decsystem10

INTRODUCTION TO DECsystem-10 SOFTWARE

DEC-10-MZDC-D

This document reflects the software of the 5.07 and 6.01 releases of the monitor.

digital equipment corporation · maynard, massachusetts

First Printing, November 1971
Revision: July 1973
Revision: August 1974

The postage prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation

CDP	DIGITAL	INDAC	PS/8
COMPUTER LAB	DNC	KA10	QUICKPOINT
COMSYST	EDGRIN	LAB-8	RAD-8
COMTEX	EDUSYSTEM	LAB-8/e	RSTS
DDT	FLIP CHIP	LAB-K	RSX
DEC	FOCAL	OMNIBUS	RTM
DECCOMM	GLC-8	OS/8	RT-11
DECTAPE	IDAC	PDP	SABR
DIBOL	IDACS	PHA	TYPESET-10
			UNIBUS

CONTENTS

ILLUSTRATIONS

PREFACE

This manual presents a general overview of the DECsystem-10. It is written for a person who is familiar with computers and computing systems and who desires to know the relationship between the various parts of the DECsystem-10. Chapter 1 introduces the components of DECsystem-10 as well as the concept of multimode computing. Chapter 2 discusses the languages, utilities, and applications; those programs that enable the user to fully utilize the resources of the computing system. The more detailed Chapter 3 describes the essential aspects of the resident operating system. Appendix A is a survey of the DECsystem-10 hardware.

Introduction to DECsystem-10 Software is not intended to be a programmer's reference manual. For complete details on the operating system, refer to DECsystem-10 Operating System Commands (DEC-10-MRDD-D) and DECsystem-10 Monitor Calls (DEC-10-MRRD-D). However, it is recommended that this manual be read at least once before reading the foregoing manuals.

CHAPTER 1

THE DECSYSTEM-10

The DECsystem-10 is more than a processor and its associated peripheral devices. Because it is a system, there are many parts, or components, working together to achieve a goal in a manner that is both convenient for the user of the system and advantageous for the operation of the system. It is a machine designed to be utilized concurrently by many users who wish to perform various operations. There are three major components of the computing system, as shown in Figure 1-1: the actual machine, or hardware; the operating system, or monitor; and the languages and utilities, or non-resident software.

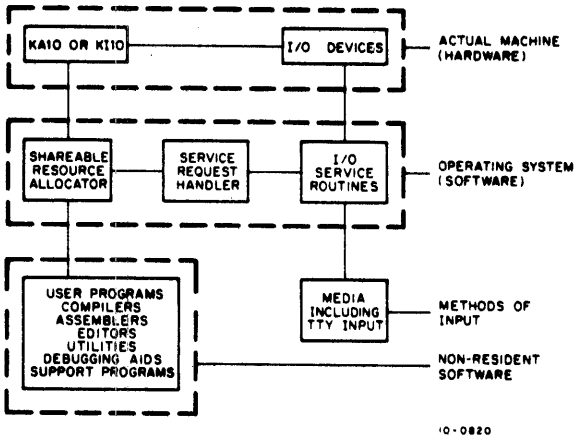

Figure 1-1

DECsystem-10 Components

1.1 DECSYSTEM-10 HARDWARE

The DECsystem-10 hardware consists of one or two central processors and various memories and input/output devices connected to these processors. There are six different systems included in the DECsystem-10 family, each system being distinguished by the hardware associated with the central processor. By adding hardware to an individual system, additional performance is achieved. However when adding hardware to expand from a small system to a larger system, no software changes are required in user programs. A single operating system and command control language can be used for all configurations of the DECsystem-10.

1.2 DECSYSTEM-10 OPERATING SYSTEM

The DECsystem-10 hardware has numerous capabilities: it is powerful, fast, and highly sophisticated. Because of its complexity, this machine is not usually directly manipulated by its users. The users communicate with an intermediary, the operating system, in order to direct their problems to the actual machine and to receive solutions back. With many users on the system, this second component of the DECsystem-10 must also keep track of what each user does and the devices and system resources that each user accesses. Though the operating system cannot be seen like the actual machine, the action of the operating system is the most important and noticeable part of the system to each user. It is true that the operating system can do nothing for the user if the actual machine does not exist, but the user normally does not think of this. If the operating system accomplishes for him what he wants the actual machine to do, he is satisfied. Therefore, it is important to the user that he can depend on the same operating system regardless of the hardware that composes the actual machine.

The operating system is always resident in the core memory of the actual machine and is composed of three parts (refer to Figure 1-1). Because there are so many services that can be obtained from the operating system, including the allocation of core memory, processor time, and devices of the actual machine; one part, the service request handler, is responsible for accepting requests for these services. The service request handler passes the requests to another part, the sharable resource allocator, which is responsible for allocating the services requested. If the requested service is for use of a device, the I/O service routines are then notified to carry out the user's request.

1.3 DECSYSTEM-10 NON-RESIDENT SOFTWARE

The third component of the DECsystem-10 is the non-resident software, those programs necessary for the varied operation of the computing system. This software includes the compilers, assemblers, editors, debugging programs, and operating system support programs. These software programs reside on a high-speed mass storage device of the actual machine and are brought into memory when needed by a user. By utilizing the non-resident software, the user of the computing system can create programs, transfer them from one device to another, compile, edit, execute, and debug them, and then receive the results of execution on any specified device.

1.4 DECSYSTEM-10 MULTIPROCESSING

The DECsystem-10 can be a single-processor system or a dual-processor system, composed of a primary processor and a secondary processor. Each processor in the dual-processor system runs user programs, schedules itself, and fields instruction traps. In addition to these tasks, the primary processor also has control of all the input/output devices and processes all requests to the operating system. The primary processor completes any job that the secondary processor could not finish because of a request to the operating system. The two processors are connected to the same memory and execute the same copy of the operating system, thereby saving core memory over a multiprocessing system in which each processor has its own copy. The primary objective in the DECsystem-10 dual-processor environment is to provide more processing power than that found in the single-processor DECsystem-10. This means that with the addition of the second processor, more users can run at the same time. Or, if more users are not allowed on the

system, the addition of the second processor reduces the elapsed time required to complete the processing of most programs.

1.5 MULTIMODE COMPUTING

The DECsystem-10 is designed for the concurrent operations of timesharing, Multiprogram Batch, real-time and remote communications in either single or dual-processor systems. In providing these multifunction capabilities, the DECsystem-10 services interactive users, operates local and remote batch stations, and performs data acquisition and control functions for on-line laboratories and other real-time projects. By dynamically adjusting system operation, the DECsystem-10 provides many features for each class of user and is therefore able to meet a large variety of computational requirements.

1.5.1 Timesharing

Timesharing takes maximum advantage of the capabilities of the computing system by allowing many independent users to share the facilities of the DECsystem-10 simultaneously. Because of the interactive, conversational, rapid-response nature of timesharing, a wide range of tasks - from solving simple mathematical problems to implementing complete and complex information gathering and processing networks - can be performed by the user. The number of users on the system at any one time depends on the system configuration and the total computing load on the system. DECsystem-10 timesharing is designed to allow for up to 512 active terminals. These terminals include CRTs and other terminals which operate at speeds of 110 to 2400 baud. Terminal users can be located at the computer center or at remote locations connected to the computer center by communication lines.

