELLO and LOGOUT/BYE—these are replaced by $JOB and $EOJ, respectively.) Table 4-3 lists the batch-specific commands.

Table 4-3
Batch-Specific Commands (RSX-11M-PLUS only)*

SUBMIT	Queues QMG batch jobs consisting of one or more user batch jobs for processing by a batch processor.
$CONTINUE	A no-operation
$DATA	Marks the beginning of a data block included in the batch job.
$EOD	Signifies the end of data in the input block following a $DATA command.
$EOJ	Marks the end of a batch job. EOJ performs the same functions as the LOGOUT/BYE command.
$GOTO	Transfers flow of control to a given labeled line.
$IF	Transfers flow of control to a given labeled line if the result of a logical comparison of symbolic values is true.
$JOB	Indicates the beginning of a batch command file and provides job control information (such as time limit). Performs the same function as LOGIN/HELLO.
$ON	Transfers flow of control to a given labeled line if an error of a given severity or greater is encountered at any time during command procedure processing.
$SET	With special batch parameters, used to override $ON or to reinstate in.
$STOP	Stops the batch job.

* Command names preceded by a $ are meaningful only in a batch command file or command procedure. All other commands listed in the DCL and MCR command tables above can be issued as part of batch jobs both interactively and in a batch command file.

The batch processor uses a software terminal called a "virtual terminal" to pass commands and data to system tasks. A batch job can do almost anything from a virtual terminal that a user can do from an interactive terminal, including compiling or assembling, taskbuilding, and running tasks.

Users can submit batch jobs at any time the Queue Manager is active. Batch jobs can run at any time as well. System management controls the availability of batch processing.

RSX-11M-PLUS supports multistream batch processing. Operations personnel can control the number of batch streams that can run. An interactive user, a program, or another batch job can submit batch jobs. When the number of batch jobs submitted exceeds the number of streams, the remainder of the batch jobs are held in a batch input queue. The operator can control the batch job queue by changing job priorities, holding a job, or cancelling a job.

Complete information on the progress of batch jobs is always available. Within specified limits, a user can specify when a job should be run (during evening hours when demands on the system are less, for example), and if the job should be held, released, or deleted from its queue.

HELP

Users can get information about various aspects of an RSX-11M/M-PLUS system through the HELP command or, for help on DCL-specific features, by typing a question mark in response to any DCL prompt.

Typing HELP on the terminal displays a list of the HELP files available. To get help on the TYPE command, for example, the user enters HELP TYPE on the terminal. In such an instance, the HELP text consists of a brief explanation of the command followed by an illustration of the syntax.

System managers can create help files that can be made available to all users, to provide information on special aspects of their installations. In addition, users can create local help files for their own use.

TEXT EDITORS

For fast and easy program data entry and modification, RSX-11M and RSX-11M-PLUS provide a choice of text editors: EDI or EDT. Both feature automatic backup of input files so that if a user accidentally deletes a large amount of text, or if the equipment fails, the latest backup file is available for quick recovery.

The user invokes the EDI and EDT text editors interactively; that is, the user creates and processes files online.

EDI Editor

EDI is a single-pass, line-oriented, interactive editor that is used to create and maintain text and source files. EDI accepts over 50 commands that determine its mode of operation and control its actions on input files, output files, and working text buffers. The commands are grouped into the following seven categories:

- *Setup commands* select operating conditions, close and open files, select data modes.
- *Locator commands* control the position of the current line pointer and thus determine which text is acted upon.
- *Text modification commands* change text lines.
- *Macro commands* define, store, recall, and use sequences of EDI commands.
- *File input/output commands* transfer text to and from input and output files and save files.
- *Device output commands* send output to the terminal or printer.
- *Close and exit commands* terminate edit operations.

EDI editing functions are on a line-by-line basis. In block-mode operation, EDI reads a block of text from the disk file. A block contains only that amount of text that will fit into the EDI buffer, and editing operations are performed on the text that is in the buffer. When work on that block is completed, a user requests the editor to renew the buffer with the next block of text.

EDT Editor

EDT has two features that distinguish it from EDI:

- It provides unlimited access to an entire file at one time, making it unnecessary to work with smaller sections of a file, as is usually necessary with EDI.
- It provides character-mode editing for users with video terminals. Character mode allows editing at the character and word level, as well as at the line level.

EDT, the standard DIGITAL editor, is an interactive text editor that is particularly useful for entering and maintaining text files. EDT is available on many DIGITAL operating systems. It is especially powerful when used on DIGITAL's VT100 and VT52 video terminals because it can take advantage of the editing keypad on these terminals. With keypad mode, a single keystroke performs an entire editing function—for example, deleting, reinserting, or replacing a word. Users can even redefine the functions of keypad keys (through key macros) to produce commonly used operations.

In change mode on a VT100 or VT52, users can edit one 22-line window (screenful) at a time so that they can observe immediately the effects of any editing operations performed. Instead of being restricted to the most recently altered line, users can see a whole screenful of text and can see the relationship of new and old lines. If the text is longer then 22 lines, it can easily be scrolled through to get to any other point in the file.

Filenames and Filetypes

In naming files, users can uniquely identify a file by specifying its filename and filetype, illustrated in the following format:

filename.typ

The filename can be from one to nine alphanumeric characters and can assume any name that is meaningful to the user.

The filetype, a three-character identifier preceded by a period, describes the nature of the file's contents. Although the filetype can consist of any three alphanumeric characters that are meaningful to the user, several filetypes have standard meanings. Some of the standard filetypes are:

Filetypes	Default Use
.BP2	BASIC-PLUS-2 source program
.CBL	COBOL source program
.FTN	FORTRAN source program
.MAC	MACRO-11 source program
.CMD	Indirect command file
.DAT	Data file
.LST	Listing file
.OBJ	Object-module output from assembler or compiler
.TSK	Task-image file

PROGRAMMING LANGUAGES

All RSX-11M/M-PLUS systems include the MACRO-11 assembler for programming the computer using its instruction set. A wide variety of language processors is optionally available to high-level language programmers: FORTRAN IV, FORTRAN-77, PDP-11 COBOL, COBOL-81, BASIC-PLUS-2, and CORAL 66. These are briefly discussed below and described further in Section 6.

PDP-11 FORTRAN-77 is a high-performance, optimizing compiler and object-time system. Upward-compatible from FORTRAN IV, FORTRAN-77 can interface with the Record Management Services (RMS). It produces direct

PDP-11 machine code that is highly optimized for execution-time efficiency on a PDP-11 with a Floating-Point Unit. FORTRAN-77 also can produce sharable code. FORTRAN-77 conforms to the most recent ANSI FORTRAN standard, X3.9-1978 (commonly referred to as FORTRAN-77), at the subset-language level.

FORTRAN IV is based on the former ANSI FORTRAN standard (X3.9-1966). It is characterized by high compilation speed and efficiency in small memory environments.

COBOL-81, a high-performance compiler that produces compact object code, is designed for business systems where high performance and ease of use are prime considerations. It not only follows Level 1 of the 1974 American National Standards Institute Standard X3.23-1974 very closely, but it also implements many items planned for the next release of the ANSI standard.

COBOL-81 provides features that are aimed at making both COBOL programmers and COBOL programs highly productive on both large and small PDP-11 systems. The compiler's extensive library facilities and Interactive Symbolic Debugger will increase programmers' productivity and enable them to produce powerful applications programs. Where available, COBOL-81 takes full advantage of the Commercial Instruction Set to generate even more efficient object code.

PDP-11 COBOL is designed specifically for sophisticated applications requiring rich functionality. Its high computational capabilities complement the system performance capabilities of RSX-11M and RSX-11M-PLUS. It provides industry-standard capabilities (as specified in ANSI-74, X3.23-1974), advanced interactive symbolic debugging facilities, and packed-decimal data support. On the PDP-11/23-PLUS, PDP-11/24, and PDP-11/44, PDP-11 COBOL takes advantage of the CIS to enhance performance in data movement and packed-decimal arithmetic.

As a true compiler, PDP-11 BASIC-PLUS-2 significantly improves the performance of compute-bound BASIC applications. Fast program execution and a variety of advanced programming features make BASIC-PLUS-2 a highly productive programming environment and make it powerful enough for a wide variety of applications. It produces files that can interface directly to the RMS record management system, enabling users to create files, do record mapping, and access records sequentially, randomly, or by key. BASIC-PLUS-2 also has a CALL statement that allows programmers to access external subroutines. A number of BASIC-PLUS-2 statements allow interactive observation and control of program execution. PDP-11 BASIC-PLUS-2 is compatible with VAX-11 BASIC.

CORAL 66 is a high-level block-structured programming language. It is the standard general purpose language prescribed by the British government for realtime and process control applications. The language is designed to replace assembly level programming in industrial and commercial applications.

RECORD MANAGEMENT SERVICES

The Record Management Services (RMS) are a collection of services that extend the programming languages by providing general purpose file- and record-handling capabilities. Programmers use RMS to handle record I/O within programs.

RMS routines provide an efficient and flexible means of handling files and their data. RMS enables programmers to choose the file organization and record access method appropriate for the data processing application. The file organization and record access method are independent of the language in which they are programmed. Every programming language, including COBOL, BASIC-PLUS-2, and FORTRAN-77, uses RMS to process files that are organized to provide sequential, random, or multikeyed indexed record accessing. For further information on RMS and the system's data management capabilities, refer to Section 7.

DEBUGGING AIDS

Debugging aids include an Online Debugging Tool (ODT), a Postmortem Dump (PMD), and a Snapshot Dump ($SNAP) facility to assist in identifying faulty code in a program.

Online Debugging Tool

ODT allows interactive control of task execution. Programmers specify to the Task Builder that they want a debugging aid included in a task.

When a task that includes ODT is run, execution begins at the ODT transfer address, rather than at the task starting address. Therefore, ODT gains control and allows users to type special commands that establish base addresses and that set breakpoint locations within the task. After users tell ODT to begin task execution, ODT saves the instructions at the breakpoint locations specified and replaces them with PDP-11 breakpoint (BPT) instructions. ODT enables the SST (Synchronous System Trap) entry point in the task. Upon encountering a BPT instruction in the task, the Executive passes control to ODT at its breakpoint routine. ODT saves task registers in special locations, restores instructions to the breakpoint locations, and transfers control to the user's terminal. By typing ODT commands, users can examine and alter any instructions or data within task memory.

If a task generates an SST error, ODT gains control at its SST entry point, prints a notice at the user terminal, and passes control to the terminal. The ODT commands can be used to discover the cause of the error, correct it, and perhaps continue executing the task.

Postmortem Dump

PMD is directed by the Executive to extract runtime-related data pertaining to a terminated task, to format it, and to request a printed listing. PMD requires that the Executive option for abnormal task termination and device-not-ready messages be selected at system generation. Normally, when a task generates an SST, such as what would be caused by an improper reference to an odd address or a reference to a nonexistent memory location, the Executive tries to transfer control to an SST entry point defined by the task. If the task does not have an SST routine defined for the particular type of trap, the Executive begins task termination.

By enabling Postmortem Dumps for a task that does not handle SSTs, users tell the Executive to supply more data at abnormal task termination. In other words, the Executive follows the abort procedure and, in addition, creates a request for PMD to create the dump. PMD examines

system and task structures to preserve status and runtime data, reads the task image from memory, and writes it to disk in a readable format. PMD then queues a request to print the file containing the dump data, after which the Executive completes the task abort procedure.

Snapshot Dump

$SNAP generates an edited dump of a running task. The snapshot dump requires users to insert special code in a task. Although more complex than PMD, $SNAP allows users to choose the location at which the dump is created and to select the extent and format of the dump. In addition, users can generate the dump from more than one location and can, therefore, generate as many as are needed during task execution.

It is often useful to include such debugging facilities as $SNAP in a task based on defining a conditional variable. The facility can be included, while debugging, by defining the variable. The facility can then be omitted by reassembling the code with the conditional variable undefined.

PROGRAM DEVELOPMENT UTILITIES

Several kinds of utilities are helpful in program development. These utilities are programs that allow programmers to work with different kinds of files and the contents of those files. Utilities can be invoked from either the DIGITAL Command Language (DCL) or the Monitor Console Routine (MCR) environment.

File Manipulation Utilities

DIGITAL provides many file manipulation utilities, two of which are the Peripheral Interchange Program (PIP) and the File Transfer Program (FLX). With these utilities, users can, among other jobs, copy and spool files and transfer files between volumes.

Peripheral Interchange Program — PIP transfers data files from one standard Files-11 device to another. PIP also performs file control functions. Some of the functions PIP performs are:

- copying files from one device to another
- deleting files
- renaming files
- listing file directories
- setting the default device and UIC for PIP operations
- unlocking files
- spooling files
- specifying file protection values

File Transfer Program — FLX transfers files with different formats from one volume to another. In addition, FLX converts the format of the transferred file to conform to the format of the target volume.

FLX allows users to:

- list directories of cassettes, DIGITAL's RT-11, or DOS-11 volumes
- delete files from DOS-11 and RT-11 file-structured volumes
- initialize cassettes, RT-11, or DOS-11 volumes

FLX performs file transfers and format conversions as appropriate from:

- DOS-11 to Files-11 volumes
- Files-11 to DOS-11 volumes
- DOS-11 to DOS-11 volumes
- Files-11 to Files-11 volumes
- Files-11 to RT-11 volumes
- RT-11 to RT-11 volumes
- RT-11 to Files-11 volumes

File Spooling Utilities

File spooling (Shared Peripheral Operations OnLine) for RSX-11M can include either the Queue Manager or the Serial Despooler Task. The choice of either tool is made at system generation. RSX-11M-PLUS supports only the Queue Manager.

Spooling on RSX-11M and RSX-11M-PLUS is gathering output on a mass storage device—usually a disk—to be sent to an output device—particularly a lineprinter—in an orderly fashion. Despooling is the orderly transfer of this output from the mass storage device to the output device.

Users can spool files by using the PRINT command. Files spooled by tasks will also be queued automatically. With command qualifiers, attributes can be set for the job, and the queues can be displayed. Users can alter, hold, or release a job after it has been placed in a queue.

The Queue Manager is a collection of programs that provides for the orderly processing of queued files. The Queue Manager allows users to specify how, when, and where a file will be despooled and to display information about the queue. It controls the printing and provides all of the services of the Serial Despooler, plus much more. On RSX-11M-PLUS only, the Queue Manager also supports batch processing. The Queue Manager is discussed in greater detail in Section 5.

The PRINT Command — The PRINT command spools print jobs and places them in a queue controlled by the Queue Manager for despooling. The most common use of this command is printing files on the system's lineprinter. Switches on the PRINT command can specify many attributes of the print job, including:

- the time when the spooling is to be done
- the device that is to accept spooled output
- the queue priority of the job
- the restartability of the job
- the forms the job is to be printed on
- the number of lines per page
- the number of copies of each file that are to be printed
- whether the job should be deleted after spooling

The Queue Manager maintains, on disk, the queue of jobs to be printed. If the system is shut down, or if it crashes, none of the files that are yet to be printed are lost.

Print spooling under RSX-11M-PLUS has the additional capability of transparent spooling, wherein output to the lineprinter is transparently written to disk. When the file is closed it is automatically queued to the lineprinter.

The following example illustrates the difference between spooling a file on RSX-11M and the transparent spooling on RSX-11M-PLUS. The commands are MCR commands.

On RSX-11M:

```
>DMP FILE.LST/SP=FILE.TMP
>PIP FILE.LST;*/DE
```

On RSX-11M-PLUS:

```
>DMP LP:=FILE.TMP
```

The Serial Despooler Task — The Serial Despooler Task provides a more primitive means of eliminating contention for the system lineprinter. Rather than waiting for the lineprinter to become available, a task directs the output intended for the lineprinter to a disk file. The task issues a Send Data directive to the serial despooler, placing a data block that identifies the file to be spooled in the serial despooler queue. A request directive is then issued by the task to activate the serial despooler, in case it is not already active. The serial despooler handles FCS-created files, but RMS files can be read only if they are sequential. All files identified in the serial despooler queue are printed in first-in/first-out (FIFO) order.

With the serial despooler, users must use the PIP /SP switch to send files to the lineprinter.

Programming Utilities

RSX-11 supports two programming utilities—the Librarian Utility Program and the File Dump Utility—that allow users to work with library files and to examine file contents.

Librarian Utility Program — The Librarian Utility Program enables users to create, update, modify, list, and maintain object, macro, and universal library files. Library files, direct access files that usually contain modules of the same type, are organized for rapid access by the Task Builder, the MACRO-11 assembler, and the system library routine.

The Librarian is invoked interactively via DCL or MCR commands. Once the Librarian is invoked, users can work with it directly or by means of indirect command files. This provides users with fast entry-point search time, easy update with minimal copying of entire files, and the ability to handle multiple module types.

There are two library types as defined below:

- macro libraries
- object libraries

DIGITAL makes system directives and system-related features available through calls. Definitions for the calls reside in macro libraries. The libraries are stored in a predefined file area known as the User File Directory (UFD). The UFD is located on the system library device.

To use these libraries, a user supplies in the source code the appropriate names of the modules as parameters of a MACRO-11 directive. This action tells the assembler to generate an entry for that call in its macro-symbol table and to search the appropriate library for the definition of the macro symbol.

These libraries provide the code that enables a task to issue system directives and to obtain File Control Services (FCS); allows software to refer to offsets for the Executive data structures; and provides the definitions for Record Management Services (RMS) calls for sequential and relative file I/O.

On RSX-11M and RSX-11M-PLUS, system object libraries provide general utility functions and special purpose Executive features. These libraries, like the macro libraries, reside in the UFD on the system library device.

System library routines reduce program development time and decrease the use of memory by making routines that perform the following functions available to all users:

- save and restore register contents for control transfer to subroutines
- perform integer and double-precision multiplication and division
- convert ASCII input data to binary and Radix-50 format, and vice versa
- convert and format output data to produce text for a readable printout or display
- manage memory and disk-file storage, to accommodate tasks that require large amounts of memory for data that must be transferred between memory and a disk work file
- obtain a command line from a terminal, indirect command file, or an online storage medium
- separate a command line into whatever appropriate dataset descriptions are required by the file system for opening a file
- separate a user-defined command line with user-defined command syntax that includes built-in variables

File Dump Utility — The File Dump Utility (DMP) program produces a printed listing of the contents of a file or volume. The listing can be directed to any suitable output device—lineprinter, terminal, magnetic tape, DECtape, or disk. DMP runs in two modes: file mode and device mode.

In file mode, one input file is specified, and all or a range of virtual blocks of the named file is dumped. A virtual block refers to a block of data in a file. Any Files-11-structured volume serves as the input device for DMP.

In device mode, only the input device is specified, and a specified range of logical blocks is dumped. A logical block refers to an actual block on disk and DECtape, and physical records on magnetic tape and cassette.

Program Maintenance Utilities

Program maintenance includes modifying, patching, and comparing files. The four program maintenance utilities are the File Compare Utility, Source Language Input Program, Object Module Patch Program, and the Task/File Patch Program.

File Compare Utility — The File Compare Utility (CMP) compares two ASCII text files, line by line, to determine whether parallel records are identical. Using CMP, a user can:

- generate a listing showing the differences between the two files. Each difference is listed as a pair; first, the lines from the first file that are being compared to lines in the second file, then the lines from the second file.
- generate a listing in the form of one list, with differences marked by change bars.

- generate output suitable for input to the Source Language Input Program (SLP) utility (described below). This output contains the SLP commands and the input that is required to make the first input file identical to the second input file.

Source Language Input Program — The Source Language Input Program (SLP) is used for source-file maintenance. SLP maintenance is usually performed on the most recent version of the source file, ensuring that the file contains the latest updates and corrections. An optional audit trail in the output file allows a user to keep a record of changes to the software.

With SLP a user can:

- update (delete, replace, add) lines in the existing file
- create source files
- run indirect command files containing SLP edit commands

Both an input file to be updated and command input that consists of text lines and edit command lines specifying the update operations to be performed are input to SLP. To locate lines to be changed, SLP uses locators that are specified as line numbers or character strings. Command input can come directly from a user's terminal or from an indirect command file that contains commands and input lines that are to be inserted into the file. SLP accepts data from any RSX-11M/M-PLUS file-structured device.

SLP output is a listing file and an updated input file. SLP provides an optional audit trail that helps keep track of the update status of each line in the file. Unless suppressed, the audit trail is shown in the listing and is permanently applied to the output file.

Object Module Patch Program — The Object Module Patch Program (PAT) allows a user to update or patch code in a relocatable object module. Input to PAT consists of two files, an input file and a correction file. The input file consists of one or more concatenated object modules that are connected individually by PAT. The correction file consists of object code that, when linked by the Task Builder, either overlays or is appended to the input object module. Unlike the Task Builder and patching options (described below), PAT allows users to increase the size of the object module, because the changes are applied before the module is linked to the Task Builder.

PAT uses corrections and/or additional instructions in the correction file to update the object module. Correction input is prepared in source form and then assembled by the MACRO-11 assembler. Output from PAT is the updated input file.

Task/File Patch Program — With the Task/File Patch Program (ZAP), users can directly examine and modify files on a Files-11 volume. Users can patch data files or task images interactively without having to reassemble and rebuild the task.

ZAP performs many of the functions performed by the RSX-11 Online Debugging Tool (ODT), including:

- *command-line switches* that allow access to specific words and bytes in a file, modify locations in a task image, list the disk-block and address boundaries for each overlay segment in a task disk image, and open a file in read-only mode
- a set of *internal registers* that include eight Relocation Registers
- *single-character commands* that, in combination with other command-line elements, allow users to display, open, close, and manipulate the values in task images and data files

Except in read-only mode, the results of ZAP commands are permanent. By using ZAP commands on a hardcopy terminal, users are assured a record of changes made during the patching process.

Using maps generated by the Task Builder, as well as listings generated by MACRO-11, users have sufficient information to make the required patches rapidly.

RTEM-11 RT-11 EMULATOR

RT-11 is DIGITAL's single-user operating system for real-time applications and program development. It is supported on most PDP-11 systems. With RTEM-11, RSX-11M and RSX-11M-PLUS systems can be used to develop applications programs to run on RT-11 systems.

RTEM-11 is an optional RT-11 emulator that provides the RT-11 program-development environment on RSX-11M and RSX-11M-PLUS systems. Several users can develop RT-11 applications concurrently on an RSX-11 host system. Users can create, edit, assemble, and link programs using RTEM-11 and then execute these programs on an appropriately configured RT-11 system. Programs developed with RTEM-11 execute on RT-11 systems the same way they would had they been developed on RT-11.

Most programs developed on RTEM-11 can also be debugged and tested on RTEM-11. The execution environment supplied with RTEM-11 is Foreground/Background only.

HARDWARE FOR PROGRAM DEVELOPMENT

There are three types of devices available for program development: disks, terminals, and lineprinters. This section briefly introduces these devices. For further information on peripherals, refer to Section 9.

Unless programmers are writing specially tailored code for these devices, the system software handles them transparently through such mechanisms as the print spooler and PIP.

Disks

Disks are the main storage medium on RSX-11M and RSX-11M-PLUS. Disk drivers are either public (accessible to all users) or private (accessible to a restricted set of users). Almost all utility programs work with disk storage as a default device. Public disk resources can be shared to create source-program files and, as needed, to allocate a user's own private drive to store reserved copies of source and documentation files.

Terminals

Terminals are the means by which users communicate with the system. Users input to the system through a typewriter-like keyboard. The system returns output to them either on a screen at a videodisplay terminal or on

paper at a hardcopy terminal. Videodisplay terminals are more convenient because they typically operate at faster rates than hardcopy devices. Hardcopy terminals, however, have the advantage of providing a record of what transpired during a session on the system.

Terminals are connected to the computer through either direct lines or modem units over dialup telephones.

Lineprinters

Lineprinters provide hardcopy output of data. On larger systems, users communicate with the lineprinter through intermediate software called spooling programs. On smaller systems, users can explicitly specify the lineprinter device to which the output should be spooled.

OVERVIEW OF PROGRAM DEVELOPMENT PROCESS

The following figure illustrates (using MACRO-11) the steps in the program development process.

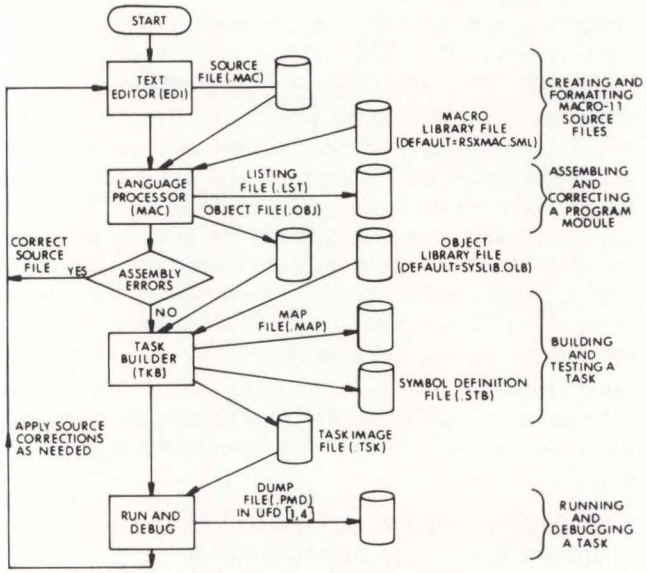


Figure 4-1
Overview of the Program Development Process

The steps normally taken to prepare a program to run on the system are:

1. Create a source program in a file on a disk.
2. Submit the source file to a language processor (assembler or compiler) to produce an object module.
3. Submit the file (or files) containing the object module to the Task Builder to create a file containing a loadable task image.
4. Request the Executive to execute the task.

A language processor creates the file of relocatable object code. MACRO-11, as an assembly language, also accesses the system macro library to include code for system directives in the object file. Compilers invoke system directives through calls to subroutines in the system object library.

The Task Builder creates the file of loadable code assuming certain default conditions about the runtime environment and building these characteristics into the task. The Task Builder also accesses system and user-specified libraries, to resolve references in the task.

Once a task image is created, a user requests the Executive to run the program. If any errors are encountered, the user must edit the source file, reassemble or recompile, build a new task image, and try again.

5 System Management and Maintenance

RSX-11 operating systems provide users with a wide variety of system utilities and tools designed to make system management, maintenance, and operation easy and efficient.

These tools include programs to monitor and control system use, authorize users, grant or restrict privileges, tune system performance, supervise day-to-day operations, and perform system backup and recovery and file restoration. Extensive tools are provided to test and maintain system and volume reliability. In addition, on RSX-11M-PLUS, system-management personnel controls batch processing and can run programs that perform shadow recording and resource accounting.

INTRODUCTION

System-management personnel are responsible for overall system control, operation, and maintenance. RSX-11 supports a wealth of programs to test, monitor, maintain, and customize RSX-11 operating systems during and after system generation.

These programs let system management create and maintain a list of valid users and User Identification Codes, broadcast messages to all terminals, and stop the system after issuing timed warning messages.

The indirect command-file processor (discussed in Section 4) allows all regularly scheduled maintenance activities to be stored as automated command sequences. Using the automatic scheduling feature of the RUN command, these procedures can be initiated automatically at specified intervals.

The system manager often designates one or more users to perform operator functions. A system does not require an operator, but it can have one or several persons performing operation functions that include:

- system startup and shutdown
- task control (changing task priorities and killing tasks, for example)
- device allocation
- volume mount and dismount request servicing
- online disk and magnetic-tape volume and file backup
- software maintenance update installation
- diagnostic execution

An operator uses a command language to control operations, check system status, and run utility programs.

SYSTEM GENERATION AND SHUTDOWN

The system generation utility, called SYSGEN, lets an operator define the specific hardware configuration and select software options. Through an online, interactive, question-and-answer session, the operator can develop a system tailored to the application's requirements. The answers to many of SYSGEN's questions are saved in a file. Later, the saved answer file can be used to modify the system, if its features need to be changed, or to generate another RSX-11 system with the same capabilities.

SYSGEN contains two options that can ease the operator's involvement in the system-generation procedure and can minimize the number of questions to be answered. The Autoconfigure program probes the hardware configuration and, in most cases, provides SYSGEN with a complete and accurate hardware configuration. This can eliminate the need to answer most or all of the SYSGEN questions concerning peripheral devices. On RSX-11M, the Standard Function System option and, on RSX-11M-PLUS, the Full-Functionality option produce a mapped operating system with most operating-system capabilities automatically. If these options are used, the number of SYSGEN questions is reduced dramatically.

User Environment Test Package

Once the SYSGEN process is complete, the User Environment Test Package (UETP) verifies the integrity and operation of the newly generated RSX-11M/M-PLUS system.

UETP consists of several indirect command files that verify the presence and operation of devices, test the basic Executive features, and verify the presence of system utilities.

UETP consists of five test modules:

- Load Test
- I/O Exerciser Test
- Utilities Test
- MCR Command Test
- Interactive Utilities Test

The operator can select which of these tests to run, can indicate how many times to run UETP, can select or omit extended comments at the start of each test, and can exclude specific devices from testing.

Virtual Monitor Console Routine

The Virtual Monitor Console Routine (VMR) is a system program that allows complete interactive configuration of an RSX-11 system.

Containing a subset of the Monitor Console Routine (MCR) commands, VMR is used to make the same changes to the system image file on a disk that can be made to the running system with MCR. Some of these changes include setting the size of pool, creating partitions, loading and unloading drivers, and installing and fixing tasks. The advantage of using VMR is that a system image file can be almost completely configured online before it is booted.

The use of VMR on RSX-11M, RSX-11M-PLUS, and VAX/VMS systems to generate an RSX-11S system is discussed in Section 3.

The SHUTUP Program

With the SHUTUP program, a privileged user can shut down an RSX-11 system in an orderly fashion. SHUTUP prompts for the number of minutes to wait before shutdown, the number of minutes between shutdown messages, and the number of minutes to wait before disabling logins.

Before halting the system, SHUTUP performs such cleanup functions as logging off all logged-in terminals; submitting a user-written SHUTUP indirect command file for execution; stopping, if present, the Queue Manager, Console Logger, and Error Logger; deallocating checkpoint space; and dismounting devices.

USER AUTHORIZATION

Multiuser protection, a system-generation option, allows RSX-11M/M-PLUS installations to monitor and control individual users of the system. This option protects against destructive interference among users.

Use of an RSX-11M/M-PLUS system is controlled by setting up accounts. RSX-11M and RSX-11M-PLUS provide an Account File Maintenance Program for the creation and maintenance of a multiuser account file.

User Identification Codes

System management assigns each RSX-11M/M-PLUS user a two-number identification code. Called a User Identification Code (UIC), it is enclosed in brackets and used (with password) for logging in. The number is in the form of <g,m>, with "g" giving the user's group number, and

"m" giving the user's member number. Under RSX-11's default file protection setup, users with the same group number can use each other's files without hindrance.

The group number also determines whether the user is privileged or nonprivileged: a group number of ten or less signifies a privileged user. Installations usually have only a few privileged users. The system manager and the operators are always privileged. Privileged users have access to every part of the operating system. Nonprivileged users can use most of the operating system, but they cannot change it. Nonprivileged users can issue a DCL SHOW TIME command, for example, and the system will respond with the currently set date and time. Privileged users can issue a SET TIME command to change the system time.

Whenever a user tries to log onto a system, the system checks the account file and determines whether or not the user should be allowed access to the system. The account file describes all the UICs that have been authorized for use and their privileges. One UIC can have several users, each having their own password.

Account File Maintenance Program

The Account File Maintenance Program (ACNT) is an interactive program that allows a privileged user—usually the system manager or an operator—to:

- create the account file
- add new accounts to the file
- examine individual account entries
- modify individual account entries
- list all the account entries in the file
- delete an account from the file

When activated, ACNT lists the options and requests the user to select one. According to the option selected, it responds by requesting further input or by displaying information.

Each account entry includes the following information:

- the UIC
- the password
- the user's system device
- the first and last name
- the date and time of user's most recent login
- the number of times the user has logged into the system
- the default command language interface (CLI)

Nonprivileged users can run the ACNT program to change the account entry descriptions or to change their own passwords.

RSX-11M-PLUS account entries also include a session identifier and a user account number. RSX-11M-PLUS maintains a record of CPU time per user, connect time, and number of pages printed. This information can be processed through user-supplied routines like a billing program.

Data privacy and system security are based on the UIC assigned by system management, volume protection codes, and file protection codes. These are discussed in Section 3 and Section 7.

MONITORING SYSTEM USE

RSX-11M and RSX-11M-PLUS include tools that monitor how a system is being used. Systems management and users can then use this information to take best advantage of the system's resources.

Resource Monitoring Display

The Resource Monitoring Display (RMD) provides information about the active tasks in the operating system and the availability of system resources. This information includes the active tasks, their location in memory, the amount of memory they occupy, and available pool space. RMD generates dynamic displays on video terminals and "snapshot" displays on hardcopy terminals.

The information is presented in easily comprehensible graphic form. RMD can also help locate certain system lockout problems or bugs in an application and/or system-level software.

Software Performance Monitors

SPM-11M and SPM-11M-PLUS are high resolution, low overhead, event-driven performance monitors. Optionally available to run under RSX-11M and RSX-11M-PLUS, these monitors allow users to access the impact of individual tasks on system resources, to determine which tasks use the greatest amount of resources and which tasks wait the longest for resources. Users can also identify resources used heavily by the total system workload, to aid in locating bottlenecks. A flexible set of controls for data collection allow the user to measure and generate reports on usage and waiting times for the CPU, memory, I/O devices, the file system, and the task loader.

Resource Accounting

On RSX-11M-PLUS systems, Resource Accounting, a system generation option, provides a transaction file of system usage information. Accounting gathers data for both the user and the system. With this data, system management can bill individual users for the used resources and can measure overall system usage.

Resource Accounting gathers and saves in the transaction file information about the following topics: users; tasks; system; logons and invalid logons; device allocation, deallocation, mount, and dismount; print jobs; cardreader jobs; system time changes; and device usage.

Console Logger

The Console Logger consists of a driver and the Console Output Task (COT) that handle I/O to the console output device and that record time-stamped system messages on a terminal or in a log file or both. Support for console logging is a SYSGEN option.

The COT handles messages sent to the console device. It allows the system manager to forward messages to an alternative terminal or to a file. The system manager can also forward messages to the console device or logging device. The Console Driver sends all messages to COT, and COT forwards them to the selected terminal or logging device. The Console Driver and COT are supported on mapped RSX-11M systems and on all RSX-11M-PLUS systems.

MAINTAINING VOLUMES

With RSX-11's many volume maintenance facilities, users can back up files onto disks and tapes, locate bad blocks on the volumes, consolidate disk data storage area, and verify the contents of the volumes. In addition, users on RSX-11M-PLUS systems can back up all information as it is being written to a Files-11 disk by utilizing Shadow Recording.

Disk Volume Formatter

The Disk Volume Formatter (FMT) formats and verifies disk cartridge, disk pack, fixed-media disk, and flexible-disk volumes under an RSX-11M/M-PLUS operating system that includes online formatting support, which is a SYSGEN option.

The disks can be completely formatted in normal operating mode or formatted on an individual sector basis in manual operating mode.

In general, FMT performs the following:

- writes a complete header for each section of the disk it is formatting
- verifies the address contents of each sector header
- sets the density for DIGITAL's RX02 floppy diskettes
- lets the user specify an error limit for the volume being formatted (FMT terminates processing if the error limit is reached)
- lets the Bad Block Locator utility to be run (spawned) if the system permits spawned tasks

Bad Block Locator

The Bad Block Locator (BAD) utility tests disks and DECtapes for the location and number of bad blocks and records this bad-block information on the volume. The MCR INI (initialize volume) command is then used to allocate the bad blocks to a specific file. The bad blocks are marked as "in use" and thus cannot be allocated to other files.

BAD supports any last-track device, as well as vendor-supplied cartridges that do not have a prerecorded manufacturer's bad-sector file on the last track. Users can use BAD in its task version, which runs at the same time as other tasks, or in its stand-alone version, which runs by itself on the computer. The stand-alone version must be used if the system has only one disk drive.

Bad Block Replacement Control Task

The Bad Block Replacement Control Task (RCT) handles bad-block replacement and recovery on Mass Storage Control Protocol (MSCP) disks like the RA80 or RA81. Bad-block handling on MSCP disks consists of four stages: detection, notification, replacement, and revectoring.

The disk controller (UDA50) detects bad blocks and notifies the driver. The driver activates RCT. RCT performs all the bad-block replacement functions that enable the controller to revector (redirect) I/O from the bad block to the replacement block.

RCT performs the following bad-block replacement functions:

- stores data from the bad block
- allocates a replacement block

- updates data structures on the disk
- initializes the replacement block

RCT also performs replacement and recovery on MSCP disks that went offline during bad-block replacement or before the contents of a write-back cache were copied to the disk. If RCT determines that bad-block replacement was partially completed when the disk went offline, RCT completes the bad-block replacement process. If RCT determines that the write-back cache was not copied to the disk before the disk went offline, RCT software write-locks the disk so that the contents of the write-back cache are preserved.

Backup and Restore Utility

With the Backup and Restore Utility (BRU), users can back up and restore Files-11 volumes. BRU transfers files from a volume to a backup volume (or volumes), to ensure that a copy of the files is available in case the original files are destroyed. If the original files are destroyed, or if for any other reason the copy needs to be retrieved, users can restore the backup files with BRU commands. Users can run BRU either at the same time as other tasks or stand-alone.

Backup and restore operations that take place on disk and tape volumes are:

- disk to tape—for backup operations
- tape to disk—for restore operations
- disk to disk—for either backup or restore operations

In addition to these basic data transfer functions, BRU provides command qualifiers to:

- initialize disks
- perform selective backup and restore operations
- control such tape processing as density, length, ANSI tape labeling, rewinding, and appending
- perform volume and data checking
- display such information as backup set names and file names

BRU reallocates and consolidates the disk data storage area. It concatenates files and their extensions into contiguous blocks whenever possible, and it can reduce the number of retrieval pointers and file headers required for the same files on the new disk volume.

A BRU operation begins with data on one disk and ends with the same data on another disk, in compressed form.

Disk Save and Compress Utility

The Disk Save and Compress (DSC) utility copies a Files-11-structured disk either to disk or to tape and from DSC-created tape back onto disk. At the same time, DSC reallocates and consolidates the disk data storage area. It concatenates files and their extensions into contiguous blocks whenever possible and, therefore, reduces the number of retrieval pointers and file headers required for the same files on the new volume.

DSC copies files that are scattered randomly over a disk volume to a new volume, without the intervening spaces. This eliminates unused space between files and also reduces the time required to access them.

After a DSC copy operation, individual files are written in available contiguous blocks, and the blocks available for new files are located in a contiguous area at the end of the new volume. If the contents of one disk are transferred to a disk with a larger capacity, the new disk takes on the attributes of the original disk, except that additional storage space is available.

File Structure Verification Utility

For Files-11 volumes, the File Structure Verification (VFY) utility can perform the following functions:

- check the readability and validity of a file-structured volume (default function)
- print the number of available blocks on a file-structured volume
- search for files in the index file that are not in any directory (that is, files that for some reason cannot be accessed by filename)
- validate directories against the files they list
- list all files in the index file, showing the file ID, filename, and owner
- mark as "used" any blocks that appear to be available, but are actually allocated to a file
- rebuild the storage allocation map so that it properly reflects the information in the index file
- restore files that are marked for deletion
- delete bad file headers
- perform a read check on every allocated block on a file-structured volume

Shadow Recording

With Shadow Recording, an RSX-11M-PLUS system backs up all new data as it is written to a Files-11 disk. A SYSGEN option, Shadow Recording creates two identical sets of disks that are called a "shadowed" pair. More than one pair of disks can be shadowed, but shadowed pairs cannot overlap. The two disks must be of the same type—both RK07s or RA80s, for example. The first disk of the pair, the primary disk, is the original disk that exists whether or not Shadow Recording is active. Any disk on an RSX-11M-PLUS system, including the system disk, can be the primary disk of a shadowed pair. The second disk of the pair, the secondary disk, becomes an exact copy of the primary disk.

Shadow Recording operates transparently—writing to and reading from the secondary disk is an Executive function. The Executive always writes the same data to the secondary disk as it writes to the primary disk. When a disk read occurs, the Executive reads the primary disk first. If a read error occurs on the primary disk, the Executive reads the secondary disk. The Executive displays all I/O errors occurring on a Shadow Recording disk pair on the operator's console.

Shadow Recording provides a dynamic backup of all blocks as they are written to the primary disk. That is an important feature for many processing environments, particularly environments in which critical information must be duplicated to safeguard against inadvertent damage or loss in the event a disk error occurs. Recovery may be quicker too, and downtime may be reduced, because disk errors do not necessarily mean an application must be halted.

Because it does backup online and provides an "instant" duplicate disk, Shadow Recording can also be used with systems on which later backup time or resources are unavailable.

CONTROLLING SYSTEM RESOURCES

On RSX-11M/M-PLUS systems, the Queue Manager (QMG) provides for the orderly processing of print jobs and, on RSX-11M-PLUS, batch jobs. Most users who use the Queue Manager are unaware of its presence. Every request to print a listing, submit a batch job, and create a spooled listing or map from a system program is passed automatically to the Queue Manager, which places them in the appropriate queues.

QMG includes privileged commands that are used by the system manager to set up the QMG for the installation. Depending on the needs of the installation, the manager can choose the number of queues to be set up and can choose where the output of these queues will be directed. A system can have as many as 16 output devices to which QMG directs output. The manager can also specify that certain kinds of print jobs will go to one or another lineprinter.

Some installations may have user-written output processors that pass jobs to devices other than lineprinters. QMG can pass jobs to most kinds of devices. These include queues for electrostatic plotters, papertape or cardpunches, or even terminals and magtapes. QMG can direct output to any record-oriented output device. It can also direct output to application tasks or networks.

When a lineprinter or other output device is controlled by the Queue Manager, it is said to be a spooled device. Spooled devices are initialized by the system manager with certain characteristics. For example, a lineprinter can be initialized to print both uppercase and lowercase characters on a previously defined special form. Through PRINT command qualifiers, users can specify the characteristics they want, such as printing with uppercase characters only.

Complete information on the progress of queued jobs is always available. Within specified limits, a user can specify when a job should be run (during evening hours when demands on the system are less, for example) and if the job

should be held, released, or deleted from its queue. An operator, and in some instances any user, can control a job queue by changing job priorities, holding a job, or cancelling a job.

Printing Files

The Queue Manager handles the orderly printing of files for an RSX-11M/M-PLUS system through software tasks called despoolers, or print processors. There is a print processor for every lineprinter on a system. Even if a system does not have a hardware device of the lineprinter type, it will have some device designated as the system output device that takes the part of the lineprinter. This output includes all requests to system tasks for maps or listings, logs from batch jobs, and print jobs entered through the PRINT command, as well as output from applications tasks unique to an installation.

A print job can specify the forms required, the number of copies, the print priority, holding an entry, deleting files after printing, and printing after a specified time.

The print job can contain one or more files to be printed. Print jobs can be submitted by an interactive user, a program, and, on RSX-11M-PLUS, by a batch job. Print jobs are automatically submitted at the end of a batch job.

Each print job has a large, easily read job header and file header burst pages, to identify print requests and files within a print request. Both types of headers contain identification and general accounting information.

Batch Processing

RSX-11M-PLUS has a multistream batch processing capability. The operations personnel can control the number of batch streams that can run.

Batch jobs can be submitted by an interactive user, a program, or another batch job. When the number of batch jobs submitted exceeds the number of streams, the remainder of the batch jobs are held in a batch input queue. As with the spool queues, the operator can control the batch job queue by changing job priority, holding a job, or killing a job.

Volume mount commands issued in a batch job can request a generic device, such as a disk, or specific device unit, such as disk-drive unit 2. The batch job waits until the operator satisfies the mount request, while other batch jobs proceed.

TUNING THE SYSTEM

On the RSX-11 systems, users have the tools they need to fine-tune their systems for maximum performance and flexibility.

Pool Monitoring

Pool monitoring support controls the use of the system's Dynamic Storage Region (DSR, or pool). Pool is a contiguous area in memory allocated by the Executive and used as a workspace for storing such system data structures as system lists and control blocks. Pool is included at the top of the Executive's permanently mapped address space in memory. The size of pool must be sufficient to handle all of the dynamic storage requests of the system. By default, SYSGEN extends the mapped system Executive to its maximum size, to create the largest possible pool space.

Pool requirements for a system are dependent on the configuration, application, and degree of system loading. Enough pool must be available to satisfy peak demands; otherwise, system performance will be degraded.

Since nearly all Executive functions require pool, a system can exhaust pool when system activity is very heavy. This can happen if too large a number of tasks are installed, if too many volumes are mounted, or if a number of other conditions are present. When this happens, the system does not appear to have crashed, but it is not functioning normally.

To avoid this condition, pool monitoring can be used. Pool monitoring oversees pool levels, restricts use, and notifies the operator if pool is near depletion. Pool monitoring has two components: the RSX-11M/M-PLUS Executive pool monitor code and the privileged Pool Monitor Task (PMT).

The pool monitor code within the Executive monitors the amount of free pool and detects major pool events. When a major pool event occurs, the Executive notifies PMT.

PMT's response to the information provided by the Executive depends on specific pool events and conditions. It executes actions appropriate to the condition detected. For example, for a low-pool state, it establishes pool-use controls, including preventing nonprivileged users from logging on and suppressing INSTALL/RUN/REMOVE sequences on nonprivileged terminals. It also sends warning messages to all logged-on terminals and displays pool statistics at the console terminal.

Pool-monitoring support is available on mapped systems only. It is a SYSGEN option on RSX-11M systems and it is included by default on RSX-11M-PLUS systems.

I/O Queue Optimization

To increase the throughput of disk subsystems, RSX-11M-PLUS supports I/O Queue Optimization, a set of optimization algorithms that RSX-11M-PLUS uses when choosing among I/O requests of equal priority. For maximum flexibility, system management can choose from three methods of I/O Queue Optimization.