1.5.1.1 Command Control Language - By allowing resources to be shared among users, the timesharing environment utilizes processor time and system resources that are wasted in single-user systems. Users are not restricted to a small set of system resources, but instead are provided with the full variety of facilities. By means of his terminal, the user has on-line access to most of the system's features. This on-line access is available through the operating system command control language, which is the means by which the timesharing user communicates with the computing system.

Through the command language, the user controls the running of his task, or job, to achieve the results he desires. He can create, edit, and delete his files; start, suspend, and terminate his job; compile, execute, and debug his program. In addition, since multiprogramming batch software accepts the same command language as the timesharing software, any user can enter his program into the batch run queue. Thus, any timesharing terminal can act as a remote job entry terminal.

1.5.1.2 Peripheral Devices - With the command language, the user can also request assignment of any peripheral device; e.g., magnetic tape, DECtape, and private disk pack, for his own

exclusive use. When the request for assignment is received, the operating system verifies that the device is available to this user, and the user is granted its private use until he relinquishes it. In this way, the user can also have complete control of devices such as card readers and punches, paper-tape readers and punches, and line printers.

1.5.1.3 **Spooling** - When private assignment of a slow-speed device (e.g., card punch, line printer, paper-tape punch, and plotter) is not required, the user can employ the spooling programs of the operating system. Spooling is a method by which output to a slow-speed device is placed on a high-speed disk or drum. This technique prevents the user from using unnecessary time and space in core while waiting for either a device to become available or output to be completed. In addition, the device is managed to a better degree because the users cannot tie it up indefinitely, and the demand fluctuations experienced by these devices are equalized.

1.5.1.4 **Mass Storage File System** - Mass storage devices, such as disks and drums, cannot be requested for a user's exclusive use, but must be shared among all users. Because many users share these devices, the operating system must ensure independence among the users; one user's actions must not affect the activities of another unless the users desire to work together. To guarantee such independence, the operating system provides a file system for disks, disk packs, and drums. Each user's data is organized into groups of 128-word blocks called files. The user gives a name to each of his files, and the list of these names is kept by the operating system for each user. The operating system is then responsible fro protecting each user's file storage from intrusion by unauthorized users.

In addition to allowing independent file storage for users, the operating system permits sharing of files among individual users. For example, programmers working on the same project can share the same data in order to complete a project without duplication of effort. The operating system lets the user specify protection rights, or codes, for his files. These codes designate if other users may read the file, and after access, if the files can be modified in any way.

The user of the DECsystem-10 is not required to preallocate file storage; the operating system allocates and deallocates the file storage space dynamically on demand. Not only is this convenient for the user because he does not have to worry about allocation when he is creating files, but this feature also conserves storage by preventing large portions of storage from being unnecessarily tied up.

1.5.1.5 **Core Utilization** - The DECsystem-10 is a multiprogramming system, i.e., it allows multiple independent user programs to reside simultaneously in core and to run concurrently. This technique of sharing core and processor time enhances the efficient operation of the system by switching the processor from a program that is temporarily stopped because of I/O transmission to a program that is executable. When core and the processor are shared in this manner, each user's program has a memory area distinct from the area of other users. Any attempt to read or change information outside of the area a user can access immediately stops the program and notifies the operating system.

Because available core can contain only a finite number of programs at any one time, the computing system employs a secondary memory, usually disk or drum, to increase the number of users serviced. User programs exist on the secondary memory and move into core for execution. Programs in core change places with the programs being transferred from secondary memory for maximum use of available core. Because the transferring, or swapping, takes place directly between core and the secondary memory, the central processor can be operating on a user program in one part of core while swapping is taking place in another. This independent overlapped operation greatly improves system utilization by increasing the number of users that can be accommodated at the same time.

To further increase the utilization of core, the operating system allows the users to share the same copy of a program or data segment. This prevents the excessive core usage that results when a program is duplicated for several users. A program that can be shared is called a reentrant program and is divided into two parts or segments. One segment contains the code that is not modified during execution (e.g., compilers and assemblers) and can be used by any number of users. The other segment contains the user's code and data that are developed during the compiling process. The operating system invokes protection for shared segments to guarantee that they are not accidentally modified.

The virtual memory option permits a user program to execute with an address space greater than the physical memory actually allocated to that program during execution. User jobs are swapped as described above. However, the entire program may not necessarily be in core during execution. Programs are divided into pages each of which is 512 words long. Some of these pages may remain on secondary storage while the program executes. When a virtual memory job attempts to access a page that is not in core, a Page Fault Handler decides which page or pages to remove from core and which to bring in from secondary storage.

1.5.1.6 General-Purpose Timesharing - Timesharing on the DECsystem-10 is general purpose; i.e., the system is designed in such a way that the command language, input/output processing, file structures, and job scheduling are independent of the programming language being used. In addition, standard software interfaces make it easy for the user to develop his own special language or systems. The general purpose approach is demonstrated by the many programming languages implemented by DECsystem-10 customers.

1.5.2 Multiprogram Batch

Multiprogram batch software enables the DECsystem-10 to execute up to 14 batch jobs concurrently with timesharing jobs. Just as the timesharing user communicates with the system by way of his terminal, the batch user normally communicates by way of the card reader. (However, he can enter his job from an interactive terminal.) Unlike the timesharing user, the batch user can punch his job on cards, insert a few appropriate control cards, and leave his job for an operator to run. In addition, the user can debug his program in the timesharing environment and then run it in batch mode without any additional coding.

1.5.2.1 Multiprogram Batch Components - The Multiprogram Batch software consists of a series of programs: the input spooler, SPRINT-10; the batch controller, BATCON; the queue manager, QMANGR; and the output spoolers, LPTSPL, CDPSPL, PTPSPL, and PLTSPL (Figure 1-2). The input spooler is responsible for reading the input from the input device and for

entering the job into the batch controller's, BATCON's, input queue. The input spooler uses the queue manager to do the actual entering into the batch input queue. Although the input spooler is oriented toward card reader input, disk and magnetic tape also can be handled. The input information is then separated according to the control commands and placed into disk files, either user data files or the batch controller's control file, for subsequent processing. In addition, the input spooler creates the job's log file and enters a report of its processing of the job, along with a record of any operator intervention during its processing. The log file is part of the standard output that the user receives when his job terminates.

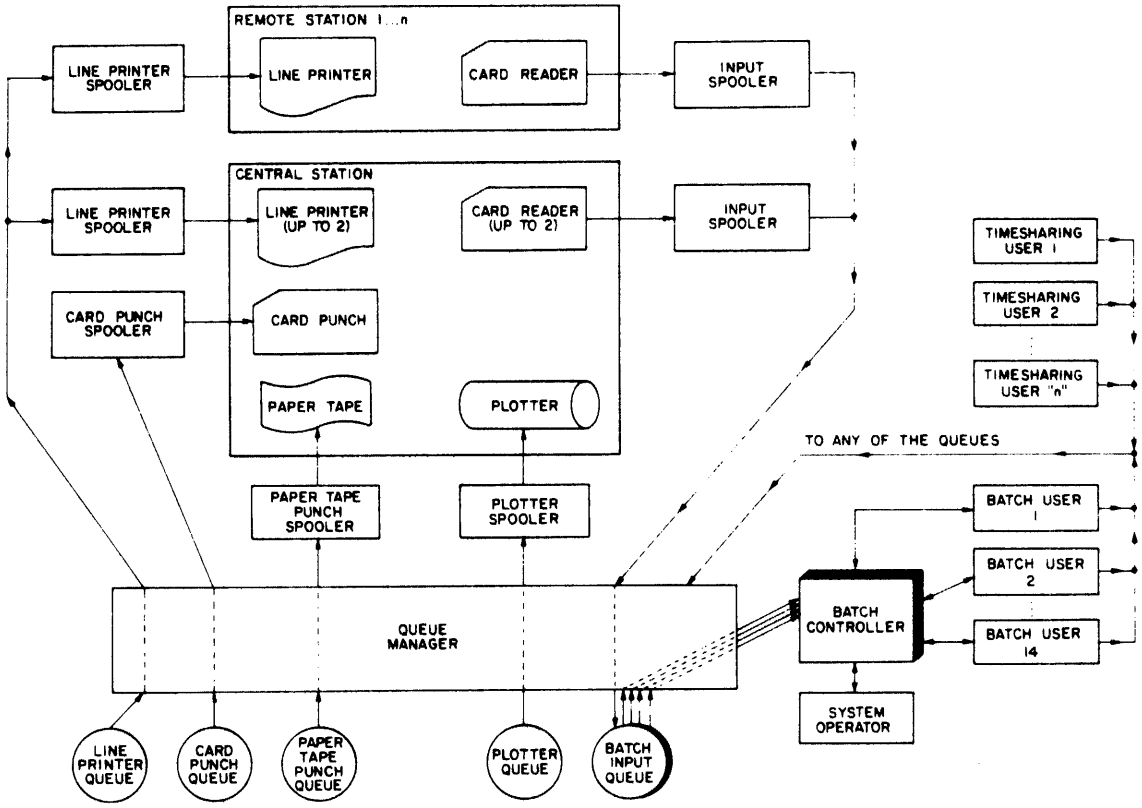

Figure 1-2
Programs in the Batch System

After the input spooler reads the end-of-file and closes the disk files, it makes an entry in the batch controller's input queue. The batch controller processes batch jobs by reading the entries in its queue. The control file created by the input spooler is read by the batch controller, and data and non-resident software commands are passed directly to the user's job. Operating system commands are detected by the batch controller and passed to the operating system for action. Most operating-system and non-resident software commands available to the timesharing user are also available to the batch user. Therefore, only one control language need be learned for both timesharing and batch. During the processing of the job and the control file, the batch controller adds information to the log file for later analysis by the user.

The queue manager is responsible for scheduling jobs and maintaining both the batch controller's input queue and the output spooling queues. A job is scheduled to run under the batch controller according to external priorities, processing time limits, and core requirements which are dynamically computed by the queue manager, and according to parameters specified by the user for his job, such as start and deadline time limits for program execution. The queue manager makes an entry for the job in the batch input queue based upon the various priorities. After the job is completed, the queue manager again schedules it for output by placing an entry in an output queue. When the output is finished, the job's entry in the output queue is deleted by the queue manager.

The output spooling programs improve system throughput by allowing the output from a job to be written temporarily on the disk for later transfer instead of being written immediately on a particular output device. The log file and all job output are placed by the queue manager into one or more output queues to await spooling. When the specified device is available, the output is then processed by the appropriate spooling program. These spooling programs may be utilized by all users of the computing system.

1.5.2.2 Batch Use of System Features - The Multiprogram Batch software employs many of the computing system's features in order to operate with maximum efficiency. Because core memory is not partitioned between batch and timesharing jobs, batch jobs can occupy any available area of core. Fast throughput for high priority batch jobs is accomplished with the same swapping technique used for rapid response to interactive users. When available core is not large enough for a high priority batch job, the operating system transfers programs of lower priority to secondary memory in order to provide space for the job. This I/O transfer is done at the same time that the processor is operating on another job. Thus, processing can be overlapped with I/O to utilize time that would otherwise be wasted. Batch jobs can also share programs with timesharing and other batch jobs. Only one copy of a sharable program need be in core to service any number of batch and timesharing jobs at the same time.

1.5.2.3 Flexibility of the Batch System - Multiprogram Batch allows the user a wide range of flexibility. The input spooler normally reads from the card reader, but can read from magnetic tape or disk in order to create a control file on disk and to enter the job into the batch controller's input queue. However, a job can be entered from an interactive terminal, in which case the user by-passes the input spooler and creates a control file on disk for the batch controller. The control file contains the operating system commands and non-resident software commands necessary to run the job. The user then enters the job into the batch controller's input queue by way of an operating system command string. In the command string, the user can include switches to define the operation and set the priorities and limits on core memory and processor time.

1.5.2.4 Job Dependency - Although jobs are entered sequentially into the batch system, they are not necessarily run in the order that they are read because of priorities, either set by the user in an input spooler control command or computed by the queue manager when determining the scheduling of jobs. Occasionally, the user may wish to submit jobs that must be executed in a particular order; in other words, the execution of one job is dependent on another. To ensure that jobs are executed in the proper order, the user must specify an initial dependency count in a control command of the dependent job. This dependency count is then part of the input queue

entry. A control command in the job on which the dependent job depends decrements the count. When the count becomes zero, the dependent job is executed.

1.5.2.5 Error Recovery - The user can control system response to error conditions by including in his job commands to the batch controller to aid in error recovery. These commands are copied into the control file by the input spooler. With error recovery commands, the user specifies the action to be taken when his program contains a fatal error, as for example, to skip to the next program or to transfer to a special user-written error handling routine. If an error occurs and the user did not include error recovery conditions in his job, the batch controller initiates a standard dump of the user's core area and terminates the job. This core dump provides the user with the means to debug his program.

Although the batch system allows a large number of parameters to be specified, it is capable of operating with very few user-specified values. If a parameter is missing, the batch system supplies a reasonable default value. These defaults can be modified by the individual installations.

1.5.2.6 Operator Intervention - Normal operating functions performed by the programs in the batch system require little or no operator intervention; however, the operator can exercise a great deal of control if necessary. He can specify the number of system resources to be dedicated to batch processing by limiting the number of programs and both the core and processor time for individual programs. He can stop a job at any point, requeue it, and then change its priorities. By examining the system queues, he can determine the status of all batch jobs. In addition, the programs in the batch system can communicate information to the operator and record a disk log of all messages printed at the operator's console. All operator intervention during the running of the input spooler and the batch controller causes messages to be written in the user's log file, as well as in the operator's log file, for later analysis.

1.5.3 Real-Time

For a system to be satisfactory for real-time operations, two important requirements must be met. The more important requirement is fast response time. Because real-time devices cannot store their information until the computing system is ready to accept it, the system would be useless for real-time if the response requirements of a real-time project could not be satisfied. The operating system must allocate system resources dynamically in order to satisfy the response and computational requirements of real-time jobs without affecting the simultaneous operations of timesharing and batch jobs. As part of its normal operation, the DECsystem-10 operating system satisfies this response requirement by overlapping I/O operations with processing time and by reacting to a constantly changing system load quickly and efficiently.