- The *nearest cylinder* method processes the I/O request closest to the current request, regardless of whether it comes before or after the current request.
- The *elevator* method processes I/O requests as it moves in one direction along the disk, then changes direction to process requests in the opposite direction.
- The *cylinder scan* method processes requests in only one direction along the disk, the direction being from the lowest cylinder number to the highest.

System management can choose the method that best suits the processing environment, the physical location of data on a disk drive, and how often tasks access data areas. All three methods attempt to minimize head-seek time, which decreases the seek-time component of the total file service time.

Without I/O Queue Optimization, RSX-11M-PLUS groups the I/O requests in the queue, by priority, and on a first-in/first-out basis. The highest-priority requests appear first in the queue and are processed in sequence. With I/O Queue Optimization, the I/O requests within priority groups are examined, and the request having the "best"

disk address is processed first. Highest-priority requests are still serviced before lower-priority requests. However, throughput is enhanced by advantageous reordering of requests within priority levels.

A "fairness count" is also set that limits the number of times an I/O request can be passed over.

By reducing erratic head movement, I/O Queue Optimization has the potential to speed disk operation.

MAINTAINING SYSTEM RELIABILITY

The RSX-11 operating systems include a variety of aids that help operators ensure system integrity. These aids are both comprehensive and easy to use.

I/O Exerciser

The I/O Exerciser (IOX) detects and diagnoses I/O problems on the disks and tapes in a system's hardware configuration. IOX exercises Files-11 disks, non-file-structured disks, magnetic tapes, DECtapes, and cassettes. Used by system management and maintenance personnel, it determines whether these units are correctly executing I/O operations. In addition, IOX measures system activity and provides a command language for executing test functions. IOX output consists of detailed error reporting and general information that describes system activity.

IOX performs three kinds of exercises. Operators use the IOX Command Language to specify and control the exercise for the units in a system. They choose an exercise appropriate to a unit and they set exercise parameters that determine how the unit is exercised.

Error Logging

The RSX-11M/M-PLUS Error Logging System records information about errors and events that occur on system hardware, either for immediate action or for eventual analysis and reporting. Error Logging handles mass storage device (disk and tape) errors, as well as memory errors. Since Error Logging is a part of the RSX-11M/M-PLUS system, it is most effective for hardware errors that allow the system to continue functioning.

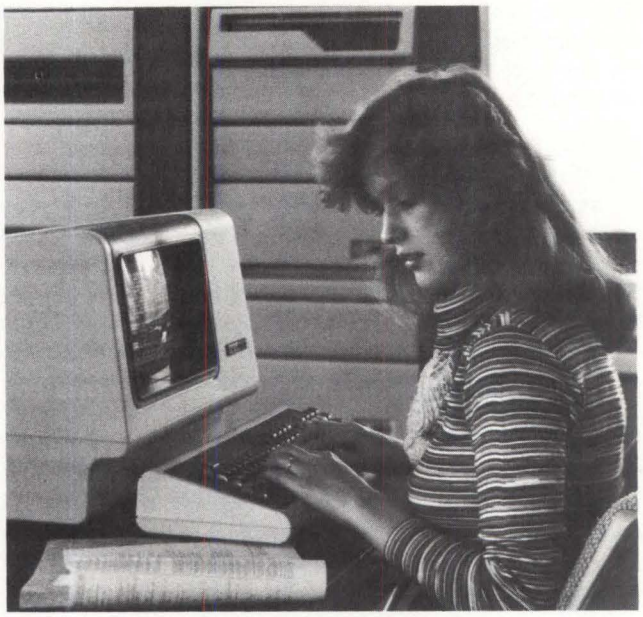
Error Logging is not used to detect information about operating-system failures or about device problems that cause the system to fail. However, it does provide information about what I/O activities occurred on a device at the time of an I/O failure. If a system includes the Crash Dump Analyzer (CDA), CDA can provide reports on operating-system failures.

Error Logging Reports can be used to determine that a device is having problems before it actually fails and causes lost data. For example, a report showing a pattern of recurring errors from different blocks on a single disk head may indicate that the head needs to be replaced.

Crash Dump Analyzer

The Crash Dump Analyzer (CDA) is a specialized utility that helps establish the cause of system crashes. It reads the contents of a system memory dump, formats the information, and outputs the following type of information to a lineprinter for evaluation:

- all memory management and hardware register contents
- the system stack



- task control blocks for each active task
- contents of the clock queue
- information on all devices in the system
- contents of physical memory
- task headers for each task in memory
- contents of each Partition Control Block
- contents of the system Dynamic Storage Region (pool)
- contents of the System Task Directory for all tasks

The CDA is a valuable tool for helping to eliminate a variety of causes of system crashes, because system crashes can occur when users make modifications to the Executive or when privileged tasks are mapped into the system data structures, but are not yet debugged.

Online Software Maintenance

Another way the system manager or operator maintains system integrity is by using the software maintenance tools that make patches and install updates to the RSX-11 binary and source code. The Autopatch (Automated Patching Facility) uses a machine-readable format; the SYSGEN process updates the system automatically with the updates supplied.

Remote Diagnostics

On PDP-11/44 and PDP-11/70 systems equipped with the remote diagnosis option, the system manager can set up the system for remote preventive maintenance or trouble-shooting. When a hardware problem is detected or suspected, the system manager mounts a diagnostic disk pack, sets a switch on the processor console, and calls DIGITAL's local service office. A system manager need not be present at the installation, once the call is made. A technician at DIGITAL's diagnostic center can then connect to the installation, run automated diagnostics, operate the diagnostic console manually, and check the error-log file. If a problem is found, one of DIGITAL's Field Service engineers can bring the proper equipment and replacement modules to make repairs.

RSX-11M-PLUS Reconfiguration Services

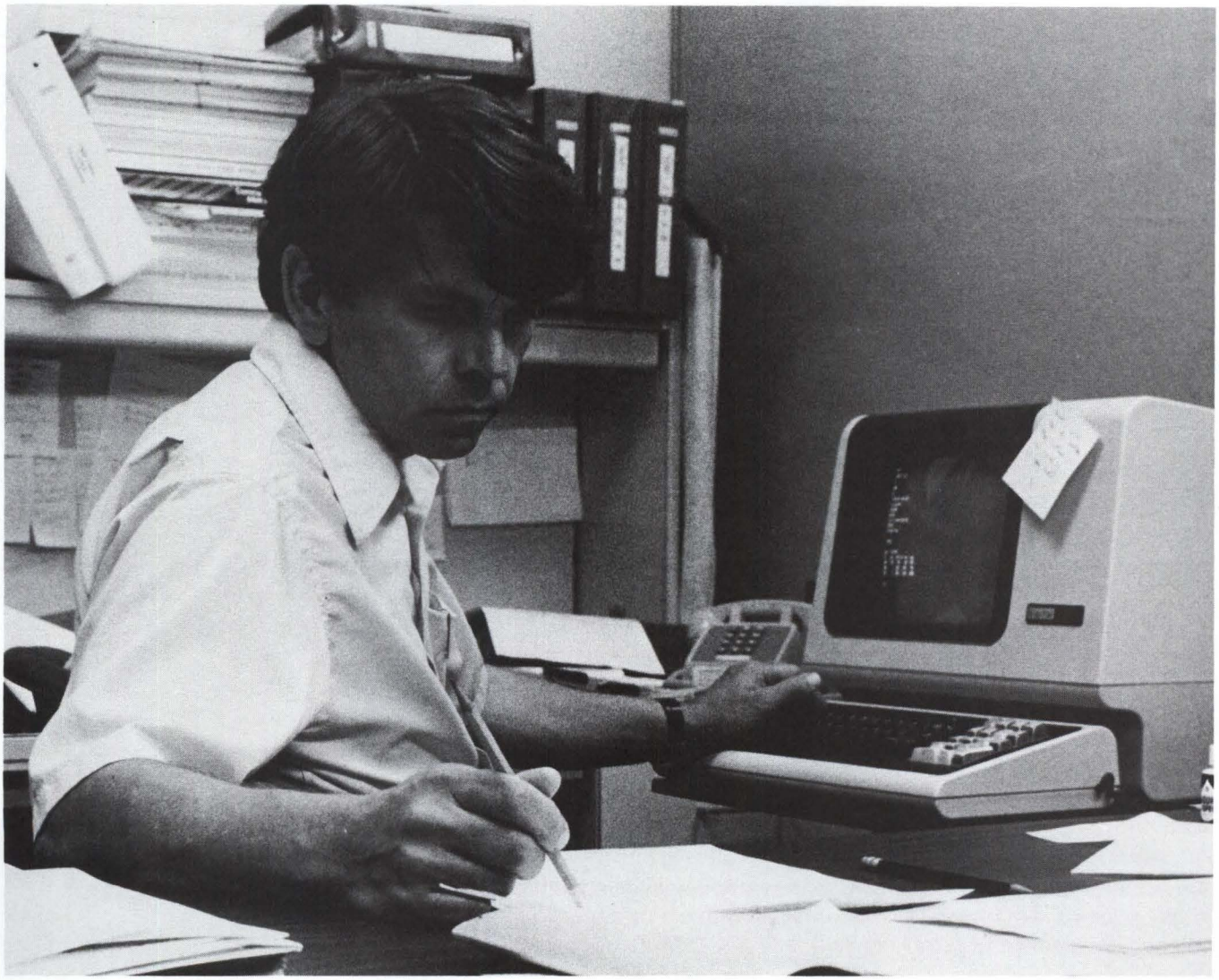
In a reconfigurable system, resources such as memory and devices can be added or removed. Reconfiguration is a means of physically and logically connecting and disconnecting various resources. Reconfiguration services are available only on RSX-11M-PLUS systems.

With the reconfiguration services, an RSX-11M-PLUS system can be reconfigured to bypass faulty hardware elements and to isolate the system from the effects of these elements. An operator can reconfigure a system interactively from a terminal or by means of indirect command files. The reconfiguration services act as an interface between the operator's terminal and the RSX-11M-PLUS system.

An operator can define a set of hardware resources that are accessible from the online system and also remove devices from the pool of resources allocated to the online system. For example, after booting the system the operator can place a failed disk drive offline and then use another drive—either one already online or one placed online specifically for this purpose—to take over for the disabled unit.

Reconfiguration services allow faulty hardware to be isolated so that it does not affect the system—and system users—adversely.

6 The Languages



RSX-11M and RSX-11M-PLUS include a complete program-development environment for a wide range of languages. In addition to the MACRO-11 assembly language, they offer programmers a choice of optional high-level programming languages—FORTRAN-77, FORTRAN IV, COBOL-81, PDP-11 COBOL, BASIC-PLUS-2, and CORAL 66.

RSX-11 systems also provide the tools necessary to write, assemble, link, and run programs and to build libraries of macro and object modules.

Programmers can use the system for development while realtime applications are in progress. They interact with the system online; can execute indirect command files; and, on RSX-11M-PLUS, can submit batch jobs.

INTRODUCTION

RSX-11 systems support a wide range of programming languages to satisfy most types of applications. These include:

- MACRO-11
- PDP-11 FORTRAN-77
- FORTRAN IV
- COBOL-81
- PDP-11 COBOL
- PDP-11 BASIC-PLUS-2
- CORAL 66

MACRO-11 ASSEMBLY LANGUAGE

RSX-11M and RSX-11M-PLUS systems support many programming languages. However, one language is distributed on all systems: the PDP-11 assembly language, MACRO-11.

MACRO-11 processes source programs written in MACRO. It accepts a disk source input file in ASCII format and can create a relocatable object module and a listing file of the source language. The object module contains all the object records and relocation information needed to link with other object modules. All symbol definition done by the assembler has a base of zero. The allocation of virtual addresses and relocation is left for the taskbuilding process.

MACRO-11 provides:

- relocatable object modules
- global symbols for linking separately assembled object programs
- device and filename specifications for input and output files
- user-defined macros with keyword arguments
- a comprehensive system macro library
- program-sectioning directives
- conditional assembly directives
- assembly and listing control functions
- alphabetized, formatted symbol table listing
- default-error listing on command output device
- a Cross-Reference Table (CREF) symbol listing

The MACRO assembler included with RSX-11 also features:

- global arithmetic, global assignment operator, global label operator, and default global declarations
- multiple macro libraries with fast access structures
- predefined (default) register definitions
- an indirect command file facility for controlling the assembly process

Source input to MACRO-11 consists of free-format statements; each line of input contains a single statement. Input statements are either PDP-11 instructions, MACRO-11 assembler directives, macro calls, or direct assignments. Statements can contain labels, so control can change locally (within the module) or so control can be passed between modules (globally).

Symbols and Symbol Definitions

Three types of symbols can be defined for use within MACRO source programs: permanent symbols, user-defined symbols, and macro symbols. Permanent symbols consisting of the PDP-11 instruction mnemonics and MACRO directives do not have to be defined by the user. User-defined symbols are those used as labels or defined by direct assignment. Macro symbols are those symbols used as macro names.

MACRO maintains a symbol table for each type of symbol. The value of a symbol depends on its use in the program. To determine the value of a symbol in the operator field, the assembler searches the macro symbol table, user symbol table, and permanent symbol table, in that order. To determine the value of the symbol used in the operand field, the assembler searches the user symbol table and the permanent symbol table, in that order. The search orders allow redefinition of permanent symbol table entries as user-defined or macro symbols.

Source input usually contains user-defined symbols. User-defined symbols are either internal to a source-program module or global (externally available). An internal symbol definition is limited to the module in which it appears. Internal symbols are local definitions that are resolved by the assembler.

A global symbol can be defined in one source-program module and referenced within another. Global symbols are preserved in the object module; they are not resolved until the object modules are linked into an executable program. With some exceptions, all user-defined symbols are internal, unless explicitly defined as being global.

The assembler allows use of both local and global symbols as labels for statements. When a global statement appears as a label, the related statement is referred to as an entry point (that is, a point at which other modules can transfer control to the current object module). Local symbols can be used as statement labels to define points to which control transfers within an object module.

The assembler evaluates all local-symbol definitions in a source file. Any symbols remaining undefined are classed as global. Thus, after an assembly, all local symbols are assigned relative locations, but the module can contain references for which definitions must be supplied. The resolution of these references is left for the taskbuilding process.

Directives

A program statement can contain one of three different operators: a macro call, a PDP-11 instruction mnemonic, or an assembler directive. MACRO includes directives for:

- listing control
- function specification
- data storage
- radix and numeric usage declarations
- location-counter control
- program termination
- program-boundary information
- program sectioning
- global-symbol definition

- conditional assembly
- macro definition
- macro attributes
- macro message control
- repeat-block definition
- macro libraries

Listing-Control Directives — MACRO includes several listing-control directives to control the content, format, and pagination of all listing output generated during assembly. Listing-control directives enable such documentation features as listing-heading lines, listing-page formatting, and table-of-contents generation.

Through qualifiers in the command string issued to the MACRO assembler, the listing-control options can also be specified at assembly time. The use of these qualifiers provides initial listing-control options that can be overridden by the corresponding listing-control directives in the source program.

Conditional Assembly Directives — By using conditional assembly directives, programmers can include or exclude blocks of source code during the assembly process, based on the evaluation of stated condition tests within the body of the program. This capability allows several variations of a program to be generated from the same source module.

The user can define a conditional assembly block of code and, within that block, issue subconditional directives. Subconditional directives can indicate the conditional or unconditional assembly of an alternate or noncontiguous body of code within the conditional assembly block. Conditional assembly directives can be nested.

Macro Definitions — With special statements called macros, programmers can reference a predefined symbol that causes the assembler to expand a single-line source statement into multiple lines of code or data and to insert the assembled result in the object module. Typically, such macro symbols are typically used for recurring coding sequences. The insertion of the code sequence occurs at each point where the the macro symbol is referenced. Definitions for such macro symbols can occur in the source file itself or can reside in a macro library. The Executive and file-processing services are made available to the program through macro symbols that are defined in a DIGITAL-supplied macro library.

Macro Calls and Structured Macro Libraries — A program can call macros that are not defined in that program. A user can create libraries of macro definitions, and MACRO will look up definitions in one or more given library files when the calls are encountered in the program. Each library file contains an index of the macro definitions it contains, so MACRO can find definitions quickly.

Program Sectioning — The MACRO program-sectioning directives are used to declare names for program sections and to establish certain program-section attributes. These program-section attributes are used when the program is linked into an image.

The program-sectioning directives allow the user to exercise complete control over the virtual memory allocation of a program, since any program attributes established through this directive are passed to the linker. For example, if a programmer is writing multiuser programs, the program sections containing only instructions can be declared separately from those sections containing only data. Furthermore, these program sections can be declared as read-only code, thus protecting them.

Assembling

MACRO-11 is a two-pass assembler. During the first pass, the assembler groups all symbols as either local or global; performs statement generation; locates all macro symbols; and, if necessary, reads the macro definitions from libraries. At the end of the first pass, the assembler must have processed all local references and determined all undefined global symbols to be resolved by the Task Builder.

During the second pass, the assembler generates the object-module and listing files, flagging with an error code in the listing file those source statements containing errors. If the programmer requested a cross-reference listing of symbols, the assembler also generates a request for the Cross-Reference Program (CRF) to create the proper information.

The MACRO-11 listing file provides both documentation for the module and a tool for debugging the code. As a reference aid, the assembler generates and includes line numbers in the listing for each statement in the source file. It also maintains a current-location counter for each program section defined in the source file. In addition, the listing includes a symbol table showing symbols, their attributes, and their values (if known at assembly time).

The location-counter value given in the listing file is vital in debugging because it provides the offsets into the module for each program section. An offset, combined with the base load address for a program section (from the Task Builder map), allows access to locations in the memory-resident task image during debugging.

PDP-11 FORTRAN-77

PDP-11 FORTRAN-77, a high-performance optimizing compiler and object-time system, produces machine code highly optimized for execution on PDP-11 systems with a Floating-Point Processor. Running under RSX-11M and RSX-11M-PLUS as well as under the RSTS/E timesharing operating system, its optimization techniques improve memory efficiency and increase program execution speed.

PDP-11 FORTRAN-77 is an extended implementation of the American National Standard Institute (ANSI) subset FORTRAN-77 standard, X3.9-1978. PDP-11 FORTRAN-77 contains all the features of the ANSI FORTRAN-77 subset, many of the full-set language features, and extensions that are not included in the ANSI FORTRAN-77 standard. It also provides optional switch-selectable support for programs conforming to the previous ANSI FORTRAN standard (X3.9-1966).

FORTRAN-77 meets the Federal Information Processing Standard Publication (FIPS PUB-69) requirement for a "flagger." The flagger optionally produces diagnostic messages for syntax and/or source-form elements that do not conform to the full-level ANSI FORTRAN-77 standard.

PDP-11 FORTRAN-77 supports the CHARACTER data type and Block IF constructs. Support for the CHARACTER data type means that character constants, variables, and arrays can be declared and used. Function subprograms or statement functions cannot be of character data type. The length of a character data element is an integer constant or expression. In addition, this feature improves portability of FORTRAN programs between PDP-11 and VAX systems.

Block IF constructs, including IF...THEN, ELSE IF...THEN, ELSE, and END IF statements, are used for conditional execution of blocks of statements. This is an adaptation of the popular IF...THEN...ELSE construct found in PL/I, ALGOL, and PASCAL.

Full-Language FORTRAN-77 Features

PDP-11 FORTRAN-77 includes the following features of full-language FORTRAN-77 as defined by the ANSI standard.

- double-precision and complex data types
- intrinsic functions, including LEN, ICHAR, and INDEX
- exponentiation forms, including double-precision and complex
- format edit descriptors, including S, SP, SS, T, TL, TR, Iw.m, and Gw.dEe
- generalized DO-loop parameters
- generic function selection based on argument data type for FORTRAN-defined functions
- upper- and lower-bounds specification in array declaration
- substrings of character variables and character-array elements
- optional syntax for I/O statements (UNIT= and FMT=)

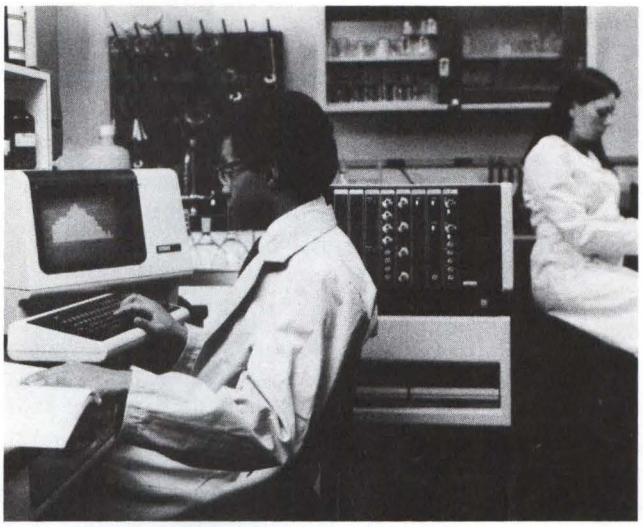
Extensions Beyond ANSI FORTRAN-77

PDP-11 FORTRAN-77 also includes the following extensions to the ANSI standard.

- Language elements for keyed and sequential access to RMS multikey ISAM files.
- DEFINE FILE, FIND, DELETE, REWRITE, and UNLOCK statements.
- TYPE and ACCEPT input/output statements.
- Comments permitted at the end of each source line.
- INCLUDE statement.
- BYTE data type.
- ENCODE and DECODE statements.
- Explicit specification of storage allocation units for data types (for example, INTEGER*4).
- Hexadecimal and octal constants.
- Virtual array support for systems with memory management directives. Virtual arrays are memory-resident and require enough main memory to contain all elements of all arrays. The use of virtual arrays in a program frees memory for executable code and other data storage.
- O- and Z-format edit descriptors.

Optimizations

FORTRAN-77 optimizations are designed to produce an object program that executes in less time and is smaller than an equivalent nonoptimized program.



PDP-11 FORTRAN-77 optimizations include:

- *Peephole optimizations.* The initial machine instructions generated by a FORTRAN program are examined to find operations that can be replaced by shorter, faster code sequences. The final code generated by the compiler contains these improved code sequences.
- *Common-subexpression elimination.* Often, the same subexpression appears in more than one computation. If the values of the operands of a common subexpression are not changed between computations, that value is computed once and substituted wherever the subexpression appears.
- *Removal of invariant expressions from DO loops.* An algorithm executes faster when computations are moved from frequently executed program sequences to less frequently executed program sequences. So computations within a loop involving only constants are moved outside the loop.
- *Allocation of processor registers across Block IF constructs and DO loops.* Wherever possible, frequently referenced variables are retained in registers to reduce the number of load and store instructions executed. Frequently used variables and expressions are also assigned to registers across block IF constructs and DO loops.
- *Sharable code.* The compiler can produce shared object code as a compile-time option. Shared tasks can then be created by using the multiuser-linker option. This improves memory utilization in multiuser systems because many users share one memory-resident task.

Object Time System

The PDP-11 FORTRAN-77 Object Time System (OTS) is a set of object modules that, when selectively linked with compiler-produced object modules by the Task Builder, produce a task ready for execution. Selective linking means that if a program performs only sequential formatted I/O, no direct-access I/O routines are included in the task.

The OTS is composed of the following routines:

- *math routines,* including the FORTRAN-77 library functions and other arithmetic routines (for example, exponentiation routines)
- *miscellaneous utility routines* (for example, ASSIGN, DATE, and ERRSET)
- *routines that handle FORTRAN-77 input/output*
- *error-handling routines* that process arithmetic errors, I/O errors, and system errors
- *miscellaneous routines* required by the compiled code

PDP-11 FORTRAN-77 can create two object-time systems. The OTS based on File Control Services (FCS) handles many file operations transparently to the user and allows sequential and direct access to sequentially organized files. The OTS based on Record Management Services (RMS) uses RMS to provide access to sequential, relative, and indexed files. The RMS OTS provides additional capabilities and is normally larger than the FCS OTS. For a given task, it is not possible to mix FCS OTS modules with RMS OTS modules.

Hardware Requirements

The FORTRAN-77 compiler is an optional language processor for the RSX-11M and RSX-11M-PLUS operating systems. For both operating systems, the hardware configuration must include a Floating-Point Processor, at least 48 Kbytes of user memory, up to 370 contiguous disk blocks for the compiler task, and an additional 150 to 250 disk blocks for the Object Time System library file and auxiliary support files, depending on installation options.

FORTRAN IV

FORTRAN IV is an extended implementation of the FORTRAN language based on the previous specification for ANSI FORTRAN, X3.9-1966. To help minimize program-development time, it was designed to both reduce the size and increase the speed of the executable programs it produces. The FORTRAN IV compiler is a fast one-pass compiler that produces object code without using temporary files. No intermediate assembly step is required.

Program portability is ensured because the same FORTRAN IV processor is available on DIGITAL's RSX-11M, RSX-11M-PLUS, RT-11, RSTS/E, and CTS-500 operating systems. The compiler is also available in compatibility mode under VAX/VMS, allowing host-system program development.

PDP-11 FORTRAN-77 is upwardly compatible from FORTRAN IV. FORTRAN-77's primary advantages over FORTRAN IV include:

- structured programming features
- a wider variety of data types
- enhanced optimization techniques
- its ability to produce sharable code

FORTRAN IV, however, requires only 16 Kbytes of memory; FORTRAN-77 requires 48 Kbytes.

Because FORTRAN IV is a superset of 1966 ANSI-standard FORTRAN, standard conforming programs written for other computer systems can run unmodified under FORTRAN IV.

Powerful FORTRAN IV extensions to the former ANSI standard simplify program coding. Some of these are:

- *Array subscripts.* Any arithmetic expression can be used as an array subscript. If the value of the expression is not an integer, it is converted to integer type.
- *Array dimensions.* Arrays can have up to seven dimensions.
- *Alphanumeric literals.* Strings of characters bounded by apostrophes can be used in place of Hollerith constants.
- *Mixed-mode expressions.* Mixed-mode expressions can contain any data type, including complex and byte. Data conversions by assignment are not needed.
- *End-of-line comments.* Any FORTRAN statement can be followed in the same line by a comment that begins with an exclamation point.
- *Read/write end-of-file or error condition transfer.* The specifications END=n and ERR=n (where n is a statement number) can be included in any READ or WRITE statement to transfer control to the specified statement upon detection of an end-of-file or error condition. The ERR=n option is also permitted in the ENCODE and DECODE statements, allowing program control of data format errors.
- *General expressions in I/O lists.* General expressions are permitted in I/O lists of WRITE, TYPE, and PRINT statements.
- *General expression DO and GOTO parameters.* General expressions are permitted for the initial value, increment, and limit parameters in the DO statement and as the control parameter in the computed GOTO statement.
- *DO increment parameter.* The value of the DO statement increment parameter can be negative.
- *Optional statement label list.* The statement label list in an assigned GOTO is optional.
- *Override field width specifications.* Undersized input data fields can contain external field separators to override the FORMAT field width specifications for those fields (called "short field termination"), permitting free-format input from terminals.
- *Default FORMAT widths.* The FORTRAN IV programmer can specify input or output formatting by type and default width and precision values will be supplied.
- *Additional I/O statements.* These include file control and attribute definitions; list-directed (free format) I/O; device-oriented I/O; memory-to-memory formatting; and unformatted direct access I/O, which allows the FORTRAN programmer to read and write files written in any format.
- *Logical operations on INTEGER data.* The logical operators .AND., .OR., .NOT., .XOR., and .EQV. can be applied to integer data to perform bit masking and manipulation.
- *Additional data type.* The BYTE data type is useful for storing small integer values as well as for storing and manipulating character information.
- *IMPLICIT declaration.* The IMPLICIT command defines the implied data type of symbolic names.

Language Statements

A FORTRAN IV program consists of statements and optional comments. There are two kinds of statements: executable and nonexecutable. Executable statements describe the action of the program. Nonexecutable statements describe the data arrangement and characteristics and provide editing and data conversion information. There are assignment statements, control statements, I/O statements, FORMAT statements, and specification statements. FORMAT and specification statements are nonexecutable.

FORTRAN IV has no statement-ordering requirements; therefore, declarations can appear anywhere within the source program. Terminal format input (using the tab character to delimit fields) makes program preparation easier.

Command String Specification Options

In the input/output file specification command string issued to the FORTRAN IV compiler to request program compilation, programmers can specify a number of switch parameter options. These include:

- generating inline code that directly supports the floating-point and extended instruction sets for processors with the additional arithmetic hardware
- suppressing internal sequence numbers, which reduces program storage requirements for generated code and slightly increases execution speed
- enabling array vectoring, which decreases the time necessary to reference elements of a multidimensional array

Optimizations

Many techniques are used to increase the execution speed of an object program. Redundant expressions that occur in the same basic block of a program are located and eliminated.

Expressions that are constant, such as multiple constant subscripts of an array, are calculated at compilation time. Constant portions of subscripts are absorbed into the array address so additional operations are not required during execution.

Branch structure optimizations improve program speed and size. And, as an option, inline code generation can be selected for integer and logical operations.

A unique automatic "array vectoring" feature of FORTRAN IV eliminates the time-consuming multiply operations required in array subscripting. Precompiled FORMAT statements make formatted input and output conversions faster and smaller.

To give a final polish to each program, FORTRAN IV does extensive local ("peephole") optimization, examining each sequence of operations output and substituting a shorter and faster sequence of instructions, if possible.

Libraries

With the operating system's librarian utility, the FORTRAN IV programmer can create and modify a library of commonly used assembly language and FORTRAN functions and subroutines. Library files can be included in the command string to the linker utility. The linker recognizes the file as a library file and links only those routines in the library that are required in the executable program. By default, the Task Builder also automatically searches the system library for any other required routines.

An RSX-11 library consists of object modules. Two types of libraries exist: shared and relocatable.

Shared libraries are located in main memory, and a single copy of each library is used by all referencing tasks. Access to a shared library is gained by specifying the name of the library at taskbuild time. Shared libraries are built using the Task Builder. They must contain sharable (reentrant) code.

Relocatable libraries are stored in files on disk. Object modules from relocatable libraries are built into the task image of each task referencing the module. The Task Builder is used to include modules from relocatable libraries in a task image. When a library specification is encountered in the command string, those modules in the library that contain definitions of any currently undefined global symbols are included in the task image. The user can construct relocatable libraries of assembly language and FORTRAN routines, using the Librarian utility.

Hardware Requirements

FORTRAN IV runs on any valid RSX-11M or RSX-11M-PLUS system configuration with at least 16 Kbytes of memory available for the FORTRAN compiler. If run in a larger partition, the compiler uses the extra space for program and symbol-table storage.

COBOL-81

COBOL-81, a high-performance compiler that produces compact object code, is designed for business systems where high performance and ease of use are prime considerations. In addition to following Level 1 of the 1974 American National Standards Institute Standard X3.23-1974 very closely, it implements many items planned for the next release of the ANSI standard. It has also been validated at the low level by the Federal Compiler Testing Center. This means that COBOL-81 is certified as complying with the 1974 ANSI standard, with certain exceptions, for use at government installations.

COBOL-81 provides features that are aimed at making both COBOL programmers and COBOL programs highly productive on both large and small PDP-11 systems. The compiler's extensive library facilities and Interactive Symbolic Debugger will increase programmers' productivity and enable them to produce powerful applications programs. COBOL-81 runs on PDP-11 processors with or without the Commercial Instruction Set (CIS) under RSX-11M and RSX-11M-PLUS, as well as under RSTS/E, DIGITAL's PDP-11 timesharing operating system.

COBOL-81 code executes at high speeds and is memory-efficient. Where available, COBOL-81 takes full advantage of CIS to generate even more efficient object code.

The compiler performance is impressive—it averages up to 500 lines per minute on a PDP-11/44. Because the compiler is designed for the Commercial Instruction Set, it generates compact, high-performance object code, resulting in highly productive applications. This design also requires less use of time-consuming overlays.

COBOL-81 has many features in common with VAX-11 COBOL. These features are implemented with the same syntax on both compilers. In this way, code developed using COBOL-81 can be migrated to VAX-11 COBOL. COBOL users can start out on a small PDP-11 system, and later, when their computing needs change and grow, they can easily move their COBOL applications to a VAX system. Additionally, a VAX/VMS system can be used to develop code that will eventually be compiled using COBOL-81.

Language Elements

COBOL-81, defined as an implementation of ANSI COBOL, supports the following features.

Module	Function	ANSI Level
Nucleus	Contains all language elements necessary for internal processing	1 (excluding ALTER statement, external switches) plus selected level 2 features
Table Handling	Defines and manipulates tabular data	1
Sequential I/O	Defines and accesses sequential files	1 plus selected I/O level 2 features
Indexed I/O	Defines and accesses indexed sequential files	2
Segmentation	Specifies overlay of the Procedure Division at object time	2 (excluding independent segments)
Interprogram Communication	Calls separately compiled subroutines and passing parameters	1
Library	Calls frequently used data descriptions and program text sections	1

Data Types

The COBOL-81 compiler supports multiple data types. These data types cover a wide range of applications, provide flexibility in system design, and allow compatibility with other vendors' COBOL implementations. The compiler supports the following data types:

- numeric DISPLAY data
- numeric COMPUTATIONAL data
- packed-decimal data (COMPUTATIONAL-3)
- alphanumeric DISPLAY data

Character String Facilities

COBOL-81 provides INSPECT, STRING, and UNSTRING verbs for character-string handling. Using these verbs, programmers can count and/or replace embedded character strings and can join together or break out separate strings with various delimiters.

File Organization

The COBOL-81 compiler supports both sequential and indexed files. The Indexed I/O Module statements enable COBOL-81 programs to use RMS multikey record management services to process files. These files can be accessed sequentially, randomly, and dynamically, using one or more indexed keys to select records. Additional I/O features include variable-length records through extensions to the RECORD VARYING clause; the ability to designate sequential input files as optional; additional FILE STATUS values; and an APPLY clause from which users can specify file characteristics that are not ordinarily available through COBOL language syntax.

Because COBOL-81 uses RMS for I/O handling, it can handle files created under other PDP-11 languages, given data type compatibility. A valuable feature, the multikey facility provides flexibility and power in developing application systems. In addition, COBOL-81 supports file sharing, an important feature required for interactive applications programs.

Library Facility

COBOL-81 supports a full Level 1 ANSI-74 library facility. Frequently used data descriptions and program text sections can be stored in library files that are available to all programs. The library files can be copied at compile time to reduce program preparation time and to eliminate errors during program development.

Resident Library Support

COBOL-81 also supports resident libraries. The use of resident libraries decreases disk storage requirements for task images, increases Task Builder performance, and increases memory availability in a multiuser environment.

CALL Facility

With the CALL statement, a COBOL programmer can execute routines that are external to the source module in which the CALL statement appears. The COBOL-81 compiler produces an object module file from a single source file. The object module file can be taskbuilt with other ob-

ject files to produce an executable image. Thus, COBOL programs can call external routines written in COBOL-81 and in the MACRO-11 assembly language.

The CALL statement facility has been extended to allow the programmer to pass arguments BY REFERENCE (the default in COBOL) or BY DESCRIPTOR.

Symbolic Interactive Debugger

COBOL-81 provides an easy-to-learn, easy-to-use interactive debugger. The COBOL Symbolic Interactive Debugger allows for faster, error-free program development. Programmers can debug COBOL programs by including the debugger when taskbuilding the program, rather than having to alter the source program during testing. Programmers can follow the program flow during the execution of a job.

The debugger offers the programmer the capability to:

- reference data items by their user-defined names
- reference section names and paragraph names
- examine and modify the value of variables during program execution
- optionally stop and restart programs at the line numbers, section name, or paragraph name specified by the programmer
- gain control at program commencement and at abnormal termination

Other Debugging Features

To further help programmers debug, the COBOL-81 compiler produces fully descriptive diagnostic messages—from simple warnings to major error detection—listed at the point of error. Many error conditions are checked at compile time. The compiler can also produce a Data and Procedures Map and a Cross-Reference Listing.

When an error occurs during execution, the type of error, program name, and line number of the source statements that caused the error are displayed at the user's terminal. If the program is executed within an active PERFORM, the line numbers of the active PERFORM statements will be produced on the user's terminal. When the error is detected during execution of a subprogram, a backwards trace of the calling programs that were active at the time of error is produced on the user's terminal.

Interactive COBOL Execution

Programmers have the flexibility to design their own screen format. COBOL-81's ACCEPT and DISPLAY statements (in the Procedure Division) allow for easy terminal-oriented interaction between a COBOL-81 program and the programmer.

COBOL-81 also can call FMS, the Forms Management System.

Utility Programs

COBOL-81 provides the REFORMAT and BLDODL utilities to help programmers develop applications.

The REFORMAT utility converts COBOL source programs that are coded using DIGITAL's terminal format into the ANSI-standard format accepted by other COBOL compilers throughout the industry. It also has the inverse option to convert programs written in ANSI-standard format to DIGITAL's terminal format. The shorter, easy-to-enter terminal format, designed for use with interactive text editors, saves disk space and compile-time processing when a user is initially migrating from a non-DIGITAL COBOL system to COBOL-81.

The BUILD Overlay Description Language (BLDODL) utility combines skeleton overlay description files generated by COBOL compilations into a single ODL file. This utility gives the user a simplified way of structuring segmented programs or subprograms into an efficient task image.

Installation

COBOL-81 was designed so that customers can install the software without the aid of one of DIGITAL's software specialists. The installation procedure determines a default compiler for the user's hardware configuration. If the compiler meets the user's requirements (indicated by a yes response to questions from the system), the default compiler is then built. If, however, the compiler is not acceptable, the system prompts the user with several more questions, in order to build a customized compiler.

Hardware Requirements

COBOL-81 runs on any valid RSX-11M/M-PLUS configuration with:

- a user area of at last 48 Kbytes of memory if the RMS resident library will be used (52 Kbytes for all other applications)
- at least 3,500 free blocks of online storage on the public disk structure and additional space for user programs and data files

PDP-11 COBOL

PDP-11 COBOL is an optional high-level language designed to provide fast direct-access data processing for commercial applications. Its high computational capabilities complement the system performance capabilities of RSX-11M and RSX-11M-PLUS.

Based on the ANSI-74, X3.23-1974, standard, PDP-11 COBOL includes advanced interactive symbolic debugging facilities and packed-decimal data support. PDP-11 COBOL takes advantage of the Commercial Instruction Set (CIS) when available to enhance performance in data movement and packed-decimal arithmetic.

PDP-11 COBOL can be used to create online terminal applications or to write batch applications. Typical PDP-11 COBOL customers either have a large library of PDP-11 COBOL programs or have applications based on other vendors' high-level ANSI-74 COBOL.

Language Elements

PDP-11 COBOL is a fully implemented compiler conforming in language element, representation, symbology, and coding format to ANSI-74 COBOL. It includes:

- a full high-level Nucleus module, providing all language elements necessary for internal processing
- a full high-level Table Handling module for defining and manipulating tabular data
- a full high-level Sequential I/O module for defining and accessing sequential files
- a full high-level Relative I/O module for defining and accessing indexed sequential files, including dynamic access

- a full high-level Indexed I/O module for defining and accessing indexed sequential files, including dynamic access and multiple alternate keys
- a low-level Interprogram Communication module for calling separately compiled subroutines and for passing parameters
- a Segmentation module for specifying overlay of the PROCEDURE DIVISION at object time
- a full low-level Library function with partial high-level REPLACING facility for copying predefined COBOL text into the source program and for changing text while copying
- conditional variables
- nested conditionals

Data Types

PDP-11 COBOL supports all the standard data types as well as most of the common COBOL data types. These data types, which are required in a wide variety of applications, provide flexibility for application specification and design.

- Numeric COMP-3, Packed-Decimal Data
- Numeric COMPUTATIONAL (COMP), Binary Data
- Numeric DISPLAY Data (ASCII)
- Alphanumeric DISPLAY Data (ASCII)

Character String Facilities

PDP-11 COBOL provides INSPECT, STRING, and UNSTRING verbs for character-string handling. Using these verbs, programmers can search for embedded character strings with TALLY and REPLACE and can join together or break out separate strings with various delimiters.

File Organization

The Sequential I/O, Relative I/O, and multikey Indexed I/O modules meet the full ANSI-74 high-level standards and include all the COBOL verbs. The Indexed I/O Module statements enable COBOL programs to use RMS multikey record management services to process files. These files can be accessed sequentially, randomly, and dynamically, using one or more indexed keys to select records. The multikey facility offers flexibility and power in the development of application systems and is a valuable language feature.

COBOL uses RMS data management services to implement user file handling. In addition, COBOL programs can create and/or read ANSI standard-format magnetic-tape files. Finally, COBOL programs can build files that the DATATRIEVE software package processes, and vice versa.

Library Facility

With PDP-11 COBOL, programmers have a full ANSI-74 low-level library facility that includes high-level extensions (COPY...REPLACING). Frequently used data descriptions and program text sections can be held in library files that are available to all programs. These files can then be copied at compile time to reduce program preparation time and to eliminate a common source of errors.

CALL Facility

The CALL statement allows COBOL programs to invoke separately compiled subprograms, passing arguments in the process. Subprograms can be written in PDP-11 COBOL or MACRO-11. The CALL facility:

- provides flexibility through modular development of application systems
- permits functional separation of small, well-defined source modules
- gives the programmer access to operating system-dependent features via subroutines written in MACRO

Symbolic Interactive Debugger

PDP-11 COBOL provides an easy-to-learn and easy-to-use interactive debugger. The Symbolic COBOL Interactive Debugger reduces the time required to test programs. Altering source programs during testing is often unnecessary because the programmer can debug COBOL programs by including the debugger when linking the program. By using the symbolic debugger, programmers can follow program flow during the execution of a job and they can:

- reference data items by their user-defined names, including qualification and subscripting
- reference section names and paragraph names
- examine and modify the value of variables during program execution
- stop and restart programs at locations specified by the programmer at the beginning or during the execution of the program
- alter program flow during the debugging session
- gain control before normal or abnormal termination

Other Debugging Tools

To make program debugging even easier, the PDP-11 COBOL compiler produces source-language listings with embedded diagnostics. Fully descriptive diagnostic messages are listed at the point of error. Over 400 different error conditions are checked—varying from simple warnings to major error detections.

Debugging large source programs is further simplified with the optional Data Division allocation map and with modular programming techniques that are offered by the segmentation and interprogram communication facilities.

Another useful debugging aid is the optional cross-reference listing produced by the compiler. It is a listing of all data names, procedure names, and the source-line numbers of those program lines containing the definitions and references. For each name, a list of ordered source-line numbers is displayed. Source-line numbers for defined items are distinguished from source-line numbers for referenced items.

Interactive COBOL Execution

The ACCEPT and DISPLAY statements of the PROCEDURE DIVISION allow easy terminal-oriented interaction between a PDP-11 COBOL program and a programmer.

The ACCEPT statement allows the programmer to enter input lines to the COBOL program. The ACCEPT statement also can retrieve the current date or time from the system.

The DISPLAY statement transfers data from a specified literal or data item to a specified device—normally the programmer's terminal. The statement can be modified by a special WITH NO ADVANCING phrase (without automatic appending of carriage return and linefeed) that allows the COBOL program to control the format of the message sent. The WITH NO ADVANCING phrase causes the device to remain positioned on the same line and the same character position following the last character displayed. This is especially useful when typing prompting messages on the terminal.

The ACCEPT and DISPLAY statements are intended primarily for use with keyboard devices. However, PDP-11 COBOL also allows the ACCEPT statement to accept cards from a cardreader and the DISPLAY statement to display data on a lineprinter.

In addition, more advanced screen facilities can be invoked by PDP-11 COBOL by calling FMS, the Forms Management System.

Source-Program Formats

The disk-resident PDP-11 COBOL compiler accepts source-program input from cards, console terminals, and disks, including input from source-text library files stored on disks.

PDP-11 COBOL accepts source programs that are coded using either the conventional 80-column card reference format or the shorter, easy-to-enter DIGITAL terminal format.

Terminal format is designed for use with interactive text editors. It eliminates the line number and identification fields and allows horizontal tab characters and short lines. These capabilities offer potential savings in disk space and allow easier interactive input of source programs.

Conventional format produces source programs that are compatible with the reference format of other COBOL compilers throughout the industry.

Utility Programs

PDP-11 COBOL offers the RFRMT and MERGE utilities to aid programmers with data processing. RFRMT (reformat) converts DIGITAL terminal-format COBOL programs into conventional-format ANSI COBOL programs. MERGE merges skeleton overlay description files generated by COBOL compilations into a single ODL file.

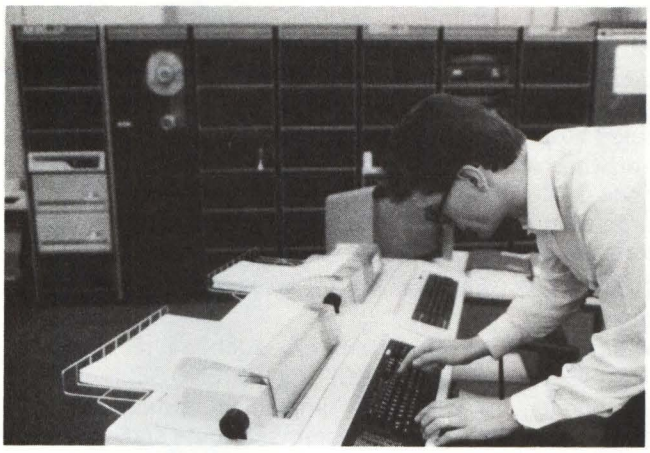
Programmers can enter source programs in the simpler terminal format and, should those programs ever require compatibility with the reference format of other COBOL compilers, use RFRMT to convert their programs to conventional format.

Hardware Requirements

PDP-11 COBOL is supported on any RSX-11M/M-PLUS configuration that includes the Extended Instruction Set, a user area of at least 62 Kbytes of memory, and at least 4,000 free blocks of online disk storage on the public disk structure.

PDP-11 BASIC-PLUS-2

DIGITAL's PDP-11 BASIC-PLUS-2 is far more than a basic language. A powerful structured programming language for PDP-11 systems running RSX-11M and RSX-11M-PLUS



as well as RSTS/E, it has the features and runtime efficiency needed to handle such large applications as payroll and accounting. At the same time, BASIC-PLUS-2 provides a highly interactive development environment friendly to both novice and professional programmers.