The second requirement is protection. Each user of the computing system must be protected from other users, just as the system itself is protected from all user program errors. In addition, since real-time systems have special real-time devices associated with jobs, the computing system must be protected from hardware faults that could cause system breakdown. And, because protection is part of the function of the operating system, the real-time software employs this feature to protect users as well as itself against hardware and software failures. Therefore, inherent in the operating system is the capability of real-time, and it is by way of calls to the

operating system that the user obtains real-time services. The services obtained by calls within the user's program include:

1. locking a job in core
2. connecting a real-time device to the priority interrupt system
3. placing a job in a high-priority run queue
4. initiating the execution of FORTRAN or machine language code on receipt of an interrupt, and
5. disconnecting a real-time device from the priority interrupt system.

1.5.3.1 Locking Jobs - Memory space is occupied by the resident operating system and by a mix of real-time and non-real time jobs. The only fixed partition is between the resident operating system and the remainder of memory. Since a real-time job needs to be in memory so as not to lose information when its associated real-time device interrupts, the job can request that it be locked into core. This means that the job is not to be swapped into secondary memory and guarantees that the job is readily available when needed. The operating system optimizes the placement of the job by positioning it in core so as to obtain the maximum amount of contiguous core space in the remaining memory. Because memory is not divided into fixed partitions, it can be utilized to a better degree by dynamically allocating more space to real-time jobs when real-time demands are high. As real-time demands lessen, more memory can be made available to timesharing and batch usage.

1.5.3.2 Real-Time Devices - The real-time user can connect real-time devices to the priority interrupt system, respond to these devices at interrupt level, remove the devices from the interrupt system, and/or change the priority interrupt level on which these devices are assigned. There is no requirement that these devices be connected at system generation time. The user specifies both the names of the devices generating the interrupts and the priority levels on which the devices function. The operating system than links the devices to the operating system.

The user can control the real-time device in one of two ways: single mode or block mode. In single mode, the user's interrupt program is run every time the real-time device interrupts. In block mode, the user's interrupt program is run after an entire block of data has been read from the real-time device. When the interrupt occurs from the device in single mode or at the end of a block of data in block mode, the operating system saves the current state of the machine and jumps to the user's interrupt routine. The user services his device and then returns control to the operating system to restore the previous state of the machine and to dismiss the interrupt. Any number of real-time devices may be placed on any available priority interrupt channel.

1.5.3.3 High-priority Run Queues - The real-time user can receive faster response by placing jobs in high-priority run queues. These queues are examined before all other run queues in the computing system, and any runnable job in a high-priority queue is executed before jobs in

other queues. In addition, jobs in high-priority queues are not swapped to secondary memory until all other queues have been scanned. When jobs in a high-priority queue are to be swapped, the lowest priority job is swapped first and the highest priority job last. The highest priority job swapped to secondary memory is the first job to be brought into core for immediate execution. Therefore, in addition to being scanned before all other queues for job execution, the high-priority queues are examined after all other queues for swapping to secondary memory and before all other queues for swapping from secondary memory.

1.5.3.4 Job Communication - TheDECsystem-10 operating system enables a real-time user to communicate with other jobs through the use of sharable data areas. This also enables a data analysis program, for example, to read or write an area in the real-time job's core space. Since the real-time job associated with the data acquisition would be locked in core, the data analysis program residing on secondary memory would become core resident only when the real-time job had filled a core buffer with data. Operating system calls can be used to allow the data analysis program to remain dormant on secondary memory until a specified event occurs in the real-time job; e.g., a buffer has been filled with data for the data analysis program to read. When the specified event occurs, the dormant program is then activated to process the data. The core space for the real-time job's buffer area or the space for the dormant job does not need to be reserved at system generation time. The hardware working in conjunction with the operating systems core management facilities provides optimum core usage.

1.5.4 Inter-Process CommunicationFacility

The Inter-Process Communication Facility (IPCF) provides the capability for independent jobs to communicate with one another. For example, if several programs are involved in processing or maintaining a data base, it is possible that one program might want to inform the others of any modifications it made to the data. A job using IPCF cannot make any changes to another job so protection is in no way sacrificed when using the IPCF feature.

In order to use the IPCF, each participating job that wishes to receive communication from other jobs must request a unique process identifier (PID) from the system. The transmitting job then may send a 'packet' of information to another job. (In addition to the information, the system automatically provides a 'return address' so that the receiving program can respond to the sender.) The monitor maintains a linear queue (the 'mailbox') for each job using IPCF. The packet (or packets) will be kept in the mailbox until the receiving job retrieves it. This queue is not created until a job sends an IPCF packet and it does not occupy any space until such time. The maximum number of packets allowed in a queue at any one time is determined by a 'receive' quota that may be set at each installation for each user. (If no quota is set by the installation, the standard default is five.)

On systems with the virtual memory option the packet could be an entire page. In this case, the monitor takes advantage of the KI10's page mapping hardware to transmit the page without actually copying it.

1.5.5 Remote Communications

Until the capability of remote communications was implemented, each remote user of the DECsystem-10 had been individually linked to the computer center by separate long distance telephone lines. Also, the only device that the remote user had available at his location was the terminal; he was able to utilize available devices at the central station, but he could not obtain output other than his terminal output at his remote site. Either he had to travel to the central station to obtain a listing, or he had to have the listings delivered to him. However, with remote communications hardware and software, listing files and data can be sent via a single synchronous full-duplex line to a small remote computer. That remote computer in turn services many remote peripherals, Including terminals, card readers and line printers. This eliminates the need for the user to travel to the central site to obtain his output. The remote computer and its associated peripherals constitute a remote station.

Remote station use of the central computer is essentially the same as local use. All sharable programs and peripherals available to local users at the central computer station are also available to remote users. The remote user specifies the resources he wants to use and, if available, these resources are then allocated in the same manner as to a local user. In addition to utilizing the peripherals at the central station, the remote user can access devices located at his station or at another remote station. Local users at the central station can also make use of the peripherals at remote stations. Therefore, by specifying a station number in appropriate commands to the operating system, each user of the DECsystem-10 is given considerable flexibility in allocating system facilities and in directing input and output to the station of his choice.

The DECsystem-10 allows for simultaneous operation of multiple remote stations. Software provisions are incorporated in the operating system to differentiate one remote station from another. By utilizing peripheral devices at various stations, the user is provided with increased capabilities. For example, data can be collected from various remote stations, compiled and processed at the central station, and then the results of the processing can be sent to all contributors of the data.

Operating system commands not only allow a user to access peripherals at other remote stations, but also allow him to pretend that his job is at a remote station different from the physical station at which he is actually located. In this case, the user has a logical station and can run entire jobs from this station. With this capability, a local user at the central station could become a remote user as far as the system was concerned by changing the location of his job to a remote station in contact with the central station.

In summary, any computer, regardless of how powerful, is only as good as the operating system that maximizes its capabilities. The DECsystem-10 enhances the speed, power, and flexibility of the PDP-10 by dynamically responding to the changing user load and, in doing so, provides the user with a truly flexible and easily-used computing system.

CHAPTER 2

NON-RESIDENT SYSTEM SOFTWARE

For the computer to execute any of the basic operations which it is capable of executing, it must be told which operation it is to perform and where to find the information on which to perform the operation. This requires that a language be established by which the user can indicate to the computer what it needs to know. This language is the machine language of the computer and is unique for each machine. This machine language is the means by which the computer's circuits are instructed to perform the desired operation and, because of this, it is the fastest and most direct method of programming. However, machine language programming is not the easiest method of programming for most users to employ. Although it is not impossible to memorize all of the operation codes recognized by the computer, it can be very difficult for the programmer to remember where each piece of information is inside memory in order to direct the computer to it. Therefore, symbolic language programming was developed to aid the programmer in manipulating the computer.