The BASIC Environment

BASIC-PLUS-2 provides a comprehensive environment with all the services and utilities programmers need to write, edit, run, debug, and compile programs. Programmers can take a program all the way from source-code entry through debugging and editing to final compilation, without once having to leave the BASIC environment.

- *BASIC-PLUS-2's line editor* lets programmers write and edit source code quickly and easily.
- *The line-by-line syntax checker* can be set to check each line for proper syntax as the line is entered.
- *The interactive HELP command* displays online documentation for explanations of various BASIC-PLUS-2 language elements.
- *The SEQUENCE command* automatically numbers the lines of a program being entered through a user terminal.
- *The RESEQUENCE utility* renumbers lines and updates all references to reflect the new line numbers.
- *The RUN command* compiles the source program currently being worked on for immediate execution.
- *The LOAD command* permits the explicit loading of precompiled modules for execution by the RUN command.
- *Extensive error-checking* with meaningful error messages makes the compiler's output listing into a vital tool in the debugging process. In addition to being displayed on the terminal, error messages appear in the listing next to the line where the error was detected.
- *The BASIC-PLUS-2 Symbolic Debugger* provides a variety of debugging commands that let a programmer control and observe program execution interactively.
- *The extensive listing-file generation facilities* include optional cross referencing.

Structured Programming

Structured programming is a synonym among programmers for straightforward, modular programs that are easy to build, to debug, to update, and to maintain. Structured programming techniques can help keep the cost of software maintenance down. The "self-documenting" code produced can be more quickly understood, and the program's modularity makes it much easier to add new modules and to modify old one.

BASIC-PLUS-2 supports structured programming in several ways.

- *Block-structured statements* such as IF...THEN...-ELSE...END IF and SELECT...CASE...END SELECT allow programmers to directly implement structured code without resorting to makeshift shortcuts and excessive commenting.
- *31-character variable and function names and statement labels* allow for self-documenting names that indicate their use in the program. And these labels can be used in place of line numbers in most parts of a program.
- *User-named program constants* let programmers give meaningful names to often-used values in a program, such as integers that represent TRUE, FALSE, SUCCESS, and FAILURE flags.
- *Flexibility in program formatting* allows programmers to arrange their source programs so that the functional blocks and the flow of the program can be easily identified. For example, blocks can be indented with tab characters; several statements can be placed on one line; and one statement can be spread over several lines.
- *Program segmentation* allows a program to be constructed of separately compiled modules headed by the SUB or FUNCTION statements, to be subsequently accessed by the main module. Segmentation makes it easier to modify the program and gives programmers more flexibility in overlaying.
- *The CALL statement* can pass parameters BY VALUE, REFerence, or DESCriptor.
- *The EXTERNAL statement* provides access to global variables, functions, and constants, and allows data typing of parameters to aid in minimizing runtime mismatches.
- *The OTHERWISE clause ON GOTO and ON GOSUB* provides a simple solution when the index expression is out of range.

Compile-Time Directives

With compile-time directives, programmers can put instructions for the BASIC-PLUS-2 compiler in their source code. These directives can be used to promote the development of applications that are portable across different systems. Programmers can use the conditional compilation directives, for example, to selectively compile small sections of code that have been inserted for debugging purposes. Compile-time directives also give programmers more control over the nature and extent of their listing files. For example, programmers can determine which sections of code they want included in a listing file.

File and Data Handling

BASIC-PLUS-2's great flexibility in file I/O facilities makes it a very practical language for developing commercial applications. The language supports both terminal format files and block I/O, in addition to the full range of RMS file-handling facilities.

BASIC-PLUS-2's support of RMS record I/O operations includes sequential, relative, and indexed file organizations, primary and alternate keys, integer keys, segmented string keys, default name, user open, fixed and variable length records, record mapping, and the high-speed record file address (RFA) access method. This host of I/O capabilities lets programmers choose the file access method best suited to the job, whether it requires fast data access or low processing overhead.

BASIC-PLUS-2 is equally flexible in the way it handles data formats.

- *DYNAMIC MAP and REMAP* statements provide runtime allocation of fields within maps.
- *Data typing* can be implicit or explicit. Explicit data types allow the declaration of variables that are of any of the supported data types.
- *Variables of all types* can be used in the same program, allowing a program to be more compact.
- *Dynamic string handling* provides for easy manipulation of alphanumeric data. No limit other than the amount of available memory is imposed on the size of string values or string elements of arrays manipulated in memory.
- *Virtual arrays* are random-access disk-resident files. A BASIC-PLUS-2 program accesses virtual arrays using element subscripts, just like a memory-resident array. Because the arrays are stored on disk, the programmer can manipulate large amounts of data without affecting program size.
- *The COMMON statement* enables one subprogram to pass data to another subprogram, much in the manner of the FORTRAN COMMON statement. Strings passed in COMMON are of fixed length, thus reducing string-handling overhead.

BASIC-PLUS-2 supports the following data types:

- REAL data types:
  - *SINGLE precision* data type provides a range of 2.9×10^{-37} to 1.7×10^{38} and a precision of six digits.
  - *DOUBLE precision* data type provides a range of 2.9×10^{-37} to 1.7×10^{38} and a precision of 16 digits.
- INTEGER data types:
  - *eight-bit bytes* with a range of -128 to +127.
  - *16-bit words* with a range of -32,768 to +32,767.
  - *32-bit longwords* with a range of -2,147,483,648 to +2,147,483,647.
- The STRING data type allows both static (in MAP or COMMON) and dynamically varying lengths.

Report-Formatting Capabilities

BASIC-PLUS-2's report-formatting facilities include specialized features for commercial and statistical applica-

tions. The PRINT USING statement and FORMAT$ function give programmers the control over the appearance and location of data on an output line that they need for creating professional-looking lists, tables, reports, and forms. Available features include:

- suppression of zero fields
- zero or blank-fill fields
- commas in large numeric values
- CR (credit) or DR (debit) indicators
- floating currency symbols for numeric fields
- asterisk-fill on numeric fields

Compiler Efficiency

The BASIC-PLUS-2 compiler produces threaded code that requires less memory than conventional in-line code and uses less CPU time at program execution than interpreted BASIC. The compiler implementation reduces the program's CPU time demands because its arithmetic computation and string handling are very efficient.

The BASIC-PLUS-2 compiler assumes certain compile and build switches and taskbuilder parameters automatically. This feature makes BASIC-PLUS-2 programs easier to enter, to compile, and to build. It also can decrease taskbuild times, and can make the disk size of a resulting program significantly smaller than with other versions of BASIC.

More than one version of the compiler can be running on a system at the same time. This can be very handy when programmers need to compile programs to run on differently equipped systems—those with the Extended Instruction Set only versus those that also have the Floating-Point Unit option, for instance.

Compatibility with Other DIGITAL BASIC Implementations

BASIC-PLUS-2 is a functional subset of VAX-11 BASIC. In fact, programmers can create programs on VAX systems that can run both on PDP-11s and VAX systems by using the BASIC-PLUS-2 subset flagger in VAX-11 BASIC. This flagger warns the programmer when any VAX-11 BASIC programming feature that is not available under BASIC-PLUS-2 is used.

And BASIC-PLUS-2 programmers can use the compile-time directives for conditional compilation to write VAX-oriented code blocks that will not be compiled on a PDP-11 and PDP-11-oriented code blocks that will be ignored by the VAX compiler.

BASIC-PLUS-2 is also a functional superset of DIGITAL's BASIC-PLUS, the traditional programming language on RSTS/E. DIGITAL's *BASIC Transportability Manual* describes how to prepare programs compatible with BASIC-PLUS, as well as how to use a variety of translation utilities available from DIGITAL.

Hardware Requirements

BASIC-PLUS-2 runs on any valid RSX-11M or RSX-11M-PLUS operating system configuration that includes the Extended Instruction Set, at least 128 Kbytes of user memory, and the RMS Record Management System. Also, between 4,800 and 5,800 contiguous disk blocks must be available for product installation.

CORAL 66

CORAL 66 is the standard, general purpose language prescribed by the British government for realtime and process-control applications. A high-level, block-structured language, it is defined in the *Official Definition of CORAL 66* published by Her Majesty's Stationery Office. CORAL 66 is supported on RSX-11M only.

The CORAL 66 language replaces assembly-level programming in modern industrial and commercial applications. It is particularly useful for products that are expected to be long-lived and that require flexibility and easy maintenance.

The CORAL 66 compiler used on RSX-11M is implemented in accordance with the *Official Definition*. In addition to the functionality prescribed in the *Official Definition*, the RSX-11 CORAL 66 compiler provides:

- BYTE, LONG (32-bit integer), and double (64-bit floating-point) numeric types
- generation of reentrant code at the procedure level
- generated code executable on any valid RSX-11S operating system that includes the Extended Instruction Set (EIS)
- switchable option to select a target PDP-11 computer instruction set
- switchable option to optimize generated code
- switchable option to check the bounds of array type variables
- conditional compilation of defined parts of source code
- English-language error messages at compile and (optionally) at runtime
- switchable option to control listing output
- INCLUDE keyword to incorporate CORAL 66 source code from user-defined files
- switchable option to read card format

CORAL 66 with Floating-Point-Unit (FPU) support generates code for floating-point and integer arithmetic.

The CORAL 66 Object Time System (OTS) is a library of object modules that are selectively linked by the Task Builder with compiler-produced object modules to produce a task ready for execution. It enables users to generate executable tasks to run under RSX-11M and RSX-11S.

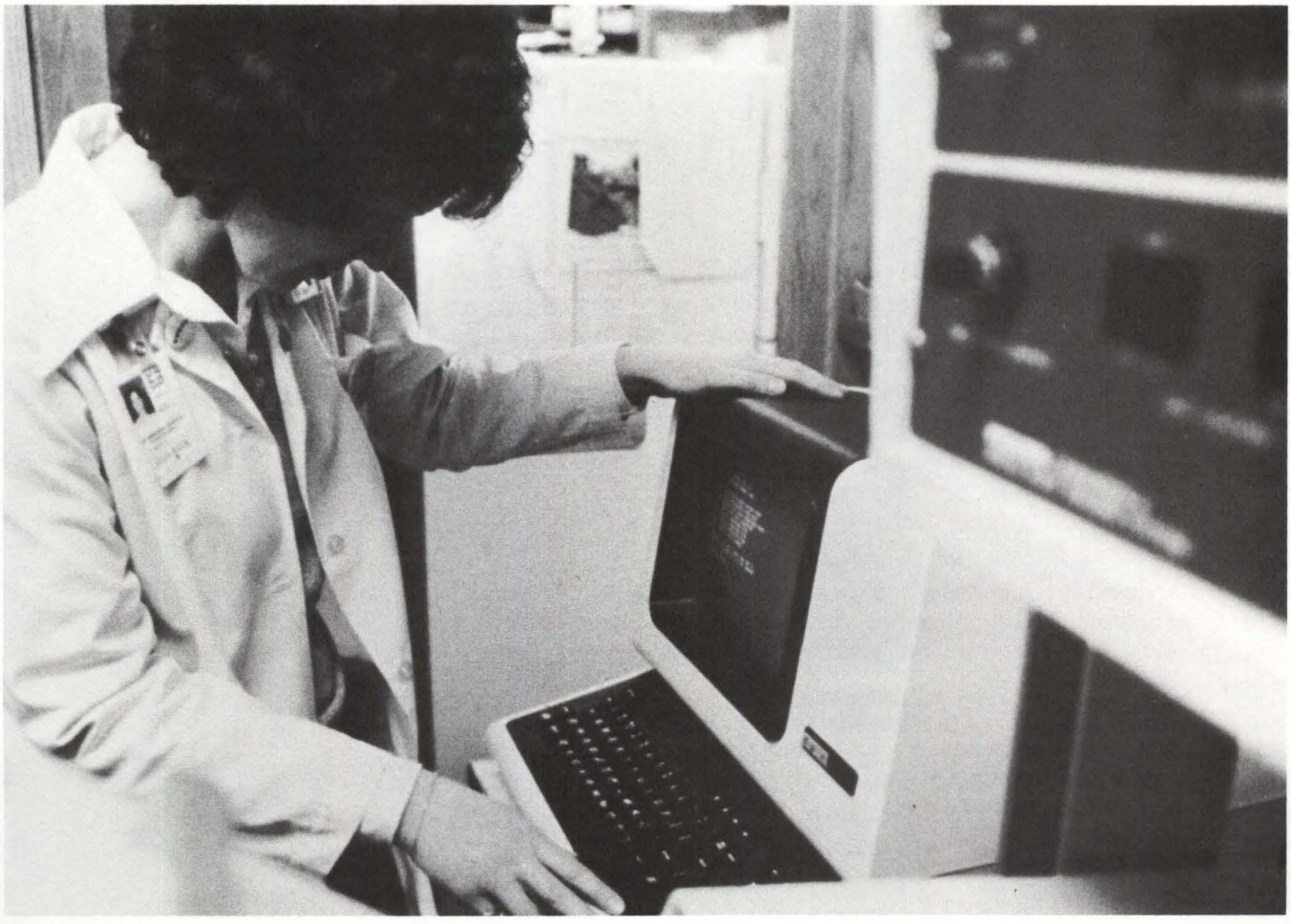
The OTS is reentrant, so part or all of the OTS can be made into a shared library for concurrent use by multiple tasks. It provides stream, record, and block input/output, using FCS, terminal input/output, mathematical procedures, and routines that implement a subset of the RSX-11M and RSX-11S system directives.

Hardware Requirements

CORAL 66 is supported on any valid RSX-11M configuration that includes:

- KT11 Memory Management Unit (or equivalent)
- KE11-E Extended Instruction Set (or equivalent)
- a 48 Kbyte main-memory partition
- a 9-track 800-bit-per-inch magnetic tape system or an RK05, RK06, RK07, or RL01 disk cartridge system

7 Data Management Facilities



RSX-11 data management facilities include a file system that provides volume structuring and directory access, file control services that enable users to perform record-oriented and block-oriented I/O operations, and record management services that provide device-independent access to all types of peripherals.

File Control Services (FCS) allows for both sequential and random access to files. The sequential access method is device-independent; that is, it can be used for record-oriented and random-access file-structured devices. The direct access method can be used only for file-structured devices.

Record Management Services (RMS) provides sequential, relative, and indexed-sequential file organizations. At the application level, RMS supports sequential, random, and record's file address record access, as well as block I/O.

Under both FCS and RMS, logical records are regarded by the user program as data units that are structured and accessed in accordance with application requirements.

RSX-11 also supports several other data management facilities: DATATRIEVE-11, an interactive query, report-writing, and data maintenance system; FMS-11, the Forms Management System utility package; and SORT-11, a sort utility.

INTRODUCTION

The RSX-11M/M-PLUS operating system's data management facilities are provided by:

- the Files-11 file system
- device drivers
- the command interpreters
- file and volume management utilities
- File Control Services
- Record Management Services
- optional utilities for data and file manipulation and inquiry

The file system provides volume structuring and directory access to disk and magnetic tape files. Programmers can use the file system as a base for building their own record processing system, or they can use the FCS File Control Services or RMS Record Management Services.

Both FMS and RMS provide device-independent access to all types of I/O peripherals. The FCS and RMS procedures enable a program to access records within files and provide the same programming interface, regardless of device characteristics. The system includes utilities for RMS file creation and maintenance.

FCS enables users to perform record-oriented and block-oriented I/O operations and to perform additional functions for file control. It supports both sequential and random access to data in files on sequential-access devices (such as magnetic tapes) and random-access devices (such as disks).

RMS, developed after FCS, allows more complex file organizations than is possible with FCS. With it, programmers have a wider choice of file organizations and record access modes and retrieval capabilities. They can use sequential, relative, or multikey indexed-sequential file organizations, and sequential, random, or record's file address access modes.

The device drivers provide the basic I/O device handling for all of the other data management services. Device drivers and their features are described in Sections 3 and 9.

RSX-11 supports DATATRIEVE-11 for data inquiry and report writing, FMS-11 for screen formatting and forms generation, and SORT-11 for reordering data.

FILE MANAGEMENT

RSX-11 includes Files-11 for overseeing the storage and handling of files on volumes. Volumes contain both user files and system files. Files-11 accepts both DCL and MCR commands for initializing (or formatting) the three types of volumes: disks, DECtapes, and magnetic tapes.

Volumes that are not in Files-11 format are described as foreign. Although Files-11 cannot access foreign volumes directly, FLX, the File Exchange program (described in Section 4), translates files from other DIGITAL formats to Files-11 formats.

Volume and file protection are based on User Identification Codes (UICs) assigned to accessors and to the file or volume. The UICs establish the accessor's relationship to the data structure as either owner, owner's group, system, or world (all others). Depending on the relationship, the accessor may or may not have read, write, extend, or delete access to any given file.

The file structure on disk volumes appears to a program to be a virtually contiguous set of blocks. The blocks of the file, however, can be physically located anywhere on a volume. Mapping information is maintained to identify all blocks that constitute a file.

Magnetic tape and DECtape contain files that consist of physically contiguous blocks. These files have ANSI-format labels.

File Directories and Directory Structures

A directory is a file containing a list of files on a given volume. A directory entry contains the name, type, version, and file identifier for a particular file. A directory can list files having the same owner UIC or files having different owner UICs.

A disk volume contains at least one directory, called the Master File Directory (MFD), that contains a list of user directory files. The utility that initializes volumes creates a volume's MFD. When users create a file, the system places the filename in a User File Directory (UFD) and stores their current UIC in the file header to indicate the owner of the file. All UFDs are listed in each volume's MFD. UFD and MFD directories list the names of files and contain pointers to each file's header. The file header contains information about the file's owner and the physical location of the file segments.

A task or user must satisfy the protection mask of both the file to be accessed and the UFD in which the file is located, in order to gain access to a file.

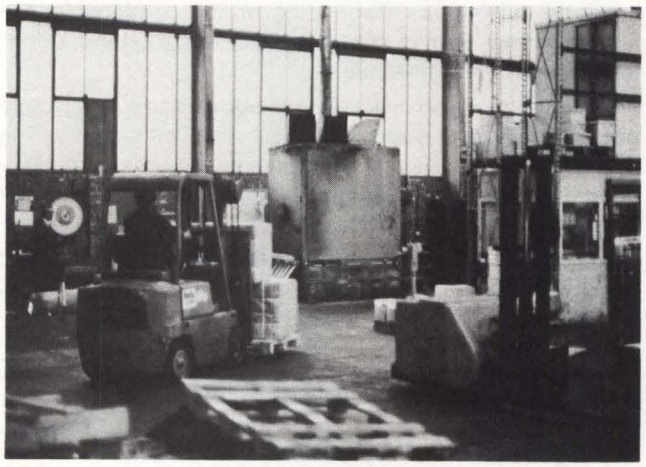
Because directories of files are files themselves, they are assigned owner UICs and can be protected from certain kinds of access, depending on the relationship established by a user's UIC. In the special case of directory files, the file protection fields control an accessor's ability to:

- look up files
- enter new files in the directory, including new versions of existing files
- remove files from the directory

File Specifications

A file specification identifies the file to be used in a file-processing operation. Programs use file specifications to identify the file they want to create, access, delete, or extend. Users supply the command interpreter with a file specification to identify the file they want to edit, compile, taskbuild, copy, or delete, for example. A file specification can be composed of a:

- *device name*—the physical or logical device unit on which the volume containing the file is mounted. The device name is followed by a single colon to delimit it from the remainder of the file specification.
- *User Identification Code (UIC)*—a specification for the User File Directory (UFD) in which the file is listed. The UIC represents the owner's group and the member number and serves as a protection for files and directories. The UIC is always enclosed in square brackets.



- *filename*—an alphanumeric string from one to nine characters in length that specifies the name of the file. A period always separates the filename from the filetype.
- *filetype*—a three-letter alphanumeric string that often is a mnemonic that identifies the nature of the file contents. A semicolon always separates the filetype from the version number.
- *version number*—an octal number that indicates the version or generation of the file.

An example of a complete file specification is:

DK0:[200,200]TECSUM.TXT;2

In this case, *DK0* is the name of the device; *[200,200]* is the UIC; *TECSUM* is the filename; *TXT* is the filetype; and *2* is the version number.

Normally users don't have to provide a complete file specification to identify files. The system sets defaults for most fields in a command-line file specification, depending on the program. When users omit one or more fields in a file specification, the system assumes the default. For example, if the device name is not present, the device is assumed to be the user's current system device. If the version number is not present, it is always assumed to be the latest.

Sometimes users can specify more than one file in a single specification by using a wildcard in one or more fields of the specification. The wildcard causes the system to select all the files that satisfy the explicitly described fields. This saves both keystrokes and time.

FILE AND VOLUME MANAGEMENT UTILITIES

RSX-11 provides many services that aid in file and volume management and maintenance. Programs to locate bad blocks, to verify file structures, and to backup and restore files and volumes are among these services. For more information on these and other services, refer to Sections 4 and 5.

FILE CONTROL SERVICES

RSX-11 File Control Services (FCS) are a set of routines that a task can use to access the file system. It enables users to perform record-oriented and block-oriented I/O operations, and to perform other functions required for file control, such as creating, deleting, opening, closing, reading, and writing. To use FCS, a task invokes the FCS macros. The macros call FCS routines, which issue the actual I/O directives (QIOs). Most of the RSX-11M and RSX-11M-PLUS tasks and utilities use FCS.

Figure 7-1 illustrates the file access operation.

The FCS routines, which can reside in the task's image, in a resident system library, or in a system object-module library, are linked with a task when the task is built. These routines, consisting of pure position-independent code, provide a user interface to the file system, enabling the user to read and write files on file-structured devices and to process files in terms of logical records.

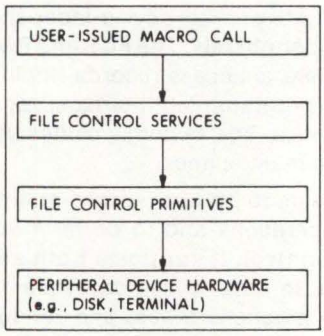


Figure 7-1
File Access Operation

FCSRES is a resident library of commonly used FCS routines. Users can build tasks to link to this single copy of the FCS routines instead of including the routines in each task image. Having only one copy of the FCS routines reduces memory usage. A task accesses a resident library by using the User-mode mapping registers. These registers are also used to map the task code and data, so each task that uses the FCSRES library must reserve some of its logical address space to map the library.

The FCSRES library uses no special hardware, and can thus be used on any of the processors that RSX-11M and RSX-11M-PLUS support.

RSX-11M-PLUS systems support FCSFSL, a Supervisor-mode library of commonly used FCS routines. (Supervisor-mode libraries are discussed in Section 3). Tasks built with FCSFSL have exactly the same functionality as tasks built with FCSRES or tasks that include the FCS routines in their task images.

A task accesses a Supervisor-mode library using the Supervisor-mode mapping registers, a hardware feature available on the PDP-11/44 and PDP-11/70. Mapping the library in Supervisor mode allows the library to reside outside the task's logical address space. Tasks built with FCSFSL can therefore be larger than the same tasks built with FCSRES.

With FCS, users can write a collection of data (consisting of distinct logical records) to a file in a way that enables

them to retrieve the data at will. Data can be retrieved from the file without having to know the exact form in which it was written to the file. FCS thus provides a sense of transparency to the user so that records can be read or written in logical units that are consistent with an application's requirements.

File Access Methods

Under FCS, RSX-11 supports both sequential and random access to files. The sequential access method is device-independent; that is, it can be used for both record-oriented and random-access devices (for example, magtapes and disks). The direct access method can be used only for file-structured random-access devices.

Data Formats for File-Structured Devices

Data are transferred between peripheral devices and memory in blocks. A data file consists of virtual blocks, each of which can contain one or more logical records. Records in a virtual block can be either fixed or variable in length.

Virtual blocks and logical records within a file are numbered sequentially, starting with one. A virtual block number is a file-relative value, while a physical block number is a volume-relative value. For example, the first virtual block in a file is always virtual block number 1, but at the same time it could also be physical block number 156.

Block I/O Operations

The READ and WRITE macro calls allow the user to read and write virtual blocks of data from and to a file without regard to logical records in a file. Block I/O operations provide a very efficient means of processing file data, since such operations do not involve the blocking and deblocking of records within the file. Also, in block I/O operations, the user can read or write files in an asynchronous manner; control can be returned to the user program before the request I/O operation is completed.

When block I/O is used, the number of the virtual block to be processed is specified as a parameter in the appropriate READ and WRITE macro call. The virtual block so specified is processed directly in a buffer reserved by the program in its own memory space.

As implied above, the user is responsible for synchronizing all block I/O operations. For this purpose, the user invokes a WAIT$ macro when it is necessary to wait for the I/O to complete. A user-selectable event flag can be used to coordinate block I/O transfers, allowing simultaneous asynchronous I/O to a number of files.

Record I/O Operations

The GET$ and PUT$ macro calls are provided for processing record-oriented files. GET$ and PUT$ operations perform the necessary blocking and deblocking of the records within the virtual blocks of the file, allowing the user to read or write individual records.

When writing a new file with PUT$ operations, the user program must specify the format of the records. For example, it must specify whether the records are fixed or variable in length, or whether records that are to be output to a carriage-control device are to contain carriage-control information, which can be either at the beginning of the records or embedded within the records.

For sequential access files, I/O operations can be performed for both fixed- and variable-length records. For direct access files, I/O operations can be performed only for fixed-length records.

In contrast to block I/O operations, all record I/O operations are synchronous; control is returned to the user program only after the requested I/O operation is performed.

However, FCS has facilities for simultaneous I/O to a ring of buffers, called multiple buffering, and I/O to a buffer of several disk blocks, called big buffering, to increase program efficiency at the cost of using additional virtual memory.

Because GET$ and PUT$ operations process logical records within a virtual block, only a limited number of GET$ or PUT$ operations result in an actual I/O transfer—that is, when the end of a data block is encountered. Therefore, all GET$ and PUT$ I/O requests do not necessarily involve a physical transfer of data.

The File Storage Region

The file storage region (FSR) is an area allocated in the user program as the working storage area for record I/O operations. The FSR consists of two program sections that are always contiguous. The first program section of the FSR contains the block buffers and the block buffer headers for record I/O processing. The user determines the size of the area at assembly time. The number of block buffers and associated headers is based on the number of files that the user intends to open simultaneously for record I/O operations.

The second program section of the FSR contains impure data that are used and maintained by FCS in performing record I/O operations. Portions of this area are initialized at taskbuild time, and other portions are maintained by FCS. This program section is intentionally isolated from the user to preserve its integrity.

Blocking and deblocking of records during input is accomplished in the FSR block buffer during output. Note also that FCS serves as the user interface to the FSR block buffer pool. All record I/O operations initiated through GET$ and PUT$ calls are totally synchronized by FCS.

Data Transfer Modes

When record I/O is used, a program can gain access to a record in either of two ways after the virtual block has been transferred into the FSR from a file.

- *Move mode.* Individual records are moved from the FSR buffer. Move mode simulates the reading of a record directly into a user record buffer, thereby making the blocking and deblocking of records transparent to the user.
- *Locate mode.* The user program accesses records directly in the FSR block buffer. Program overhead is reduced in locate mode, since records can be processed directly within the FSR block buffer.

Shared Access to Files

FCS permits shared access to files according to established conventions. Two macro calls, among several available in FCS for opening files, invoke these functions. The OPNS$ macro call is used specifically to open a file for

shared access. On the other hand, the OPEN$ call invokes generalized open functions that have shared access implications only in relation to other I/O requests then issued.

By default, several tasks can read the same task simultaneously. Restricted or shared access to files being read or written is possible. Shared access during reading does not necessarily imply the presence of read requests from several separate tasks. The same task can open the same file using different logical unit numbers.

Spooling Operations

FCS provides facilities at both the macro and subroutine level to queue files for subsequent printing. A task issues the PRINT$ macro call to queue a file for printing on the system lineprinter.

FCS Macros and Macro Use

FCS includes four basic kinds of macros that simplify the user's interface to the system's file control primitives. The four kinds are:

- initialization macros
- file-processing macros
- command-line-processing macros
- the CALL macro

The initialization and file-processing macros are used to establish the database description and the necessary temporary storage areas needed to perform I/O operations. The command-line-processing macros are used to dynamically process I/O commands entered from a terminal. The CALL macro is used to invoke file-control routines.

The initialization and file-processing macros set up the following structures to define the database:

- A file data block (FDB) that contains execution-time information necessary for file processing. It defines the basic characteristics of a file—for example, record type, record size, and access privileges, among others.
- A data set descriptor that is accessed by FCS to obtain the file name, type, version number, and location necessary to open a specified file. The data set descriptor is used when a program accesses a given set of known or predefined files.
- A default file name block that is accessed by FCS to obtain default file information required to open a file. This is accessed when complete file information is not specified in the data set descriptor. It is used by programs written to access a general set of files.

There are two types of initialization macros: assembly-time macros and runtime macros. Data supplied during assembly of the source program establish the initial values in the FDB. Data supplied at runtime can either initialize additional portions of the FDB or change values established at assembly time. Furthermore, the data supplied through the file-processing macros can either initialize portions of the FDB or change previously initialized values. The user not only has a broad range of control over defining the database characteristics, but also has control over when the definitions are made.

File-processing macros also determine the way in which files are processed. These macro calls are invoked and expanded at assembly time. The resulting code is then executed at runtime to perform the following operations:

OPEN$	opens and prepares a file for processing
OPNS$	opens and prepares a file for processing; allows shared access to the file (depending on the mode of access)
OPNT$	creates and opens a temporary file for processing
OFID$	opens an existing file using the file identification provided in the filename block
GET$	reads logical records from a file
GET$R	reads fixed-length records from a file in random access mode
GET$S	reads records from a file in sequential-only access mode
PUT$	writes logical records to a file
PUT$R	writes fixed-length records to a file in random mode
PUT$S	writes records to a file in sequential mode
READ$	reads virtual blocks from a file
WRITE$	writes virtual blocks to a file
DELET$	removes a named file from the associated volume directory and deallocates the space occupied by the file
WAIT$	suspends program execution until a requested block I/O is performed
PRINT$	queues a file for printing on a special terminal or lineprinter

In summary, the file-processing macros allow the user to specify random access or sequential access to files, and perform block-oriented or record-oriented file processing. In addition, the PRINT$ macro allows the user to spool files to a lineprinter or terminal device.

The command-line-processing macros allow the user to access special routines available in the system object library. The Get Command Line (GCML) routine accomplishes all the logical functions associated with the entry of a command line from a terminal, an indirect command file, or an online storage medium. The Command String Interpreter (CSI) routine takes command lines from the GCML input buffer and parses them into appropriate data set descriptors required by FCS for opening files.

The CALL macro allows the user to access a special set of file control routines. Among other operations, these routines allow a MACRO program to perform the following: find, insert, or delete a directory entry; rename a file; extend a file; mark a temporary file for deletion; and delete a file.

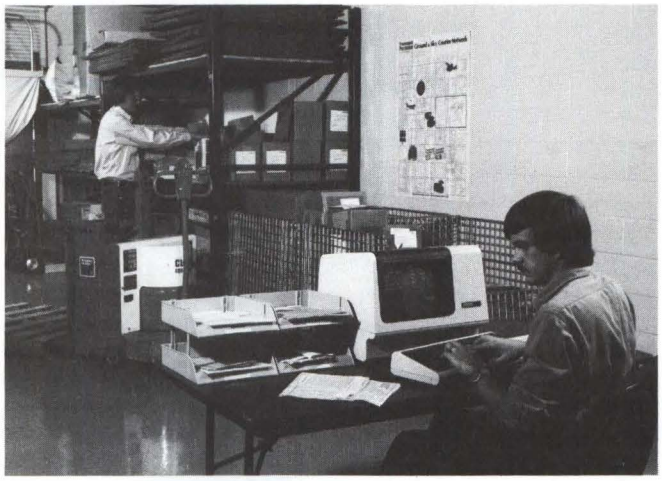
RECORD MANAGEMENT SERVICES

Record Management Services (RMS), a set of general purpose file-handling capabilities, combines with the host RSX-11M or RSX-11M-PLUS operating system to provide

efficient and flexible data storage, retrieval, and modification. When writing programs, users can select processing methods suitable to their application from among several RMS file structuring and accessing techniques.

Not only does RMS handle such functions as file organization and access methods, but it also manages the other file attributes (for example, storage medium and record format) and the runtime environment. By accomplishing most of its work transparently, RMS relieves programmers of many of the complexities associated with file and record manipulation.

Included in both RSX-11M and RSX-11M-PLUS, RMS is also part of all operating systems available from DIGITAL for the PDP-11 family.



File Organization

A *file* is a collection of related information whose requirements are established by the nature of application programs needing the information. For example, a company might maintain personnel information (employee names, addresses, job titles) in one file and product information (part numbers, prices, specifications) in another file. Within each of these files, the information is divided into *records*. In the personnel file, it would be logical for all the information on a single employee to constitute a single record and for the number of records in the file to equal the number of employees.

Similarly, each record in the product information file would represent a description of a single product. The number of records in a file reflects the requirements of a particular application; in this case, a central registry of products sold by a company.

Each record in the personnel and product files would be subdivided into discrete pieces of information known as *data fields* whose number, location within the record, and logical interpretation are defined by the programmer. Program applications then interpret a particular data field in records of the personnel file as the name of an employee. They would interpret another data field in records of the product file as a part number. Figure 7-2 illustrates records that might occur in a personnel file and in a product file.

Thus, the relationships among data fields and records are known and are embedded in the logic of the programs. RMS has no awareness of such logical relationships; rather, RMS processes records as single units of data. Programs either build records and pass them to RMS for storage in a file or issue requests for records while RMS performs the necessary operations to retrieve the records from a file.

The purpose of RMS, then, is to ensure that every record written into a file can subsequently be retrieved and passed to a requesting program as a single logical unit of data. The structure, or *organization*, of a file establishes the manner in which RMS stores and retrieves records. The way a program requests the storage or retrieval of records is known as the *access mode*. Legal access modes depend on the organization of a file.

RMS File Organizations

When creating a file, users have a choice of three file organizations: sequential, relative, and indexed.

Sequential File Organization — In sequential file organization (see Figure 7-3), records appear in a physical sequence that is always identical to the order in which the records were originally written to the file by an application program.

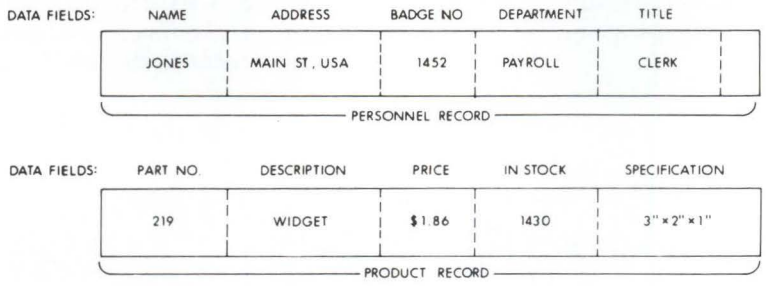


Figure 7-2
Personnel and Product Records

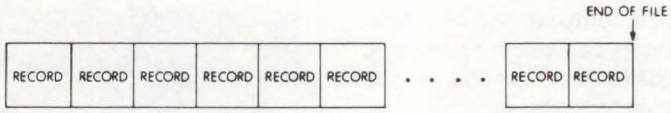


Figure 7-3
Sequential File Organization

Relative File Organization — When relative organization is selected, RMS structures a file as a series of fixed-size record cells. Cell size is based on the size specified as the maximum permitted length for a record in the file. RMS considers these cells as successively numbered from 1 (the first) to n (the last), and the cell's number represents its location *relative* to the beginning of the file.

Each cell in a relative file can contain a single record. There is no requirement, however, that every cell contain a record. Empty cells can be interspersed among cells containing records.

Since cell numbers in a relative file are unique, they can be used to identify both a cell and the record (if any) occupying that cell. Thus, record number 1 occupies the first cell in the file, record number 17 occupies the seventeenth cell, and so forth. When a cell number is used to identify a record, it is also known as a *relative record number*. Figure 7-4 depicts the structure of a file with relative organization.

Indexed File Organization — Unlike the physical ordering of records in a sequential file or the relative positioning of records in a relative file, the location of records in indexed files is transparent to the program. RMS completely controls the placement of records in an indexed file. The presence of *keys* in the records of the file governs this placement.

The key, chosen by the programmer, is a data type present in every record of a particular indexed file. The location and length of this data type are attributes of the file, and therefore are identical for all records in the given file. Legal key data types are: string, integer, unsigned binary, and packed decimal. Selecting a data type indicates to RMS that the content (that is, key value) of that key in any particular record written to the file can be used by a program to identify that record for subsequent retrieval. Since the key is the arbitrary choice of the programmer, it can, of course, be equal to a field. Therefore, in the inventory file, the part-number field could be a key; in the personnel file, the last-name field could be a key.

At least one key, the *primary key*, must be defined for every indexed file. Optionally, up to 254 *alternate keys* can be defined. Each alternate key represents an additional character string in records of the file. The key value in any one of these additional strings can also be used as a means of identifying the record for retrieval.

As programs write records into an indexed file, RMS locates the values contained in the primary and alternate keys. From the values in keys within records, RMS builds a tree-structured table known as an index. The index consists of a series of entries, each of which contains a key value copied from a record that a program wrote into the file. With each key value is a pointer to the location in the file of the record from which the value was copied. RMS builds and maintains separate indexes for the primary and alternate keys defined for the file. Each index is stored in the file. Figure 7-5 shows the general structure of an indexed file that has been defined with only a single key. Figure 7-6 depicts an indexed file defined with two keys: a primary key and one alternate key.

RMS Access Modes

The various methods of retrieving and storing records in a file are called access modes. A different access mode can be used to process records within the file each time it is opened. Additionally, a program can change access mode during the processing of a file, by a procedure known as dynamic access.

RMS provides three record access modes: sequential, random, and record's file address (RFA). For logical reasons, RMS permits only certain combinations of file organization and access mode. Table 7-1, on page 7-8, lists these combinations.

Sequential Access Mode — Sequential access mode can be used with any RMS file. Sequential access means that records are retrieved or written in a particular sequence. The organization of the file establishes this sequence.

Sequential Access to Sequential Files In a file organized sequentially, physical arrangement establishes the order in which records are retrieved when using sequential ac-

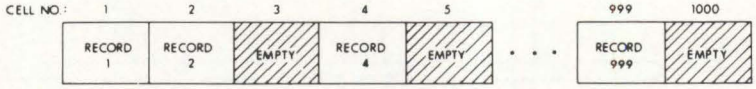


Figure 7-4
Relative File Organization

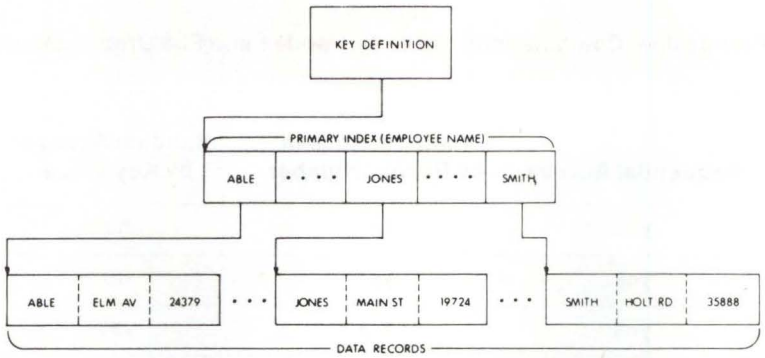


Figure 7-5
Single-Key Indexed File Organization

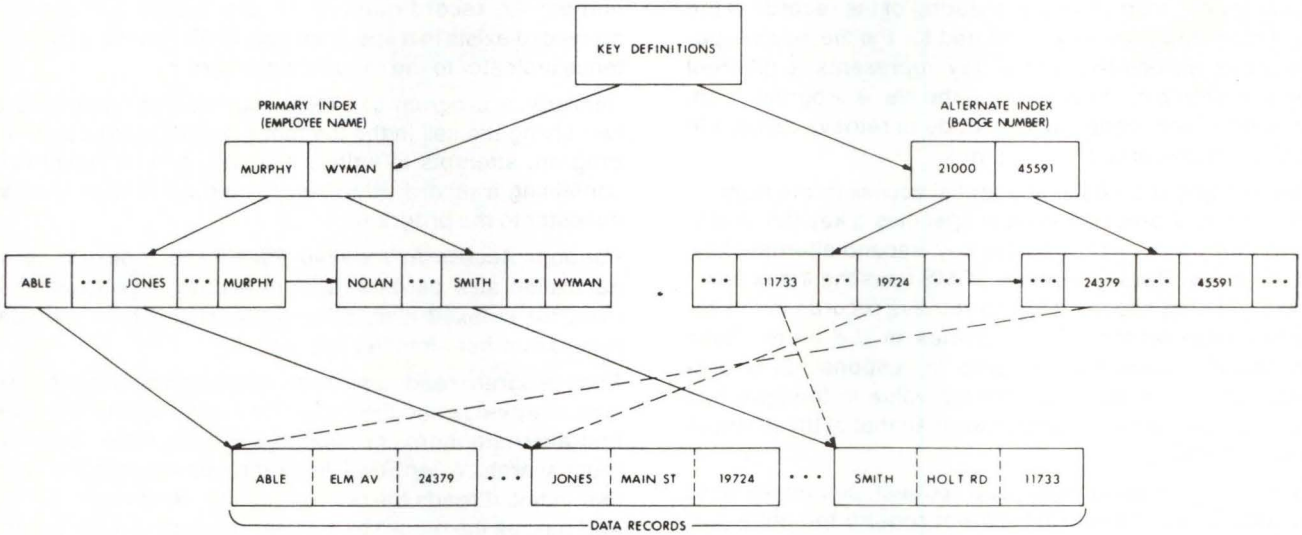


Figure 7-6
Multikey Indexed File Organization

cess mode. To read a particular record in a file—for example, the fifteenth record—a program must open the file and access the first fourteen records before accessing the desired fifteenth. Each record in a sequential file can be retrieved only by first accessing all records that physically precede it. Similarly, once a program has retrieved the fifteenth record, it can read all the remaining records (from the sixteenth on) in physical sequence. It cannot, however, read any preceding record without beginning again with the first record.

When writing new records to a sequential file in sequential access mode, a program must first request that RMS position the file immediately following the last record. Then, each time a program issues a sequential write operation, a record is written following the previous record.

Sequential Access to Relative Files During the sequential access of records in the relative file organization, the contents of the record cells in the file establish the order in which a program processes records. RMS recognizes whether successively numbered record cells are empty or contain records.

When a program issues read requests in sequential access mode for a relative file, RMS ignores empty record cells and searches successive cells for the first one containing a record. If, for example, a relative file contains records only in cells 3, 13, and 47, successive sequential read requests cause RMS to return relative record number 3, then relative record number 13, and finally relative record number 47.

When a program adds new records in sequential access mode to a relative file, the order in which RMS writes the records depends on ascending relative cell numbers. Each write request causes RMS to place a record in the cell whose relative number is one higher than the relative number of the previous request, as long as that cell does not already contain a record. If the cell already contains a record, RMS rejects the write operation. Thus, RMS allows a program to write new records only into empty cells in the file.

Sequential Access to Indexed Files In an indexed file, the presence of one or more indexes permits RMS to determine the order in which to process records in sequential

Table 7-1
Permissible Combinations of Access Modes and File Organizations

File Organization	Sequential Access	Random Access by Record Number	Random Access by Key Value	Access by Record's File Address
sequential	yes	no	no	yes*
relative*	yes	yes	no	yes
indexed*	yes	no	yes	yes

* disk files only

access mode. The entries in an index are arranged in ascending order by key values. Thus, an index represents a logical (rather than physical) ordering of the records in the file. If more than one key is defined for the file, each separate index associated with a key represents a different logical ordering of the records in the file. A program, then, can use the sequential access mode to retrieve records in the order represented by any index.

When reading records in sequential access mode from an indexed file, a program initially specifies a key (for example, primary key, first alternate key, second alternate key, and so on) to RMS. Thereafter, RMS uses the index associated with that specified key to retrieve records in the sequence represented by the entries in the index. Each successive record RMS returns in response to a programmed read request contains a value in the specified key field that is equal to or greater than that of the previous record returned.

In contrast to a sequential read request, sequential write requests to an indexed file do not require the initial key specification. Rather, RMS uses the stored definition of the primary key field to locate the primary key value in each record to be written to the file. When a program issues a series of sequential write requests, RMS verifies that each successive record contains a key value in the primary key field that is equal to or greater than that of the preceding record.

Random Access Mode — In random access mode, the program, rather than the organization of the file, establishes the order in which records are processed. Each program request for access to a record operates independently of the previous record accessed. Associated with each request in random mode is an identification of the particular record of interest. Successive requests in random mode can identify and access records anywhere in the file.

Random access mode cannot be used with sequentially organized files. Both the relative and indexed file organizations, however, permit random access to records. The subsections that follow describe the use of random access with these organizations. Each organization provides a distinct way programs can identify records for access.

Random Access to Relative Files Programs can read or write records in a relative file by specifying relative record numbers. RMS interprets each number as the corresponding cell in the file. A program can read records at random by successively requesting, for example, record number 47, record number 11, and record number 31. If no record exists in a specified cell, RMS returns a nonexistence indicator to the requesting program.

Similarly, a program can store records in a relative file by identifying the cell in the file that a record is to occupy. If a program attempts to write a new record in a cell already containing a record, RMS returns a record-already-exists indicator to the program.

Random Access to Indexed Files The indexed file organization also permits random access of records. However, for indexed files, a key value rather than a relative record number identifies the record.

Each program read request in random access mode specifies a key value and the index (for example, primary index, first alternate index, or second alternate index) that RMS must search. When RMS finds the key value in the specified index, it reads the record that the index entry points to and passes the record to the user program. Under random access the programmer could, for example, instruct RMS to return all records with SMITH in the key-equal-to-last-name field.

In contrast to read requests, which require a program-specified key value, program requests to write records randomly in an indexed file do not require the separate specification of a key value. All key values (primary and, if any, alternate key values) are in the record itself. When an indexed file is opened, RMS retrieves all definitions stored in the file. Thus, RMS knows the location and length of each key field in a record. Before writing a record into the file, RMS examines the values contained in the key fields and creates new entries in the indexes. In this way RMS ensures that the record can be retrieved by any of its key values. Thus, the process by which RMS adds new records to the file is precisely the process it uses to construct the original index or indexes.

Record's File Address Access Mode — Record's file address (RFA) access mode can be used with any file organization as long as the file resides on a disk device. This access mode is further limited to retrieval operations only. Like random access mode, however, RFA access allows a specific record to be identified for retrieval.

As the name suggests, every record within a file has a unique address. The actual format of this address depends on the organization of the file. In all instances, however, only RMS can interpret this format.

The most important feature of RFA access is that the address (RFA) of any record remains constant while the record exists in the file. After every successful read or write operation, RMS returns the RFA of the subject record to the program. The program can then save this RFA to use again to retrieve the same record. It is not required that this RFA be used only during the current execution of the program. RFAs can be saved and used later at any time.

Dynamic Access — Dynamic access is not strictly an access mode. Rather, it is the capability to switch from one access mode to another while processing a file. There is no limitation on the number of times such switching can occur. The only limitation is that the file organization (or, in the case of RFA access, the device containing the file) must support the access mode selected.

As an example, dynamic access can be effectively used immediately following a random or RFA access mode operation. When a program accesses a record in one of these modes, RMS establishes a new current position in the file. Programs can then switch to sequential access mode. By using the randomly accessed record (rather than the beginning of the file) as the starting point, programs can retrieve succeeding records in the sequence established by the file's organization.

File Attributes

The logical and physical characteristics of an RMS file are known as its attributes. These characteristics are defined by the source language statements of an application program or by the RMS utility program DEFINE. RMS uses this information about the attributes to structure a file on the storage medium.

The most important attribute of any RMS file is its organization. A file for use in a particular application can be tailored by making the proper selection of this and other attributes. In addition to file organization, the user can specify the following attributes:

- storage medium on which the file resides
- file and protection specification of the file
- format and size of records
- size of the file
- size of a particular storage structure, known as the bucket, within relative and indexed files
- definition of keys for indexed files

Storage Media — Selection of a storage medium on which RMS builds a file is related to the organization of the file. Permanent sequential files can be created on disk devices or ANSI magnetic tape volumes. Transient sequential files can be written on devices such as lineprinters and terminals. Relative and indexed files can reside only on disk devices.

File Specifications — The name assigned to a file enables RMS to find the file on the storage medium. RMS allows for the assignment of a protection specification to a file at the time it is created.

When a file is created, the user must provide the format and maximum size specifications for the records the file will contain. The specified format establishes how each record appears physically in the file on a storage medium. The size specification allows RMS to verify that records written into the file do not exceed the length specified when the file was created.

RMS Record Formats — RMS supports four record formats: fixed, variable, variable-with-fixed-control (VFC), and stream.

Like the selection of a storage medium, the choice of a format for the records of a file depends on a file's organization. Table 7-2 shows the allowed combinations of record format and file organization.

Fixed Length Record Format The term fixed length record format refers to records of a file that must all be one specified size. Each record occupies an identical amount of space in the file.

Variable Length Record Format In variable length record format, records in a file can be either equal or unequal in length. To allow retrieval of variable length records from a file, RMS prefixes a count field to each record it writes. The count field describes the length (in bytes) of the record. RMS removes this count field before it passes a record to the program.

Variable-with-Fixed-Control Record Format Variable-with-fixed-control (VFC) records consist of two distinct parts: the fixed control area and the user data record. The

Table 7-2
File Organizations and Record Formats

File Organization	Fixed Record Format	Variable Record Format	VFC Record Format	Stream Record Format
sequential	yes	yes	yes	yes*
relative*	yes	yes	yes	no
indexed*	yes	yes	no	no

* disk files only

size of the fixed control area is identical for all records of the file. The contents of each fixed control area are completely under the control of the program and can be used for any purpose. As an example, fixed control areas can be used to store the identifier (for example, relative record number or RFA) of related records.

The second part of a VFC record is similar to a variable length record. It is a user data record, variable in length and composed of individual data fields.

Stream Format Records Records in stream format can vary in size. However, no count field precedes each record. Instead, RMS considers the entire file a stream of contiguous ASCII characters. Each record in the file is delimited by a form feed, vertical tab, linefeed, or carriage return immediately followed by a linefeed.

Stream format records are supported for file interchange with non-RMS application programs. Since this format is not very efficient, it should be used only when such interchange is a concern.

Size of Records — The programmer provides RMS with record size information along with the selected record format. How RMS uses this information depends on the record format chosen.

When fixed format records are chosen, the actual size of each record in the file must be indicated. This size specification becomes part of the information stored and maintained by RMS for the file. Thereafter, if a program attempts to write a record whose length differs from this specified size, RMS will reject the operation.

When creating a file with variable length format records, the user can specify a maximum record size greater than zero or, for sequential and indexed files, a maximum record size equal to zero. If the specified size is greater than zero, RMS interprets the value as the size of the largest record that can be written into the file.

VFC format records require two size specifications. The first size identifies the length of the fixed control area of all records in the file; the second size specification represents the maximum length of the data portion of the VFC records. RMS handles this second size specification in a manner similar to its handling of the size specification for variable format records.

For stream format records, RMS permits the user to specify the same record size information as for variable format records. That is, a nonzero value represents the maximum permitted size of any record written in the file, while a zero value suppresses RMS size checking.

Size of RMS Files — The size of an RMS file is expressed as a number of *virtual blocks*. Virtual blocks are physical storage structures. That is, each virtual block in a file is a unit of data whose size depends on the physical medium on which the file resides. For example, the size of virtual blocks in files on disk devices is 512 bytes.

The operating system assigns ascending numbers to a file's virtual blocks. This numbering scheme allows a file to appear as a series of adjacent virtual blocks. In reality, though, the successive numbering of virtual blocks and the physical placement of these blocks on a storage medium need not correspond.

The virtual blocks of a file contain the records that programs write into the file. Depending on the size of records, a virtual block can contain one record, more than one record, or a portion of a record.

When creating an RMS file, users can specify an initial allocation size. If no file size information is given, RMS allocates the minimum amount of storage needed to contain the defined attributes of the file.

Buckets in Relative and Indexed Files — RMS uses a storage structure known as a *bucket* for building and maintaining relative and indexed files. Unlike a virtual block, a bucket can never contain a portion of a record. That is, RMS does not permit records to span bucket boundaries.

The size of buckets in a file is defined at the time the files are created. A large bucket size will serve to increase sequential-mode processing speed of a file, since fewer actual I/O transfers are required to access records. Minimizing bucket size, on the other hand, means that less I/O buffer space is required to support file processing.

Key Definitions for Indexed Files — To define a key for an indexed file, the position and length of character data in the records of the file must be specified. At least one key, the primary key, must be defined for an indexed file. Additionally, up to 254 alternate keys can be defined. Each primary and alternate key represents from one to 255 characters in each record of the file.

When identifying the position and the length of keys to RMS, users can define either simple or segmented keys. A simple key is a single, contiguous string of characters in the record; in other words, a single data field. A segmented key, however, can consist of from two to eight data fields within records. These data fields need not be contiguous, and RMS treats the separate data fields (segments) as a logically contiguous character string.

At file creation, two characteristics for each key can be specified: duplicate key values are allowed and key value can change.

If duplicate key values are allowed, the programmer indicates that more than one record in the file can have the same value in a given key.

The personnel file can serve as an example of the use of duplicate keys. At file creation time, the department-name field could be defined as an alternate key. As programs write records into the file, the alternate index for the department-name key field would contain multiple entries for each key value (for example, PAYROLL, SALES, ADMINISTRATION), since departments are composed of more than one employee. When such duplication occurs, RMS stores the records so that they can be retrieved in first-in/first-out order.

An application could be written to list the names of employees in any particular department. A single execution of the application could list the names of all employees working, for example, in the department called SALES. By randomly accessing the file by alternate key and the key value SALES, the application would obtain the first record written into the file containing this value. Then, the applica-

tion could switch to sequential access and successively obtain records with the same value, SALES, in the alternate key field. Part of the logic of the application would be to determine the point at which a sequentially accessed record no longer contained the value SALES in the alternate key field. The program could then switch back to random access mode and access the first record containing a different value (for example, PAYROLL) in the department-name key field.

The second key characteristic, key value can change, indicates that records can be read and then written back into the file with a modified value in the key. When such modification occurs, the appropriate index is automatically updated to reflect the new key value. This characteristic can be specified only for alternate keys. Further, when specifying this characteristic, the user must also specify that duplicate key values are allowed.

If the sample personnel file were created with the department-name field as an alternate key, the creator of the file would need to specify that key values can change. This allows a program to access a record in the file and change the contents of a department-name data field to reflect the transfer of an employee from one department to another.

The user can also declare the converse of either of these two key characteristics. That is, the user can specify for a given key that duplicate key values are not allowed or that key values cannot change. When duplicate key values are not allowed, RMS rejects any program request to write a record containing a value in the key that is already present in another record. Similarly, when the key value cannot change, RMS does not allow a program to write a record back into the file with a modified value in the key.

Program Operations on RMS Files

After RMS has created a file according to the user's description of file characteristics, a program can access the file to store and retrieve data. The organization of the file determines the types of record operations permitted.

If the record accessing capabilities of RMS are not utilized, programs can access the file as a physical structure, in which case RMS considers the file simply as an array of virtual blocks. To process a file at the physical level, programs use a type of access known as *block I/O*.

Record Operations on RMS Files — The organization of a file, defined when the file is created, determines the types of operations that the program can perform on records. Depending on file organization, RMS permits a program to perform the following record operations:

- *Read a record.* RMS returns an existing record within the file to the program.
- *Write a record.* RMS adds a new record that the program constructs to the file. The new record cannot replace an already existing record.
- *Find a record.* RMS locates an existing record in the file. It does not return the record to the program, but establishes a new current position in the file.
- *Update a record.* The program modifies the contents of a record read from the file. RMS writes the modified record into the file, replacing the old record.

Sequential File Organization Record Operations In sequential file organization, a program can read existing records from the file using sequential or RFA access modes. New records can be added only to the end of the file and only through the use of sequential access mode. The find operation is supported in both sequential and RFA access mode. In sequential access mode, the program can use a find operation to skip records. In RFA access mode, the program can use the find operation to establish a random starting point in the file for sequential read operations. The sequential file organization does not support the delete operation, since the structure of the file requires that records be adjacent in and across virtual blocks. A program can, however, update existing records in disk files as long as the modification of a record does not alter its size.

Relative File Organization Record Operations Relative file organization permits programs greater flexibility in performing record operations than does sequential organization. A program can read existing records from the file using sequential, random, or RFA access mode. New records can be sequentially or randomly written as long as the intended record cell does not already contain a record. Similarly, any access mode can be used to perform a find operation. After a record has been found or read, RMS permits the delete operation. Once a record has been deleted, the record cell is available for a new record. A program can also update records in the file. If the format of the records is variable, update operations can modify record length up to the maximum size specified when the file was created.

Indexed File Organization Record Operations Indexed file organization provides the greatest flexibility in performing record operations. A program can read existing records from the file in sequential, RFA, or random access mode. When reading records in random access mode, the program can choose one of four types of matches that RMS must perform using the program-provided key value. The four types of matches are:

- exact key match
- approximate key match
- generic key match
- approximate and generic key match

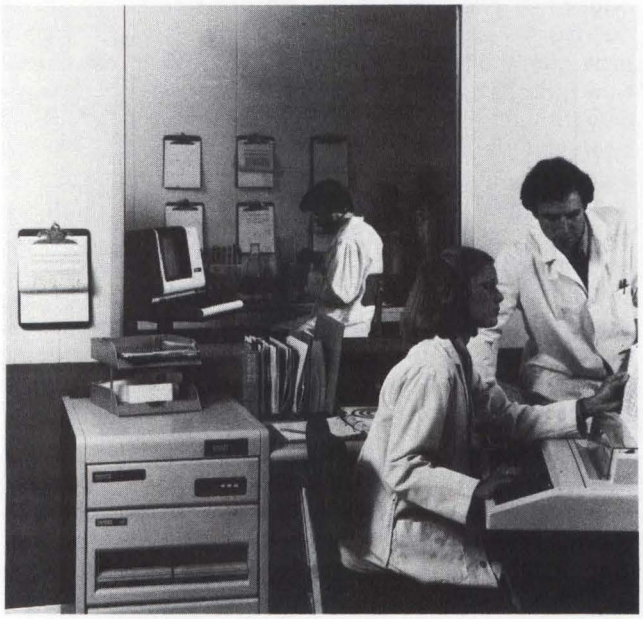
Exact key match requires that the contents of the key in the record retrieved will precisely match the key value specified in the program read operation.

The approximate match facility allows the program to select either of the following relationships between the key of the record retrieved and the key value specified by the program:

- equal to or greater than
- greater than

The advantage of this kind of match is that if the requested key value does not exist in any record of the file, RMS returns the record that contains the next higher key value. This allows the program to retrieve records without knowing an exact key value.

Generic key match means that the program need specify only an initial portion of the key value, thereby forming a logical truncation of the key. RMS returns to the program



the first occurrence of a record whose key contains a value beginning with those characters. This capability is useful in applications where a series of records must be retrieved according to the contents of only a part of the key field. In an indexed inventory file, for example, a company might designate its part numbers in such a way that the first three digits represent the vendor from whom the part is purchased. In order to retrieve the record associated with a particular part, the program would normally supply the entire part number. Generic selection permits the retrieval of the first record representing parts purchased from a specific vendor.

The final type of key match combines both generic and approximate facilities. The program specifies only an initial portion of the key value, as with generic match. Additionally, a program specifies that the key data field of the record retrieved must be either equal to or greater than the program-supplied value, or greater than the program-supplied value.

In addition to versatile read operations, RMS allows any number of new records to be written into an indexed file. It rejects a write operation only if the value contained in a key of the record violated a user-defined key characteristic (for example, duplicate key values not permitted).

The find operation, similar to the read operation, can be performed in sequential, RFA, or random access mode. When finding records in random access mode, the program can specify any one of the four types of key matches provided for read operations.

In addition to read, write, and find operations, the program can delete any record in an indexed file and update any record. The only restriction RMS applies during an update operation is that the contents of the modified record must not violate any user-defined key characteristic (for example, key values cannot change and duplicate key values are not permitted).

Block I/O — Block I/O allows a program to bypass the record processing capabilities of RMS entirely. Rather than performing record operations through the use of supported access modes, a program can process a file as a physical structure consisting solely of virtual blocks.

Using block I/O, a program reads or writes multiple virtual blocks by identifying a starting virtual block number in the file. Regardless of the organization of the file, RMS accesses the identified block or blocks on behalf of the program.

Since RMS files, particularly relative and indexed files, contain internal information meaningful only to RMS itself, DIGITAL does not recommend that a file be modified by using block I/O. The presence of the block I/O facility, however, does permit user-created file structures. The resultant structures must be user-maintained using specialized programs. The structures cannot be accessed using RMS record access mode and record operations.

RMS Runtime Environment

The environment within which a program processes RMS files at runtime consists of two levels: the file processing level and the record processing level.

At the file processing level, RMS and RSX-11M/M-PLUS provide an environment that permits concurrently executing programs to share access to the same file. RMS ascertains the amount of sharing permissible from information provided by the programs themselves. Additionally, at the file processing level, RMS provides facilities that allow programs to minimize buffer space requirements for file processing.

At the record processing level, RMS allows programs to access records in a file through one or more record access streams. Each record access stream represents an independent and simultaneously active series of record operations directed toward the file. Within each stream, programs can perform record operations synchronously or asynchronously. That is, RMS allows programs to choose between receiving control only after a record operation request has been satisfied (synchronous operation) or receiving control before the request has been satisfied (asynchronous operation).

For both synchronous and asynchronous record operations, RMS provides two record transfer modes: move mode and locate mode. Move mode causes RMS to copy a record from an internal RMS I/O buffer into a user buffer. Locate mode allows programs to address records directly in an internal RMS I/O buffer.

File Processing Environment

RMS provides two major facilities at the file processing level: file sharing and buffer handling.

File Sharing — Timely access to critical files requires that more than one program executing concurrently be allowed to process the same file at the same time. Therefore, RMS allows executing programs to share files rather than process files serially. The manner in which a file can be shared depends on the organization of the file. Program-provided information further establishes the degree of sharing of a particular file.

File Organization and Sharing With the exception of magnetic tape files, which cannot be shared, an RMS file can be shared by any number of programs that are read-

ing the file. Under certain circumstances, an RMS file can also be shared by any number of programs that are writing the file.

Sequential files on disk can be accessed by a single writer or shared by multiple readers. Relative and indexed files, however, can be shared by multiple readers and multiple writers.

Program Sharing A file's organization establishes whether it can be shared for reading with a single writer or for multiple readers and writers. A program specifies whether such sharing actually occurs at runtime. The user controls the sharing of a file through information the program provides RMS when it opens the file. First, a program must declare what operations it intends to perform on the file. Second, a program must specify whether other programs can read the file or both read and write the file concurrently with that program.

The combination of these two types of information allows RMS to determine if multiple user programs can access a file at the same time. Whenever a program's sharing information is compatible with the corresponding information another program provides, concurrent access is allowed.

Bucket Locking RMS uses a bucket-locking facility to control operations to a relative or indexed file that is being accessed by one or more writers. The purpose of this facility is to ensure that a program can add, delete, or modify a record in a file without another program's simultaneously accessing the same record.

When a program opens an indexed or relative file with the declared intention of writing or updating records, RMS locks any bucket accessed by the program. This locking prevents another program from accessing any record in the bucket until the program releases it, and remains in effect until the program accesses another bucket. RMS then unlocks the first bucket and locks the second.

Buffer Handling — To a program, record processing under RMS appears as the movement of records directly between a file and the program itself. Transparently to the program, however, RMS reads or writes virtual blocks or buckets of a file into or from internal memory areas known as I/O buffers. Records within these buffers are then made available to the program.

In addition to buffers that contain virtual blocks or buckets, RMS requires a set of internal control structures to support file processing. The combination of these buffers and control structures is known as the space pool.

Record Processing Environment

After opening a file, a program can access records in the file through the RMS record processing environment. This environment provides three facilities: record access streams, synchronous or asynchronous record operations, and record transfer modes.

Record Access Streams — In the record processing environment, a program accesses records in a file through a record access stream, a serial sequence of record operation requests. For example, a program can issue a read request for a particular record, receive the record from RMS, modify the contents of the record, and then issue an update request that causes RMS to write the record back into the file. The sequence of read and update record operation requests can then be performed for a different record, or other record operations can be performed, again in a serial fashion. Thus, within a record access stream, there is at most one record being processed at any time. However, for relative and indexed files, RMS permits a program to establish multiple record access streams for record operations to the same file. The presence of such multiple record access streams allows programs to process in parallel more than one record of a file. Each stream represents an independent and concurrently active sequence of record operations. Further, when such streams update records in the file, RMS employs the same bucket-locking mechanism among streams that it uses to control the sharing of a file among separate programs.

Synchronous and Asynchronous Record Operations — Within each record access stream, a program can perform any record operation either synchronously or asynchronously. When a record operation is performed synchronously, RMS returns control to a program only after the record operation request has been satisfied (for example, a record has been read and passed to one program). When a record operation is performed asynchronously, RMS can return control to one program before the record operation request has been satisfied. A program, then, can utilize the time required for the physical transfer between the file and memory of the block or bucket containing the record to perform other computations. However, a program cannot issue a second record operation through the same stream until the first record operation has completed. To ascertain when a record operation has actually been performed, a program can issue a wait request and regain control when the record operation is complete.

Record Transfer Modes — In addition to specifying synchronous or asynchronous operations for each request in a record access stream, a program can utilize either of two record transfer modes to gain access to each record in memory:

- *Move mode record transfers.* RMS permits move mode record operations for all file organizations and record operations. Move mode requires that an individual record be copied between the I/O buffer and a program. For read operations, RMS reads a block or bucket into an I/O buffer, finds the desired record within the buffer, and moves the record to a program-specified location.

  Before a write or update operation in move mode, the program builds or modifies a record in its own work space. Then the program issues a write or update record operation request, and RMS moves the record to an I/O buffer.

- *Locate mode record transfers.* RMS supports locate mode record transfers for read operations to all file organizations. However, it permits locate mode on write operations for sequential files only.

  Locate mode reduces the amount of data movement, thereby saving processing time. This mode enables programs to access records directly in an I/O buffer. Therefore, there is normally no need for RMS to copy records from the I/O buffer to a program. To allow the program

to access a record in the I/O buffer, RMS provides the program with the address and size of the record in the I/O buffer.

DATATRIEVE-11

DATATRIEVE-11 is an optional data maintenance inquiry language and report writing system. It gives users direct, fast, and easy access to the data in sequential, relative, and indexed Record Management Services (RMS) files. DATATRIEVE-11's substantial help facilities ensure that all users can take advantage of its capabilities.

DATATRIEVE-11 accepts simple words and phrases to extract, modify, and update RMS data. With fewer than ten commands, users can find, print, update, and sort records. In addition, DATATRIEVE's advanced command forms can save both time and keystrokes for more sophisticated users.

There are many advantages to using DATATRIEVE-11 over application programs to generate ad hoc queries and reports.

- Requests can be tailored interactively to meet the user's specific needs.
- Sessions are shorter with less coding time.
- Time-consuming compilations are not needed.
- Users can respond immediately to errors or unexpected variations in results at execution time.

Because DATATRIEVE-11 is so easy to learn and simple to use, all users can access data without the services of a programmer. So, by eliminating the need for many specialized application programs and their time-consuming compilations, DATATRIEVE-11 helps to maximize both programmer and system productivity.

The three major categories of DATATRIEVE-11 capabilities are:

- data access and update facilities
- report generation facilities
- Data Dictionary

Data Access and Update Facilities

With DATATRIEVE's inquiry and update commands, users can perform record and file manipulation. DATATRIEVE offers both simple and advanced commands. Novices can use the simple commands to find, update, and sort records. More experienced users can use the advanced commands to perform more complex functions, such as combining commands to form procedures.

DATATRIEVE-11's flexible "value-based" data access/update capabilities can eliminate much programming overhead in many ad hoc situations. Information is returned to the user in the form of collections of records that can be manipulated and/or displayed on the terminal or printer using the DATATRIEVE-11 report writing facility.

The following features make DATATRIEVE easy to learn and use.

- GUIDE MODE is a tutorial aid with automatic prompting. This permits the novice to retrieve and display data by stepping through a subset of commands.
- The documentation set for DATATRIEVE-11 includes a *Beginner's Primer* designed to introduce the novice to DATATRIEVE-11, and a *User's Guide* that uses examples to present the various DATATRIEVE-11 functions.
- The commands are simple words and phrases instead of confusing acronyms.
- Simple arrays are supported.
- A data type is provided that recognizes data formats and facilities, entering and displaying dates in any one of several formats.
- DATATRIEVE-11 provides a full set of arithmetic operators (addition, subtraction, multiplication, division, and negation), statistical operators (total, average, maximum, minimum, and count), and conversion between data types used in DIGITAL's FORTRAN, COBOL, and BASIC-PLUS-2 languages.

Report Generation

DATATRIEVE-11 also provides a report writing facility to generate reports from RMS files. The data can come directly from the files or can be preselected and manipulated through a series of DATATRIEVE-11 commands. Users can specify such parameters as spacing, titles, headings, and totals on their reports. As in the inquiry and update facility, errors in commands are discovered immediately, which avoids printing wrong or incomplete reports.

Data Dictionary

The Data Dictionary maintains definitions of record structures and domain names. A record structure describes the format of the records in a file. A domain is a named group of data containing records of a single type. Record structures and domain names must be defined before DATATRIEVE-11 can be used to access data.

The definitions provide a substantial level of data and program independence because the record definitions (or views) can cross file boundaries. Thus, by providing a single value-based DATATRIEVE-11 query, users can access information from multiple files and records. DATATRIEVE-11 also provides commands to list the contents of the Data Dictionary, to delete entries, and to control access to individual entries.

Data Protection — Data protection is accomplished through two independent mechanisms: the protection systems of the RSX-11 operating system and those within DATATRIEVE-11. The DATATRIEVE protection system uses passwords and User Identification Codes (UICs) to allow a user to regulate access to domains, records, procedures, and tables through access requirements recorded in the Data Dictionary. Thus, each resource has its own security system to ensure that access is not granted to unauthorized users.

Data Input Validation — DATATRIEVE-11 provides the ability to encode and decode data and validate input through tables stored in the Data Dictionary. Validation criteria can also be stored in the record definition to verify the accuracy of input. DATATRIEVE automatically reprompts during data entry if an input error is detected.

Procedures — Procedures, which are groups of frequently used statements or commands, can be stored in the Data Dictionary. After storing a procedure in the Data Dictionary, a user can execute a sequence of commands by calling the procedure name, rather than entering the entire sequence manually. Procedures can be defined by a user for personal use, or can be made available to all users. Defining system-wide procedures not only saves typing, but also provides a means to establish and distribute complicated sequences of commands to all users. Procedures can be edited online, allowing for application development.

View Facility — By using the DATATRIEVE-11 view capability users can define logical records that cross file boundaries. These view definitions are stored in the Data Dictionary and are used by DATATRIEVE to provide a relational data access capability.

FMS-11 FORMS MANAGEMENT SYSTEM

FMS-11 is an optional set of software tools that provides a screen management front-end for applications using VT100-series video terminals. Forms serve as the interactive interface between a terminal user and an application program. With FMS-11, programmers can create applications that use one or more displayed forms to handle user inquiry/response operations.

FMS-11 makes it easy to use the distinctive video attributes of the VT100 series: reverse video, bold, blink, underline, 132-column lines, jump or smooth scrolling, and split or reverse screen. Character data types within fields (pictures) are checked on a character-by-character basis. And special symbols used for formatting can be embedded within a field without breaking the field into smaller fields.

Programmers can use FMS-11 to develop application programs in MACRO and in most high-level languages on RSX-11M and RSX-11M-PLUS as well as on DIGITAL's RSTS/E, RT-11 and VAX/VMS operating systems. Programs can be coded to be completely independent of the forms layout, since form and field names are not bound to the program until execution time.

FMS features include:

- Use of VT100-series terminal features, including reverse video, bold, blink, and underline characteristics; and split-screen and scrolling capabilities.
- HELP for forms as well as for fields within forms, to provide immediate assistance to users without complicating the development process.
- Extensive field protection and validation features to ensure the integrity of data returned to the application program.
- The ability to design forms directly and interactively on the video screen, eliminating the need to lay out the form on paper, code form-language statements, compile, debug, and enter corrections.
- The design and storage of forms independent of the application program. The individual field descriptions can be modified without having to reconstruct the form, recompile, relink the application, or reprocess collected data.
- The Form Driver, which provides a wide range of terminal I/O functions including both field and record level I/O calls and flexible manipulation of scrolled regions on the screen.

FMS-11 Forms

FMS-11 forms include a video screen image that comprises data fields and constant background text and protection and validation information for individual data fields. The data fields and background text can be highlighted by using the VT100 video attributes. Split-screen and scrolling capabilities mean more data can be viewed than can be displayed on a screen at one time. A scrolled area provides a window into an amount of data that is too large to appear on the screen at one time, thereby allowing applications to create and use forms of unlimited length.

Individual data fields can be display-only, enter-only (no echo), or can be restricted to modification by privileged users. Data fields can be formatted with fill characters, default values, and formatting characters (such as a dash in a phone number), all of which assist the user, but are not visible to the application program. Fields can be right- or left-justified or can use a special fixed-decimal field type to align data properly.

Field validation includes checking each keystroke in a field for the proper data type: alphanumeric, alphabetic, numeric, signed numeric, or any character. Fields can also be defined as "must enter" or "must complete."

HELP

A line of HELP information can be associated with each field, and a chain of HELP screens can be associated with each form. If users need HELP while using a form, they can press the HELP key to view a line of information pertinent to the current field. Subsequent key depressions will yield more HELP information.

FMS-11 Components

The FMS-11 software includes three components used for developing and executing form applications programs:

- *Form Editor*—a program used to design, create, and modify forms directly on the terminal screen
- *Form Utility*—a program used to create memory-resident forms and to help debug and maintain form descriptions
- *Form Driver*—a set of reentrant subroutines called by application programs to control the user's use of the form

Form Editor — The Form Editor is an interactive program that uses many of the special capabilities of the VT100 video terminal as a means of entering and modifying FMS-11 form descriptions and storing them in forms files. Users can specify video display characteristics for both constant text and field characters by using the keypad and keyboard functions.

When the Form Editor is used, the user's screen always shows the current state of the form that is being edited. Fields are defined on the screen with COBOL-like picture characters. As an aid to users, the form description can contain brief, helpful explanations about individual fields and about each form as a whole.

Fields and forms are accessed by name. Because the relationship between fields and programs is determined at execution time, users can change the form's design without having to change the application program that uses it.

When creating a form, the designer can define a number of attributes that control how the user should enter data into the form. The designer can also define attributes that specify how the application program is to use the data. The designer can assign these attributes at any time during the editing session by filling in appropriate questionnaire forms on the video screen. The four categories of attributes are:

- *form-wide attributes*—characteristics that affect the form as a whole, such as its name, where it will be located on the screen, and the name of any related help form.
- *field attributes*—characteristics that describe a form's individual data fields and how they will be set up and used; for example, whether a field will be right-justified or zero-filled and whether any help text is available.
- *named data*—constant data that are not displayed for the user's benefit, but that the application program can access; for example, the name of another form or a range of values against which the user's input is checked. By using named data, the programmer can write a more general, more maintainable application.
- *display attributes*—information determining how areas of the screen (both text and data fields) are to be displayed: blank, bold, underline, reverse video, or no display.

Form Utility — The Form Utility is used to create versions of form descriptions that are suitable for hardcopy listings, for creating and modifying form libraries, for listing names of forms contained in the form library, and for producing object modules of form descriptions. In addition, the Form Utility generates COBOL data division code that corresponds to a form definition and that is suitable for copying into a COBOL source program.

Form Driver — The Form Driver, a set of subroutines, permits an application program to access form descriptions created by the Form Editor. Form-Driver calls embedded in a task of an application program and written in the source code of the task invoke the form description from the form-library file. All Form-Driver calls refer to specific forms and/or fields within forms by specifying the form name assigned during the form's creation process. Under the direction of the calling program, the Form Driver displays forms, performs all screen management the forms require, handles all terminal I/O for application programs, and validates each user response by checking it against the field and form descriptions. In response to a HELP request, the Form Driver displays the appropriate help text associated with the form and field being processed.

SORT-11

The optional SORT-11 utility program allows users to reorder data from any input file into a new file in either ascending or descending sequence based upon control or key

fields within the input data records themselves. In addition to running under RSX-11M/M-PLUS, SORT-11 runs under all other PDP-11 operating system that include RMS.

If users do not wish to sort the data in a particular file, SORT can still be used to extract key information, sort that information, and store the sorted information in another permanent file. Later that file can be used to access the data in the order of the key information in the sorted file. The contents of the sorted file may be entire records, key fields, or record indexes relative to the position of each record within the file (the first record on the database is record 1, the second, 2, and so on). SORT provides four sorting techniques, which are described later in this section.

The SORT utility program can be controlled by a command string and an optional specification file. There is a simple format for each. If the SORT application does not require that records be restructured or that only a subset of the input file be sorted, then only a command string is needed to control SORT.

Data Files

SORT can accept a file from any one of the peripheral devices available in the system configuration: disk units, magtape units, or terminals.

A record is usually divided into several logical areas called data fields. The data in each field may or may not be relevant to SORT. SORT uses record identifiers to distinguish the various types of records in a file, while it uses the key fields in each record to reorder an input file. The key fields may be any one of a number of different data types, including character, zoned decimal, two's-complement binary, and two- or four-word floating point. Any other data field in a record may be retained in the output file or ignored.

Command String and Specification File

The user can direct the SORT program by entering a command string, which serves three functions:

- references devices in the system for each file in the current sort
- specifies switches that define file parameters used in the sorting process
- references a specification file or specifies other switches to control the sort

Several command string switches define the sorting process parameters. One switch describes record formats and the maximum record size. Another delimits the internal work files. Others provide detailed file information to RMS.

Normally, the sort must be directed with a specification file, but two additional switches can be used instead. The first specifies the sorting process option; the second identifies the key fields. The use of these switches is limited to sorting an input file of uniform format. The key fields must reside in the same location in every record of the input file. And the file must contain only the records to be included in the sort. Figure 7-7 illustrates a general sort that would require only a command string and switches.

The specification file is the supplement to the command string, which provides the basis for controlling and directing the sorting process.

The specification file provides a variety of controlling features, which are listed and illustrated below.

Record Selection — Users can include or omit any records from the sorting process. The output file will contain only the specified records. Figure 7-8 illustrates this type of sort.

Alternate Collating Sequence — If necessary, users can specify an alternate collating sequence. The normal sequence is that implied in ASCII code. One alternate choice is EBCDIC values. The other is an individual alternate collating sequence (ALTSEQ). An ALTSEQ can be used to change the ASCII values of the normal sequence. It applies to all the alphanumeric key data in the records, but only during the actual sorting process. The output record remains unchanged. See Figure 7-9 for an example of this type of sort.

Figure 7-7
Sort Using Command String and Switches

Figure 7-8
Record Selection

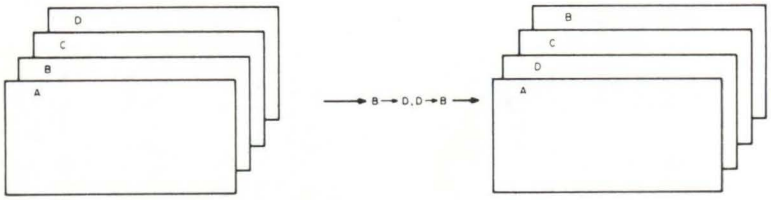


Figure 7-9
Alternate Collating Sequence

Forced Keys — An ALTSEQ applies to all positions of the key. Forced keys allow the user to specify an alternate sequence for particular positions within the key. An alternate can be specified by substituting a lower-valued character, such as the slash (/) in Figure 7-10 below. Since the slash comes before 0, the 300-series records in the example are brought to the front of the file. Notice that the records so treated are in sequence and in front of the rest of the sorted file. The net effect is that of two sorted files, one behind the other.

Input Format Variation — If the input file contains records with several different formats, the user can identify those records by type so that they may be properly handled. Note that only one type may be selected and sorted per run.

In Figure 7-11, A and N are record identifiers.

Output Format Variation — Users can change the format of the data file during the sort, but they cannot change the contents of any given data item. Figure 7-12 demonstrates this point.

Sort Operation

The SORT program consists of two basic parts: a control program and a subroutine package called SORTS. The control program directs the overall processing, while SORTS serves as a collection of subroutines available to the control program during its processing. The subroutine package can be invoked from a user-written program. This is supported in most PDP-11 programming languages.

There are three phases of operation in the SORT control program. In the first phase, SORT reads the command string, decodes it, and stores the switch values and the specification file, if present. Any errors in the command string or specification file are reported at this point.

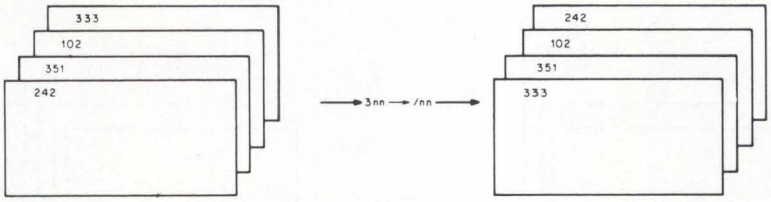


Figure 7-10
Forced Key

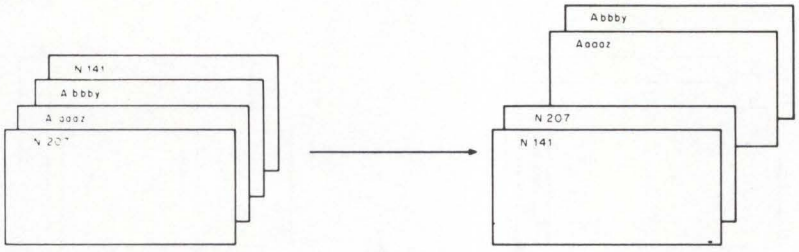


Figure 7-11
Input Format Variation

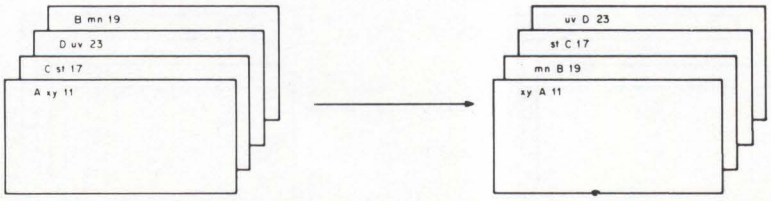


Figure 7-12
Output Format Variation

Phase two begins the presort operation. The control program is called to open and read the input file and establish the keys. The SORTS subroutine begins the initial sorting process. At this point, the amount of available internal storage space becomes important to the efficiency of the sort. If that space is not sufficient to hold all the records, SORT builds strings of sorted records and transfers them to scratch files on bulk storage devices. In order to merge these files and complete the sort, space for at least three scratch files must be available. The SORT program normally provides for a maximum of eight scratch files. Either a switch in the command string or the amount of available internal workspace can reduce the number of scratch files used.

The final merge phase rebuilds the intermediate scratch files into a merged file. Another subroutine reads the records in the proper sequence. The records are then written into the output file. If there are no scratch files to merge because main memory was sufficient to hold all the records, the sorted records are written directly into the output file. After the last record is written, the control program cleans up the scratch files and returns to the first phase; SORT is then ready to accept another job.

SORT Processing Options

SORT offers four processing options. Table 7-3 highlights how the sorting processes differ.

Record Sort (SORTR) — SORTR outputs all specified record data in a sorted sequence. Each record is kept intact throughout the entire sorting process. Since it moves the whole record, SORTR is relatively slow and may require considerable main memory or external storage workspace for large files.

Tag Sort (SORTT) — SORTT produces the same kind of output file as SORTR, but it handles only record pointers and key fields. Since SORTT moves a smaller amount of data than SORTR, SORTT usually performs a faster sort than SORTR. The input file must be randomly reaccessed to create the entire output file, which can be a lengthy process for large files.

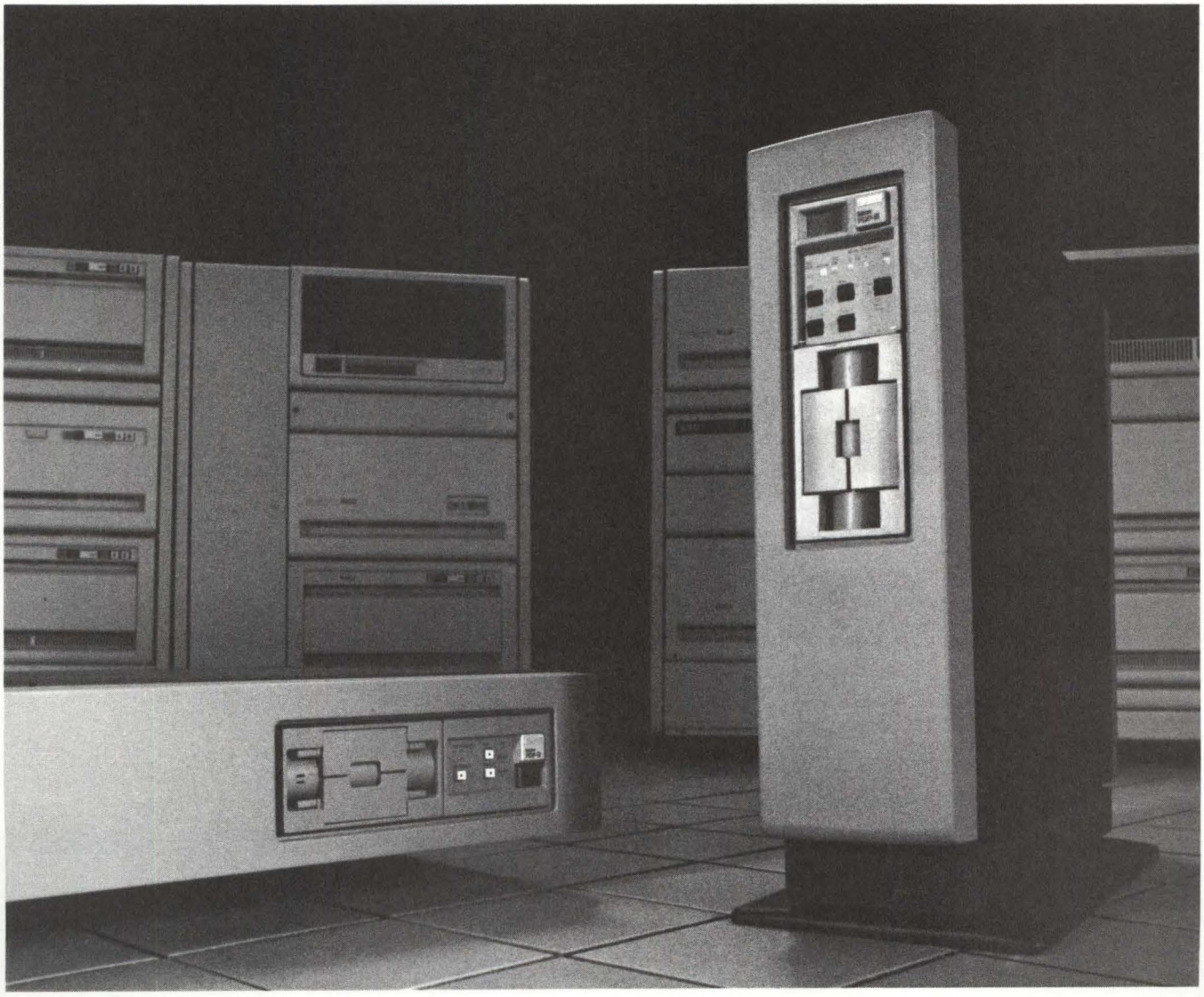
Address Routing Sort (SORTA) — SORTA produces address routing files, which consist of relative record pointers, beginning at one, in binary words. These files can be used as a special index file to access randomly the data in the original file. It is possible to maintain only one data file, but several different index files may be needed. Like SORTT, SORTA uses the minimum amount of data necessary in the sorting process. Once the input phase is completed, the input file is not read again. The output data are in a restricted mode. This means that SORTA is the fastest sorting method in the sort package.

Index Sort (SORTI) — SORTI produces an index file consisting of relative record pointers, as in SORTA, and index keys. This makes it slightly slower than SORTA. During processing, SORTI handles only the relative record pointers and two forms of the key fields. One form is used for sorting and the other is left as it was in the original data.

Table 7-3
Sorting Process Options

Type of SORT	Type of File	Record Size and Format	Speed	Device
Record Sort (SORTR)	Input and Output	Any	Slowest	Any
Tag Sort (SORTT)	Input Output	Any Any	Slow for large file	Disk Any
Address Routing Sort (SORTA)	Input Output	Any Fixed, 6 bytes	Fastest	Disk Any
Index Sort (SORTI)	Input Output	Any Fixed, 6-byte pointer plus original key	Fast	Disk Any

8 The Processors



The RSX-11 operating systems run on a wide variety of DIGITAL's interactive minicomputers and microcomputers. Users can choose the PDP-11 system that best meets their application needs.

The LSI-11 and PDP-11 processors, on which the PDP-11 systems are built, are largely compatible, so they provide an upward migration path for growing application needs. And they can be used together in distributed processing networks. The basic differences between PDP-11 minicomputers—which are based on PDP-11 processors—and PDP-11 microcomputers—which are based on LSI-11 processors—are in the implementation of the circuit design and in the bus structure.

PDP-11 processors offer a comprehensive instruction set; an integral Memory Management Unit that provides hardware memory protection through various operating modes and access to extended memory; an optional Floating-Point Unit; and, on some processors, a fast cache memory.

The LSI-11 microprocessors, while built with highly reliable large-scale integrated (LSI) circuits, are true PDP-11s. Based on the PDP-11 architecture, they use the same instruction set and operating systems and they support many of the same peripherals as do the PDP-11 processors. When an LSI-11 central processor board is packaged with a backplane, powersupply, memory, memory management, and interfaces, it provides many of the same capabilities as the larger PDP-11s.

INTRODUCTION

The RSX-11 family of operating systems can be used with DIGITAL's PDP-11 and LSI-11 processors. Instruction-set compatibility means that users with LSI-11-based microcomputers can easily upgrade to larger PDP-11 minicomputer systems. And, in a network, applications developed on PDP-11 minicomputers can run on the smaller LSI-11-based microcomputers. It is in the implementation of the circuit design and in the bus structure that the PDP-11 minicomputers and PDP-11 microcomputers primarily differ.

The PDP-11 processors are part of the PDP-11/24, PDP-11/34A, PDP-11/44, and PDP-11/70 minicomputer systems. The LSI-11 processors are the basis for the PDP-11/23, PDP-11/23-PLUS, and MICRO/PDP-11 microcomputer systems.

This section discusses the major features of the processors that run RSX-11. For more detailed information, ask your sales representative for the *PDP-11 Processor Handbook* and the *Microcomputers and Memories Handbook.*

PROCESSOR COMPONENTS

The integrated physical components of both the PDP-11 and LSI-11 processors are:

- the instruction set, an Extended Instruction Set, and, for some processors, an optional Commercial Instruction Set
- Central Processing Unit
- main memory with Memory Management Unit and, on some processors, cache memory
- the optional Floating-Point Unit
- the front console
- the peripheral controller interfaces: the LSI-11 bus on the LSI-11-based microcomputers; the UNIBUS on the PDP-11 minicomputers; and the MASSBUS on the PDP-11/70

INSTRUCTION SET

The PDP-11 instruction set and addressing modes, common to the PDP-11 and LSI-11 processors, produce over 400 unique instructions. The instruction set offers a wide choice of operations so that a single instruction frequently accomplishes a task that would require several on another computer. These instructions allow byte and word addressing in both single- and double-operand formats, to save memory space and to simplify the implementation of user applications.