With symbolic language programming, programs are written using symbols which, when translated, equal the machine language of the computer. Symbol operation codes (mnemonics that specify which operation the user wants the computer to perform) are translated to the actual, or absolute, operation codes that the computer understands. Addresses of core are designated with symbolic labels and are converted into absolute core addresses so that the computer can locate the information on which to perform the desired operation.

There are three kinds of translators used in symbolic language programming: assemblers, compilers, and interpreters. An assembler is a program that is able to take another program written in symbolic language and translate it, item by item, into machine language. Therefore, to assemble a program means to substitute one absolute value for one symbolic notation until the entire program has been translated. A compiler also translates a symbolic language program into a machine language program, but the substitution is not one-to-one. A program written in a compiler language is freer in format than an assembly language program, and the language elements usually resemble English words. The compiler is larger and more complex than most assemblers, because it translates a program that is farther away from the machine language. Generally, one statement written in a compiler language is translated into several machine language instructions. Although a compiler occupies more space in memory and is generally slower than an assembler, a program written in a compiler language is more compatible with other computer models and the language itself is easier to learn and write because of its general statements and freer format. An interpreter differs from a compiler in that a binary version of the program is not produced for storage. Each statement of the source text is translated into machine language and then executed immediately before the next statement is processed. Interpreters are generally useful when the program to be translated will probably be executed only once or twice and therefore not need some of the time consuming features of a compiler, such as program optimization and output of the machine language instructions.

2.1 DECSYSTEM-10 ASSEMBLER

MACRO is the symbolic assembly program on the DECsystem-10. It makes machine language programming easier and faster for the user by

1. translating symbolic operation codes in the source program into the binary codes needed in machine language instructions

2. relating symbols specified by the user to numeric values

3. assigning absolute core addresses to the symbolic addresses of program instructions and data, and

4. preparing an output listing of the program which includes any errors detected during the assembly process.

MACRO progams consist of a series of statements that are usually prepared on the user's terminal with a system editing program. The elements in each statement do not have to be placed in certain columns nor must they be separated in a rigid fashion. The assembler interprets and processes these statements, generates binary instructions or data words, and performs the assembly.

MACRO is a two-pass assembler. This means that the assembler reads the source program twice. Basically, on the first pass, all symbols are defined and placed in the symbol table with their numeric values, and on the second pass, the binary (machine) code is generated. Although not as fast as a one-pass assembler, MACRO is more efficient in that less core is used in generating the machine language code and the output to the user is not as long.

MACRO is a device-independent program; it allows the user to select at runtime standard peripheral devices for input and output files. For example, input of the source program can come from the user's terminal and output of the assembled binary program can go to a magnetic tape, and output of the program listing can go to the line printer.

The MACRO assembler contains powerful macro capabilities that allow the user to create new language elements. This capability is useful when a sequence of code is used several times with only the arguments changed. The code sequence is defined with dummy arguments as a macro instruction. Thus, a single statement in the source program referring to the macro by name, along with a list of the real arguments, generates the correct and entire sequence. This capability allows for the expansion and adaption of the assembler in order to perform specialized functions for each programming job.

2.2 DECSYSTEM-10 COMPILERS

2.2.1 ALGOL

The ALGOrithmic Language, ALGOL, is a scientific language designed for describing computational processes, or algorithms. It is a problem-solving language in which the problem is expressed as complete and precise statements of a procedure.

The DECsystem-10 ALGOL system is based on ALGOL-60. It is composed of the ALGOL processor, or compiler, and the ALGOL object time system. The compiler is responsible for reading programs written in the ALGOL language and converting these programs into machine language. Also any errors the user made in writing his program are detected by the compiler and passed on to the user.

The ALGOL object time system provides special services, including the input/output service, for the compiled ALGOL program. Part of the object time system is the ALGOL library, a set of routines that the user's program can call in order to perform calculations. These include the mathematical functions and the string and data transmission routines. These routines are loaded with the user's program when required; the user need only make a call to them. The remainder of the object time system is responsible for the running of the program and providing services for system resources, such as core allocation and management and assignment of peripheral devices.

2.2.2 BASIC

The Beginner's All-purpose Symbolic Instruction Code, BASIC is a problem-solving language that is easy to learn because of its conversational nature. It is particularly suited to a timesharing environment because of the ease of interaction between the user and the computer. This language can be used to solve problems with varying degrees of complexity, and thus has wide application in the educational, business, and scientific markets.

BASIC is one of the simplest of the programming compiler languages available because of the small number of clearly understandable and readily learned statements that are required for solving almost any problem. The BASIC language can be thought of as divided into two sections: one section of elementary statements that the user must know in order to write simple programs and a second section of advanced techniques for more powerful programs.

The BASIC user types in computational procedures as a series of numbered statements that are composed of common English terms and standard mathematical notation. After the statements are entered, a run-type command initiates the execution of the program and returns the results.

The BASIC system has several special features built into its design:

BASIC contains its own editing facilities. Existing programs and data files can be modified directly with BASIC instead of with a system editor by adding or deleting lines, by resequencing the line numbers, or by combining two files into one. The user can request a listing of all or part of any of his files on either the line printer or the terminal.

At the editing level, BASIC allows various peripheral devices (e.g., disk, magnetic tape, DECtape, card reader and punch, high-speed paper-tape reader and punch, and paper-tape reader and punch attached to the user's terminal) to be used for storage or retrieval of programs and data files. Within a program, information can be read from or written to the terminal and to the disk (in the latter case, either sequentially or by random access).

Output to the terminal can be simply formatted by tabs, spaces and columnar headings or more precisely formatted by using the advanced PRINT USING statement.

BASIC has statements designed exclusively for matrix computations.

An advanced string handling capability includes a concatenation operator, substring and search functions, and other string intrinsic functions. Mathematical intrinsic functions are contained in BASIC, along with methods by which the user can define his own functions.

2.2.3 COBOL

The COmmon Business Oriented Language, COBOL, is an industry-wide data processing language that is designed for business applications, such as payroll, inventory control, and accounts-receivable.

Because COBOL programs are written in terms that are familiar to the business user, he can easily describe the formats of his data and the actions to be performed on this data in simple English-like statements. Therefore, programming training is minimal, COBOL programs are self-documenting, and programming of desired applications is accomplished quickly and easily.

The COBOL system is composed of a number of software components. The first is the COBOL compiler which is responsible for initializing the program, scanning the command strings for correct syntax, generating the code, listing, and final assembly. The second component is the object time system, LIBOL, which consists of subroutines used by the code generated by the compiler. These subroutines include the I/O, conversion, comparison, and mathematical routines available to the COBOL user. Another component is the source library maintenance program, which builds and maintains a library of source language entries that can be included in the user's source program at compile time. A fourth component is the stand-alone report generator, COBRG, which produces COBOL source programs. When compiled and loaded, these programs generate reports. The stand-alone program, SORT, accepts commands from the user's terminal in order to perform simple sorting of files. The RERUN program is used to restart a COBOL program that was interrupted during execution because of a system failure, device error, or disk quota error. COBDDT is a utility that debugs COBOL programs. Finally, ISAM builds and maintains indexed sequential files for the user.