Even though there are numerous instructions, the instruction set is a natural programming language that is easy to learn. Some of the instructions correspond directly to high-level-language statements, and the assembler mnemonics are readily associated with the instruction function. The instruction set contains a full set of conditional branches that eliminate excessive use of jump instructions—again saving memory space.

PDP-11 instructions fall into six functional categories.

- *Single-Operand* instructions specify the operation to be performed (the opcode) and provide information for locating the operand.
- *Double-Operand* instructions specify the operation to be performed and provide information for locating two operands. The format of most double-operand instructions is similar to that of single-operand instructions, except that it has two fields for locating operands—the source field and the destination field. Each field is divided into addressing mode and selected register. The mode and register used by one field can be completely different from the mode and register used by another field.
- *Branch* instructions control the operation to be performed next: this based on current processor status.
- *Jump and Subroutine Call* instructions have an opcode and address part and, in the case of the Jump-to-Subroutine (JSR) instruction, a register for linkage. The Return-from-Subroutine (RTS) instruction uses the link to return control to the main program, once the subroutine is finished.
- *Trap* instructions provide a means of interrupting normal program execution to perform special processing. Typical uses of trap instructions are calling the monitor or runtime system, performing error processing, or calling a debugger.

  In addition, user code can include trap instructions to be handled by user routines. Whether the trap instructions are handled by the system or by the user program, program flow can be returned to the point where the trap occurred or it can be changed altogether, including terminating the program.
- *Miscellaneous* instructions such as HALT and RESET control the central processor and are restricted to use in Kernel (system) mode. Only system code can use these instructions, so the integrity of the system is maintained.

Condition code bits indicate a negative condition (N), a zero condition (Z), an overflow condition (V), or a carry condition (C). These four bits are part of the processor status word, described below. The result of any single- or double-operand instruction can affect the condition code bits. Condition code bits can be checked explicitly and with branch instructions to determine the result of an operation. The condition codes are also used by various software modules to check software conditions.

Extended Instruction Set

The Extended Instruction Set (EIS), an integral part of the processor, adds four extra instructions to the basic instruction set. This feature implements the hardware facilities necessary for executing integer multiply/divide and multiple arithmetic shifts for both single- and double-precision computation.

Commercial Instruction Set

The Commercial Instruction Set (CIS), an optional instruction set for the MICRO/PDP-11, PDP-11/23-PLUS, PDP-11/24, and PDP-11/44, comprises ten instructions that are especially useful in commercial data processing and text processing applications. DIGITAL's COBOL implementations for the PDP-11 use the Commercial Instruction Set to increase both compile and execution speeds.

CIS includes instructions that operate on character strings and decimal numbers and that load operands into the processor's general registers.

- *Character-string instructions* move and search character, character-string, and character-set data types. Character-string manipulation is the most commonly used function in commercial data processing and text processing applications.
- *Decimal-string instructions* manipulate strings of decimal data. Several numeric (byte) and packed-decimal data types are supported. Instructions are included for basic arithmetic operations and for compare, shift, and convert functions.
- *Commercial load descriptor instructions* augment the character-string and decimal-string instructions. They load the general registers with two- or three-string descriptors, making the setup of character-string and decimal-string instructions easier.

Each character-string and decimal-string instruction has two forms: a register instruction form and an inline instruction form. The essential difference between the two forms is delivery of operands to the instruction.

A register instruction obtains operands from the processor's general registers. Character-string operands include character, character-string-descriptor, character-set-descriptor, and translation-table-address. Decimal-string operands include decimal-string descriptors, long binary integers, and shift-descriptor words.

An inline instruction finds operands or pointers to operands in the instruction stream immediately following the opcode word. Operands that appear directly in the character-string instruction stream include characters, translation-table addresses, and shift-descriptor words. Decimal-string operands include decimal-string descriptors and long binary integers.

CENTRAL PROCESSOR

The Central Processing Unit (CPU) contains arithmetic and control logic for a wide range of operations. These include high-speed fixed-point arithmetic with hardware multiply and divide (optional with EIS), extensive test and branch operations, and other control operations. The CPU also provides room for the addition of the high-speed Floating-Point Unit and the Commercial Instruction Set (CIS) used on the MICRO/PDP-11, PDP-11/23-PLUS, PDP-11/24, and PDP-11/44.

The LSI-11 central processor unit is implemented using two LSI chips—control and data. The Memory Management Unit, Commercial Instruction Set, and Floating-Point Unit are implemented on LSI chips as well.

Operating Modes

The CPU provides the basis for a fully protected multiprogramming environment through access modes. Depending on the processor, two or three processor access modes are recognized—Kernel, Supervisor, and User. Access modes provide two levels of privileges under which programs can run. Kernel mode is used by the RSX-11 operating system; User mode and, on RSX-11M-PLUS, Supervisor mode are used by user programs.

Programs operating in Kernel mode have complete control of the processor, including the capability to execute HALT and RESET instructions (which cannot be done while in User or Supervisor mode). And only in Kernel mode can the processor execute instructions that access the internal processor registers that control memory management or interrupt processing, unless allowed by the operating system. This protects user jobs.

In general, code executing in Kernel mode can protect itself and any portion of its data structures from read and/or write access by code executing in User and Supervisor mode. Routines that run in Kernel mode are generally part of the runtime operating system software and thus should not be corrupted by other programs. RSX-11 uses the processor's Kernel mode for the resident Executive, interrupt service routines, and device handlers.

Supervisor mode can be used for the mapping and execution of user-sharable programs that still require hardware protection. This could include command interpreters, logical I/O processors, and parts of the runtime systems. The PDP-11/44 and PDP-11/70 processors offer Supervisor mode.

Routines that run in User mode are generally part of application programs. RSX-11 uses the processor's User mode for the system utility programs and applications programs and their completion routines.

The memory protection afforded by User, Supervisor, and Kernel operating modes provides the basis for system data structure integrity.

General Registers and Addressing Modes

Generally, the CPU contains eight general registers. Some processors contain more than eight, although only eight are available at a time. The eight general registers can be used as:

- *accumulators* that hold data to be manipulated.
- *pointers* that designate the contents of the register as the address of the data (operand address), rather than the operand itself.
- *index registers* that permit the contents of the register to be added to a word from an instruction, thus producing the address of the operand. This capability allows easy access to variable entries in a list.

Two registers have special significance depending on the instruction being executed. These special registers are:

- the *hardware Stack Pointer (SP)* that keeps track of the last item added to the stack
- the *Program Counter (PC)* that contains the address of the next word to be processed in the instruction stream

Special addressing mode combinations permit temporary data storage for convenient dynamic handling of frequently accessed data. This is known as stack addressing.

Four direct and four indirect (deferred) addressing modes are provided.

- *Register mode* provides the fastest instruction execution. There is no need to reference memory to retrieve an operand. Any of the general registers can be specified, although changing the PC must be done with care. The operand is contained in the selected register (low order, byte-for-byte operations).
- In *register-deferred mode* the address of the operand is stored in a general purpose register. The address contained in the general purpose register directs the CPU to

the operand. The operand is located either in memory or in an I/O register. This mode is used for sequential lists, indirect pointers in data structures, stack manipulations, and jump tables.

- In *autoincrement mode* the register contains the address of the operand. The address is automatically incremented after the operand is retrieved. The address then references the next sequential operand. This mode allows automatic stepping through a list or series of operands stored in consecutive locations. The address stored in the register is incremented by one, if byte instructions are used, and by two, if word instructions are used. SP and PC are always incremented by two.
- In *autoincrement-deferred mode* the register contains a pointer to an address. The pointer is incremented by two (for both word and byte operations) after the address is located. Autoincrement is used only to access operands that are stored in consecutive locations. Autoincrement-deferred is used to access lists of operands stored anywhere in the system; that is, the operands do not have to reside in adjoining locations. Autoincrement is used to step through a table of operands. Autoincrement-deferred is used to step through a table of addresses.
- In *autodecrement mode* the register contains an address that is automatically decremented. The address is decremented by two (for both word and byte operations) and is then used to locate an operand. This mode is similar to autoincrement mode, but it allows stepping through a list of words or bytes in reverse order. The address is decremented by one for bytes and by two for words. SP and PC are always decremented by two.
- In *autodecrement-deferred mode* the register contains a pointer. The pointer is first decremented by two (for both word and byte operations); then, the new pointer is used to retrieve an address operand stored outside the CPU. This mode is similar to autoincrement-deferred mode, but it allows stepping through a table of addresses in reverse order. Each address then redirects the CPU to an operand. Operands do not have to reside in consecutive locations.
- In *index mode* a base address is added to an index word to produce the effective address of an operand. The base address specifies the starting location of the table or list. The index word then represents the address of an entry in the table or list, relative to the starting (base) address. The base address can be stored in a register, in which case the index word follows the current instruction. Alternatively, the locations of the base address and index word can be reversed (index word in the register, base address following the current instruction).
- In *index deferred mode* a base address is added to an index word. The result is a pointer to an address, rather than the actual address. This mode is similar to index mode, except that it produces a pointer to an address. The content of that address then redirects the CPU to the desired operand. Index-deferred mode provides for the random access of operands using a table of operand addresses.

Processor Status Word

The Processor Status Word is a special register that contains information on the current status of the processor. The information includes current processor priority; current and previous operational modes; condition codes that describe the results of the last CPU instruction; an indicator for detecting the execution of an instruction to be trapped during program debugging; and, on the MICRO/PDP-11, PDP-11/23-PLUS, PDP-11/24, and PDP-11/44, an indicator that a commercial instruction is in process.

Hardware Interrupts

The CPU provides full-vectored interrupts that eliminate polling. Each interrupt is assigned a priority to allow programs with higher priority to interrupt programs with lower priority.

CPU Priority Levels

The CPU, depending on the model, can be set to up to eight priority levels—four hardware and four software levels—under software control. Priority levels allow the CPU not to service devices of lower priority until more critical functions are completed. A high-priority device can request an interrupt to gain control of the bus. The device then can use the instruction set to manipulate data and status registers. The device-servicing routine interrupts the task being performed by the processor. After the device request is satisfied, the processor returns to its former task.

MEMORY

The MICRO/PDP-11, PDP-11/23-PLUS, PDP-11/24, and PDP-11/34A have parity MOS (Metallic Oxide Semiconductor) memory; the PDP-11/44 and PDP-11/70 have ECC (Error-Correcting Code) MOS memory.

Parity Memory

Parity memory, used in main memory and cache memory, ensures both the integrity of data storage and transfer and the reliability of system operation. Cache memory and main memory have byte parity. Parity is generated and checked on all transfers between main memory and cache, between cache and the CPU, and between high-speed mass storage devices and their controllers. RSX-11 automatically logs parity errors, handles recovery from errors, and provides information on system reliability and performance.

ECC Memory

Error-Correcting Code (ECC) is a technique for checking the contents of memory in order to detect errors and correct them before the data is sent to the processor. Checking is done by combining the bits in a number of unique ways so that error-correction bits are generated for each unique combination and stored along with the data bits in the same word as the data. The memory word length is extended to store these unique bits.

When memory is read, the data word is checked against the correction bits stored with the word. If they match, the word is sent on to the processor. If they do not match, an error exists, and the mismatch of the correction bits determines which data bit is in error. The bit in error is then corrected and sent on to the processor.

The ECC code used in MOS memory detects and corrects single-bit errors and detects double-bit errors in a word. If a double-bit error is detected, the processor is notified, as happens with a parity error.

Battery Backup

Because MOS memory is volatile, it cannot retain data without proper dc voltages being applied. In case of temporary power interruption, a battery backup unit helps retain MOS memory data for a short time. This battery is charged by the main ac power during normal system operation. Battery backup is optional on the PDP-11/24, PDP-11/34A, and PDP-11/44 processors and standard on the PDP-11/70.

Memory Management

The Memory Management Unit (MMU) provides the hardware facilities necessary for complete memory management and process protection. It enables programs to access extended memory by mapping their virtual addresses to physical locations in memory. See Section 3 for more information on memory management.

Virtual memory address space consists of all possible 16-bit addresses that can be exchanged between a program and the processor to identify a byte location in physical memory. The memory management hardware translates a virtual address into a physical address. The PDP-11/34A provides 18-bit physical addressing capabilities; the MICRO/PDP-11, PDP-11/23-PLUS, PDP-11/44, and PDP-11/70 provide 22-bit physical addressing. An extended 22-bit memory addressing option is available for the PDP-11/24, which provides 18-bit physical addressing capabilities as a standard feature.

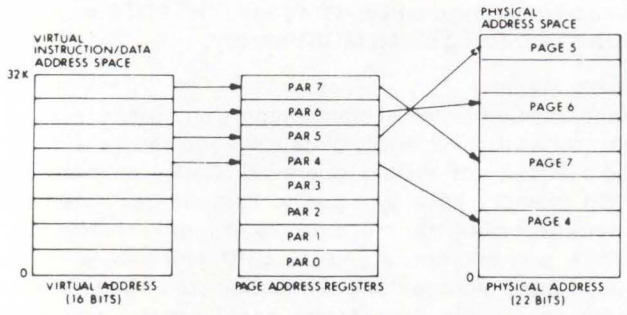


Figure 8-1
Virtual Address Mapping into Physical Address

A physical address is the address exchanged between the processor and memory and between the processor and peripheral adapters. Physical address space is the set of all possible physical addresses the processor can use to express unique memory locations and peripheral control registers.

When the MMU converts a 16-bit virtual address to an 18-bit or 22-bit physical address, it relocates the virtual address. This means that two or more programs can have the same virtual addresses, but different physical addresses. The MMU divides the 64 Kbytes of virtual address space into eight sections called pages. It assigns a page to a section of physical memory. Since the MMU relocates each page of virtual address space separately, a program can reside in disjointed sections of memory.

In addition to its primary function of managing the address space, the MMU also handles the User, Supervisor, and Kernel operating modes described above.

Cache Memory

Cache memory is a high-speed memory that buffers data between the processor and main memory. Main memory and cache appear as a single unit of memory to programs. Cache memory maintains a copy of portions of main memory, for fast access of instructions and data. For some programs, information needed by the CPU can be found in the cache 80 to 95 percent of the time. This can effectively halve execution time and significantly improve system performance.

Whenever a request is made to fetch data from memory, cache memory is checked first. If the data is in cache, it is fetched from there, without reading main memory. If the data is not in cache memory, data is fetched from main memory and stored in the cache, and the requested word or byte is passed directly to the CPU. When data is written, it is automatically stored in both cache and main memory, ensuring that both memories are updated.

Cache memory is standard on the PDP-11/44 and PDP-11/70. It is available as on option for the PDP-11/34A processor.

FLOATING-POINT UNIT

The optional Floating-Point Unit (FPU) performs all floating-point arithmetic operations and converts data between integer and floating-point formats. This can provide significant performance improvements, if floating-point data is needed.

The FPU features:

- high-speed operation
- single- and double-precision (32- or 64-bit) floating-point modes
- 17-digit precision in 64-bit mode; eight-digit precision in 32-bit mode
- flexible addressing modes
- six 64-bit floating-point accumulators
- error-recovery aids

Based on its own set of six 64-bit floating-point accumulators and additional instructions that can reference the floating-point accumulators, the central processor's gen-

eral registers, or memory locations, the FPU carries out high-speed calculations in either single-precision (32-bit) or double-precision (64-bit) mode.

The basic floating-point add, subtract, multiply, and divide instructions are complemented by a range of additional hardware instructions. These instructions allow faster, more compact coding of computation routines.

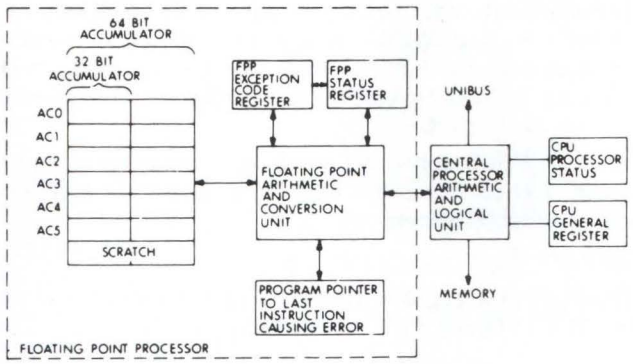


Figure 8-2
Floating-Point Unit

FRONT CONSOLE

The console is the operator's interface to the central processor. Using the console, the operator can start, stop, reset, and debug the CPU; examine and deposit data in memory locations or the processor registers; halt the processor; step through instruction streams; and boot the operating system.

In addition to the console, a bootstrap and diagnostics module is provided. The module contains one to three Kbytes (depending on the model) of read-only memory (ROM) that can be used for:

- diagnostic routines
- the console emulator routine that lets a user type commands on the console terminal to perform normal console functions
- bootstrapping programs without the operator keying in the initial bootstrap program

PERIPHERAL CONTROLLER INTERFACES

The peripheral controller interfaces on the LSI-11 and PDP-11 processors convey signals to and from such I/O devices as lineprinters, disks, cardreaders, terminals, and interprocessor communication links. On the LSI-11 processors, peripherals are connected to the processor through the LSI-11 bus. On the PDP-11 processors, peripherals are connected to the processor through the UNIBUS. The PDP-11/70 uses, in addition to the UNIBUS, a MASSBUS peripheral interface to accommodate high-speed storage devices.

The LSI-11 bus, like the UNIBUS, is a simple, fast, easy-to-use master/slave communication interface between system components. It provides vectored priority interrupts, programmed I/O transfers, and direct-memory-access (DMA) I/O data transfers. All modules operate asynchronously at their highest possible speed.

All modules connected to the LSI-11 bus receive the same interface signals. LSI-11 bus control and data lines are open-collector lines, which are asserted when low. All data and most control lines are bidirectional. All transactions on the bus are asynchronous. The LSI-11 processors use 44 LSI-11 bus signals. Sixteen of these are multiplexed data/address lines; two are multiplexed address/parity lines; four are extended address lines (not implemented on the PDP-11/23); six are data transfer control lines; six are system control lines; and ten are interrupt and DMA control lines.

With bidirectional and asynchronous communications on the LSI-11 bus, devices can send, receive, and exchange data at their own rates. The bidirectional nature of the bus allows use of common bus interfaces for different devices—which simplifies interface design.

Communication between two devices on the bus happens through a master-slave relationship. At any time, there is one device controlling the bus. This controlling device is termed the bus master. The master device controls the bus when communicating with another device on the bus, the slave. A typical example of this relationship is the processor, as master, fetching an instruction from memory (which is always a slave). Another example is a DMA device interface, as master, transferring data to memory, as slave. Bus master control is dynamic. The bus arbitrator, the CPU, controls the time allocation of the LSI-11 bus for peripherals.

Interrupt and direct memory access are implemented with two daisy-chained grant signals that provide a priority-structured I/O system. The highest-priority device is the module located electrically closest to the CPU. A device passes grant signals to lower-priority devices only when it is not requesting service.

The LSI-11 bus provides a vectored interrupt interface for any I/O device and permits DMA transfers directly between I/O devices and memory, without disturbing the

processor registers. When an interrupting device receives a grant, the device passes an interrupt vector to the processor, which points to a new processor status word and the starting address of an interrupt service routine for the device.

The Extended LSI-11 bus, implemented on the PDP-11/23-PLUS and MICRO/PDP-11, supports the extended addressing feature of these processors, while retaining a high level of compatibility with the LSI-11 bus. The Extended LSI-11 bus redefines four bus lines that are designated as "spare" on the LSI-11 bus, for use as address bits. This means that MICRO/PDP-11 and PDP-11/23-PLUS systems can include up to four Mbytes of memory, to support more active users with better system performance than an equivalent PDP-11/23 system.

The UNIBUS

The UNIBUS lets devices send, receive, or exchange data, without processor intervention and, in some cases, without intermediate buffering in memory. The PDP-11 UNIBUS consists of signal lines, to which all devices are connected in parallel.

Devices on the bus communicate as master and slave. During any bus operation the bus master controls the bus when communicating with the slave device. Master-slave relationships are dynamic. For instance, the processor can pass bus control to a disk; then the disk can become master and communicate with memory. Priority arbitration takes place asynchronously in parallel with data transfer. Every device on the bus except memory can be bus master.

When two or more devices try to obtain control of the bus at once, priority circuits choose between them. Devices have unique priority levels fixed at system installation. If requesting devices have equal priority levels, the one closest to the processor electrically on the bus takes precedence over those further away.

A device uses the bus if it needs to request the processor or transfer a word or byte of data to or from another device without involving the processor. There are two ways of requesting bus control—nonprocessor requests (NPRs) and bus requests (BRs).

An NPR is issued when a device wishes to perform a data transaction. An NPR device does not use the CPU once the running program has set up parameters of buffer address and byte count; therefore, the CPU can relinquish bus control while an instruction is being executed.

A BR is issued when a device needs to interrupt the CPU for service. A BR interrupt is serviced only when the processor finishes executing its current instruction.

The MASSBUS

The PDP-11/70 provides dedicated, prewired space for up to four high-speed I/O MASSBUSes. Each group of mass storage peripherals communicates directly to cache memory through the MASSBUS. The MASSBUS consists of signal lines for data, control, status, and parity. A high transfer rate is achieved by using synchronous transfer of data simultaneously with asynchronous control information.

Data are transferred in direct memory access (DMA) mode. An internal 32-bit-wide data bus transfers four Kbytes in parallel between memory and the high-speed controllers. The priority-arbitration logic within the cache memory controls the timing of data transfers; the cache itself, though, is not used for storage. Data transfers are between main memory and the mass storage peripheral.

The architecture of the PDP-11/70 allows overlapping of some operations, providing faster program execution speed. CPU and UNIBUS read hits with the cache memory are overlapped with mass storage device reads from main memory. The read cycles of several mass storage devices can be overlapped.

The MASSBUS generates and checks parity for data, address, and control information, to ensure the integrity of the information transferred.

PROCESSOR DESCRIPTIONS

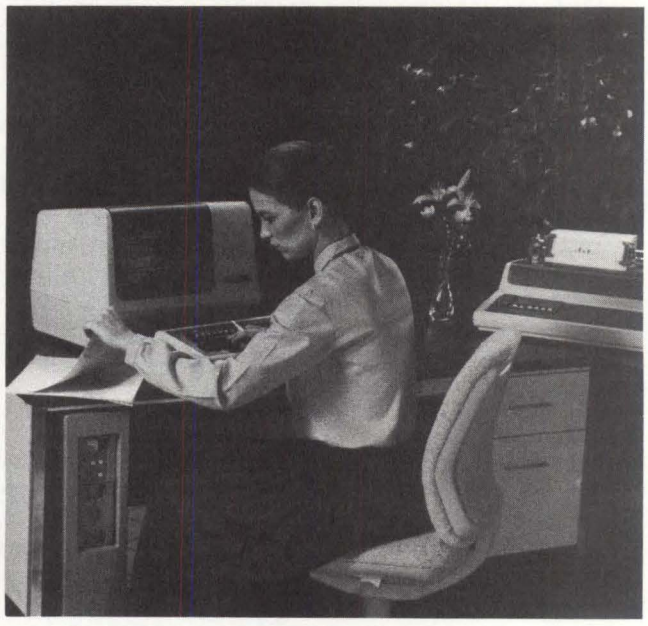
The PDP-11 and LSI-11 processors on which members of the RSX-11 family run are summarized in Table 8-1.

PDP-11/23

In many applications, the PDP-11/23 microcomputer approximates the power and performance of a PDP-11/34A minicomputer. Its features include memory management with dynamic relocation, segmentation and protection capabilities, up to 256 Kbytes of addressing, and an optional Floating-Point Unit.

The diagnostic/bootstrap/terminator module contains switch-selectable diagnostics for the processor, memory, and serial line unit. A loop-on-test switch allows repeated execution of a particular diagnostic for fault isolation. This module can also be used to terminate bus signals and to automatically load up to 32 Kbytes of user programs on powerup.

Most of the peripherals supported on the PDP-11/23 are program-compatible with the UNIBUS versions. The PDP-11/23 supports RSX-11S and RSX-11M.



MICRO/PDP-11

While the MICRO/PDP-11 includes a PDP-11/23-PLUS CPU and features many of the same capabilities, it differs in many ways from a PDP-11/23-PLUS microcomputer system. In a MICRO/PDP-11, the CPU, 256 Kbytes of parity memory, a ten-Mbyte 5¼-inch mini-Winchester system disk, a 800-Kbyte dual-diskette system for backup and media exchange, and an extended LSI-11 backplane are all contained in a single enclosure. Because it's available packaged to stand on the floor, sit on a tabletop, or be rack-mounted, it is especially suited to offices, factories, or laboratories. Users can easily install and service the system and its options because all major components plug into the system with quick-disconnect fasteners and edge connectors.

The MICRO/PDP-11 has less extensive expansion capabilities than the PDP-11/23-PLUS because it has fewer backplane slots. It supports less disk space than the PDP-11/23-PLUS, but the same amount of memory—up to four Mbytes of parity memory. RSX-11M-PLUS is the only RSX-11 operating system supported on the MICRO/PDP-11.

PDP-11/23-PLUS

The PDP-11/23-PLUS is a powerful, compact microcomputer that extends the performance capability of the PDP-11/23. Designed to increase system performance while providing efficient backplane utilization, the PDP-11/23-PLUS addresses up to four Mbytes of parity memory through the Extended LSI-11 bus. Extended addressing means that the PDP-11/23-PLUS, compared to the PDP-11/23, can, in many applications, support more memory-resident tasks, for more active users and better system response.

With its support of the optional Commercial Instruction Set (CIS), the PDP-11/23-PLUS can be used in multitasking, multiuser applications traditionally reserved for larger

Table 8-1
Processor Summary

	PDP-11/23	MICRO/PDP-11	PDP-11/23-PLUS	PDP-11/24	PDP-11/34A	PDP-11/44	PDP-11/70
RSX-11S	X	—	X	X	X	X	X
RSX-11M	X	—	X	X	X	X	X
RSX-11M-PLUS	—	X	X	X	—	X	X
Peripheral Controller Interface	LSI bus	extended LSI bus	extended LSI bus	UNIBUS	UNIBUS	UNIBUS	UNIBUS MASSBUS
Memory Management (addressing capacity)	standard (18-bit)	standard (22-bit)	standard (22-bit)	standard (18-bit); optional (22-bit)	standard (18-bit)	standard (22-bit)	standard (22-bit)
Maximum Memory	256 Kbytes	4 Mbytes	4 Mbytes	4 Mbyte	256 Kbytes	4 Mbytes	4 Mbytes
Memory Type	nonparity	parity	parity	parity	parity	ECC	ECC
Cache Memory	—	—	—	—	2 Kbytes optional	8 Kbytes	2 Kbytes
Processor Operating Modes	kernel user	kernel user	kernel user	kernel user	kernel user	kernel user supervisor	kernel user supervisor
Instruction Set*	standard EIS	standard EIS CIS (opt.)	standard EIS CIS (opt.)	standard EIS CIS (opt.)	standard EIS	standard EIS CIS (opt.)	standard EIS
Battery Backup	—	—	—	optional	optional	optional	standard
Floating-Point Unit	optional	optional	optional	optional	optional	optional	optional
Powerfail Automatic Restart	standard	standard	standard	standard	standard	standard	standard

* EIS: Extended Instruction Set; CIS: Commercial Instruction Set

minicomputers. Yet the PDP-11/23-PLUS also meets the demands of complex realtime applications that are sensitive to performance, packaging, and reliability considerations. The PDP-11/23-PLUS supports RSX-11S, RSX-11M, and RSX-11M-PLUS.

PDP-11/24

As compact as a microcomputer, the PDP-11/24 is a UNIBUS system that supports a wider variety of large-capacity mass-storage devices—including Winchester disks—than do the LSI-11-based systems. The PDP-11/24 is priced as an entry-level system, but has midrange capabilities. It contains a powerful single-hex module processor with an optional extended 22-bit-memory-addressing facility, so it supports up to four Mbytes of memory.

Other optional features complement the PDP-11/24 processor's standard instruction set. Both the optional Floating-Point Unit and the Commercial Instruction Set option improve program performance.

PDP-11/34A

The PDP-11/34A is a midrange member of the PDP-11 family of processors. Its entire CPU logic is contained on two circuit boards.

The PDP-11/34A supports a number of "large system" features. Its standard features include a memory management facility that provides program protection, memory relocation, and addressing of up to 256 Kbytes of memory and a multifunction ROM (Read Only Memory) with virtual console capability, diagnostics, and bootstraps. A cache memory that can increase program execution speeds up to 60 percent; a floating-point processor that enables the CPU to perform high-speed floating point arithmetic operations ten times faster and more efficiently than software routines; and serial communications line interfaces, a realtime clock, and a battery backup unit are all available as options. This long list of standard and optional features provides considerable flexibility in the use, maintenance, and expandability of PDP-11/34A-based systems.

PDP-11/44

The PDP-11/44 offers capabilities and performance features previously unavailable from a computer in its price range. It provides such high-performance, large-machine features as a high-speed central processor, support for up to four Mbytes of main memory, a large eight-Kbyte cache memory, and an optional hardware Commercial Instruction Set.

The PDP-11/44's high-speed, eight-Kbyte bipolar cache memory provides a 275-nanosecond cycle time that accelerates program execution and increases system throughput. In addition, it helps insulate main memory from CPU fetches, making more bandwidth available to direct memory access devices.

The PDP-11/44 was designed to meet rigorous reliability and maintainability standards, to ensure enhanced system uptime. A built-in front-end microprocessor controls the ASCII console, provides extensive system diagnostics, and controls a dual TU58 cartridge tape subsystem. The TU58, described in Section 9, is used to load system diagnostic programs and software updates and to provide additional storage capacity.

PDP-11/70

The PDP-11/70 was designed to operate in large, sophisticated, high-performance systems. It is a powerful computational tool for high-speed realtime applications and for multiuser, multitasking applications that require large amounts of addressable memory space.

Cache memory, standard in the PDP-11/70, is a high-speed bipolar memory with a two-Kbyte capacity. Its cycle of only 240 nanoseconds significantly accelerates program execution. It also reduces the number of fetches from main memory, because most of the memory references can be found in cache.

To facilitate the transfer of data and instructions in and out of cache memory, the PDP-11/70 utilizes high-speed 32-bit data paths. This, in turn, makes more bandwidth available for the high-performance peripherals.

The integrated MASSBUS controllers are high-speed mass storage controllers. I/O devices—such as RM05 removable-disk drives and TE16 and TU77 magnetic tapes—connect to the 32-bit internal data paths. The controllers provide high throughput because:

- all controllers can transfer data simultaneously since each controller is connected to the memory system with its own built-in data path
- while one device on a controller is transferring data, control operations like seek or rewind can be issued to another device on the same controller. The operation can be initiated, and an interrupt is generated when it is complete

9
The Peripherals

RSX-11M and RSX-11M-PLUS systems support a wide range of peripheral devices—disks, magnetic tapes, terminals and terminal interfaces, lineprinters, cardreaders, and communications interfaces. For realtime applications RSX-11 supports such laboratory and industrial devices as analog-to-digital converters and laboratory peripheral systems. Peripherals can be added to a system according to the needs of the system users and the application. DIGITAL supplies drivers for these devices as part of system software.

INTRODUCTION

RSX-11M and RSX-11M-PLUS support the following types of peripherals:

- mass storage peripherals—disks and magnetic tapes
- unit record peripherals—lineprinters and cardreaders
- terminals and terminal line interfaces
- realtime and sensor I/O devices such as analog-to-digital converters

Because it is a memory-only system, RSX-11S does not support disks or tapes as file-structured devices. However, it does support storage on all RSX-11M peripheral devices. Floating-Point Units, parity memory, and Memory Management Units are also supported.

MASS STORAGE PERIPHERALS

The mass storage peripherals include various capacity removable-media and fixed-media disk drives, and various speed magnetic tape transports.

All RSX-11M and RSX-11M-PLUS systems are disk-based. Disk selection depends on the type of applications, the database, and the number of users.

RSX-11M and RSX-11M-PLUS support a variety of magnetic tape devices. In most instances magnetic tapes provide backup for disk files, archival storage, and/or a journal function for larger RSX-11 systems. Programming for magtape and magnetic tape cassettes is similar; however, magtape can handle variable-length records, and with it users can select a parity mode.

Disks

In the past, technologies that affect data management and storage—software, controllers, and disk drives—have changed at different speeds. To overcome technical incongruities that can develop among these elements, DIGITAL has developed an architectural approach to subsystem design. The DIGITAL Storage Architecture (DSA) describes disks, intelligent controllers, connections, and software protocols for attaching DSA disks to DIGITAL's computer systems. Currently, the DSA components offered for PDP-11 systems are the UDA50 microprocessor-based disk controller, the RA60 removable-media disk drive, and the RA80 and RA81 fixed-media disks.

DIGITAL continues to offer our traditional disk subsystems: the RX02, RL02, RK07, and RM05. These disk subsystems provide high performance and reliability, and they include accurate servopositioning, error correction, and offset positioning recovery.

Table 9-1 summarizes the major disk drives supported by RSX-11M and RSX-11M-PLUS.

RX02 — The RX02 double-density dual floppy disk subsystem provides industry-compatible, highly reliable mass storage. Available for both LSI-11- and UNIBUS-based PDP-11s, it is a low-cost, random-access-memory disk device specially suited to smaller systems. In single-density mode each device can store up to 256 Kbytes of data; in double-density mode it can store up to 512 Kbytes. Each RX02 controller is capable of supporting two drives for a total storage capacity of one Mbyte of formatted data. Direct memory access (DMA) provides rapid data transfer and efficient utilization of the host processor.

Table 9-1
Disk Devices

Disk	Type	Capacity	Peak Transfer Rate (KB/s)	Average Access Time (ms)*	Tracks per Surface	Interface	Maximum Drives per Controller
RX02	flexible diskette	512 KB	61	262	77	LSI bus, UNIBUS	2
RL02	removable cartridge	10.4 MB	512	67.5	512	LSI bus, UNIBUS	4
RK07	removable cartridge	28 MB	538	49	815	UNIBUS	8
RA80	fixed disk	121 MB	1,200	33.3	546	UNIBUS	4
RA60	removable disk	205 MB	1,980	50	1,600	UNIBUS	4
RM05	removable disk	256 MB	1,200	38.3	823	MASSBUS	8
RA81	fixed disk	456 MB	2,200	36.3	1,248	UNIBUS	4

K = 1,024; M = K^2; B = byte; s = second; m = 10^{-3}

* Average access time is the sum of the average seek time plus the average latency.

The RX02 is available in two packages: in a standard chassis to be mounted in a cabinet or as a tabletop unit.

RL02 — Available on both LSI-bus- and UNIBUS-based PDP-11s, the RL02 single-drive buffered cartridge disk combines reliability and maintainability in a low-cost mass storage device. An embedded closed-loop servopositioning system improves data integrity by continuously sampling servo information with the same head that reads and writes the data. To further ensure data integrity, a cyclic redundancy check (CRC) is performed on data transfers between the drive and controller. Also, a phase-lock-loop clock system and modified frequency modulation (MFM) recording provide reliable reading and recording techniques. Direct memory access is used to provide rapid data transfer and efficient utilization of the host processor.

The RL02's disk cartridge has one platter containing two recording surfaces. Up to four disk drives, each with a formatted capacity of 10.4 Mbytes, can be added to expand the system. The RL02 features a 512-Kbyte-per-second transfer rate. Average access time* is 67.5 milliseconds.

The subsystem features a universal power supply. Substituting a different line cord plug and inverting a connector located at the rear of the unit is all that is required to convert the drive from 115 volts/50-60 Hz to 230 volts/50-60 Hz.

RK07 — The RK07 single-drive, buffered disk subsystem is a reliable storage device and controller for UNIBUS PDP-11s. Data integrity features include a phase-locked-loop clock system and modified frequency modulation (MFM); Error Correction Code (ECC); a hardware write-check capability and verification of sector, track, and cylinder positioning; and a software-controlled diagnostic mode (DMD) for extensive status/error reporting. Direct memory access is used to provide rapid data transfer and efficient utilization of the host processor.

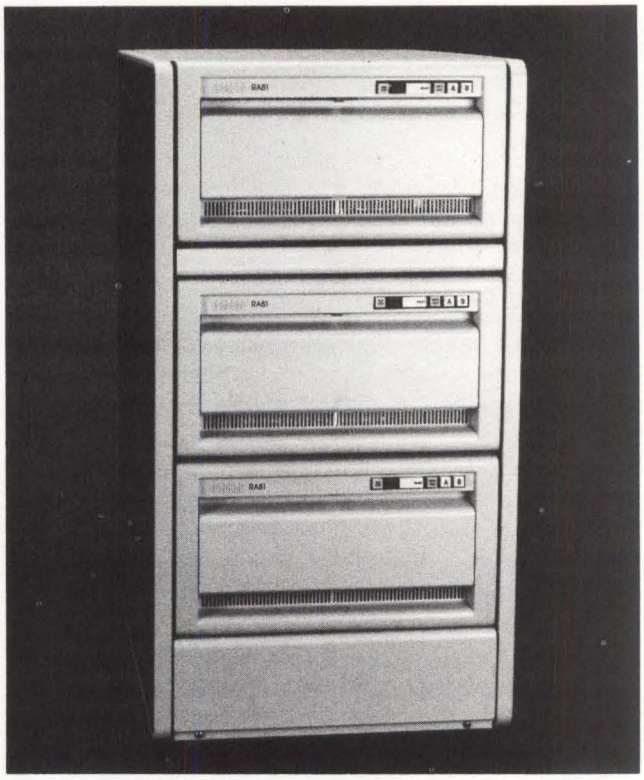
Up to eight drives, each with a capacity of 28 Mbytes of formatted data, can run off one controller. Average access time* is 49 milliseconds.

All disk cartridges are preformatted with 22 sectors per track in 16-bit format and 20 sectors per track in 18-bit format. Bad blocks are preintialized during manufacturing to ensure that all sectors can be read by any drive. This saves valuable system time and increases data integrity. Cartridges can also be reinitialized on site without destroying the original bad block marking.

The RK07 is well suited for PDP-11/24s, PDP-11/34s, and PDP-11/44s, because these systems usually require less storage than the larger PDP-11/70.

RM05 — The RM05 is a large-capacity, high-performance disk drive with a 256-Mbyte removable disk pack and a range of field-proven reliability and maintainability features. These features, together with its 38.3-millisecond access time* and a peak transfer rate of 1.2 Mbytes per second, make the RM05 a wise choice for high-throughput I/O intensive applications where abundant storage and high disk performance are an important factor in total system performance.

* Average access time is the sum of the average seek time plus the average latency.



The RM05 is housed in two cabinets. One is for the drive and the other is a utility cabinet containing the drive bus adapter. Extra space is provided in the utility cabinet for a second drive bus adapter for an additional drive. The drive connects to the PDP-11/70 MASSBUS disk controller through the drive bus adapter.

The RM05 has a dual-port option that provides multipath access to the database and high system availability where redundant system configurations are needed.

UDA50 — An implementation of the DIGITAL Storage Architecture, the UDA50 intelligent controller contains a 16-bit high-speed processor that executes host interface and drive interface programs simultaneously. It can handle data rates up to three Mbytes per second. Four disk drives can be connected to it in a radial fashion.

The UDA50 controller and its associated DSA-series disks offer significant improvements in I/O throughput, data integrity, and subsystem availability. The UDA50:

- supports high-speed disk technology
- provides powerful error-correcting systems for high-density recording
- provides multiple-level performance optimizations for both single- and multiple-drive subsystems
- supports both Winchester removable-media and fixed-media disks of varying capacities and transfer rates
- unburdens the host system of the overhead associated with error handling and I/O throughput optimization

The UDA50 controller, which is included in RA60, RA80, and RA81 subsystems, performs a variety of performance optimizations to improve disk throughput. A seek-ordering

algorithm reorders up to twenty I/O requests in the UDA50's command queue to minimize seek time in both single- and multiple-drive subsystems. When requests are present for several disks, the controller performs overlapped seek operations. This means throughput on multidrive subsystems is increased since one disk can be track-seeking while another disk is transferring data. In addition, if two drives are both on-cylinder, the UDA50 selects the drive nearest its beginning block to perform data transfers. This process is known as rotational optimization.

Firmware contained in the UDA50 controller performs data transfers, performance optimization, and error detection/correction. The controller also uses device-independent software—the Mass Storage Control Protocol (MSCP). Disk-dependent requirements, such as disk geometry and error recovery strategies, are isolated from the host operating system code by the MSCP device driver located in the CPU. Thus, future disk drives will be supported by the UDA50 controller without modifications to operating system software.

With the UDA50, which supports up to four disks, users can "mix and match" up to a total of three RA60, RA80, and/or RA81 disk drives in a 42-inch- (108.7 cm)-high cabinet. Two UDA50 controllers can be configured on a single CPU. These capabilities give users the flexibility to choose whatever combination of Winchester removable-media and fixed-media disks that best suits their application.

RA60 — The RA60 removable-media disk drive provides 205 Mbytes of storage for UNIBUS PDP-11 systems. The RA60's peak transfer rate is 1.98 Mbytes per second, and its average access time* is 50 milliseconds.

The RA60 disk drive incorporates a variety of innovations. These include:

- the use of enhanced servo technology, which eliminates the need for alignment packs
- new recording methods
- microprocessor-controlled diagnostics
- 170-bit error-correcting code
- modular design for easy maintenance

In most removable-media disk drives, a misaligned stack of heads will cause unreliable pack interchange. The RA60's embedded servo system (head-positioning information on every data track) overcomes this problem and always allows highly reliable pack interchange. In the RA60, periodic head-alignment checking and realignment are not necessary.

RA80 — The RA80 disk combines the advantages of Winchester fixed-disk technology with a high-performance, high-reliability microprocessor-based controller, the UDA50. As the result of an advanced mechanical design that incorporates a rotary positioner, computer-designed positioner arms, and the lightweight Winchester head suspension, the RA80 offers exceptional throughput performance—a peak transfer of 1.2 Mbytes per second and an average access time* of 33.3 milliseconds. The RA80 has a storage capacity of 121 Mbytes.

* Average access time is the sum of the average seek time plus the average latency.

The RA80 features a sealed Head Disk Assembly (HDA) and a microprocessor that controls major drive functions and resident fault isolation diagnostics. Each RA80 drive uses a direct disk-to-controller cable, which prevents a failure on one drive from affecting other drives on the controller.

RA81 — The RA81 features a high-performance Winchester-technology disk drive and an intelligent UDA50 controller that accelerates I/O throughput, performs expanded error recovery, and contains a 51-sector data buffer to match the disk's 2.2 Mbyte-per-second burst data rate to the host system.

The RA81's high capacity of 456 Mbytes is achieved through innovative engineering in the read/write and positioner control systems. The read/write system employs an encoding/decoding scheme that yields a third more storage capacity than drives using conventional modified frequency modulation (MFM) encoding. Positioning information on a dedicated (servo) surface enables high-speed seeking. Additional positioning information is embedded between sectors on every track for high-precision positioning.

The RA81 features outstanding data reliability, including an industry-leading error correction code, automatic sector reallocation, error detecting code, quadruplicated header addresses, data compare commands, access commands, and error reporting. All error recovery routines are initiated and completed in the subsystem, thus off-loading the host system.

Tapes

Lightweight and small in size, tape cartridges and magnetic tapes provide a convenient medium for backup and private file storage. RSX-11's system utilities allow users to mount and dismount private files that are stored offline.

All tape devices, except the TU58, use Mylar-based, oxide-coated magnetic tapes with reflecting marker strips to indicate the beginning and the end of the tape. Adjacent files are separated by formatted interrecord gaps.

Several common features ensure data integrity and reliability, optimize performance, and facilitate maintenance.

All tape devices provide a write-protect capability to protect the integrity of data that is read and written onto the tape. The TU58 cartridge has a plastic write-lockout tab. Users need only flip the tab on the cartridge to write-protect tapes for short or long term. The other tape devices provide an industry-standard write-protect ring on the tape reel. The tape drive can sense the write protection, thus prohibiting data from being written.

Accurate data recording and retrieval is ensured by a checksum on the TU58 and a read-after-write check on the other tape devices. A checksum is a character generated when a block of data is written and checked when the block is read. If an error is found, the TU58 automatically rereads the tape up to eight times before a hard error is indicated. A read-after-write check prevents data from being written on worn or damaged tape sections. If an error is detected in the read-after-write check, a message is sent to the host processor.

Parity, longitudinal, and cyclic redundancy checking further ensure the accurate transfer of data in all magnetic tape systems. Parity is checked character by character while reading and writing on tape at 1,600 bits per inch. Recording density of 800 bits per inch includes a cyclic redundancy check character and a longitudinal redundancy check character. These characters are calculated when a block is written and checked when the block is read. If an error is detected, the host processor is notified.

All magnetic tape systems minimize tape error problems through a runaway timer that allows the system to recover from bad tape sections on the reel.

Magnetic tape controller/formatter operations include:

- moving the tape to new positions in forward and reverse
- monitoring tape operation
- fetching, formatting, and sending data
- handling error conditions and drive-servicing requirements

The major tape drives supported by RSX-11M and RSX-11M-PLUS are summarized in Table 9-2.

TU58 — A TU58 DECtape II cartridge tape subsystem consists of one or two TU58 cartridge drives, a controller, and one DL11-type serial line interface. It is supported on both LSI-bus- and UNIBUS-based PDP-11s. All command and data transfers are via programmed I/O and interrupt-driven routines. TU58 DECtape is a convenient pocket-sized cartridge that functions as a random-access mass storage subsystem capable of storing 512 preformatted blocks of 512 Kbytes each.

DECtape II is a fixed-address system with an average access time of ten seconds and a read/write speed of 30 inches per second. Block formatting compacts the storage area into related groups of records, eliminating the need to address individual records. The controller performs read/write check operations using a 16-bit checksum and signals the driver that there is an error if more than one attempt is made to perform a successful read. Serving as a transport subsystem for auxiliary data storage, DECtape II is able to read and write 256 Kbytes of data per drive at a rate of 800 bits per inch. Each transport has a high-quality read/write head with a full erase gap for reliable recording.

TSV05 — The TSV05 is a microprocessor-based magnetic tape subsystem that incorporates reel-to-reel technology. Designed for data interchange, archiving, journaling, and disk backup, it is the only magnetic tape DIGITAL offers for LSI-11-based PDP-11s. It offers industry-standard 1,600 bits-per-inch Phase Encoded (PE) format, ANSI compatibility, and a storage capacity of 28 Mbytes per 2,400-foot reel. Reading and writing are performed at 25 inches per second, and characters are transferred at a maximum rate of 40 Kbytes per second.

The TSV05 accepts seven-to-10½-inch reels of half-inch tape. For easy of operation, tapes are front-loaded and threaded automatically.

Reel-to-reel technology involves writing data on tape without stopping and starting between each record block. Interrecord gaps, as required in the ANSI format, are inserted automatically while the tape is in motion.

Table 9-2
Tape Drives

Tape	Type (Size)	Capacity (MB)	Recording Density (bits/in)	Read–Write Speed (in/s)	Maximum Data Transfer Rate (KB/s)	Rewind Speed (in/s)	Interface	Maximum per Formatter
TU58	2-track (2.4 × 3.2 × 0.5″ cartridge)	0.25	800	30	3.7	60	LSI bus,UNIBUS*	2†
TSV05	9-track (10½″ reel)	28	1600	25	40	180	LSI bus	1
TE16	9-track (10½″ reel)	20, 40	800, 600‡	45	72	150	UNIBUS,MASSBUS	8
TS11	9-track (10½″ reel)	40	1600	45	72	150	UNIBUS	1
TU77	9-track (10½″ reel)§	20, 40	800, 1600‡	125	200	440	UNIBUS,MASSBUS	4

K = 1,024; M = K²; B = byte; in = inch; s = second

* Via DL11-E serial-line asynchronous interface

† Only one TU58 can operate at a time

‡ Program selectable

§ Also loads and threads Easy Load #1 and Easy Load #2 cartridges.
(Easy Load #1 and Easy Load #2 are trademarks of IBM Corporation.)

The TSV05 is highly reliable. It features:

- dual-microprocessor design
- simplified mechanical design
- built-in self-test and fault isolation diagnostics
- read-after-write parity check
- automatic single-track error correction
- in-line microdiagnostics

The TSV05 is program compatible with DIGITAL's TS11 magnetic tape subsystem at the DIGITAL-supplied device drive level. While a TSV05 tape reel offers nearly three times the capacity of an RL02 floppy disk, a TSV05 tape reel is significantly less expensive.

TE16 — The TE16 magnetic tape is a fully integrated storage system for UNIBUS PDP-11 systems that uses industry-standard recording formats with densities of 1,600 (PE—phase encoded) and 800 (NRZI—non-return to zero inverted) bits per inch, selectable under program control. Reading and writing are performed at 45 inches per second, and characters are transferred at a maximum rate of 72,000 per second. The TE16 features bidirectional tape reading; writing occurs only while the tape is moving forward. The TE16 is controlled by the TM03 magnetic tape formatter.

The TE16 uses an oxide-coated magnetic tape with reflecting marker strips that indicate the beginning and the end of the tape. The formatter includes a runaway timer that is used as a failsafe mechanism. Tape motion is stopped if no record is detected during the timer's interval, which is roughly equivalent to 25 feet of tape.

The TE16 provides many features that protect the integrity of data that is read from and written onto the tape. An industry-standard write-protect ring located on the tape reel prevents accidental writing of data to the reel. Parity, longitudinal, and cyclic redundancy checking further ensure accurate data transfer. Parity is checked character by character when reading from and writing on tape in PE (1,600-bit-per-inch) operation. The recording of bad data is checked for and prevented by the use of 12-character data blocks. If there are fewer than 12 characters in a block, the data will not be acknowledged by the host computer.

The TE16 is available as a subsystem or as an add-on drive. Subsystems include the TE16 transport and the TM03 formatter and controller. The transport and formatter are integrated into one cabinet. The controller mounts in the CPU chassis. A TM03 formatter supports up to eight TE16 transports.

TS11 — The TS11 is a medium-performance nine-track magnetic tape storage system that features microprocessor-controlled electronics for high data reliability and maintainability. Performance features include 45-inch-per-second read/write speed, up to 72,000-byte-per-second transfer rate, 150-inch-per-second rewind speed, and forward and reverse read.

To ensure accurate transfer of data the TS11 has a built-in microprocessor to monitor and control voltage levels, tape speed, and timing. Additionally, the TS11 features single-track error correction, read-after-write checks, retry algorithms, and parity checking on both the data path and microprocessor memory transfers. The TS11 tape subsystem automatically verifies its own operational integrity by continually running diagnostics to inspect all critical drive functions.

The TS11 consists of a tape transport with integrated formatter and a single hex-sized UNIBUS interface/controller module. This module plugs into any hex-sized UNIBUS small peripheral controller slot and communicates with one tape transport. A universal power supply is standard. Voltage and frequency are easily converted by changing one small assembly consisting of a power cord, line filter, and circuit breaker.

TU77 — The TU77 is a high-performance nine-track tape storage system packaged with its associated interface and power supply. The TU77 subsystem consists of a TU77 tape drive, a dual-density TM03 tape formatter, and a MASSBUS controller that mounts in the CPU chassis. Reading and writing are performed at 125 inches per second, and data is recorded at either 1,600 bits per inch (PE) or 800 bits per inch (NRZI). Data is transferred at up to 200,000 bytes per second using ANSI standard formats. Automatic tape threading maximizes operator convenience and minimizes tape handling.

To guarantee security of data, the TU77 has a wide range of data integrity features. The formatter, the tape drive's interface to the MASSBUS, oversees the handling of error conditions and drive servicing. The TU77 features automatic detection and correction of single-track errors in both PE and NRZI formats plus parity checking on the entire data path. Additional data integrity features include a read-after-check write protect feature, self-testing of error correction circuits, and detection of bad tape errors by way of a runaway timer.

The combined features of the TU77 make it ideally suited for such demanding applications as transaction processing and disk-to-tape backups. In addition to the first drive, the TM03 formatter can attach to three additional transports.

UNIT RECORD PERIPHERALS

To satisfy their output requirements, DIGITAL offers users a variety of unit record devices—lineprinters, cardreaders, and a papertape reader/punch.

Lineprinters

DIGITAL offers a wide range of lineprinters for a variety of environments and applications—from low-cost 300-lines-per-minute band printers with optional European and Japanese character sets to heavy-duty 1,200-lines-per-minute Charaband printers designed for data processing environments. DIGITAL also offers versatile lineprinter/plotters.

LP11-AA/LPV11-AA and LP11-BA/LPV11-BA — As DIGITAL's low-cost steel-band printers, the LP11-AA/LPV11-AA and LP11-BA/LPV11-BA feature user-replaceable font bands, horizontal-font printing that provides clear print appearance, and improved reliability and maintainability. The LP11-AA and LP11-BA are the UNIBUS versions of these printers; the LPV11-AA and LPV11-BA are the LSI-bus versions.

The LP11-AA/LPV11-AA includes a 64-character font band and prints at a minimum speed of 285 lines per minute. The LP11-BA/LPV11-BA includes both 64- and 96-character font bands that are easily interchangeable. It prints at a minimum rate of 285 lines per minute with the 64-character font band and at 214 lines per minute with the 96-character band. The British pound sign (£) and the U.S. dollar sign ($) are contained on the standard 64-character band, so only a map PROM change is required to print the pound sign.

Additional bands for other European and Japanese character sets are available for both printers. Another optional font band can compress 15 characters into an inch, permitting many output formats to be printed on 8½-inch-wide (21.25 cm) paper. A maximum of three map PROMs can coreside.

LP11-EA/LPV11-EA and LP11-EB/LPV11-EB — The LP11-EA/LPV11-EA and LP11-EB/LPV11-EB are freestanding band printers that use a flat steel band with raised letters and a hammer bank with 132 hammers (one for each column). (In comparison, the band on the LP11-AA/LPV11-AA and LP11-BA/LPV11-BA has only 66 columns.) The LP11-EA/LPV11-EA and LP11-EB/LPV11-EB accept single or multiple-part forms of pinfeed, continuous fanfold, edge-perforated paper. They can produce up to six copies on multipart forms. Featuring a wide choice of user-selectable and interchangeable character fonts, these printers are both economical and versatile. Optional fonts include those for German, Finnish/Swedish, Danish/Norwegian, and Spanish/Portuguese.

The LP11-EA and LP11-EB are the UNIBUS versions of these printers. The LSI-bus versions are the LPV11-EA and LPV11-EB. The LP11-EA and LPV11-EA operate at speeds up to 600 lines per minute using a 64-character set; the LP11-EB and LPV11-EB operate at either 600 lines per minute using a 64-character set or 445 lines per minute using a 96-character set.

LP11-C and LP11-D — The LP11-C and LP11-D lineprinters are fast, reliable drum printers that reduce operation noise. They have variable forms control with top-of-form, but do not have optional character sets. The LP11-D prints at 900 lines per minute with the 64-character set and 600 lines per minute with the 96-character set, while the LP11-C, which has the 64-character set, prints at 900 lines per minute. These printers are available for UNIBUS systems only.

LP11-GA and LP11-GB — The LP11-GA and LP11-GB are freestanding, heavy-duty, high-performance Charaband printers designed to handle high-volume throughput. These printers offer excellent print quality during normal operation. For extra high-quality printing, the print rate can be changed. The LP11-GA and LP11-GB also feature optional fonts for foreign languages. In addition, both of these printers include a tape vertical format unit, which provides for selectable form length up to 143 print lines.

The LP11-GA and LP11-GB are available for UNIBUS PDP-11s only. The LP11-GA operates at either 715 or 905 lines per minute (switch-selectable) with a 96-character set. The LP11-GB operates at either 990 or 1,220 lines per minute (switch-selectable) with a 64-character set.

LXY11, LXY21, and LXV11 — The LXY11, LXY21, and LXV11 lineprinter/plotters are versatile printing devices that combine the benefits of a dot-matrix printer with a plotter. They can plot standard line drawings (for example, graphs, histograms, and charts) as well as plots requiring shaded or solid areas (for example, bargraphs and barcodes). Explanatory text or large, solid characters for labeling can be incorporated where desired.

These printer/plotters print the full 96-character ASCII set as well as double-height characters and underlines. They are ideally suited to provide hardcopy output of designs formulated on a graphics terminal. The LXY11, LXY21, and LXV11 print on single-part or multiple-part forms of continuous fanfold edge-perforated paper. Up to six copies of multipart forms can be produced.

The LXY11 and LXV11 produce high-quality output at speeds up to 300 lines per minute for printing and up to 16.7 inches per minute for plotting. The LXY21 prints and plots at higher speeds: up to 600 lines per minute for printing and up to 33.3 inches per minute for plotting.

In addition to a controller, cable, and pedestal with basket and paper guide, these printer/plotters include the PLXY-11 Graphics Software Package and, for RSX-11M systems only, the BCP Barcode/Block Character Software Package.

The PLXY-11 software package provides RSX-11M/M-PLUS users with access to the plotting capabilities of the LXY (for UNIBUS systems) and LXV (for LSI-bus systems) printer/plotters. The package consists of a library of FORTRAN-callable graphics subroutines and a postprocessing task (to create a plot file).

The PLXY-11 subroutine library includes routines for character and line drawing. As all operations are program-controlled, either one axis or both axes can be addressed in positive or negative incremental steps. Output to the printer/plotters is directed through the standard LP11 lineprinter drivers.

BCP, the Barcode/Block Character Plotter software package, provides RSX-11M users with the ability to generate CODE 39 barcode on the LXY11, LXY21, and LXV11 printer/plotters. The package consists of an interactive

user program that permits online generation of graphic figures. A library of graphic routines can also be linked to user application programs written in FORTRAN-77.

Cardreaders

RSX-11's cardreader driver interprets encoded, punched information using the American National Standard eight-bit card code and uses a special punch outside the data representation to indicate end-of-file.

DIGITAL'S CR11 and CR11-B cardreaders are designed to meet different throughput requirements and to read various data formats. Both cardreaders use the industry-standard EIA card that has 80 columns and 12 zones (rows). The CR11 reader processes punched cards at 285 cards per minute; the CR11-B processes cards at up to 600 cards per minute.

These cardreaders have a high tolerance to cards that have been nicked, warped, bent, or subjected to high humidity. A short card path—that allows only one card in the track at a time—and a vacuum pick mechanism—that keeps cards from sticking together by blowing a stream of air through the bottom half-inch of cards—decrease the likelihood of card jams and keep card wear to a minimum. The input hopper on the CR11 holds 550 cards, while the CR11-B's input hopper holds 1,000 cards. Cards can be loaded and unloaded while the readers are operating.

PC11 Papertape Reader/Punch

The PC11 is high-speed papertape reader/punch that features automatic fan-fold operation of industry-compatible papertape. The PC11 reads eight-channel unoiled perforated papertape at 300 characters per second and punches at 50 characters per second.

When reading tape, a set of photodiodes in the PC11 unit indicates the presence or absence of holes. When punching tape, logic levels representing ones and zeros are converted into punched holes. Any information read or punched is parallel-transferred through the PC11 control. When an address is placed on the UNIBUS the control decodes the address and determines if the reader or punch has been selected.

The controller lets users operate the reader and punch independently of each other. Either device can operate under direct program control or can operate without direct supervision, using interrupts to maintain continuous operation.

TERMINALS

Interactive terminals can be connected to RSX-11 systems. As SYSGEN options in RSX-11M systems, both half-duplex and full-duplex terminal drivers are available to handle hardcopy and videodisplay terminals. RSX-11M-PLUS systems support only full-duplex drivers.

Programs can control the operation of terminals through the terminal driver. The terminal driver supports many special operating modes for terminal lines.

The compact half-duplex terminal driver is generally used in RSX-11M and RSX-11S systems where small driver size is essential and where the additional functional capability provided by the larger full-duplex terminal driver is not required.

On RSX-11 systems that require a larger driver, the full-duplex driver offers these features:

- full-duplex operation
- type-ahead buffering
- eight-bit characters
- detection of hard receive errors
- increased byte transfer length
- additional terminal characteristics
- additional terminal types
- optional time-out on solicited and/or unsolicited input
- device-independent cursor control
- added hardware support

Terminal driver support is provided for the following DIGITAL terminal devices: LA12, LA34, LA38, LA50, LA100, and LA120 hardcopy terminals; and VT100, VT101, VT102, VT125, and VT131 videodisplay terminals.

User Interface

All interaction between a user and RSX-11 takes place on a terminal, which can be either a hardcopy terminal or a videodisplay terminal. The type of terminal chosen depends largely on the kind of work the user will be doing. For example, a hardcopy terminal is useful for keeping a log of events, whereas a video terminal's special function keys make it convenient for text editing.

RSX-11 accepts input either a line at a time or a character at a time. In line mode, characters typed at the terminal are stored in the input ring buffer until a line terminator such as a carriage return or linefeed is typed. When the system receives the line terminator it sends the line of data to the currently running program. A line of data is limited only by the column width set for the terminal. Although the default width is 80 columns, a user can reset the terminal width from 30 to 255 characters.

In character mode, the running program receives each character immediately after it is typed. Any number of characters can be entered without using a line terminator. Character mode is used, for instance, when running the EDI text editor.

RSX-11 responds to control characters that are typed at the terminal. Control characters are available to:

- correct typing errors
- direct all keyboard input to a foreground or background job
- terminate program execution
- cause terminal output to appear on both a videodisplay screen and the terminal
- suppress terminal output during program execution
- temporarily suspend output to a terminal

Foreground and background terminal I/O are independent of one another; a user can type input to one job while another job prints output. If a user is typing input to the background job when the foreground job is ready to print output, the foreground job interrupts the user by printing a message on the terminal. The system then suppresses the echo of the input until the job with higher priority completes. When the background again gains control of the system, all accumulated input is echoed.

If two jobs are ready to print output at the same time, the job with higher priority is serviced. For example, output from the foreground job would be printed before a request from the background job is serviced. When the foreground job terminated, control would revert automatically to the background job.

RSX-11 also has a type-ahead feature that lets a user enter terminal input while a program is executing. While the first command line is executing, the second line can be typed. Any characters entered while the previous command is executing are echoed immediately on the screen, stored in the input ring buffer, and executed when the previous operation completes. Type-ahead is particularly useful for specifying multiple command lines to system utility programs.

Terminal characteristics are initially established during SYSGEN. Through a command issued at the terminal, users can modify the defaults set for the terminal.

Hardcopy Terminals

DIGITAL offers a wide variety of hardcopy terminals. Both keyboard send/receive (KSR) interactive terminals and receive only (RO) printing terminals are available.

LA12 Correspondent — The LA12 Correspondent is a briefcase-sized portable interactive terminal that prints up to 132 columns at 150 characters per second on plain paper. At about 19 pounds, the lightweight Correspondent is truly portable, and small enough to fit under an airline seat. Its nine-by-nine-dot matrix and bidirectional printing produce clear characters and fast output. The Correspondent also offers full bit-map graphics with resolution of 132-by-72 dots per inch.

The Correspondent's communications capabilities are extremely flexible. It supports full-duplex modem connections, hardline connections, half-duplex supervisor control mode, and more.

A loop-back connector and internal self-tests provide extensive diagnostic capabilities.

The Correspondent also features:

- auto-answer/auto-answerback (for unattended message reception)
- 300 baud acoustic coupler
- 1200/300-baud direct-connect modem
- 9600-baud EIA-RS232 interface
- local echo selection (for attachment to non-DIGITAL systems)
- shoulder-strap carrying case

There are four models of the Correspondent, each providing a different combination of communication interfaces.

LA34 and LA38 DECwriter IVs — The LA34 and LA38 DECwriter IVs are low-cost desktop microprocessor-driven terminals capable of processing data at a rate of up to 30 characters per second. The LA34 is a receive-only version; the LA38 is a keyboard send/receive (KSR) interactive terminal.

Both DECwriters include a universal power supply, a standard EIA interface and EIA null modem cable, a paper-out switch, and a customer-assistance documentation package. The LA34 accommodates roll-feed paper, and the LA38 has a paper-feed tractor for multipart forms and an 18-key numeric keypad.

Both DECwriters have a permanently stored format for a computer printout. When powered up they automatically assume a ten-character-per-inch character pitch, six-line-per-inch vertical spacing, tab stops every eight spaces, left margin set at column 1, and right margin set at column 132. A user can, however, set and change these characteristics, choosing among such features as horizontal tabs, horizontal margins, and a choice of four character sizes and six line spacings.

DECwriters operate at 300 baud and can print at burst speeds up to 45 characters per second. An alternate speed of 110 baud and ten characters per second can be selected from the keyboard. The desktop size and sculptured keyboards are so similar to standard typewriters that the transition from typewriter to terminal is natural.

The terminal's basic design contributes to its reliability and maintainability. A single logic/power amplifier board with custom LSI microprocessor reduces the component count and increases circuit reliability.

The highly reliable printhead has been designed and tested to print over 100 million characters and can be adjusted to adapt to various forms thicknesses. Should a printhead jam occur, the LA34 and LA38 immediately remove the power from the printhead drive until the jam is corrected and the terminal is restarted. This action prevents motor overloads and blown fuses. If the DECwriters run out of paper, a paper-out sensor generates a signal.

These terminals were designed to operate reliably without scheduled preventive maintenance. They disassemble simply and quickly for easy access to all components if a problem occurs. Printing self-test diagnostics allow quick and accurate identification of any faulty components. In addition, snap-in ribbon cartridges let users change ribbons quickly and easily.

Wherever possible, ANSI-standard escape sequences are used. The same escape sequences are implemented on DIGITAL's LA120 and VT100, ensuring compatibility among DIGITAL's terminal products.

LA100 Letterprinter and LA100 Letterwriter — The LA100 Letterprinter 100 and LA100 Letterwriter 100 are desktop printing terminals. The Letterprinter 100 is a receive-only version, while the Letterwriter 100 is a keyboard send/receive (KSR) interactive terminal. Both printers offer a variety of printing speeds and print qualities, use standard or custom-designed paper forms, and can change type fonts at any time during operation, manually or under host-program control. Margins, tabs, and forms length are also software controllable.

The Letterprinter 100 and Letterwriter 100 use the same graphics software as the LA34 hardcopy terminal. They can be attached to the optional printer ports of VT100 family video terminals or connected directly to a host CPU.

The Letterprinter 100/Letterwriter 100 has four print modes. Each mode includes underlining and descenders (letter tails). These modes are:

- 240 characters per second with a seven-by-nine-dot matrix for high-speed draft-quality output
- 80 characters per second with a 33-by-nine-dot matrix that produces a denser character width for more pronounced, darker characters suitable for printing memos
- 30 characters per second with a 33-by-18-dot matrix that produces printing that is ideal for professional-looking business correspondence
- bit-map graphics at 132-by-72 dots per inch

Options include tractor-driven form feeder, VT100 line-drawing character set, and up to five resident character fonts such as Courier and Orator, which are standard on some models. User-selectable communication speeds range up to 9600 baud.

LA120 DECwriter III — The LA120, an interactive hard-copy terminal, offers exceptional throughput and a number of advanced keyboard-selectable formatting features. It has a contoured typewriter-style keyboard, an additional numeric keypad, and a prompting LED (light-emitting diodes) display for infrequently used features.

The LA120's high throughput is achieved through its:

- 180-character-per-second print speed
- 14 data-transmission speeds ranging up to 9600 baud
- a 1,000-character buffer to equalize differences between transmission speeds and print speeds
- smart and bidirectional printing that causes the printhead to always take the shortest path to the next print position
- high-speed horizontal and vertical skipping over white space

The LA120 is also distinguished by its printing features. The terminal offers eight font sizes ranging from an expanded (five characters per inch) to a compressed (16.5 characters per inch) font. Using the compressed font, a user can print 132 columns on an 8½-inch-wide sheet. Other print features include six line spacings ranging from two to 12 lines per inch; user-selectable form lengths up to 14 inches; left/right and top/bottom margins; and horizontal and vertical tabs.

The LA120 is designed for easy use. Terminal characteristics are selected via clearly labeled keys and simple mnemonic commands. Once terminal characteristics are selected, a STATUS key can be pressed that causes the terminal to print a listing of the selected settings.

Videodisplay Terminals

The VT100 series of videodisplay terminals has proven to be among the most popular, and most imitated, video terminals ever offered. The VT100 series comprises the VT100, VT101, VT102, VT125, and VT131. All these terminals offer the basic VT100 capabilities. The VT100 can be upgraded with a variety of useful features, while the VT101 and VT102 cannot. The VT101 and VT102 have a local-echo feature, and the VT102 and VT131 have a printer-port and advanced video features as standard equipment. The VT102 and VT131 also offer more versatile communications capabilities. A graphics terminal, the VT125 provides full VT100 capabilities, bit-map graphics, and the ability to produce color output for a detached color monitor.

VT100 Video Terminal — The VT100, an uppercase and lowercase ASCII video terminal, offers a variety of user-controllable character and screen attributes. The VT100 features a typewriter-like detachable keyboard that includes a standard numeric/function keypad for data entry applications and seven LEDs, four of which are program-controlled, that can be used as operator information and diagnostic aids.

The VT100 offers a number of advanced features. These features let a terminal user:

- choose between two screen sizes, either 24 lines by 80 columns or 14 lines by 132 columns
- select either double-width/single-height characters or double-width/double-height characters on a line-by-line basis
- have smooth-scrolling and split-screen capability
- set baud rates, tabs, and Answer-Back messages from the keyboard and store these in random access memory
- display simple graphics with 32 special line-drawing and graphic characters for creating solid vertical and horizontal lines and square corners
- select black-on-white or white-on-black characters on a full-screen basis

The VT100 can be upgraded with several options that further extend its capabilities. An advanced video option adds selectable blinking, underline, dual-intensity (bold), and reverse-video attributes on a character-by-character basis as well as additional memory allowing 24 lines of 132 characters. Another upgrade option transforms the VT100 into a full-featured VT125 graphics terminal.

VT101 Video Terminal — The VT101 is a low-cost member of the VT100 video terminal family. It offers the basic functions of a VT100 terminal and adds the local-echo feature, for use with non-DIGITAL computer systems. (Some non-DIGITAL systems do not echo typed characters to the terminal).

VT102 Video Terminal — The VT102 is a powerful and versatile general purpose terminal. As standard features it includes all the basic VT100 functions plus the advanced video and printer-port options. The VT102 also offers:

- advanced, in-terminal editing features for inserting and deleting text by the character and by the line

- United States and British character sets as standard features, with others optional
- United States and European full- and half-duplex communication with five modem control modes
- local echo, allowing the VT102 to be used with non-DIGITAL computers
- local printer control for printing without host computer intervention

VT125 Graphics Terminal — The VT125 graphics terminal adds ReGIS-based graphics capabilities to all the standard VT100 attributes. The VT125 directly executes DIGITAL's Remote Graphics Instruction Set (ReGIS) for creating and storing pictorial data as simple ASCII text. Its bit-map architecture provides two full-screen picture planes, each with resolution of 768 by 240 pixels. At any one time either plane or both planes can be displayed on the screen.

Using ReGIS commands, pictures and text can be assigned such individual attributes as:

- color or intensity
- shading
- variable height, width, spacing, and writing direction
- italics
- overlay, replace, complement, and erase modes for writing the screen image

The VT125 contains a printer port to connect an optional LA34 graphics printer for hardcopy output. An RGB monitor can be attached to display color graphics. The VT125 can display 64 colors, four at a time, on the color monitor.

VT131 Video Terminal — The VT131 incorporates all the functions of the VT102 plus block-mode operation. Block mode allows each screenful of data to be edited locally before transmittal to computers supporting block I/O (such as DIGITAL's VAX/VMS system and some non-DIGITAL systems). On PDP-11 computers running RSX-11, a VT131 is identical to a VT102.

DECmate I System — The DECmate I system provides word processing capabilities and the ability to transfer documents to and from DIGITAL host computers. It performs word processing in its standalone mode and can also be used as a data processing terminal in interactive mode with a host system. In terminal mode, it acts as a VT100. It has many of the same features as the VT100: underline, bolding, and extended 132 column mode.

In standalone mode, the DECmate I is programmable in BASIC, FORTRAN, and DIBOL. It also supports such optional application packages as financial modeling, math, mailing, and office management. With optional software designed for commercial applications programming, this system can emulate an IBM 2780/3780.

A communications package, which is part of the word processing software, turns the DECmate I into an enhanced video terminal wherein documents can be sent to host computers, and output from host computers can be captured on the DECmate's display, printer, and disk. The communications capability, CX and DX, works with RSX-11M and RSX-11M-PLUS. Because the DECmate I is a fully functional computer, its use does not reduce the capabilities of the host computer.

The configuration consists of a videodisplay, a dual-diskette unit, and an optional matrix printer and communications option. It is user installable. The office automation software includes word processing, list processing, sort, and communications.

TERMINAL AND COMMUNICATIONS INTERFACES

RSX-11 supports a variety of synchronous and asynchronous interfaces for connecting terminals and communications packages to PDP-11 systems. Some of these interfaces handle only a single line, while others can handle multiple lines. Some interfaces have modem control capabilities ranging from auto-dial and auto-answer to auto-disconnect. Others are capable of multiplexing—that is, they combine several signals to be transmitted over the same line. And, finally, some of these interfaces are intelligent, so they relieve the computer of much of the workload involved in communications.

With this tremendous range of interfaces to choose from, users can select the interface or interfaces that best suit their particular communications requirements.

DH11 Asynchronous Multiplexers

The DH11 asynchronous multiplexers connect UNIBUS PDP-11s with 16 serial communications lines operating with individually programmable parameters.

These multiplexers use 16 double-buffered receivers to assemble the incoming characters and to support variable speeds. An automatic scanner takes each received character plus the line number and deposits that information in a first-in/first-out buffer memory called the "silo." The bottom of the silo is a register that is addressable from the UNIBUS.

The transmitter in the DH11 also uses double-buffered units. They are loaded directly from message tables in the PDP-11 memory by means of single-cycle direct memory transfers.

DL11 and DLV11 Asynchronous Serial-Line Interfaces

The DL11 is an interface between a single asynchronous serial communication line and the UNIBUS. The DLV11 is a version of the DL11 used to connect an asynchronous serial line to the LSI-11 bus. Both provide full-duplex (simultaneous two-way) operation and half-duplex (also two-way, but one way at a time) operation.

The DL11 and DLV11 perform serial-to-parallel and parallel-to-serial conversion of serial start/stop data with a double-character buffered MOS/LSI circuit called a UART (Universal Asynchronous Receiver-Transmitter). This 40-pin dual-inline package includes all of the circuitry necessary to double-buffer characters in and out, serialize/deserialize data, provide selection of character length and stop-code configuration, and present status information about the unit and each character.

Using a DL11 or DLV11 interface, a PDP-11 or LSI-11 computer can communicate with a local terminal such as a console teleprinter, with a remote terminal via data sets (modems), or with public switched telephone facilities.

Users can configure the data rate from a selection of 13 standard rates up to 9,600 bits per second or can configure a nonstandard rate device. With most of the standard rates, the interface can offer split-speed operation for faster, more efficient handling of computer output. Character size is strap-selectable or switch-selectable, and parity checking (even, odd, or none) and stop-code length (one, 1.5, or two bits) are selectable.

A four-channel asynchronous serial DLV11 interface is also available. Instead of connecting four single-channel DLV11 devices to the bus, one four-channel DLV11 can be connected, increasing system performance and decreasing system overhead.

DMP11 Network Link

The DMP11, supported by RSX DECnet, is a synchronous line communications interface that connects a PDP-11 UNIBUS system to other DIGITAL computers in distributed network applications. It operates full- or half-duplex over common carrier or private lines, or over local shielded cables. Full-duplex communication speeds up to 500,000 bits per second are possible, versus 1,000,000 bits per second for half-duplex.

The DMP11, a direct-memory-access device containing a microprocessor, implements DDCMP protocol in microcode. This reduces demands on the system CPU, which need not waste valuable time executing instructions to implement the protocol or manage the details of data transfer to memory.

DMR11 Network Link

The DMR11, supported by RSX DECnet, is a UNIBUS synchronous line communications interface designed to interconnect PDP-11 UNIBUS systems with other local or remote DIGITAL computers using synchronous serial lines. The DMR11 is capable of high-speed operation over inexpensive twinaxial or triaxial cables in full- and half-duplex modes. Reliability is increased using 16-bit cyclic redundancy checking to discover data transmission errors. Transmission speeds vary from 56,000 bits per second to 1,000,000 bits per second.

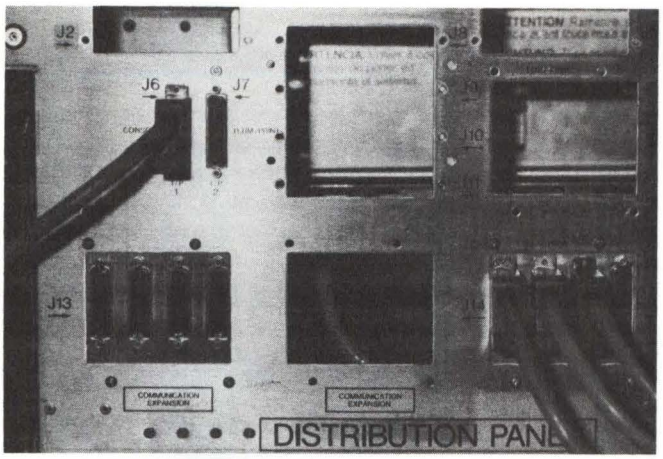
The DMR11 is a direct-memory-access device that implements DIGITAL's DDCMP communications protocol in microcode. This reduces demands on the system CPU, which need not waste valuable time executing instructions to implement the protocol or manage the details of data transfer to memory.

DMV11 Network Link

Supported by RSX DECnet, the DMV11 is a synchronous line communications interface that connects an LSI-bus system to other DIGITAL computers in distributed network applications. The DMV11 has much the same capabilities as the DMR11 and DMP11 communications interfaces. Unlike the DMR11 and DMP11, however, DMV11 operates at speeds of up to 56,000 bit per second (full duplex).

DPV11 Synchronous Serial-Line Interface

The DPV11 is a character-buffered synchronous serial-line interface that provides two-way simultaneous communications. It translates between serial data and parallel data. Output characters are transferred in parallel from the LSI-11 bus in the DPV11 where they are serially shifted to



the communications line. Input characters from the modem are shifted into the DPV11 and made available to the LSI-11 bus on an interrupt basis.

This allows a full character time in which to service transmitter and receiver interrupts. The clocking necessary to serialize the data is provided by the associated synchronous modem.

Modem control is a standard feature of the DPV11. The signals needed to establish communications with the Bell Series 200 synchronous modems are present in the receive status register.

The DPV11 transmits data at a maximum speed of 56,000 bits per second when the speed of the LSI-11 and operating software allow. The modem interface is compatible with EIA RS423 for use with higher-speed modems. A data interface with EIA RS422 is implemented for high-speed links using data leads only.

DUP11 and DUV11 Synchronous Line Interfaces

The DUP11 interface connects a serial line to the PDP-11 UNIBUS for synchronous communications. The DUP11 can operate in full-duplex mode using both byte and bit communication protocols. It is well-suited both to applications using medium speed synchronous lines for remote data collection and concentration and to networking.

Data transmission and reception is double-character-buffered at speeds of up to 9,600 bits per second. Output characters are transferred in parallel from the PDP-11 UNIBUS into the DUP11 where they are serially shifted to the communication line. Input characters from the resident modem are shifted into the DUP11 and made available to the PDP-11 on an interrupt basis. Data transmission errors are efficiently detected by cyclic redundancy checking. Modem control for remote communications via Bell Series 200 synchronous modems is a standard feature.

The DUV11 interface is used with the LSI-11 bus. It too is a single synchronous line, double-buffered communications device. It also offers full- and half-duplex operation, and modem control for remote communications at speeds of up to 9,600 bits per second. The DUV11 detects data transmission errors by parity checking. It has auto-answering capability, allowing its use as an unattended network station.

For communication flexibility, the DUV11 can be used with character-oriented protocols. It allows program control over character size (five, six, seven, or eight bits) in addition to programmable sync characters. Output characters are transferred in parallel from the LSI bus into the DUV11 where they are serially shifted to the communication line. Input characters are shifted into the DUV11 and made available to the bus on an interrupt basis as parallel data.

The DUV11 is also capable of isochronous operation, which is essentially asynchronous data transmission over a synchronous modem. For isochronous operation, start/stop bits are added to each transmitted character. Isochronous transmission data rates are higher than asynchronous rates.

DZ11 and DZV11 Serial-Line Interfaces

The DZ11 is a serial-line multiplexer that connects the UNIBUS with up to eight asynchronous serial lines. Character formats and operating speeds are programmable on a per-line basis—each line can run at any one of 15 speeds. The DZV11 is a version of the DZ11 that connects up to four asynchronous serial data communications lines to the LSI-11 bus. These interfaces are a low-cost method of connecting multiple local remote terminals for moderate throughput.

Local operation with EIA terminals is possible at speeds up to 9600 baud. The DZ11 and DZV11 can be used with dialup, full-duplex terminals that operate through most industry-standard modems that run at 300 bits per second or at 1,200 bits per second. Incoming characters are stored in a receiver silo buffer. Outgoing characters are processed on a programmed interrupt-request basis. An optional feature of these interfaces is parity checking. A parity bit is generated on output and checked on input.

DZS11 Statistical Multiplexer

The DZS11 Statistical Multiplexer is a terminal concentrator that connects up to eight asynchronous terminals to a UNIBUS PDP-11 system using one synchronous, full-duplex communication link. It substantially reduces the costs of cables, modems, and leased lines by combining several low-speed lines into a single high-speed data path.

Designed for a variety of communications applications, the DZS11 can be used to connect terminals at local and remote sites. A route-through capability also makes it possible for a cluster of terminals at one remote location to pass data on to other terminals at a second remote site.

The DZS11 can replace up to 16 modems and eight separate telephone lines with two modems and one synchronous line. For connections less than 1,056 yards (one kilometer) apart, all modems can be eliminated by using one RS422 "long line" cable.

By allocating one high-speed link dynamically among users in proportion to the communication traffic each generates, the statistical multiplexer makes it economically feasible to have remote terminals communicate as fast as local terminals. Instead of separate 300-baud dialup lines, a single multiplexed line operating at up to 19,200 baud can service each terminal at 4800 or 9600 baud.

More than one DZS11 can be configured on a system. To the host, all eight terminals linked to a DZS11 appear to be attached by one standard DZ11 multiplexer. It makes no difference whether the terminals are connected by cables or combinations of cables and modems or if they are clustered in one location or spread across two sites.

The terminal concentrator is composed of two parts. One is installed in the host processor as a UNIBUS device, and the other fits inside a VT100 video terminal. The remote module inside the VT100 acts as the cluster controller and accepts up to seven additional user terminals as input devices. A second remote multiplexer can be installed at another site. In this arrangement, the first remote multiplexer services the terminals connected directly to it and transparently routes all messages to the second terminal cluster.

KMC11 Auxiliary Communications Processor

The KMC11, an auxiliary processor complete with memory, interfaces to the PDP-11 UNIBUS. It improves the performance of PDP-11 systems by performing time-consuming system functions in parallel with the PDP-11 CPU, thereby offloading the main CPU. It is especially suited to controlling I/O operations that require extensive intelligence—for example, data communications and analog I/O.

The KMC11's architecture is specifically suited to data movement, character processing, address arithmetic, and other functions necessary for controlling I/O devices, formatting data, and processing communications. The functions performed by the KMC11 are determined primarily by the microprogram in the KMC11's control memory. This control memory is volatile and can be changed whenever desired by the PDP-11 processor. In normal operation, the operating system loads the microprogram into the KMC11 control memory as part of system initialization; it remains in the control memory until the system is reinitialized.

Software support of the KMC11 consists of two parts: the KMC11 microprogram and the operating system driver. The microprogram must be tailored to the specific processing to be performed by the KMC11. The operating system driver interfaces the microprogram to the rest of the PDP-11 software. Communication between the microprogram and the operating system uses the KMC11 control status registers and is entirely defined by the software. Different applications may require different types of microcode/software interfaces.

PCL11-B Parallel Communications Interface

PCL11-B Parallel Communications Link hardware is supported via two drivers. One driver supports the transmitter function and the other driver supports the receiver function. The PCL11-B is a hardware interface that functions as a time-division-multiplexed (TDM) interface over which several PDP-11 computers can transfer data to each other. Each PCL11-B consists of a transmitter, receiver, and master section. The transmitter section can transfer parallel 16-bit words along the TDM bus to a receiver section of a separate PCL11-B on a different PDP-11 computer's UNIBUS. One of the PCL11-B units attached to the TDM bus must have its master section enabled to effect the data transfer.

The PCL11 transmitter driver has two basic functions. First, it receives data sent by the attached task and stores

it in a silo buffer in the PCL11 hardware. Second, the driver passes proper receiver address and command information to the PCL11 transmitter hardware to effect the actual transfer over the TDM bus.

The PCL11 receiver driver also performs two basic functions. It must remove data from the receiver silo and send it to the connected task. In addition, the receiver driver must acknowledge a transmitter when a data transmission is requested by that transmitter. Subsequent requests by other transmitters in the TDM bus are ignored until all message transactions with the current transmitter are completed.

REALTIME I/O DEVICES

RSX-11 was designed to satisfy the time-critical requirements of realtime applications. Optional software application packages are available to support I/O devices that take advantage of RSX-11's fast, efficient use of system resources and priority-driven operating environment. Software can range from simple FORTRAN IV extensions to nuclear medicine application packages.

DIGITAL's realtime and sensor I/O devices, as well as users' custom I/O devices, can be easily interfaced to any RSX-11 system through:

- *Programmed I/O*, where a program polls the hardware to coordinate the data transfer. This method is useful for a time-critical device to which the program must respond as soon as data are available.
- *Inline interrupt service routines*, where a program starts an I/O transfer but may continue processing. When the transfer completes, the device issues an interrupt. An interrupt service routine in the program determines whether the transfer is incomplete, complete, or has encountered an error. It then takes the appropriate action. Interrupt-driven I/O enables two or more processes to run concurrently. It does not monopolize system resources and is often the best way to service sensor or control devices such as analog-to-digital (A/D) converters.
- A *device driver* that exists as a memory image file on a mass storage device and resides in memory when it is needed to perform device I/O. Device drivers provide device independence, share processor time with other processes, and are simple to use.

The method used to interface devices depends largely on how the user wants the device to appear to system and application programs. RSX-11 documentation is available to simplify the decision-making process and to help users write both interrupt service routines and device drivers.

Realtime I/O Devices for LSI-11-Based PDP-11s

ADV11-A A/D Converter — An option for LSI-11-based processors, the ADV11-A is a 12-bit successive approximation analog-to-digital converter that samples analog data at specified rates and stores the digital equivalent value for processing. A multiplexer section can accommodate up to 16 single-ended or eight quasidifferential inputs. The converter section uses a patented auto-zeroing design that measures the sample data with respect to its own circuitry offset and therefore cancels out its own offset error.

A/D conversions are initiated by program command, clock overflow, or external events. The program control is determined by the control and status register (CSR). The clock overflow command is supplied by the KWV11-A option. External event inputs can originate at the user's equipment or from the Schmitt trigger output on the KWV11-A clock. The digital data output is routed through a buffer register to the bus, from which it can be transferred into memory. This buffer optimizes the throughput rate of the converter.

DRV11 Parallel Line Interface Units — The DRV11s are general purpose interface units for connecting parallel-line TTL or DTL devices to the LSI-11 bus. The DRV11-LP permits program-controlled data transfers at rates up to 40 Kwords per second. The DRV11-B/DRV11-BP provides direct memory access and permits data transfers at rates up to 250 Kwords per second in single cycle mode and up to 500 Kwords per second in burst mode.

The DRV11s provide LSI-11-bus interface and control logic for interrupt processing and vector generation. Data is handled by 16 diode-clamped input lines and 16 latched output lines. The device address is user-assigned, and control/status registers (CSRs) and data registers are compatible with PDP-11 software routines.

The DRV11-J program-controlled parallel-line interfaces contain 64 bidirectional input/output lines configured as four 16-bit ports. They are bit interruptable up to 16 lines. Interrupt vectors can have fixed or rotating priorities.

KWV11-A Programmable Realtime Clock — A programmable clock/counter for LSI-11 processors, the KWV11-A provides a variety of means for determining time intervals or counting events. It can be used to generate interrupts to the processor at predetermined intervals, or to synchronize the processor ratios between input and output events. It can also be used to start the ADV11-A analog-to-digital converter either by clock counter overflow or by the firing of a Schmitt trigger.

The clock counter has a resolution of 16 bits and can be driven from any of five internal crystal-controlled frequencies (100 Hz to one MHz), from a line frequency input or from a Schmitt trigger fired by an external input. The KWV11-A can be operated in any of four programmable modes: single interval, repeated interval, external event timing, and external event timing from zero base.

The KWV11-A includes two Schmitt triggers, each with integral slope and level counters. The Schmitt triggers permit the user to start the clock, to initiate A/D conversions, or to generate program interrupts in response to external events.

Realtime I/O Devices for UNIBUS-Based PDP-11s

LPA11-K — The LPA11-K is an intelligent (dual-microprocessor) direct-memory-access controller that buffers realtime data and transfers them to RSX-11 memory. Because the LPA11-K has automatic buffer-switching capability, transfers can occur continuously. Via a system call, the programmer can instruct the LPA11-K to take samples from a data source at specified time intervals. Sampling is handled by the microprocessors without the intervention of the CPU. Under RSX-11, the LPA11-K can be accessed via FORTRAN, BASIC-PLUS-2, and MACRO.

The LPA11-K operates in two distinct modes: multirequest mode and dedicated mode.

In multirequest mode, up to eight requests can be active concurrently. Each user's sampling rate is a user-specified multiple of the common realtime clock rate; thus, independent rates can be maintained for each user. Each request specifies the device so that analog-to-digital, digital-to-analog, or digital I/O can be synchronously sampled. The transition of a bit in a digital word can synchronize the sampling with a user event. In multirequest mode, throughput is determined by the number and types of requests. The aggregate throughput rate for all users is typically 15,000 samples per second.

In dedicated mode, one user can sample from analog-to-digital converters or can output to a digital-to-analog converter. Two analog-to-digital converters can be controlled simultaneously. Sampling is initiated by an overflow of the realtime clock or by an external signal. Two sampling algorithms are implemented. One, at each overflow, samples both of the analog-to-digital converters in parallel, allowing two channels to be sampled simultaneously. The other algorithm samples the two converters on an interleaved basis, beginning with the first whose sampling begins on alternate clock overflows.

The LPA11-K supports the following K-series laboratory peripherals on RSX-11:

- AA11-K four-channel 12-bit D/A converter with scope control
- AD11-K 16-channel 12-bit A/D converter
- ADK11-KT 16-channel 12-bit A/D converter with realtime clock, distribution panel, and cables
- AM11-K multiplexer board
- DR11-K 16-bit parallel general device interface
- KW11-K realtime clock

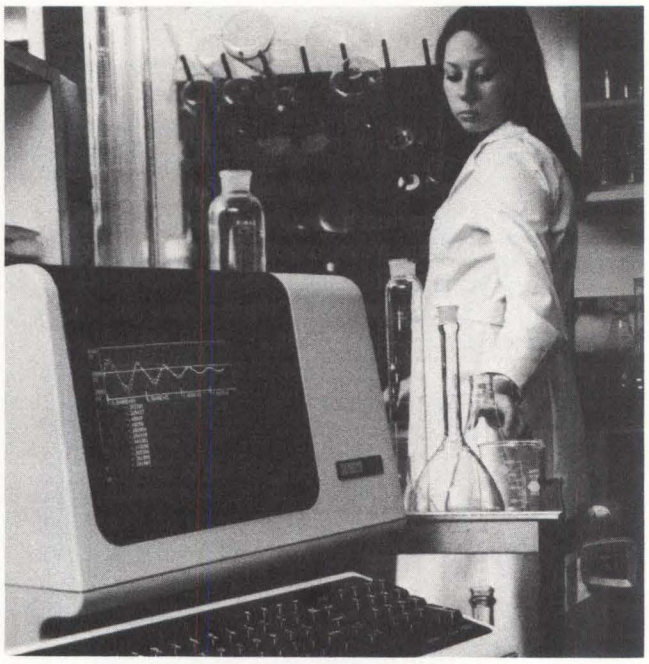
K-Series Laboratory Peripherals — K-series laboratory realtime I/O options are supported through a set of program-callable routines that are linked with the user's task at taskbuild time. These routines are highly modular; therefore, a particular task contains only the code necessary for the facilities actually used. There is also support for directives that allow a user's task to bypass normal QIO processing and perform I/O almost completely independent of the Executive, thus increasing I/O throughput.