DECsystem-10 COBOL accepts two source program formats: conventional format and standard format. The conventional format is employed when the user desires his source programs to be compatible with other COBOL compilers. This is the format normally used when input is from the card reader. The standard format is provided for users who are familiar with the format used in DECsystem-10 operations. It differs from conventional format in that sequence numbers and identification are not used because most DECsystem-10 programs require neither. The compiler assumes that the source program is written in standard format unless the appropriate switch is included in the command string to the compiler or the special sequence numbers created by the symbolic editor LINED are detected by the compiler.

DECsystem-10 COBOL is the highest level of ANSI COBOL available and because it operates within the operating system, it offers the user the many features of the DECsystem-10 in addition to the business processing capability of the language. These features enable the COBOL user to run programs in either, or both, timesharing or batch processing environments, to perform on-line editing and debugging of his programs with the system programs available, to choose various peripheral devices for input and output, and to write programs that can be shared with other users.

2.2.4 FORTRAN

The FORmula TRANSlator language, FORTRAN, is a widely used procedure-oriented programming language. It is designed for solving scientific-type problems and is thus composed of mathematical-like statements constructed in accordance with precisely formulated rules. There-

fore, programs written in the FORTRAN language consist of meaningful sequences of these statements that are intended to direct the computer to perform the specified computations.

FORTRAN has a varied use in every segment of the computer market. Universities find that FORTRAN is a good language with which to teach students how to solve problems via the computer. Scientific markets rely on FORTRAN because of the ease with which scientific problems can be expressed. In addition, FORTRAN is used as the primary data processing language by timesharing utilities.

Because of this wide market, DECsystem-10 FORTRAN-10 is designed to meet the needs of all users. FORTRAN-10 is compatible with and encompasses the ANSI standard. FORTRAN-10 also provides many extensions and additions to this standard which greatly enhance its usefulness and increase its compatibility with other FORTRAN language sets.

FOROTS, the FORTRAN-10 object-time system, implements all program data file functions and provides the user with an extensive runtime error reporting system. An additional feature is that the association between FORTRAN logical units and the file descriptions to which they refer may be made either within the source program or deferred until runtime.

DECsystem-10 FORTRAN-10 also supports FORDDT, an interactive program that is used as an aid in debugging FORTRAN programs.

The FORTRAN system is easy to use in both the timesharing and batch processing environments. Under timesharing, the user operates in an interactive editing and debugging environment. Under batch processing, the user submits his program through the Multiprogram Batch software in order to have the compiling, loading and executing phases performed without his intervention.

FORTRAN programs can be entered into the FORTRAN system from a number of devices: disk, magnetic tape, DECtape, user terminal, paper-tape reader, and card reader. In addition to data files created by FORTRAN, the user can submit data files or FORTRAN source files created by the system programs LINED, PIP, or TECO. The data files contain the data needed by the user's object program during execution. The source files contain the FORTRAN source text to be compiled by the FORTRAN compiler. Commands are entered directly to the FORTRAN compiler with a run-type command or indirectly through a system utility program that accepts and interprets the user's command string and passes it to the compiler. Output can then be received on the user's terminal, disk, DECtape, magnetic tape, card punch, or paper-tape punch.

2.3 DECSYSTEM-10 INTERPRETERS

2.3.1 AID

The Algebraic Interpretive Dialogue, AID, is the DECsystem-10 adaptation of the language elements of JOSS, a program developed by the RAND Corporation. To write a program in the AID language requires no previous programming experience. Commands to AID are typed in via the user's terminal as imperative English sentences. Each command occupies one line and can be

executed immediately or stored as part of a routine for later execution. The beginning of each command is a verb taken from the set of AID verbs. These verbs allow the user to read, store and delete items in storage; halt the current processing and either resume or cancel execution; type information on his terminal; and define arithmetic formulas and functions for repetitive use that are not provided for in the language. However, many common algebraic and geometric functions are provided for the user's convenience.

The AID program is device-independent. The user can create external files for storage of subroutines and data for subsequent recall and use. These files may be stored on any retrievable storage media, but for accessibility and speed, most files are stored on disk.

2.3.2 APL

APL (A Programming Language) is a concise programming language especially suitable for dealing with numeric and character array-structured data. APL is a completely conversational system which tends to increase programmer productivity and expertise by allowing the user to interact with the APL system and his running programs. APL is rich in operators that facilitate array calculations. This higher-level programming is accomplished by suppressing much of the programming detail inside single APL operators. One operator may be used to sort a vector of values in ascending order, thereby making "sort" a primitive operation rather than a tedious subroutine. APL is intended for use as a general data processing language as well as a mathematician's tool.

2.4 DECSYSTEM-10 EDITORS

2.4.1 LINED

The line editor for disk files, LINED, is used to create and edit source files written in ASCII code with line numbers appended. These line numbers allow LINED to reference a line in the file at any time without having the user close and then reopen the file. The user has the option of either specifying the beginning line number and the increment to the next line number when inserting lines or allowing LINED to assume a beginning line number and increment if the user specification is omitted.

Commands to LINED allow the user to create a new file or edit an existing file by inserting, replacing, or deleting lines within the file. Single or multiple lines of the file can be printed on the user's terminal for an aid in editing. When the user has edited the file to his satisfaction, he closes the file and can either open a new file or return to monitor level to assemble or compile his file.

2.4.2 TECO

The Text Editor and COrrector program, TECO, is a powerful editor used to edit any ASCII text file with a minimum of effort. TECO commands can be separated into two groups: one group

of elementary commands that can be applied to most editing tasks, and the larger set of sophisticated commands for character string searching, text block movement, conditional commands, programmed editing, and command repetition.

TECO is a character-oriented editor. This means that one or more characters in a line can be changed without retyping the remainder of the line. TECO has the capability to edit any source document: programs written in MACRO, FORTRAN, COBOL, ALGOL, or any other source language; specifications; memoranda, and other types of arbitrarily-formatted text. The TECO program does not require that line numbers or other special formatting be associated with the text.

Editing is performed by TECO via an editing buffer, which is a section within TECO's core area. Editing is accomplished by reading text from any device (except a user's terminal) into the editing buffer (inputting), by modifying the text in the buffer with data received from either the user's terminal or a command file (inserting), and by writing the modified text in the buffer to an output file (outputting).

A position indicator, or buffer pointer, is used to locate characters within the buffer and its position determines the effect of many of TECO's commands. It is always positioned before the first character, between two characters, or after the last character in the buffer. Various commands, such as insertion commands, always take place at the current position of the buffer pointer.

There are TECO commands to manipulate data within the editing buffer. Input and output commands read data from the input file into the buffer and output data from the buffer to the output file. There are other commands to have one or more characters inserted into the editing buffer, deleted from the buffer, searched for, and/or typed out. In addition, the user can employ iteration commands to execute a sequence of commands repeatedly and conditional execution commands to create conditional branches and skips.

2.4.3 SOUP

The SOftware Updating Package, SOUP, is a set of programs that facilitates the updating of system or user source files. Because software is constantly being updated to reflect changes and improvements made by DEC, a method to make the updating process easier and faster for all concerned was developed. SOUP enables DEC to distribute a file containing only the differences to the software routine instead of redistributing the entire routine. In addition, since customers frequently maintain system files that are modified to reflect their individual needs, SOUP can be used to update these modified files as well. Although SOUP was implemented to update system files, it can be employed to update any source file with more than one version.