AA11-K Digital-to-Analog Converter The AA11-K is a four-channel, 12-bit digital-to-analog converter that interfaces to the PDP-11 UNIBUS. The AA11-K includes four D/A converters and a display control. The display control permits users to display data in the form of a 4,096-by-4,096 dot array. Under program control a dot can be produced at any point in the array, and a series of sequential dots can be programmed to produce graphic output to a chart recorder, X/Y recorder, or video terminal.

Each of the AA11's four D/A converters are driven from a 12-bit buffer register and from circuitry that provides all controls necessary to output the analog signals to an external oscilloscope or other analog device. D/A output is nominally ±5 V; however, this can be set to ±0.5 V or ±10 V. Outputs are capable of driving up to 5,000 pF of load.

Output operations are accomplished by loading the display status register and the D/A buffer registers. Through display status register bits, a user can:

- change the contents of the D/A registers
- provide delays necessary for some oscilloscope applications
- provide erase, write-through, and nonstore control functions for storage oscilloscope applications
- enable interrupt on completion of oscilloscope intensification, erase, and external delay



The display control offers four program-controlled modes in which the oscilloscope can intensify a point. Jumpers provide additional display-control flexibility by allowing a user to select the desired delay, intensification pulse polarity, magnitude, and duration.

AD11-K A/D Converter The AD11-K A/D converter can be switch-selected to operate as a 16-channel single-ended, 16-channel pseudodifferential, or eight-channel true differential A/D converter. It can convert an analog voltage from within the input ranges of ±5 V, ±5.12 V, ±10 V, or ±10.24 V, 0 to 10 V—or from zero to 10.24 V—to a digital number for processing. These input ranges are jumper-selectable.

The AD11-K can be started under program control, on overflow of the KW11-K programmable clock (described below), or from external input. With this versatility, this package can be adapted to most applications.

ADK11-KT Analog Realtime Data Acquisition Package The ADK11-KT is a low-cost, 12-bit analog-to-digital converter that accepts analog signals from laboratory systems and converts them to digital values.

The ADK11-KT package consists of:

- an AD11-K A/D converter

- a KW11-K dual programmable realtime clock
- a distribution panel
- cables

The rack-mountable distribution panel provides a complete instrumentation interface package. The distribution panel is interfaced to the AD11-K and the KW11-K. Signals from user devices are interfaced to the panel by screw terminals.

AM11-K Expander or Switch Gain Multiplexer The AM11-K is a multiplexer expander that supplements the 16-channel single-ended (eight differential) analog input multiplexer in the AD11-K. The expansion is done in three independent groups on the AM11-K. Each group can be set to 16 single-ended or pseudodifferential input channels or eight differential input channels. Each group can have a gain of one, four, 16, or 64 assigned to it by a switch in the amplifier.

The AD11-K and AM11-K together can provide 64 (32) channels of A/D input or 16 (eight) channels of programmable gain inputs. The inputs to the AM11-K can be selected in groups of 16 (eight) channels. If the three groups of channels are selected to provide a gain of four on channels 16–31 (20_8–37_8), a gain of 16 on channels 32–47 (40_8–57_8), and a gain of 64 on channels 48–64 (60_8–77_8), the combination of the AD11-K and AM11-K can provide programmable gain for 16 (eight) channels.

DR11-K General Device Interface The DR11-K is a general purpose interface capable of parallel transfer of up to 16 bits of data between a UNIBUS and an external device. All interfacing lines to and from the DR11-K are fused and have over-voltage protection.

KW11-K Dual Programmable Realtime Clock The KW11-K is a dual programmable realtime clock used in PDP-11 UNIBUS computers. Its clocks can be used to generate interrupts to the processor at predetermined intervals, synchronize the processor to external events, or measure time intervals.

Clock A is a 16-bit clock that can be program-selected to operate at one of eight clock rates. Clock B is an eight-bit programmable realtime clock that counts intervals of time or events. Clock B can generate a program-controlled interval or can provide an input frequency for clocking Clock A. Both clocks have five crystal-controlled clock rate frequencies (one MHz, 100 KHz, ten KHz, one KHz, and 100 Hz). Other clock rates can be determined by an external input, line frequency, or, in Clock A, the overflow of Clock B.

AR11 Analog Realtime System

The AR11 Laboratory Peripheral System is a modular realtime subsystem used for the acquisition and/or output of laboratory analog data. It is a compact, program-controlled analog realtime system useful for such applications as biomedical research, analytical instrumentation, data collection, monitoring, data logging, industrial testing, and engineering.

The AR11 consists of three major components:

- a 16-channel single-ended ten-bit analog-to-digital converter
- a programmable realtime clock
- a display control with dual ten-bit digital-to-analog converters

Through the A/D converter the AR11 can sample analog data at specified rates and store the equivalent digital value for subsequent processing. The basic subsystem consists of a 16-channel multiplexer (16 single-ended inputs), sample-and-hold circuitry, and a ten-bit A/D converter. The analog inputs can be programmed for unipolar (zero to five V) or bipolar (±2.5 V) operation.

An A/D conversion can be started under program control, on overflow of the realtime clock, or by an external input.

The programmable clock offers several methods for accurately measuring and counting time intervals or events. With it, many operations can be performed concurrently. These include synchronizing the central processor to external events, counting external events, providing interrupts at programmable intervals, and starting the A/D converter.

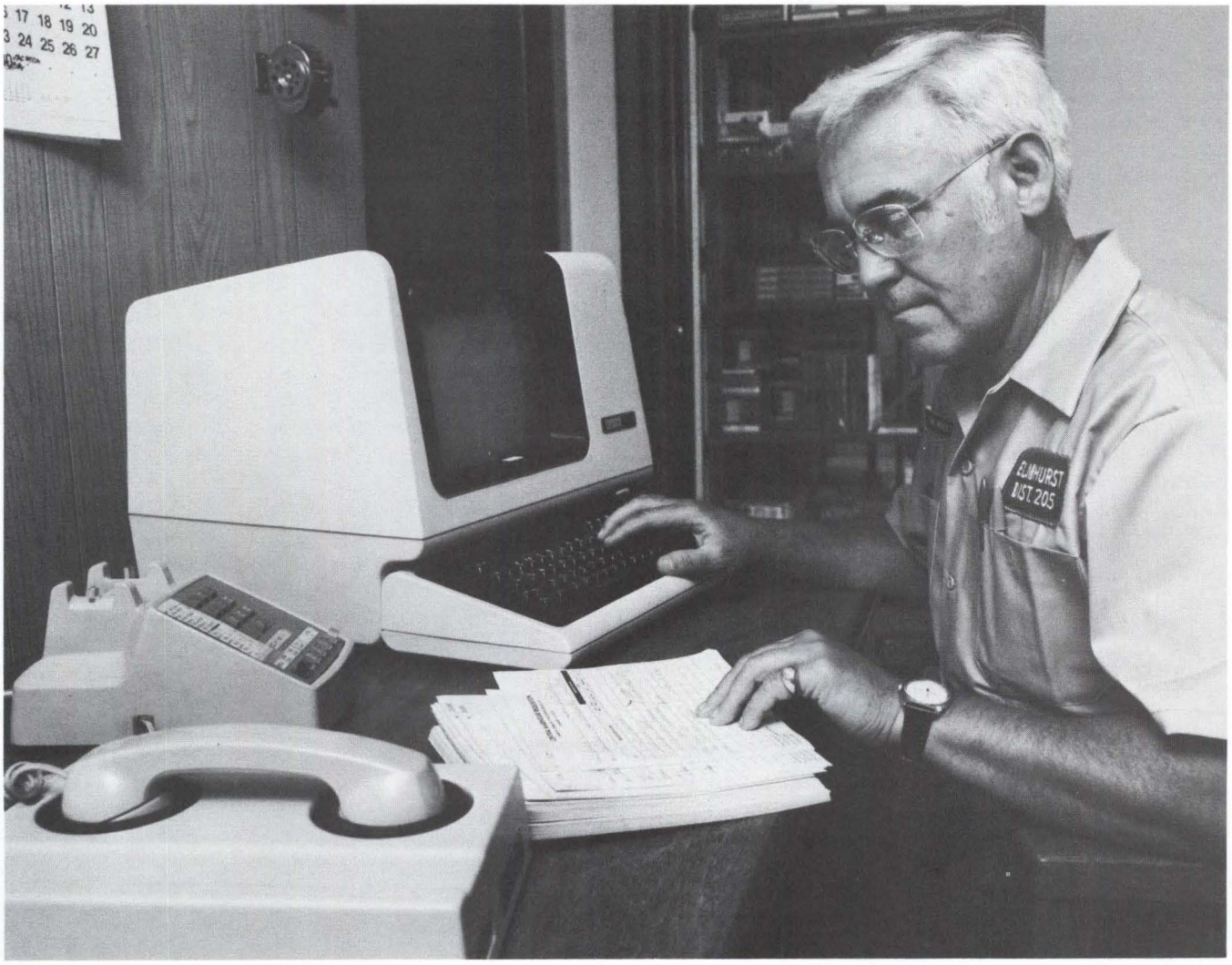
The clock can be programmed to operate in either single-interval or repeated-interval modes and to run at any of five crystal-controlled frequencies (one MHz, 100 kHz, ten kHz, one kHz, or 100 Hz).

The display control is used to display data in a 1,024-by-1,024 dot array. Under program control, a bright dot can be produced at any point in the array, and a series of dots can be programmed to produce graphic output. The display control can output to either an X/Y recorder or a video terminal display unit.

The display control consists of two ten-bit D/A converters, each driven from a ten-bit buffer register, and circuitry that provides all controls necessary to output the analog signals to an external oscilloscope. D/A output is nominally ±5 V; however, this can be set to ±0.5 V by means of jumpers and to any range between ±0.5 V and ±5 V by means of resistors.

Through the AR11's display status register bits, a user can intensify a point; provide delays necessary for some scope applications; provide erase, write-through, and nonstore control functions for storage scope applications; and enable an interrupt on completion of scope intensification.

10 Data Communications



Distributed data processing can improve and organize the flow of information throughout an organization. In a distributed processing network, different computers in different locations work together, exchanging information and sharing resources.

DIGITAL has developed the DIGITAL Network Architecture (DNA) to provide a wide variety of products that increase a user's networking flexibility. DNA allows a DIGITAL computer to be linked to another DIGITAL computer; a DIGITAL computer to be linked to another vendors' computer; and a DIGITAL network to be linked to non-DIGITAL networks.

In addition, DNA permits these systems to be connected in whatever configuration best suits the user's organization and networking needs. People at terminals, workstations, or personal computers at any point in a DIGITAL network can log on and perform network functions with a single set of commands, regardless of whether their computers are linked by public packet-switched networks, by Ethernet cables, or by DIGITAL Data Communications Message Protocol (DDCMP) multipoint or point-to-point connections.

INTRODUCTION

DIGITAL's networking architecture offers a broad range of compatible networking options. This varied set of hardware and software products means that wherever processors, terminals, and other information processing equipment are located—within a manufacturing plant or spread around the world—they can be connected in ways that allow both the exchange of information, files, programs, and control and the sharing of peripheral devices. When combined in networks, small systems can access the powerful capabilities of mainframes, and large systems can take advantage of small systems dedicated to specialized applications.

DIGITAL-to-DIGITAL computer networks are established through DECnet, a family of software products, protocols, interfaces, and support services designed specifically to link DIGITAL computer systems. Such a network can contain computers that run the same or different operating systems. A DECnet network can include, within certain constraints, DIGITAL's Professional 350 personal computers; PDP-11s running any of the RSX-11 operating systems, the RSTS/E timesharing operating system, and/or the RT-11 single-user realtime operating system; any of the VAX systems running the VAX/VMS multifunction virtual memory operating system; and DECsystem-10 and DECSYSTEM-20 mainframe systems running TOPS-10 and TOPS-20. Each DIGITAL computer and operating system in a DECnet network can specialize in solving local problems.

Using DECnet, various kinds of networks can be constructed to facilitate remote communications, resource sharing, and distributed computation. DECnet is highly modular and flexible. It is a set of services from which a user selects those appropriate to build a network that will satisfy the requirements of a particular application.

The connection of DIGITAL's systems to computers built by other manufacturers to form a network is supported by a family of products called Internets. The Internet products emulate communications protocols recognized and supported by IBM, UNIVAC, and Control Data Corporation.

The DECnet/SNA Gateway is a special purpose computer system used to connect a DECnet network to an IBM SNA network. Using the SNA Gateway, users on DIGITAL systems in a DECnet network can access or update their IBM mainframe database, perform remote-job-entry tasks, and initiate program-to-program communication, without sacrificing the flexibility and ease of use of their DECnet network.

Packetnet products interconnect computer systems, networks, or terminals through public packet-switched networks. Public packet-switched networks provide an inexpensive means of moving low- to medium-volume data traffic between different sites. Rather than paying for the line, users pay only for the data actually sent. The common carrier provides the communications line and ensures system-to-system message delivery. The Packetnet products implement the X.25 recommendation for interfacing equipment to the packet-switched network.

DIGITAL also has incorporated Ethernet local area network technology into the DIGITAL Network Architecture (DNA) and DECnet. Developed jointly by DIGITAL, Intel, and Xerox, Ethernet is a specification for the physical implementation of local area communications. It allows high-speed, reliable, high-bandwidth communication among a large volume of information processing equipment in a limited geographical area. DIGITAL's Ethernet program will produce a complete set of products for local area networks over the next few years.

Before these networking options are discussed in more detail, it is important to understand some basic networking concepts.

NETWORK CONCEPTS

In a network, computers and other information processing equipment communicate among themselves either remotely or locally. A network is a configuration of two or more independent information processing devices, called nodes, that are linked together to facilitate remote communication, to share resources, and/or to perform distributed processing.

A node is a point in a network through which a user can gain access to the network or where network work may be done. In network schematics, nodes are the endpoints of the communications links that join the network. Communications links may be dedicated, leased, or dialed-up local or remote lines or satellite or microwave beams.

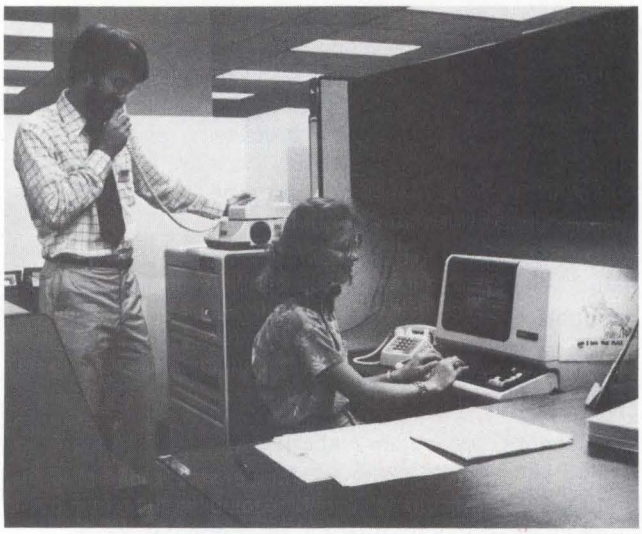
Adjacent network nodes are linked together via carriers known as physical links. Physical links can be relatively permanent bonds, such as telephone lines or cable wires laid from one node to another, or they can be temporary connections that change with each use, such as dialed-up telephone calls.

In a network, several tasks can use the same physical link to exchange data. This means more than one data path can be handled simultaneously by a physical link. The data path is called a logical link.

A variety of computer networks can be implemented using DECnet, Internet, Packetnet, and Ethernet products. Networks typically fall into one of three classes, described below. The variety of networking products available on RSX-11 provides a choice of ways to implement each type of network.

Communications networks exist to move data from one, often distant, physical location to another. The data can be file-oriented (as is often the case for remote job entry systems) or record-oriented (as occurs with the concentration of interactive terminal data). Interfaces to common carriers, using both switched and leased-line facilities, are normally a part of such networks. These networks are often characterized by the concentration of all user application programs and databases on one or two large host systems in the network. Figure 10-1 illustrates a communications network.

Resource-sharing networks enable expensive computer resources to be shared by several computer systems. Shared resources not only include such peripherals as mass storage devices, but also include logical entities, such as a centralized database that is made available to other systems in the network. These networks are often



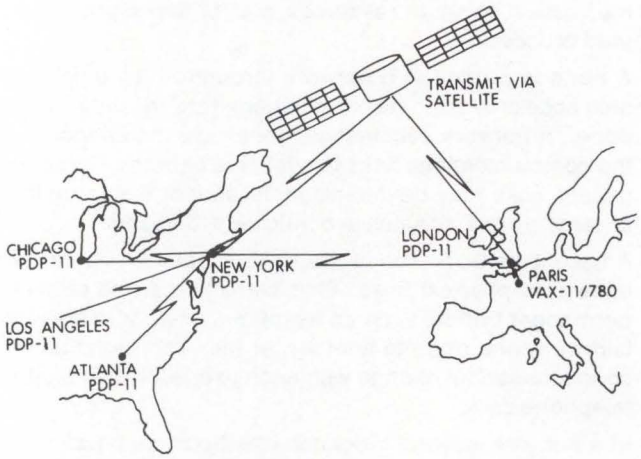


Figure 10-1
Communications Network

characterized by the concentration of high-performance peripherals, extensive databases, and large programs on one or two host systems in the network; typically, the satellite systems have less expensive peripherals and smaller programs. Figure 10-2 illustrates a resource-sharing network.

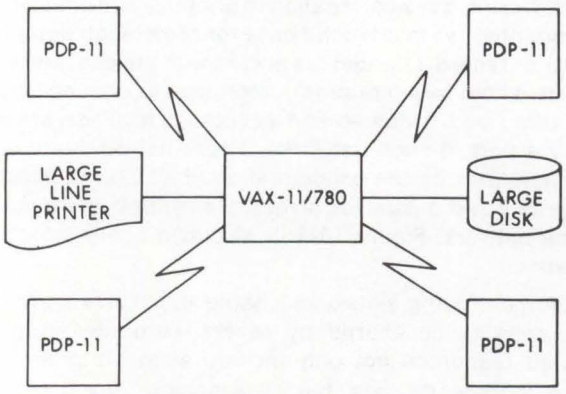


Figure 10-2
Resource-Sharing Network

Distributed computing networks coordinate the activities of several independent computing systems and the exchange of data among them. Networks of this nature can have specific geometries (star, ring, or hierarchy), but often have no regular arrangement of links and nodes. These networks are usually configured so that the resources of a system are close to the users of those resources. Distributed computing networks are usually characterized by multiple computers with applications programs and databases distributed throughout the network. Figures 10-3 and 10-4 illustrate two such networks.

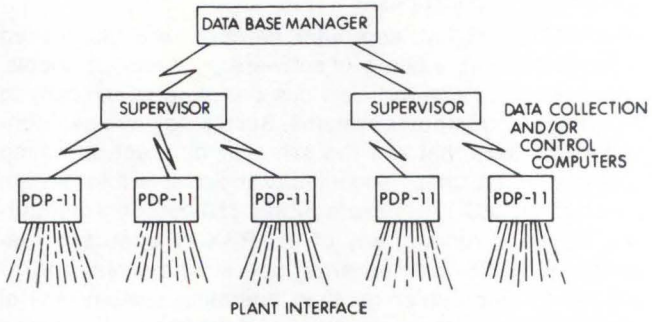


Figure 10-3
Typical Manufacturing Network

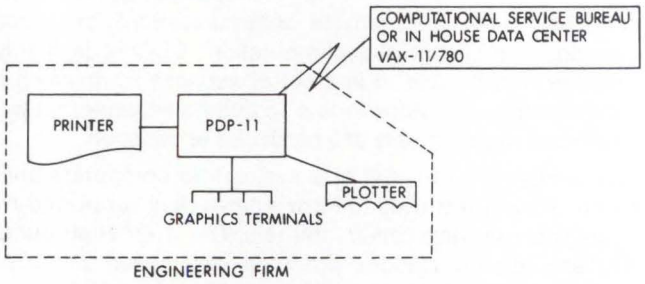


Figure 10-4
Computational Network

DIGITAL NETWORK ARCHITECTURE

The DIGITAL Network Architecture (DNA) is the framework for all of DIGITAL's communications products. It is a model of structure and function upon which DECnet implementations are based.

Consistent with DIGITAL's approach of implementing layered products based on international standards, DIGITAL designed the layers of its network architecture to correspond to those of the International Standards Organization's Open System Model.

Each DNA layer is designed to fulfill specific functions within the network. Collectively, these protocols govern the format, control, and sequencing of message exchange for all DNA implementations. DNA controls all data that travels throughout a network and provides a modular network design.

DNA consists of eight layers.

- The *User* layer, the highest layer in the architecture, contains most user-supplied functions. It also contains the Network Control Program, a Network Management module that gives system managers interactive terminal access to lower layers.
- Modules in the *Network Management* layer provide user control of and access to operational parameters and counters in lower layers. Network Management also performs downline loading, upline dumping, remote system control, and test functions. This layer is the only one that has direct access to each lower layer.
- The *Network Application* layer provides generic services to the User layer. Services include remote file access, remote file transfer, remote interactive terminal access, gateway access to non-DNA systems, and resource managing programs. This layer contains both user- and DIGITAL-supplied modules. Modules execute simultaneously and independently in this layer. The Network Application layer contains functions of both the Application and Presentation layers of the ISO Reference Model.
- The *Session Control* layer defines the system-dependent aspects of logical link communication. A logical link is a virtual circuit (as opposed to a physical one) on which information flows in two directions. Session Control functions include name-to-address translation, processing addressing, and, in some systems, process activation and access control. The Session Control layer corresponds to the Session layer of the ISO Reference Model.
- The *End Communication* layer is responsible for the system-independent aspects of creating and managing logical links for network users. End Communication modules perform data-flow control, end-to-end error control, and the segmentation and reassembly of user messages. The End Communication layer corresponds to the Transport layer of the ISO Reference Model.
- Modules in the *Routing* layer route user data, called a datagram and contained in a packet, to its destination. Routing modules also provide congestion control and packet lifetime control. The Routing layer corresponds to the Network layer of the ISO Reference Model.
- Modules in the *Data Link* layer create a communication path between adjacent nodes. Data Link modules ensure the integrity of data transferred across the path. Data link modules for Ethernet local area networks, X.25 public packet-switched networks, and synchronous or asynchronous lines execute simultaneously and independently in this layer.
- Modules in the *Physical Link* layer manage the physical transmission of data over a channel. The functions of modules in this layer include monitoring channel signals, clocking on the channel, handling interrupts from the hardware, and informing the Data Link layer when a transmission is completed. Implementations of this layer encompass parts of device drivers for each communication device as well as the communication hardware itself. The hardware includes devices, modems, and lines. In this layer, industry standard electrical signal specifications such as EIA RS-232C, CCITT V.24, the Ethernet layer, or CCITT X.25 level 1 operate rather than peer protocols.

DNA determines how DECnet software modules interact vertically with one another. The interfaces between modules in separate layers of the same node are precisely defined. Reflecting the structure of DNA, DECnet modules are like building blocks. Within each node, a layer contains only those modules required to support modules in higher layers.

In addition to defining vertical interfaces, DNA also defines the relationship between modules in separate nodes. A module in one node communicates only with a module in the same layer that is serving the same function in another node.

Communication between these modules is governed by protocols. The protocols define the form and content of messages to be exchanged by modules. Table 10-1 lists and briefly describes the function of each DNA protocol.

Table 10-1
DNA Protocols

Protocol	Description
Network Management Layer	
Network Information and Control Exchange Protocol (NICE)	Used for triggering downline loading, upline dumping, testing, reading parameters and counters, setting parameters, and zeroing counters.
Event Logger Protocol	Used for recording significant events in lower layers. An event could result from a line coming up, a counter reaching a threshold, and a node becoming unreachable, for example.
Maintenance Operation Protocol	Performs data-link-level loopback tests, remote control of unattended systems, and downline loading and upline dumping of computer systems without mass storage.
Network Application Layer	
Data Access Protocol (DAP)	Used for remote file access and transfer.
Network Virtual Terminal Protocol	A family of protocols used for terminal access through the network.
X.25 Gateway Access Protocol	Allows a node that is not connected directly to a public data network to access the facilities of that network through an intermediary gateway node.
SNA Gateway Access Protocol	Allows a node that is not connected directly to an IBM SNA network to access the facilities of the SNA network for terminal access and remote job entry.
Loopback Mirror Protocol	Used for Network Management logical-link loopback tests.

Session Control Layer	
Session Control Protocol	Used for such functions as sending and receiving logical-link data and disconnecting and aborting logical links.
End Communication Layer	
Network Services Protocol (NSP)	Handles all the system-independent aspects of managing logical links.
Routing Layer	
Routing Protocol	Handles routing and congestion control.
Data Link Layer	
DIGITAL Data Communications Message Protocol (DDCMP)	Ensures the integrity and correct sequencing of messages between adjacent nodes.
X.25 Protocol	Implements the X.25 packet level (level 3) and X.25 frame level (level 2) of the CCITT X.25 recommendation for public data network interfaces.
Ethernet Protocol	Implements the Ethernet data link protocol for communication between adjacent nodes connected by an Ethernet local area network.

DNA does not define protocols for all functional layers. For example, User Layer programs communicate over the network according to rules defined by the programmer. Furthermore, more than one protocol can be defined for the same layer because some layers support more than one function. For instance, the Network Application layer can include modules that use the Data Access Protocol (DAP) as well as modules that use a protocol defined by users for a specific application.

DECNET

DECnet is a family of communications software and hardware products that enable DIGITAL operating systems and computers to function in a DECnet network. A DECnet network is a group of DIGITAL computer systems with associated operating systems, DECnet software, and communication hardware that are connected to each other by physical channels, called lines.

Since 1974, DECnet has been growing as new functions have been phased in on a regular, planned basis.

DECnet Phase I, which allowed similar DIGITAL computer systems to work together, included task-to-task transactions, point-to-point communications, downline system loading, and downline task loading.

DECnet Phase II extended Phase I functions to heterogeneous networks so DIGITAL's computers running different operating systems could work together. DECnet Phase II also provided the capabilities and security features needed in many small networks: file transfer, remote resource sharing, and remote command submission.

DECnet Phase III provided alternatives for configuration flexibility and communications cost saving. Adaptive routing and multipoint communications reduce the need for direct links between every pair of communicating computers in a network. True network support tools—Network Management—were also available with DECnet Phase III. With Network Command Terminals, users at terminals could access remote systems running the same operating system as if they were local.

Phase IV, the most recent addition to DECnet's capabilities, provides the ability to support up to 1,000 nodes and extends the Network Command Terminal facility to include heterogeneous systems. In addition, DECnet Phase IV products now support connections to remote DECnet nodes over a public packet-switched network using the X.25 protocol and over Ethernet lines.

RSX DECnet Communication Software

The following networking products are available to link RSX-11 systems to other DIGITAL computers:

- DECnet-11M, for RSX-11M systems
- DECnet-11M-PLUS, for RSX-11M-PLUS systems
- DECnet-11S, for RSX-11S systems

These RSX DECnet products offer Phase IV networking capabilities. RSX DECnet Phase IV is completely compatible with Phase III, and RSX DECnet Phase IV products provide all Phase III capabilities.

RSX DECnet interfaces are standard with the RSX DECnet products. To program task-to-task communication or remote file access, programmers use identical calls whether the tasks or data are on the same or on different systems. The logical link between two programs is like an I/O channel over which programs can send and receive data. Using DECnet for task-to-task communication is like doing I/O with an existing driver.

RSX DECnet Capabilities

The network functions available to the RSX DECnet user depend, in part, on the configuration of the rest of a DECnet network. DIGITAL offers DECnet products for all our major PDP-11 operating systems, for VAX/VMS, and for TOPS-10 and TOPS-20. Each DECnet product offers its own functions and its own set of features to the user. Networks that combine RSX DECnet nodes with other DECnet products could limit the functions available to the RSX DECnet user because some RSX DECnet features may not be supported by all DECnet products. Conversely, the user of another DECnet implementation will not necessarily have access to all RSX DECnet functions. Users should refer to DIGITAL's Software Product Descriptions for the most current information on supported capabilities of all DECnet implementations.

The goal of RSX DECnet is to provide a network capability that is extremely easy to use and that eases the incorporation of added processing power when user requirements dictate. One example of making DECnet easier to use is the use of node name aliases (that is, users can assign shortened forms to node names), which reduces typing

time when specifying remote node names and access control information. Task-to-task communication and file access between systems is virtually transparent; these intersystem facilities appear to be no different from the intrasystem communication and file access utilities.

Below are descriptions of the user/program-level and communication-level capabilities provided by RSX DECnet Phase IV.

User/Program Level —

Task-to-Task Communication Cooperating programs can exchange data using RSX DECnet. An RSX-11 user program written in MACRO-11, COBOL, FORTRAN-77, FORTRAN IV, or BASIC-PLUS-2 can exchange messages with other network user programs. These two user programs can be on the same node, on adjacent Phase III/Phase IV nodes, or on any two nonadjacent Phase III/Phase IV nodes. The messages sent and received by the two user programs can be in any data format.

Access Control Access control is the method by which network users are screened before gaining access to network facilities. With the appropriate access-control information, a user program can log into a remote system and access any of the remote system's resources. The accessing program must have either an account or access to a guest account on the remote system, to log in successfully. The access-control information consists of a user identification code or name, a password associated with the user identification, and additional accounting information that may be required by the remote system.

Remote File Access All DECnet systems support exchange of sequential ASCII and binary files. DECnet nodes do not have to use compatible file syntaxes. The DECnet software translates the file syntax of the sending node into a common network syntax and then retranslates at the receiving end appropriately for that node.

The Remote File Access capability is implemented by such features as file transfer, remote command file submission/execution, downline taskloading, and interterminal communication.

Both DECnet-11M and DECnet-11M-PLUS support file transfers between locally supported File Control Services (FCS) devices and the file system of other DECnet nodes. Wildcards can be used for the user identification code, filename, filetype, and version number for local-to-remote file transfers. Directory listings are another supported feature.

RSX DECnet also supports Record Management Services (RMS). RMS support on RSX DECnet means a remote DIGITAL node with RMS can access sequential, relative, and indexed files on the local RSX node. Indexed files allow the user to access multikeyed files by a variety of keys.

In addition, the RSX DECnet Network File Transfer program supports block-mode transfers between RSX DECnet nodes. These transfers send files in 512-byte blocks across the network, rather than sending them record by record. Block mode increases the speed of the transfer, and allows RMS relative and indexed files to be transferred between RSX DECnet nodes. With this program, users can also rename files and change their protection across the network.

Additional facilities available on DECnet-11M and DECnet-11M-PLUS allow system command files to be submitted to a remote node. The list of commands must be in a format acceptable to the node responsible for the execution. Similarly, command files can be received from other systems and then executed. On DECnet-11M-PLUS systems, support for batch files can also be selected.

RSX DECnet also supports the remote File Transfer Spooler, which allows a series of file transfers to be queued for sequential transmission of files to and from remote nodes.

With downline taskloading, programs to be executed on DECnet-11S nodes in the network can be stored on host RSX-11M, RSX-11M-PLUS, or VAX/VMS systems and loaded into the DECnet-11S system in response to local-program or operator requests. Programs already executing on DECnet-11S nodes can be checkpointed to the host file system and later restored to main memory of the DECnet-11S node. Overlays for DECnet-11S tasks can also be stored on RSX-11M/M-PLUS and VAX/VMS file system devices. These features simplify the operation of RSX-11S systems in the network that do not have mass storage devices.

For interterminal communication, users can send and receive messages to and from terminals at remote or local nodes with the TLK utility. Because TLK does not access files, no access control information is required. There are two modes for TLK: single-message mode and dialogue mode.

Network Command Terminals Terminal users can log onto a remote DECnet node and execute commands as if they were actually typing at a terminal on that node. Once logged on, they can perform all normal commands on the remote system. For example, they can use the editor EDT over the network. RSX Phase III supported homogeneous virtual terminal communications—the local user could log onto remote systems running the same operating system. RSX Phase IV supports heterogeneous virtual terminals communications—the local user can log onto remote systems running different operating systems. This means that everyone on a DECnet network can make use of all the network's computer resources, not just those on the system to which they're directly connected.

An operator can use all the standard system and network utilities supported by the remote system. Interactive access is independent of physical connection; accessing can be across routing and/or multidrop networks.

Network Management Tools for monitoring and controlling network operation are the substance of Network Management. To make generating the network easier, RSX DECnet supports Dry Run Mode and Restore Mode. A dry run allows the operator to answer all network generation questions without actually generating a network. Restore Mode can then be used to generate a network configuration based on the specifications entered during the dry run. Restore Mode can also be used to recover from a system crash during network generation or to repeat a network generation to change the system generation.

For day-to-day operations, RSX DECnet includes facilities for tuning parameters, logging events, and testing nodes,

lines, modems, and communications interfaces. Network Management features of particular interest are loopback testing, downline system loading, and upline system dumping.

With loopback testing, a network manager can send and receive test messages over individual lines—either between nodes or through other loopback arrangements—and then compare messages. Utilities are included for a logical series of tests that aid in isolating software and hardware problems. During testing, for example, a suspected problem line can be set to service state, and messages can be routed along other paths. With the free line, the network manager can then perform test procedures, isolate the problem, and fix the line without affecting, and being transparent to, users on the network.

Downline system loading enables initial memory images for DECnet-11S nodes in the network to be stored on host RSX-11M, RSX-11M-PLUS, or VAX/VMS file system devices and loaded into adjacent nodes across point-to-point links. The initial memory image can be generated on the host first. Should the need arise, the host system can also analyze crash data dumped upline from the DECnet-11S node.

RSX DECnet also supports network management features for both Ethernet and X.25 functions. In addition, RSX DECnet interprets the full set of network management commands to control activities on remote non-RSX nodes, even if a given command is not supported in RSX DECnet.

Communications Level —

Point-to-Point A point-to-point node communicates only with adjacent nodes to which it is directly connected.

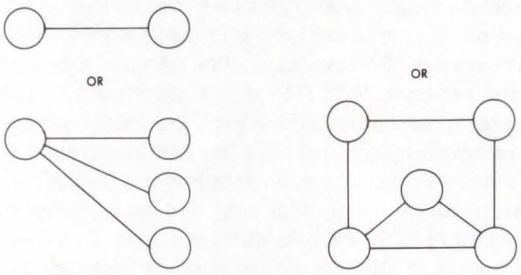


Figure 10-5
Point-to-Point (physical)

Multipoint A network party line shares time on one line with several nodes. This type of multipoint topology can reduce line costs. Multipoint configurations include a control station and tributaries. The control station controls network traffic by polling; it queries the tributary computer stations to determine if they have messages to send.

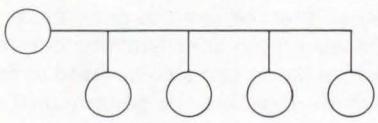


Figure 10-6
Multipoint (physical)

Routing Routing is a method for sending messages from source to destination through intermediate nodes. RSX DECnet provides adaptive path routing: messages are routed through the network over the least-cost path, as defined by the user. If either a line or a node in this preferred path fails, the network automatically routes over the next-least-cost path.

When a network is brought up or is subsequently under user control, line costs are assigned for each line leaving a node in the network. Therefore, networks can be tailored to make each node's interactions with other nodes efficient. The same line can have two different path costs depending on the direction in which the message is traveling.

Transparently to the user, this feature enables network managers to reroute traffic easily, to avoid a troublesome line, and to run diagnostics on such a line. Adaptive routing also makes it possible to install backup links, which results in fewer connections than with traditional point-to-point networks.

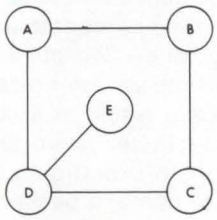


Figure 10-7
Routing

RSX DLX-11

Although not a DECnet product per se, RSX DLX-11, available on RSX-11S and RSX-11M systems, provides a low-overhead communications software line interface to a DECnet-11M or RSX-11M-PLUS Phase III/Phase IV node in a network. RSX DLX-11 is especially suited for users of small systems who want access to the network. Because of its small size, RSX DLX-11 extends the memory available to network users of DIGITAL microcomputer systems.

Representative applications include exchange of short inquiry/response messages, transfer of summary data, and transmission of status information to a central host system.

RSX DLX-11 supports a single physical line in a point-to-point or multipoint connection. A user-written MACRO-11 program at each end of the line controls the communications line directly. RSX DLX-11 provides the following user task functions.

- open the line (assign the line to a task)
- initialize the line (start the DDCMP protocol)
- close a line (deassign the line from a task)
- receive a message on the line
- transmit a message on the line
- hang up the line

Like DECnet, RSX DLX-11 is an implementation of the DIGITAL Network Architecture. It maintains the integrity and the sequence of data sent over the channel, using the DECnet DIGITAL Data Communication Message Protocol.

Unlike DECnet, it does not direct the data to the appropriate task at the receiving end. The user is responsible for all task and data synchronization, task invocation, logical link management, flow control, and session control. If these capabilities are required, DECnet should be considered.

INTERNETS

A family of products called Internets supports the connection of DIGITAL's computers to systems built by IBM, UNIVAC, and Control Data Corporation. Internets give data processing managers the freedom to choose mainframes and minicomputers on the basis of application needs, with the assurance that reliable links can be established between systems.

Internet products emulate common communications protocols. They are data transfer facilitators rather than hardware emulators. While they appear to other vendor's computers to be supported devices, they are, in fact, parts of powerful DIGITAL systems. They provide transparent communication with the equipment of other vendors and, at the same time, offer the flexibility of local file systems, many different languages, and a wide selection of computing power.

The Binary Synchronous Communications (BSC) protocol emulators can coreside with DECnet on a DIGITAL system. A user application program that communicates with an IBM system through one of DIGITAL's protocol emulators can also communicate across a DECnet network. Distributing network applications to local DIGITAL systems can result in increased local productivity.

The following lists the Internet products that are available on RSX-11 systems.

IBM Protocol Emulators
batch to system: RSX-11 2780/3780 Emulator
RSX-11M RJE/HASP
RSX-11M-PLUS RJE/HASP
interactive to system: RSX-11/3271 Protocol Emulator
interactive to network: RSX-11M/SNA Protocol Emulator

CDC Protocol Emulator
multiterminal batch/interactive to system: MUX200/RSX

UNIVAC Protocol Emulator
batch to system: UN1004/RSX

RSX-11 2780/3780 Emulator

The RSX-11 2780/3780 emulator emulates the features of an IBM 2780 or 3780 data transmission system. It accepts data transmitted from cardreaders and provides the flexibility of data transmission to and from one of DIGITAL's file-structured systems. The RSX-11 2780/3780 can print received character data on a lineprinter or can write it as files on such mass storage devices as disk. By spooling output to disk rather than transmitting at slower lineprinter speeds, connect time on the communications line can be reduced and, hence, costs can be reduced as well.

The physical unit transmitted is a block that, in turn, is divided into logical units called records. Users can transmit data in character or binary (transparent) mode.

If the user selects character format, the RSX-11 2780/3780 Emulator converts the ASCII characters to their EBCDIC equivalents before transmission. For printing, the emulator can receive a subset of EBCDIC that has an ASCII equivalent and automatically convert the EBCDIC to ASCII. The user can choose to have the EBCDIC data written to a file in binary format for later processing—for example, for refreshing a database.

RJE/HASP Protocol Emulators

The RSX-11M and RSX-11M-PLUS RJE/HASP protocol emulators are software packages for performing the standard functions of the IBM HASP Remote Job Entry Workstation. Supersets of the 2780/3780 emulators, the RJE/HASP protocols compare to the 2780/3780 protocols as follows:

	RJE/HASP	2780/3780
console support	full support	none
character compression	all characters	none
number of I/O streams	7 (input and output)	1

The RJE/HASP protocol emulators support video terminal, cardreader, cardpunch, and lineprinter. Through the DIGITAL RJE/HASP protocol emulator, operators can communicate directly with the IBM mainframe from a local terminal. This remote console support capability can be used to control and check the status of a job on the IBM host.

The RJE/HASP protocol emulator supports multiple I/O streams. Several devices and/or file transfers can be active at the same time. With this capability, called multileaving, a short job can be interleaved while a longer job is running.

RJE/HASP controls file-structured devices through use of the standard file control system. Non-file-structured devices (for example, cardreader, lineprinter, or terminal) are attached during use, and input/output requests are issued.

RSX-11/3271 Protocol Emulator

The RSX-11/3271 protocol emulator provides facilities for both program-to-program interactive communication and data pass-through 3270 terminal emulation. With it, terminal users and application programs can exchange data with a program running under IMS or CICS on an IBM 370 or 303X host. Users can benefit from distributing their applications to RSX-11M and RSX-11M-PLUS systems and still have online interactive access to IBM database facilities for information entry, retrieval, and updating. The RSX-11 system appears to the host as an IBM 3277 Model 2 terminal and 3271 Model 2 control unit connected to a multidrop synchronous line.

The terminal-emulation facility transforms VT100 terminals attached to a PDP-11 into virtual 3270 terminals so that a single terminal can be used to access both DIGITAL and IBM systems. The RSX-11 3271 Protocol Emulator allows the system manager to predefine all parameters required to connect to specificed IBM applications. Once the user invokes the emulator, the terminal appears to be connected to an IBM network. Just one keystroke returns the terminal to normal VT100 operation.

Most users adapt quickly to the minor differences between the VT100 terminal used as an emulator and an actual IBM 3270. Existing application programs will operate, in most cases, without modification.

The application-program interface to the protocol emulator provides program-to-program communications for such purposes as online database access. Programs on the IBM host use the standard interactive data communications mechanisms normally used to support 3270 terminals. On the PDP-11 end, the protocol emulator maintains the multipoint BSC protocol with the IBM host. It ensures data integrity and performs such protocol functions as processing polling requests and address selection sequences. The application program issues standard RSX-11 I/O calls to maintain data flow and is responsible for interpreting and generating meaningful data.

The RSX-11/3271 Protocol Emulator has facilities for specifying such parameters as number of units, buffer levels, and line-on and line-off. Maintenance and test features include data loopback testing, error and event logging, and a validation exercise.

RSX-11M/SNA Protocol Emulator

The RSX-11M/SNA protocol emulator provides the SNA protocol emulation required for RSX-11 systems to participate in an IBM SNA (Systems Network Architecture) environment. Users who require access to SNA systems can also take advantage of the flexibility, functionality, and data processing power of an RSX-11M general purpose computing system.

The RSX-11M/SNA protocol emulator:

- allows interactive access between a program in an IBM host and a program in an RSX-11 system
- appears to be a programmable cluster controller (a supported device) to SNA
- supports up to four lines at speeds up to 9,600 bits per second per line and up to 32 user sessions distributed across the four lines
- can coexist on a multipoint line with IBM SNA devices
- provides the flexibility of three levels of user interfaces

MUX200/RSX Multiterminal Emulator

MUX200/RSX provides communication between an RSX-11M system and a CDC 6000, CYBER series, or other host computer systems capable of using the 200 UT Mode 4A communications protocol. The product can be used to communicate to the host computer either interactively or in remote job entry (RJE) mode.

Up to 16 terminals can be connected to the host through MUX200/RSX. They can be a combination of CDC 200 UT terminals and RSX-11 terminals. Note that, in some cases, the host software restricts this number.

MUX200/RSX provides the following features.

- Output received from the host CDC system can be spooled to a lineprinter upon detection of a text string predefined by the user.
- Up to eight RSX-11 datasets can be specified for transmission to the host in a single command.
- RSX-11 terminals can be detached and used for other purposes while the software package is operating. Data received from the host that is directed to the terminal is saved to be printed out when the terminal is reattached.
- User-written tasks can replace the RSX-11 terminal and control the emulator as if the task were a terminal.

UN1004/RSX Terminal Emulator

UN1004/RSX provides communication between an RSX-11M system and a UNIVAC 1100 series or other host computer system capable of using the UNIVAC 1004 RMS-1 communications protocol.

UNIVAC designed the UN1004 protocol to provide communication between a host UNIVAC 1100 series mainframe and a remote batch terminal consisting of a keyboard, a cardreader/punch, and a lineprinter. The input for the UN1004/RSX can be from any RSX-11M-supported peripheral that can store a UNIVAC batch stream.

UN1004/RSX communicates with the host by using the RMS-1 communications protocol supporting ASCII line codes. The terminal emulator provides for one synchronous communication circuit to a host computer system. The line can be a single switched or dedicated leased-line carrier facility at speeds up to 4,800 bits per second.

The product provides the capability to send data in 80-column-card format and to receive data in line format or card format. Data transfers are controlled by console commands entered at the emulator terminal. The emulator permits the terminal operator to direct received data to any RSX-11M-supported device. This product can be compared functionally to the 2780/3780 protocol emulator.

DECNET/SNA GATEWAY

The DECnet/SNA Gateway, a small packaged PDP-11 system, links DIGITAL and IBM network environments, rather than merely providing single-function communications emulation between two computers. In effect, the DECnet/SNA Gateway extends a DECnet network to include IBM systems connected by an SNA network, allowing users to take advantage of the complementary strengths of both environments. It does so by combining three essential translator functions (remote job entry, 3270 terminal emulation, and applications program interface) into a small packaged PDP-11 front-end processor that's attached as a DECnet node.

Through the Gateway, users can access an IBM mainframe database to perform remote job entry tasks and to initiate program-to-program communication, without sacrificing the flexibility and ease of use of their DIGITAL network. The Gateway frees users to expand either the SNA or the DNA side of their total distributed processing operation without jeopardizing their present hardware and software investment.