The SOftware Updating Package consists of three programs. The first program, CAM, is responsible for

1. comparing the new version of DEC's system file to the previous version to produce a correction file, and

2. merging two correction files derived from the same system file to produce a single correction file.

The correction file contains a series of editing changes that reflect the differences between the old and new versions of the system files. The two functions of CAM can be performed simultaneously or one at a time depending on the user's command string to CAM.

The second program, COMP10, is used when the customer has modified DEC's file to such an extent that CAM cannot compare the modified file to the original file due to buffer overflow. COMP10 has extremely large buffers and can, therefore, be used to generate the correction file.

The third program, FED, reads the correction file and edits the copy of the system file by making the changes indicated in the correction file. When FED has completed its processing, the user has an updated file. As a software manufacturer, DEC sends the user a correction file, and he, in turn, need only run the FED program in order to update his system files.

2.4.4 RUNOFF

RUNOFF facilitates the preparation of typed or printed manuscripts by performing line justification, page numbering, titling, indexing, formatting, and case shifting as directed by the user. The user creates a file with an editor and enters his material through his terminal. In addition to entering the text, the user includes information for formatting and case shifting. RUNOFF processes the file and produces the final formatted file to be output to the terminal, the line printer, or to another file.

With RUNOFF, large amounts of material can be inserted into or deleted from the file without retyping the text that will remain unchanged. After the group of modifications have been added to the file, RUNOFF produces a new copy of the file which is properly paged and formatted.

2.5 DECSYSTEM-10 UTILITIES

2.5.1 CREF

The cross-reference listing program, CREF, is an aid in program debugging and modification. It produces a sequence-numbered assembly listing followed by tables showing cross-references of all operand-type symbols, all user-defined operators, and all machine op codes and pseudo-op codes.

The input to CREF is a modified assembly listing created during assembly or compilation. The command string entered by the user specifies the device on which this assembly listing is located along with the output device on which to list the cross-reference tables and assembly listings. Switches can also be included in the command string in order to control magnetic tape positioning and to select specific sections of the listing output.

2.5.2 DDT

The Dynamic Debugging Technique, DDT, is used for on-line program composition of object programs and for on-line checkout and testing of these programs. For example, the user can

perform rapid checkout of a new program by making a change resulting from an error detected by DDT and then immediately executing that section of the program for testing.

After the source program has been compiled or assembled, the binary object program with its table of defined symbols is loaded with DDT. In command strings to DDT, the user can specify locations in his program, or breakpoints, where DDT is to suspend execution in order to accept further commands. In this way, the user can check out his program section-by-section and, if an error occurs, insert the corrected code immediately. Either before DDT begins execution or at breakpoints, the user can examine and modify the contents of any location. Insertions and deletions can be in source language code or in various numeric and text modes. DDT also performs searches, gives conditional dumps, and calls user-coded debugging subroutines at breakpoint locations.

2.5.3 FAILSAFE

The FAILSAFE program enables both the system operator and the user to recover from a system failure or other unintentional destruction of data on the disk by

1. preserving disk files on magnetic tape and

2. later retrieving these files and placing them back onto the disk.

In addition FAILSAFE provides the system administrator with a method of making more efficient use of available disk space by storing infrequently-used files on magnetic tape instead of allowing these files to occupy needed disk space.

2.5.4 FILEX

The file transfer program, FILEX, converts between various core image formats and reads or writes various DECtape directory formats and standard disk files. Files are transferred as 36-bit binary data with no processing performed on the data except that necessary to convert the core image representation. The core image formats that can be used in conversions are:

1. saved-file format

2. expanded core image file format

3. dump format

4. simple block format (Project MAC's equivalent of DEC's .SAV format), and

5. binary file format.

The desired core image format is determined by the specific extension associated with the file but this extension may be overridden by the use of switches in command strings to FILEX.

DECtapes can be read or written in binary, PDP-6 DECtape format, MIT Project MAC PDP-6/10 DECtape format, PDP-11 DOS format, or PDP-15 format. In the latter two cases, ASCII

files will be converted. The DECtape can be processed quickly via a disk scratch file, which is a much faster method for a tape with many files. This scratch file can be preserved and reused in later command strings. In addition, the DECtape directory can be listed on the user's terminal or zeroed in the appropriate format on the tape. These DECtape format and processing specifiers are indicated by command string switches.

2.5.5 LINK-10

LINK-10, the DECsystem-10 linking loader, merges independently-translated modules of the user's program into a single module and links this module with system modules into a form that can be executed by the operating system. It provides automatic relocation and loading of the binary modules producing an executable version of the user's program. When the loading process has been completed, the user can request LINK-10 either to transfer control to his program for immediate execution or to output the program to a device for storage in order to avoid the loading procedure in the future.

While the primary output of LINK-10 is the executable version of the user's program, the user can request auxiliary output in the form of map, log, save, symbol, overlay plot, and expanded core image files. This additional output is not automatically generated: the user must include appropriate switches in his command strings to LINK-10 in order to obtain this type of output. The user can also gain precise control over the loading process by setting various loading parameters and by controlling the loading of symbols and modules. Furthermore, by setting switches in his command strings to LINK-10, the user can specify the core sizes and starting addresses of modules, the size of the symbol table, the segment into which the symbol table is placed, the messages he will see on his terminal or in his log file, and the severity and verbosity levels of the messages. Finally, he can accept the LINK-10 defaults for items in a file specification or he can set his own defaults that will be used automatically when he omits an item from his command string.

LINK-10 has an overlay facility to be used when the total core required by a user's program is more than the core available to the user. The user organizes his program so that only some portions of the program are required in core at any one time. The remaining portions reside in a disk file and are transferred in and out of core during execution. Because the portion brought into core may overlay a portion already core-resident, the amount of core required by the entire program is reduced. Overlays can be invoked either by runtime routines called from the user's program or by automatic calls to subroutines outside the current overlay link.

2.5.6 PIP

The Peripheral Interchange Program, PIP, is used to transfer data files from one I/O device to another. Commands to PIP are formatted to accept any number of input (source) devices and one output (destination) device. Files can be transferred from one or more source devices to the destination device as either one combined file or individual files. Switches contained in the command string to PIP provide the user with the following capabilities:

1. naming the files to be transferred
2. editing data in any of the input files

3. defining the mode of transfer

4. manipulating the directory of a device if it has a directory

5. controlling magnetic tape and card punch functions, and

6. recovering from errors during processing.

2.5.7 MACY11 and LNKX11

MACY11 is a cross-assembler which operates on the DECsystem-10 and assembles MACRO-11 source code for the PDP-11 family of computers. MACY11 will produce either an absolute binary or a relocatable object module, depending upon the assembly mode. The resulting absolute binary files are transferable to the PDP-11 for execution or, the relocatable object modules may be used as input to the LNKX11 linker on the DECsystem-10 and then transferred to the PDP-11 operating environment. MACY11 will append a symbol table and (optimally) a cross-reference table to the listing file. MACY11 produces a "side-by-side" assembly listing of symbolic source statements, their octal equivalents, assigned addresses and error codes.

2.6 DECSYSTEM-10 MONITOR SUPPORT PROGRAMS

2.6.1 MONGEN

The monitor generator, MONGEN is a dialogue program that enables the system programmer to define the hardware configuration of his individual installation and the set of software options that he wishes to select for his system. This program is a prerequisite for creating a new monitor.