Like DECnet itself, the Gateway is transparent to end users. They can perform many tasks from DIGITAL workstations and VT100 terminals as if these devices were integral parts of the SNA network.

The DECnet/SNA Gateway comprises the PDP-11 system with floppy disks and a terminal; DECnet software; an SNA communications kernel; and four server software modules, one for each of the supported DECnet/SNA functions: remote job entry, 3270 terminal emulation, user application interface, and network management.

Remote Job Entry

The Remote Job Entry facility effectively turns a DECnet node into a remote SNA workstation or group of workstations that can transmit batch jobs to an IBM host and receive job output. Users on DECnet nodes can prepare batch jobs in files, submit the jobs through the Gateway, and obtain the job's output.

3270 Terminal Emulation

The 3270 terminal emulation server and access routine allow a VT100 user in a DECnet network to interact with application programs executing under IMS or CICS that have been designed to interact with 3270 terminals.

The 3270 terminal is a block-mode terminal. A formatted screen, much like a form, is transmitted to the terminal. After the user fills in any data required by the "form," the whole block of data or modified fields is sent to the host computer as a single unit. This mechanism is emulated by buffering the "form" in memory and displaying it on a VT100. When the screen is complete, it is sent from the memory buffer to the IBM system via the DECnet/SNA Gateway.

User Application Interface

The User Application Interface enables a user-written application on a DECnet node to exchange messages with a cooperating application in an IBM host. It supports the IMS and CICS DataBase/Data Communication systems.

The User Application Interface appears to an application running on a DECnet system as a set of subroutines that the application calls to request the following operations:

- establish an SNA session with an application running in the IBM host
- listen for a session-initiation request from the IBM application
- accept an SNA BIND request from the IBM application
- transmit messages to the IBM system on the SNA normal or expedited flow
- receive messages from the IBM system on the normal flow
- reject a BIND request from the IBM application
- abort an active session

Network Management

Network management for the Gateway includes normal DECnet network management activities, installation of the Gateway and its access routines, and control, monitoring, and troubleshooting of the Gateway. All network management functions are performed from a node connected to the Gateway, rather than from the Gateway itself.

PACKETNET SYSTEM INTERFACES

The DIGITAL Network Architecture incorporates the X.25 communications protocol to help users reduce the cost of communications over geographically remote systems and networks. X.25 is the international standard upon which various public packet-switched networks (PPSNs) are now based. In the United States, these are privately owned networks—Tymnet and GTE's Telenet, for example. In other countries, they are nationally owned networks.

DIGITAL has already developed and implemented Packetnet System Interfaces (PSIs) for use in the United States, France (Transpac), the United Kingdom (PSS), and Germany (Datex-p). We are committed to support other PPSNs as they become available to the user community.

Interfacing to packet-switched networks offers customers vendor independence. Any equipment that complies with the X.25 recommendation can be used to communicate via the packet-switched network. The packet-switched network handles all line-speed differences. In addition, the responsibility for maintaining transparent, complex configurations belongs to the public utility.

The packet-switched networks were designed to benefit the medium-volume user. Tariffs are largely distance-independent.

DIGITAL's Packetnet program provides customers yet another choice for distributing their computing in creative, cost-saving ways.

RSX-11 PSIs

RSX-11 PSI/M and RSX-11 PSI/M-PLUS allow suitably configured RSX-11M and RSX-11M-PLUS systems to connect to PPSNs conforming to the CCITT recommendation X.25 (June 1980). Access to the RSX-11 PSIs is supported for RSX-11 programs written in MACRO-11, FORTRAN-77, and FORTRAN IV. The RSX-11 PSIs support task-to-task and remote terminal communications via the network.

The RSX-11 PSIs can coexist with or operate as a layered product under DECnet to allow use of DECnet facilities over X.25 as well as over private leased lines or switched telephone networks.

Task-to-Task Communications — For intertask communication, application programs use RSX-11 executive calls to set up and break connections with the network, to send and receive data, and to issue control and synchronization requests.

Remote Terminal Communications — To remote terminal users, RSX-11M/M-PLUS appears similar to what it would if they were using it locally. And applications programs need not be aware that access to the terminal is via the RSX-11 PSI software, although there may be some restrictions imposed by the network itself.

Virtual Circuits — The RSX-11 PSIs offer communications over both Permanent and Switched Virtual Circuits (PVCs and SVCs). Subject to memory availability, the maximum total number of virtual circuits supported is 255. The maximum number of remote terminals supported is 40. Note, however, that each active remote terminal uses a virtual circuit. Up to two physical connections can be made to the PPSN.

Line Discipline — The communications discipline used is the CCITT recommendation X.25. Specifically, the RSX-11 PSIs support V.24 (EIA-RS232) at the hardware level; the symmetric LAPB variant of the X.25 frame-level protocol; and the X.25 packet-level protocol over point-to-point, four-wire, synchronous, full-duplex lines.

User Program Interface — The user program interface provided with the RSX-11 PSI software allows application programs to use the X.25 functions, including setting up and breaking connections, transmitting and receiving data, sending and receiving interrupt messages, and resetting virtual circuits. This interface provides, in addition to CCITT X.25 level 3 functions, translation of symbolic DTE addresses, transfer of calls between tasks, and splitting and recombination of messages that are longer than the packet size selected for the circuit. National extensions that are not part of the international recommendations are not provided; this interface does not include, for example, file transfer, file access, or task addressing.

When using this interface, an application program can communicate with any other system—DIGITAL or non-DIGITAL—connected to the network. This interface conforms to the DIGITAL Network Architecture specification for X.25 access.

Interactive Terminal Interface — The RSX-11 PSIs support access to remote terminals according to the CCITT recommendations X.3, X.28, and X.29 (1978 and 1980). Terminals are supported in "Remote X.29 Terminal" mode, in which code conversions between ASCII and the actual code used by the terminal are performed by the network. Note that terminals are connected via a Packet Assembly Disassembly (PAD) facility provided by the network. The PSIs offer no facility whereby a terminal connected to an RSX-11M/M-PLUS system can act as a network terminal to other systems connected to the network.

The interface presented to an application program, using a remote X.29 terminal, is compatible with that to a local terminal, except for those facilities where adequate support is not provided by the network. For example, the support provided by the PAD for data forwarding upon input of an escape sequence is not compatible with that required by DIGITAL's standard software. This means that DIGITAL's Forms Management System (FMS) and some of the screen editors may not function. Additional facilities are provided to allow a program to manipulate the terminal parameters maintained by PAD.

Network Management — A Network Control Program and Configuration File Editor are provided for the control of the operation of the X.25 software. This includes loading and unloading the software, defining the lines, specification of addressing information for incoming calls, and access to error counters and other maintenance functions. This interface provides a subset of the DIGITAL Network Architecture specification for network management.

ETHERNET

Ethernet is a specification for the physical implementation of local area communications. It was developed jointly by DIGITAL, Intel, and Xerox and published in September, 1980. Since then, hundreds of companies have requested license applications, and many have announced their intentions to market Ethernet products.

Ethernet provides a common communications path by which systems and associated terminals attached by a single connection can communicate with all other attached nodes at speeds comparable to those on the bus that links devices within a single system. It provides ten-Mbits-per-second communications and a common bus topology, with all nodes communicating as peers. Because there is no centralized control or switching, the reliability of a network is significantly enhanced. Nodes can be added or removed from a network while it remains in operation.

Ethernet uses baseband technology with coaxial cable and a standard link-level protocol (CSMA/CD—Carrier Sense Multiple Access with Collision Detection). The cable and protocol correspond closely to the first two levels of the International Standards Organization (ISO) Open Systems Architecture.

Ethernet technology allows cable segments up to 1,650 feet (500 meters) long. By using repeaters, multiple segments can be connected. Local area networks of up to 1,000 nodes can be constructed, provided no two nodes are separated by more than two repeaters. A node can be up to 165 feet (50 meters) away from the cable. These parameters allow a building or a complex of buildings to be wired; users can simply tap into the cable wherever they want to place computing power. The main advantage of Ethernet is that a minimum of preplanning is necessary—a network can grow with no loss in efficiency, no need to reconfigure.

DIGITAL's Ethernet product provides for the integration of Ethernet into the overall distributed data processing and

networking capabilities we already offer. The Ethernet link-level protocol has been incorporated into the DIGITAL Network Architecture.

The full spectrum of high-level DECnet functions is available to users accessing nodes on an Ethernet. This includes virtual terminals, user-to-user and task-to-task communications, and remote file and record access. DECnet's extensive network management and network maintenance facilities are also available.

Ethernet support will be offered on DECnet Phase IV host VAX/VMS systems, Professional 350 personal computers, and DECSYSTEM-20s, as well as on RSX-11-based PDP-11 systems.

The first products to be offered as part of the Ethernet program include the physical cables, a transceiver, and a communication controller for attaching VAX systems or RSX-11-based UNIBUS PDP-11 systems. In the future, we'll extend Ethernet's capabilities with many more products—products that will cover a broad range from the physical channel hardware to sophisticated communication servers for connecting Ethernets to other DECnet networks and to foreign networks.

Ethernet physical channel hardware includes the cable itself, transceivers, repeaters, terminal servers, and routers.

Ethernet communication controllers connect the transceiver cable to the network-node system bus. DIGITAL will build controllers including those for the UNIBUS, the LSI-11 bus, and the Professional 300 series of personal computers.

Transceivers provide the physical and electrical connection to the Ethernet coaxial cable. They transmit signals onto the coaxial cable, receive signals from the cable, and detect any message collisions that occur. Transceivers also contain a connector to attach the transceiver cable to the Ethernet node.

Terminal servers connect clusters of terminals and unit-record equipment—lineprinters, for example—to an Ethernet network.

Routers connect DECnet systems with remote DECnet systems, with another Ethernet, or with other DECnet networks. Routers are useful for linking Ethernets to DIGITAL systems that do not directly support Ethernet connections.

NETWORK SUPPORT

No two networks are quite the same. There's no such thing as a standard network package. The configuration flexibility inherent in DIGITAL's network products makes network support especially important.

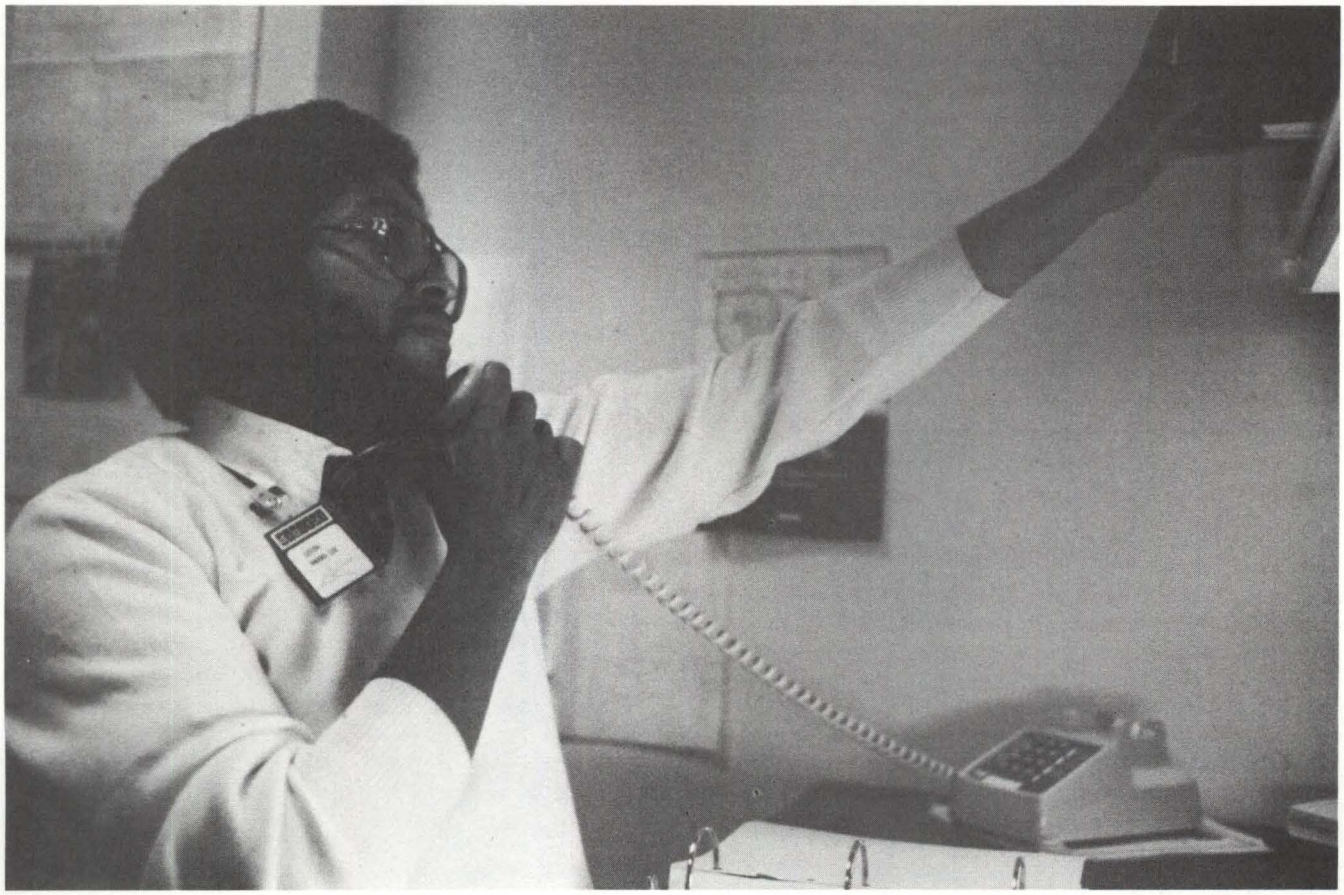
DIGITAL places particular importance on careful preplanning of networks. To this end, two special programs have been implemented in support of DIGITAL's networking products—the Network Profile and the Customer Support Plan.

The Network Profile is the network configuration document. It spells out the problems to be solved and it provides the necessary technical information about the proposed network—its applications, implementation, and future growth. Technical details include:

- number and location of nodes
- number and location of terminals
- types of transactions to be processed
- volume of data to be transferred between nodes
- data urgency and importance
- acceptable error rate
- line and system reliability
- network security
- costs
- network management

The Customer Support Plan summarizes DIGITAL's recommended services to meet the customer's various goals. The Customer Support Plan covers installation, startup, application development, training, network maintenance, and troubleshooting. It can include such things as suggested Software Maintenance Services, Educational Services courses, and Software Professional Services. (These services are described in Section 11.) It identifies the customer's needs, the purpose and benefits of the support services, and the details of the recommended services. The Customer Support Plan is customized for each customer and reflects the uniqueness of each network.

11
Support Services



DIGITAL offers comprehensive customer services to complete our commitment to meeting users' needs. Our customer service organizations include:

- *Software Services*, whose specialists are experienced in analyzing and designing systems, in modifying DIGITAL's software to meet special needs, and in developing customer application software
- *Educational Services*, whose instructors and self-paced courses train users of DIGITAL's computers, at training centers and customer sites around the world
- *Field Service*, whose representatives provide hardware maintenance services worldwide, both onsite at customer installations and offsite at service centers
- *Computer Special Systems*, whose analysts, engineers, programmers, and manufacturing specialists provide hardware and software to customers whose needs are not met by DIGITAL's standard offerings
- *Accessories and Supplies Group*, which maintains Accessories and Supplies Centers in major metropolitan areas and who provide direct factory ordering through the *Direct Sales Catalog*
- *Computer Supplies*, which offers a complete line of supplies designed specifically for use with DIGITAL's systems
- *Customer Financing*, whose representatives provide financial counseling to help customers decide what type of financial arrangement best meets their needs
- *DECUS*, the Digital Equipment Corporations Users' Society, one of the largest and most active user groups in the computer industry

Customers work with DIGITAL's sales representatives to determine how DIGITAL's people, products, and services can best be used to meet the customer's computing requirements.

INTRODUCTION

DIGITAL offers comprehensive services to help customers before, during, and after system installation. DIGITAL's sales representatives are the primary contact for all products and services.

DIGITAL provides customers with support right from the beginning of the sales cycle. Our sales representatives work closely with customers, studying each application and determining specific computing needs. Trained software and hardware specialists, well-versed in designing systems using DIGITAL's standard and special products, are available to supplement the sales representative's product knowledge.

Once the exact system requirements have been determined, the sales representative helps the customer select a system configuration. Together they review such site requirements as floor space, electrical capacity, air conditioning, and humidity control. Customers can choose among various Field Service and Software Service maintenance plans to suit individual needs and budgets.

For complex applications, the customer and a representative of DIGITAL's Software Services organization can prepare a Customer Support Plan. The CSP may consist of Software Product Services, Educational Services courses, hardware maintenance requirements, and software consulting services. A CSP identifies the customers' needs. It spells out the purposes, benefits, and details of the services recommended, specifying costs and how the services will be delivered.

Even before the system arrives, customers can train their personnel through DIGITAL's comprehensive educational programs. When a system is purchased, customers also gain training credits that can be applied to the cost of DIGITAL's courses.

When a system is delivered, members of DIGITAL's hardware Field Service and Software Services organizations are on hand to ensure smooth installation. Specialists install hardware and software and run tests to determine that the system has been installed correctly and performs properly.

Following installation, DIGITAL's support organizations are available to help with special needs that may arise both during and after the warranty period.

Purchase of a DIGITAL computer automatically qualifies a customer for membership in the Digital Equipment Computer Users Society (DECUS). It is an independent, international organization run entirely by user members.

INSTALLATION

At system delivery, a Field Service account representative schedules installation of the hardware components. During installation, Field Service engineers supervise uncrating and placement of equipment, connecting the cables, and applying power to the components. They test the hardware by running a system exerciser—a complete diagnostic package. Once hardware reliability is confirmed, the Field Service engineers coordinate with Software Services to install and test the operating system. They use the User Environmental Test Package (UETP) to exercise the system software components. This package runs compilations and assemblies and serves as both a final test of the system and as a demonstration of system operation.

Finally, Field Service completes DIGITAL forms certifying successful installation. The customer acknowledges system acceptance by signing the Field Service Labor Accounting and Reporting System form.

SOFTWARE SERVICES

The specialists in DIGITAL's Software Services organization are committed to providing top-quality support for RSX-11 software. They are specially trained in RSX-11 software to provide the expert knowledge and experience necessary to analyze customer's needs and to identify those DIGITAL services that will help meet those needs. Local software specialists are backed up by regional and corporate support personnel as well. This means that DIGITAL's total software resources and expertise are available to support RSX-11 and its various dependent products.

Software Warranty

As DIGITAL-supported software products, the RSX-11 operating systems are engineered according to corporate quality standards. They operate in accordance with their Software Product Descriptions (SPDs), and carry DIGITAL's commitment to provide product support services. The RSX-11 operating systems are DIGITAL-installed products; they must be installed by a qualified DIGITAL representative to qualify for software support.

DIGITAL-supported software products have a 90-day warranty period following installation. If, during the 90 days of warranty, a problem with the software is encountered that DIGITAL determines to be caused by a defect in the current unaltered release of the product, the following remedial services are provided.

- If the software is inoperable, DIGITAL will apply a temporary correction or make a reasonable attempt to develop an emergency bypass.
- DIGITAL will help the customer prepare a Software Performance Report (SPR). Via an SPR, users can report problems with, or suggest enhancements to, DIGITAL's software and documentation.

After the initial purchase of a DIGITAL-supported product license, the customer can purchase additional copies of the software. The additional licenses can include support services or can be purchased as "License-to-Copy Only," in which case neither media nor support services are included.

The RSX-11 operating systems include standard services as defined in the RSX-11 SPDs.

Software Product Services

After the 90-day warranty period, three levels of Software Product Services are available to provide continued software support to complement DIGITAL's hardware services. These Software Product Services offer the most comprehensive post-warranty support in the industry.

The three contractual services are:

- *Self-Maintenance Service for Software.* This service includes software product and documentation updates and a newsletter that provides up-to-date information on the software product. Optionally available under this service are machine-readable Program Change Orders.
- *Basic Service.* This service builds on Self-Maintenance Service for Software by providing telephone support and machine-readable Program Change Orders. A customer can talk directly with a DIGITAL software specialist for help with a software problem. Onsite support can be contracted on a per-call basis. The Autopatch software tool is a Basic Service option. Autopatch automatically installs patches to RSX-11 modules, upgrading the operating system software to the latest level. (See Section 5 for a description of Autopatch.)
- *DECsupport.* The most comprehensive service offered, DECsupport provides all those services included in the Basic Service plus scheduled services to install Program Change Orders and Software Product Updates.

For customers who install their own software, Software Product Updates include the Software Updates themselves—distributed on the customer's choice of available media—along with the most recent documentation. Support services are provided upon request.

Professional Services

Whenever expert software assistance is needed, DIGITAL's software consultants are available. These software specialists are experienced designers and writers of custom software who can tailor DIGITAL software to meet specific needs. Their expertise can be applied to every phase of an application—from analysis through implementation.

Software specialist services are available on a resident or per-call basis.

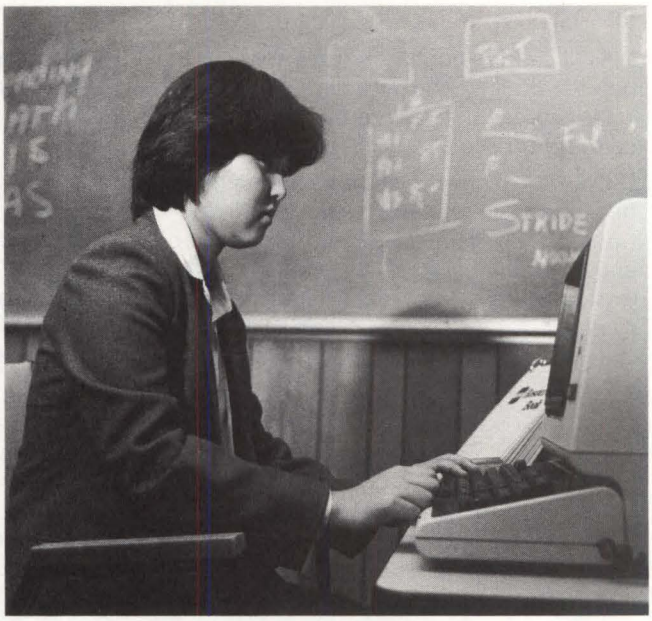
- *Resident service* is for those customers who need full-time onsite support. Resident consultants are particularly useful in new, complex installations or in critical, long-term projects. They are available for a minimum of six months; however, arrangements can be made to extend the length of service to suit individual needs.
- *Per-call service* is for customers with irregular or infrequent consulting needs. Per-call (hourly) services are ordered as needed and generally extend from a few days to a few weeks.

For detailed information on DIGITAL's software services, contact a local DIGITAL office.

EDUCATIONAL SERVICES

To train users before, during, and after system installation, DIGITAL provides comprehensive educational programs. Qualified instructors conduct lecture/lab courses in system management, operations, hardware, and software at DIGITAL's training centers around the world.

Special onsite training sessions and custom courses can also be arranged. Another alternative is self-paced instruction course packages, which allow students to learn



individually, at their own pace, wherever and whenever it's convenient. Self-paced instruction packages are available as printed manuals, audiovisual cassettes, and computer-based instruction.

Courses fall into three general categories.

- *Computing courses*, while not geared to a specific DIGITAL system or product, offer a technical foundation for personnel who have little or no previous computer experience.
- *Software systems courses* are designed to train users, programmers, and operators in the efficient and knowledgeable use of DIGITAL's operating systems, languages, and utilities. Courses are available for both beginning and advanced users. For the most part, these courses are based on the assumption that the student has general computer and programming knowledge.
- *Hardware courses* are for customers who intend to service their own equipment or who simply want a general understanding of the components in their system. Courses are available in general hardware familiarization, hardware troubleshooting, and hardware maintenance.

DIGITAL's Educational Services organization offers a comprehensive series of courses that provide specific RSX-11 training to all levels of users. These range from introductory courses describing the PDP-11 hardware and RSX-11 operating systems concepts to advanced courses designed for high-level systems programmers. The following are brief descriptions of the courses offered.

Introduction to Minicomputers discusses computer system concepts, peripherals, and memory principles. It is available both as a lecture/lab course and as a self-paced audiovisual course.

Commercial Programming Concepts, a lecture/lab course, introduces commercial computer concepts and data processing principles. This course includes basic commercial programming techniques, the fundamentals of programming in BASIC, and table and array handling.

RSX-11M Operator, available as a lecture/lab course and as a self-paced instruction course, covers startup, shutdown, recovery, backup, and restoration procedures through supervised hands-on training in normal and system-crash environments.

PDP-11 Concepts introduces, in a lecture/lab course, the PDP-11 hardware and software features that are necessary to prepare the high-level language programmer for further training on the PDP-11 operating systems.

RSX-11M/M-PLUS Utilities and Commands teaches users, either in a lecture/lab course or through self-paced instruction, the skills necessary to interface effectively with an RSX-11 system.

RSX-11M System Management and *RSX-11M-PLUS System Management*, both lecture/lab courses, cover the skills required to perform RSX-11 system management tasks, including starting up and shutting down the system, performing system resource management and diagnostics, and creating file structures.

BASIC-PLUS-2 teaches BASIC-PLUS-2's language syntax and capabilities. Practical exercises and examples in this lecture/lab course teach students to write functional BASIC-PLUS-2 programs.

BASIC-PLUS-2 Programming with RMS-11, a lecture/lab course, is designed for BASIC-PLUS-2 programmers who want to use the Record Management Services (RMS) utilities effectively. Emphasis is placed on RMS data structures.

PDP-11 COBOL teaches how to program in PDP-11 COBOL, including a careful study of the syntax, format, and structure of the language. Supervised laboratory sessions in this lecture/lab course include testing and running programs written by students.

FORTRAN IV teaches, in both a lecture/lab course and a self-paced instruction course, how to program in FORTRAN IV, including a careful study of the syntax, format, and structure of the language. Supervised laboratory sessions include testing and running programs written by students.

PDP-11 Assembly Language Programming teaches the architecture of the PDP-11 computer, showing how hardware and software components work together at the machine level. Available in both lecture/lab and self-paced audiovisual formats, it explains addressing modes and the common instruction set. Sample programs in assembly language are written to demonstrate the use of the PDP-11 instruction set.

MACRO-11 provides the skills assembly language programmers need to use assembler directives and to write macros. Available as both a lecture/lab course and as a self-paced instruction course, this course introduces the program development cycle, and it teaches the use of MACRO libraries, how to write programs using macros, and documentation techniques.

Programming RSX-11M/M-PLUS in FORTRAN/MACRO is for FORTRAN and MACRO programmers who want to develop systems or applications programs. Before taking this course as either a lecture/lab or as a self-paced instruction course, students should be able to write working programs in FORTRAN or MACRO-11.

RSX-11M/M-PLUS DECnet User is a lecture/lab course that teaches system programmers and MACRO programmers to design and program networks using RSX DECnet software. FORTRAN IV, COBOL, and BASIC-PLUS-2 programmers can also benefit from this course.

Design of Applications under RSX-11M/M-PLUS is a four-day seminar that focuses on the system features that maximize the capabilities of the RSX-11 hardware and software. Three case studies are included.

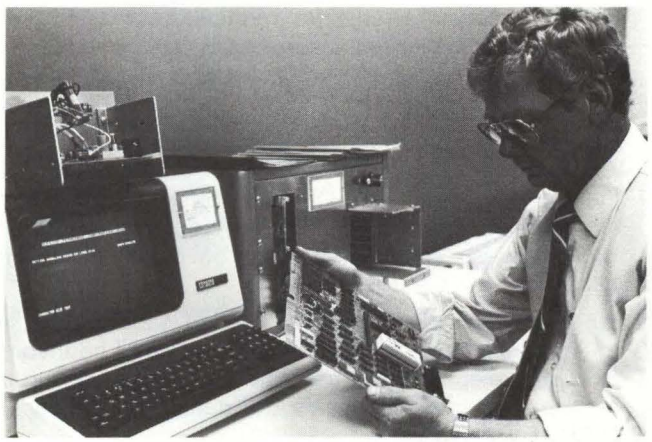
RSX-11M Operating System Internals and *RSX-11M-PLUS Operating System Internals* are lecture/lab courses for programmers who need to modify or enhance RSX-11M or RSX-11M-PLUS by writing privileged tasks or by adding user-written system services. These courses provide students with an understanding of the design and philosophy of the RSX-11 operating systems to help them make design and performance tradeoffs.

RSX-11M Device Driver and *RSX-11M-PLUS Device Driver* are lecture/lab courses for system programmers who need to add or modify device drivers on an RSX-11 configuration. Students should be fluent in MACRO-11 and experienced systems programmers on RSX-11. Students will learn to write a working device driver to the system, to debug driver code, and to write privileged tasks using the Connect-to-Interrupt Directive to service interrupts.

Additional information about course content and availability of training credits can be obtained by contacting your DIGITAL sales representative.

HARDWARE SERVICES

DIGITAL's Field Service organization offers a range of onsite and offsite post-warranty hardware maintenance services. Over 12,500 Field Service personnel in more than 400 locations worldwide are ready to provide the support needed for continuous productivity.



Onsite Services

DECservice is DIGITAL's most comprehensive onsite maintenance plan. It is designed for customers who require uninterrupted system performance. DECservice includes:

- committed response time
- continuous-effort remedial service
- priority problem escalation
- extended preventive maintenance hours
- parts, labor, and material
- installation of the latest engineering change orders
- an assigned service representative
- comprehensive site management guide

For customers who do not require a fixed response time and continuous remedial efforts, Field Service offers Basic Service. Basic Service provides:

- priority response (typically one day)
- remedial service during coverage hours
- priority problem escalation
- preventive maintenance during coverage hours
- parts, labor, and materials
- installation of the latest engineering change orders
- an assigned service representative
- comprehensive site management guide

Although standard coverage for both of these onsite service plans is eight hours a day, five days a week, customers can choose to extend DECservice coverage to 12, 16, or 24 hours a day and to include weekends and holidays.

Offsite Services

DIGITAL offers offsite services for customers who have sufficient expertise to maintain their own systems. These services include module-level repair, terminal repairs, and major system refurbishment. DIGITAL's offsite services include Customer Returns Centers, Product Repair Centers, and Terminal Service Centers. All these centers have their own parts inventories, special diagnostic systems, and repair kits designed by DIGITAL engineers.

Customer Returns Center — The Customer Returns Center (CRC) performs factory-level repairs on modules for customers who can trace equipment problems to the module or subassembly level. The CRC honors DIGITAL's Return-to-Factory Warranty for modules, and offers post-warranty DECmailer repair services.

DECmailer provides qualified customers with fast, dependable return-to-factory repairs for any of over 1,000 modules and subassemblies. Customers with a DECmailer Agreement receive a supply of preaddressed cartons for mailing failed modules. A replacement is shipped within five working days of receipt of the modules. All DECmailer repairs are warrantied for 60 days from date of return shipping.

When especially fast service is needed, DIGITAL can ship an emergency replacement module within 24 hours of receiving a telephone call from a customer requesting this service. The customer ships the defective part back to DIGITAL within a week.

When turnaround time is not critical or the repair items are not eligible for DECmailer service, Loose Piece Module Repair provides service on an "as needed" basis. Refer to the Accessories and Supplies Group's *Direct Sales Catalog* (described below) for estimated module repair time, or contact your local DIGITAL sales representative for details.

Product Repair Centers — DIGITAL's Field Product Repair Centers (PRCs) provide fast, low-cost, offsite repairs on all DIGITAL-supplied systems, options, and peripherals. PRC services include:

- A *Return-to-PRC Agreement* that provides repairs on a selected group of DIGITAL's processors and peripherals for a fixed quarterly charge. The term of the agreement is one year.
- *Individual Product* services that let customers choose between Fixed Quote and Time-and-Materials repair services. Under the Fixed Quote service, DIGITAL quotes the repair cost before any repairs are performed. Customers can then decide if they want DIGITAL to perform the repair.

  With the Time-and-Materials service, users can choose, step by step, the extent of repairs to be done. This can range anywhere from minimal repairs to total equipment refurbishment.

Terminal Service Centers — Terminal Service Centers (TSCs) provide carry-in service for DIGITAL's terminal products on a contractual or per-call basis. The TSCs also offer self-maintenance customers over-the-counter module swap service for terminals. Repairs are warrantied for 60 days. Payment can be made by credit card.

You can obtain more information on DIGITAL's Field Service hardware maintenance offerings by contacting your local DIGITAL sales representative.

COMPUTER SPECIAL SYSTEMS

DIGITAL's Computer Special Systems (CSS) group provides design services and resulting repeat products to DIGITAL's customers. Analysts, engineers, programmers, and manufacturing specialists can provide hardware and software to customers whose needs are not met by DIGITAL's standard offerings. CSS is represented by Application Engineers in the field.

CSS has nine engineering/manufacturing sites, 12 business units operating out of ten plants, over 450 Technical Specialists, Applications Engineers in 40 cities, and 19 offices with over 1,000 people worldwide. The design staff has over 1,000 man-years of experience, and has been involved in over 8,000 computer installations.

CSS provides systems and products in all of DIGITAL's markets. Products and systems are analyzed, designed, and implemented according to the customer's goals and requirements. These can range from simple processor interfaces to complete hardware and software systems. All products carry DIGITAL's high standard of quality, documentation, and field support.

CSS design and project management services, available on a contract basis, include:

- *Hardware.* CSS interfaces DIGITAL hardware with that of other manufacturers, modifies standard hardware, and designs and builds new equipment.
- *Software.* CSS designs and produces diagnostic systems and applications software, modifies and expands standard DIGITAL software systems, and builds new software according to individual needs.
- *Systems.* CSS builds complete hardware and software systems for special applications. CSS project managers oversee the analysis, design, and implementation of the system and work with the customer to ensure proper installation and startup.
- *Support.* Customers can receive DIGITAL's broad range of support services with any CSS system. Hardware is supported by DIGITAL's extensive Field Service organization, and training is available on all aspects of CSS systems.

CSS provides DIGITAL's customers with cost-effective solutions to their special system and application problems, and backs the final product with high-quality service. For detailed information on how to take advantage of the services provided by CSS, contact a DIGITAL sales representative.

ACCESSORIES AND SUPPLIES GROUP

DIGITAL's Accessories and Supplies Group (A&SG) maintains Accessories and Supplies Centers (ASCs), provides direct factory ordering through the *Direct Sales Catalog*, and supports their products worldwide.

ASCs are full-service centers established to fulfill the needs of DIGITAL's customers in major metropolitan areas. The ASCs hold a local inventory of the most requested accessories, supplies, documentation, and add-on products for fast delivery. Full order-processing capability provides access to A&SG's central inventory in Nashua, New Hampshire. The ASCs provide first-class service and convenience to DIGITAL's customers.

A&SG maintains a tollfree telephone number for customers to use when ordering accessories and supplies. Most products are shipped within 48 hours of receipt of order.

A&SG's *Direct Sales Catalog* offers a broad range of computer accessory and supply items. These include small computer systems and their complementary options, accessories, and operating supplies. The *Direct Sales Catalog* also sells such DIGITAL hardware options as DECwriters, microcomputers, and their associated options. Hardware and software documentation is also offered.

A&SG has a team of worldwide specialists and business managers to support sales.

COMPUTER SUPPLIES

DIGITAL's Computer Supplies group maintains a complete line of supplies specifically designed for use with DIGITAL's computer systems. These items facilitate reliable and efficient system operation and include:

- a family of magnetic media such as disk cartridges, disk packs, and floppy diskettes
- self-contained disk cartridge cleaners designed to accomplish fast and efficient cleaning of front- or top-loading magnetic disk cartridges
- word processing supplies such as nylon and mylar ribbons, a choice of 11 print wheels, and filter screens for video terminals
- ribbons for DECwriter and DECprinter terminals

The Computer Supplies group also offers cabinets for maintaining supplies and protecting magnetic media when not in use. Cabinet interiors can be customized with various options to meet individual needs. Options can be conveniently rearranged at any time to adapt to changing requirements. In addition, there are paper baskets, work shelves, terminal tables, tape racks, papertape trays, and multipurpose binders.

Relying on DIGITAL for computing needs means saving time, money, and paperwork. Contact a DIGITAL sales representative for more information.

CUSTOMER SPARES

Customer Spares is dedicated to supporting customers who maintain their own computers. Customer Spares is organized into three distinct businesses: self-maintenance products (hardware and documentation); system accessories; and low-volume LSI-11 products. System accessories include products for the hardware builder that facilitate easy expansion and reconfiguration of DIGITAL's systems and options.

Customer Spares sells modules, subassemblies, components, tools, and test equipment. Related services include providing assistance in selecting the proper parts and expediting delivery during emergencies.

CUSTOMER FINANCING

To simplify the financial considerations involved in acquiring a new computer, DIGITAL provides leasing and financial counseling. The Customer Finance Department can help customers acquire a DIGITAL system through a lease, conditional sale, or similar financing agreement, rather than through outright cash purchase.

For commercial businesses or private organizations, DIGITAL has developed a program, known as DIGITAL Leasing, with the U.S. Leasing Corporation of San Francisco. DIGITAL Leasing, a division of U.S. Leasing, is committed solely to financing DIGITAL's computers. Representatives are located in or near many of DIGITAL's District Sales Offices.

Federal, state, and local government agencies have special contractual needs and, in some cases, can benefit from special tax privileges. For example, a state or municipal agency qualifies for special interest rates on Conditional Sales Agreements that are significantly below those charged to commercial customers. The interest income is free from federal, and, in some cases, state income taxes.

The following financing is available.

- *Full Payout Leases* are used primarily by commercial customers. They involve a noncancellable, three-to-five-year term, usually with a ten percent purchase option at the end. No down payment is required, and title remains with the lessor. Lease payment schedules are flexible and can be tailored to specific needs.
- *Conditional Sales Agreements* are used primarily by state and local governments. They involve a noncancellable, one-to-five-year term. Title passes to the customer, but DIGITAL retains a security interest. The customer owns the equipment free and clear at the end of the term. Fiscal funding provisions are available for state and local governments.
- *Federal Government Lease to Ownership Agreements* are available only to approved federal government agencies. They involve a one-to-five-year term, cancellable at the end of each fiscal year for nonappropriation of funds. Ownership passes to the customer at the end of the term.

DIGITAL's Customer Financing group can provide financial counseling to help customers decide which arrangement best meets their needs. For more information, contact a local DIGITAL sales office.

DECUS

DECUS, the Digital Equipment Computer Users Society, is one of the largest and most active user groups in the computer industry. It is a not-for-profit association, supported and administered by DIGITAL, but actively controlled by individuals who have purchased, leased, ordered, or used a DIGITAL computer or who have a bona fide interest in DECUS. Membership is free and voluntary.

The goals of DECUS are to:

- advance the art of computation through mutual education and exchange of ideas and information
- establish standards and provide channels to facilitate the exchange of computer programs
- provide feedback to DIGITAL on hardware and software needs
- advance the effective use of DIGITAL computers, peripherals, and software by promoting the interchange of information

DECUS headquarters, located in Marlborough, Massachusetts, administers all international policies and activities. DECUS is subdivided into four chapters:

DECUS AUSTRALIAN CHAPTER (Australia, Brunei, New Zealand, Malaysia, Singapore, Indonesia, PNG)
DECUS Australia
P.O. Box 384
Chatswood
NSW 2067
Australia

DECUS CANADIAN CHAPTER
DECUS Canada
P.O. Box 13000
Kanata, Ontario
K2K, 2A6, Canada

DECUS EUROPEAN HEADQUARTERS (Europe, Middle East, North Africa, Eastern Europe)
DECUS Europe
P.O. Box 510
12, Av. Des Morgines
CH-1213 Petit-Lancy 1/GE
Switzerland

DECUS UNITED STATES CHAPTER (for U.S. and all others)
DECUS International Headquarters
Digital Equipment Corporation
MR02-3/E55
One Iron Way
Marlborough, Massachusetts 01752 U.S.A

To further the goals of the society, DECUS serves its members by holding symposia; maintaining a program library; publishing an association newsletter, technical newsletters, and books; and supporting a number of subgroups, called Special User Groups, for special interests and locations.

The symposia are regularly scheduled meetings held in each of the four chapters. They provide a forum in which users of DIGITAL's products can meet with other users and with DIGITAL's management, engineering, and support personnel. Symposia give users an opportunity to participate in DIGITAL product workshops and product-planning feedback sessions. Many of the technical papers and presentations from each symposium are published as a book, the *DECUS Proceedings. Proceedings* copies are supplied to symposia attendees and can be purchased by DECUS members.

A major activity of DECUS is the Program Library, which contains over 1,700 software packages written and submitted by users. A wide range of software is offered, including languages, editors, numerical functions, utilities, display routines, games, and other types of application software. Library catalogs are available for the PDP-8, PDP-11/VAX, and DECsystem-10/20. Catalogs are updated yearly and contain program descriptions and ordering information. The programs are available for a nominal service charge that covers the cost of reproduction and media.

Each DECUS chapter publishes an association newsletter that covers general DECUS news; it is distributed to all chapter members.

Special User Groups focus on operating systems, languages, processors, and applications. Local User Groups, National User Groups, and Regional User Groups, which are formed basically by geographical proximity, may also share common specific interests. Many of these subgroups also publish newsletters.

To obtain a membership form for DECUS, contact a DIGITAL sales representative or the appropriate chapter office.

Index

M

N

O

P

T

U

DIGITAL EQUIPMENT CORPORATION, Corporate Headquarters: Maynard, MA 01754, Tel. (617) 897-5111 — SALES AND SERVICE OFFICES; UNITED STATES — ALABAMA, Birmingham, Huntsville ARIZONA, Phoenix, Tucson ARKANSAS, Little Rock CALIFORNIA, Costa Mesa, El Segundo, Los Angeles, Modesto, Monrovia, Oakland, Pasadena, Sacramento, San Diego, San Francisco, Santa Barbara, Santa Clara, Santa Monica, Sherman Oaks, Sunnyvale COLORADO, Colorado Springs, Denver CONNECTICUT, Fairfield, Meriden DELAWARE, Newark, Wilmington FLORIDA, Jacksonville, Melbourne, Miami, Orlando, Pensacola, Tampa GEORGIA, Atlanta HAWAII, Honolulu IDAHO, Boise ILLINOIS, Chicago, Peoria INDIANA, Indianapolis IOWA, Bettendorf KENTUCKY, Louisville LOUISIANA, Baton Rouge, New Orleans MAINE, Portland MARYLAND, Baltimore, Odenton MASSACHUSETTS, Boston, Burlington, Springfield, Waltham MICHIGAN, Detroit, Kalamazoo MINNESOTA, Minneapolis MISSOURI, Kansas City, St. Louis NEBRASKA, Omaha NEVADA, Las Vegas, Reno NEW HAMPSHIRE, Manchester NEW JERSEY, Cherry Hill, Parsippany, Princeton, Somerset NEW MEXICO, Albuquerque, Los Alamos NEW YORK, Albany, Buffalo, Long Island, New York City, Rochester, Syracuse, Westchester NORTH CAROLINA, Chapel Hill, Charlotte OHIO, Cincinnati, Cleveland, Columbus, Dayton OKLAHOMA, Tulsa OREGON, Eugene, Portland PENNSYLVANIA, Allentown, Harrisburg, Philadelphia, Pittsburgh RHODE ISLAND, Providence SOUTH CAROLINA, Columbia, Greenville TENNESSEE, Knoxville, Memphis, Nashville TEXAS, Austin, Dallas, El Paso, Houston, San Antonio UTAH, Salt Lake City VERMONT, Burlington VIRGINIA, Arlington, Lynchburg, Norfolk, Richmond WASHINGTON, Seattle, Spokane WASHINGTON D.C. WEST VIRGINIA, Charleston WISCONSIN, Madison, Milwaukee INTERNATIONAL — EUROPEAN AREA HEADQUARTERS: Geneva, Tel: [41] (22)-93-33-11 INTERNATIONAL AREA HEADQUARTERS: Acton, MA 01754, U.S.A., Tel: (617) 263-6000 ARGENTINA, Buenos Aires AUSTRALIA, Adelaide, Brisbane, Canberra, Darwin, Hobart, Melbourne, Newcastle, Perth, Sydney, Townsville AUSTRIA, Vienna BELGIUM, Brussels BRAZIL, Rio de Janeiro, Sao Paulo CANADA, Calgary, Edmonton, Hamilton, Halifax, Kingston, London, Montreal, Ottawa, Quebec City, Regina, Toronto, Vancouver, Victoria, Winnipeg CHILE, Santiago COLOMBIA, Bogota DENMARK, Copenhagen EGYPT, Cairo ENGLAND, Basingstoke, Birmingham, Bristol, Ealing, Epsom, Leeds, Leicester, London, Manchester, Newmarket, Reading, Welwyn FINLAND, Helsinki FRANCE, Bordeaux, Lille, Lyon, Marseille, Paris, Puteaux, Strasbourg HONG KONG INDIA, Bangalore, Bombay, Calcutta, Hyderabad, New Delhi IRELAND, Dublin ISRAEL, Tel Aviv ITALY, Milan, Padova, Rome, Turin JAPAN, Fukuoka, Nagoya, Osaka, Tokyo, Yokohama KOREA, Seoul KUWAIT, Safat MEXICO, Mexico City, Monterrey NETHERLANDS, Amsterdam, The Hague, Utrecht NEW ZEALAND, Auckland, Christchurch, Wellington NIGERIA, Lagos NORTHERN IRELAND, Belfast NORWAY, Oslo, PERU, Lima PUERTO RICO, San Juan SAUDI ARABIA, Jeddah SCOTLAND, Edinburgh REPUBLIC OF SINGAPORE, SPAIN, Barcelona, Madrid SWEDEN, Gothenburg, Malmoe, Stockholm SWITZERLAND, Geneva, Zurich TAIWAN, Taipei TRINIDAD, Port of Spain VENEZUELA, Caracas WEST GERMANY, Berlin, Cologne, Frankfurt, Hamburg, Hannover, Munich, Nuremberg, Stuttgart YUGOSLAVIA, Belgrade, Ljubljana, Zagreb

PRINTED IN USA EZ 22656 18/82 012 07 35.0