The system programmer defines his configuration in one of four dialogues by answering MONGEN's questions in conversational mode. MONGEN transmits one question at a time to the user's terminal, and the user answers appropriately depending on the content of each question. After all questions have been answered, MONGEN produces MACRO source files containing the user's answers. These source files are then assembled and loaded with the symbol definition file and the monitor data base to yield a monitor tailored to the individual installation.

2.6.2 OPSER

The operator service program, OPSER, facilitates multiple job control from a single terminal by allowing the operator or the user to initiate several jobs, called subjobs, from his terminal. The OPSER program acts as the supervisor of the various subjobs by allowing monitor-level and user-level commands to be passed to all of the subjobs or to individually selected subjobs. Output from the various subjobs can then be retrieved by OPSER.

The subjobs of OPSER run on pseudo-TTY's, a simulated terminal not defined by hardware. All initializations of the pseudo-TTY's are performed by OPSER; the operator need only supply a subjob name. By running system programs, which ordinarily require a dedicated terminal, as subjobs of OPSER, output from the various programs can be concentrated on one hardware terminal instead of many. In addition, OPSER is able to maintain an I/O link between the running jobs and the operator - a feature that is not available when programs run on their own dedicated terminals.

2.6.3 LOGIN

LOGIN is the system program used to gain access to the DECsystem-10. This program consults system administration files in order to determine whether or not a potential user currently is authorized to use the system. If he is not, LOGIN will not permit him access to the system. If the user is authorized, LOGIN informs him of messages of the day, reports any errors detected in his disk files and allows the user to proceed with his job.

2.6.4 KJOB-LOGOUT

The user invokes the system programs KJOB and LOGOUT when he has finished running on the DECsystem-10. The many functions of these programs include saving the user's disk files in the state in which he desires them, enforcing logged-out quotas on all disk file structures, terminating the user's job, and returning the resources allocated to the user back to the system. These resources include the user's job number, his allocated I/O devices, and his allocated core.

2.7 DECSYSTEM-10 APPLICATIONS

2.7.1 Data Base Management System

The Data Base Management System (DBMS-10) is a facility of the DECsystem-10 that permits the user to consolidate his data files into one or more data bases. A data base is a collection of non-redundant data items that can be accessed by a variety of programs and/or applications that have common processing requirements and functional relationships. The data base is created and maintained through modules of DBMS-10. These modules permit the user to structure the data in a way such that each application can access it an optimum fashion, yet no data item is actually duplicated in the data base. This arrangement is accomplished by the data base administrator who structures the data base in a manner such that each application can access it through a search pattern most suited to its needs. Once the data base has been established users can access the data through COBOL programs containing special data base syntax.

2.7.2 TYPESET-10

TYPESET-10 performs a variety of tasks concerned with changing raw text into formatted text acceptable to a typesetting machine. The editors COPYED and PRUFED provide specialized editing functions for typesetting and have the capability of sending a prepared text file on-line to be typeset. JUSTIF is the program that creates a formatted output file by implementing commands embedded in the text. This normally includes justification and hyphenation, plus specification of type face, type size, column width and various vertical and horizontal spacing parameters. Output is accomplished by the ALLOT program, which allows users to specify a typesetting machine appropriate to the copy specifications.

In addition to these basic features, there are programs of particular interest in newspaper applications. WIPTIN, for example, accepts input from wire services, making all necessary character conversions and preparations for rejustification. CLASAD is a program that handles all aspects of classified ad production, from creation, control, and deletion, to billing of customers.

CHAPTER 3

THE RESIDENT OPERATING SYSTEM

The resident operating system is made up of a number of separate and somewhat independent parts, or routines (Figure 3-1). Some of these routines are cyclic in nature and are repeated at every system clock interrupt (tick) to ensure that every user of the computing system is receiving his requested services. These cyclic routines are:

1. the command processor, or decoder
2. the scheduler, and
3. the swapper.

The command decoder is responsible for interpreting commands typed by the user on his terminal and passing them to the appropriate system program or routine. The scheduler decides which user is to run in the interval between the clock interrupts, allocates sharable system resources, and saves and restores conditions needed to start a program interrupt by the clock. The swapper rotates user jobs between secondary memory (usually disk or drum) and core memory after deciding which jobs should be in core but are not. These routines constitute the part of the operating system that allows many jobs to be operating simultaneously.

The non-cyclic routines of the operating system are invoked only by user programs and are responsible for providing these programs with the services available through the operating system. These routines are:

1. the UUO handler
2. the input output routines, and
3. the file handler.

The UUO handler is the means by which the user program communicates with the operating system in order to have a service performed. Communication is by way of programmed operators (also known as UUOs) contained in the user program which, when executed, go to the operating system for processing. The input/output routines are the routines responsible for directing data transfers between peripheral devices and user programs in core memory. These routines are invoked through the UUO handler, thus saving the user the detailed programming needed to control peripheral devices. The file handler adds permanent user storage to the computing system by allowing users to store named programs and data as files.

3.1 THE COMMAND DECODER

The command decoder is the communications link between the user at his terminal and the operating system. Because all the requests for system resources are initiated via the command decoder, it is the most visible part of the system to each user. When the user types commands

and/or requests on his terminal, the characters are stored in an input buffer in the operating system. The command decoder examines these characters in the buffer, checks them for correct syntax, and invokes the system program or user program as specified by the command.

On each clock interrupt, control is given to the command decoder to interpret and process one command in the input buffer. The command appearing in the input buffer is matched with the table of valid commands accepted by the operating system. A match occurs if the command typed in exactly matches a command stored in the system, or if the characters typed in match the beginning characters of only one command. When the match is successful, the legality information (or flags) associated with the command is checked to see if the command can be performed immediately. For instance, a command can be delayed if the job is swapped out to the disk and the command requires that the job be resident in core; the command is executed on a later clock interrupt when the job is back in core. If all conditions as specified by the legality flags are met, control is passed to the appropriate program.

3.2 THE SCHEDULER

The DECsystem-10 is a multiprogramming system; i.e., it allows several user jobs to reside in core simultaneously and to operate sequentially. It is then the job of the scheduler to decide which jobs should run at any given time. In addition to the multiprogramming feature, the DECsystem-10 employs a swapping technique whereby jobs can exist on an external storage device (e.g., disk or drum)as well as in core. Therefore the scheduler decides not only what job is to be run next but also when a job is to be swapped out onto disk or drum and later brought back into core.

All jobs in the system are retained in ordered groupings called queues. These queues have various priorities that reflect the status of each job at any given moment. The queue in which a job is placed depends on the system resource for which it is waiting and, because a job can wait for only one resource at a time, it can be in only one queue at a time. Several of the possible queues in the system are:

1. run queues for jobs waiting for, or jobs in, execution.

2. I/O wait queues for jobs waiting for data transfers to be completed.

3. I/O wait satisfied queues for jobs waiting to run after data transfers have been completed.

4. resource wait queues for jobs waiting for some system resource, and

5. null queue for all job numbers that are not currently being used.

The job's position within certain queues determines the priority of the job with respect to other jobs in the same queue. For example, if a job is first in the queue for a sharable device, it has the highest priority for the device when it becomes available. However, if a job is in an I/O wait queue, it remains in the queue until the I/O is completed. Therefore, in an I/O wait queue, the job's position has no significance. The status of a job is changed each time it is placed into a different queue.

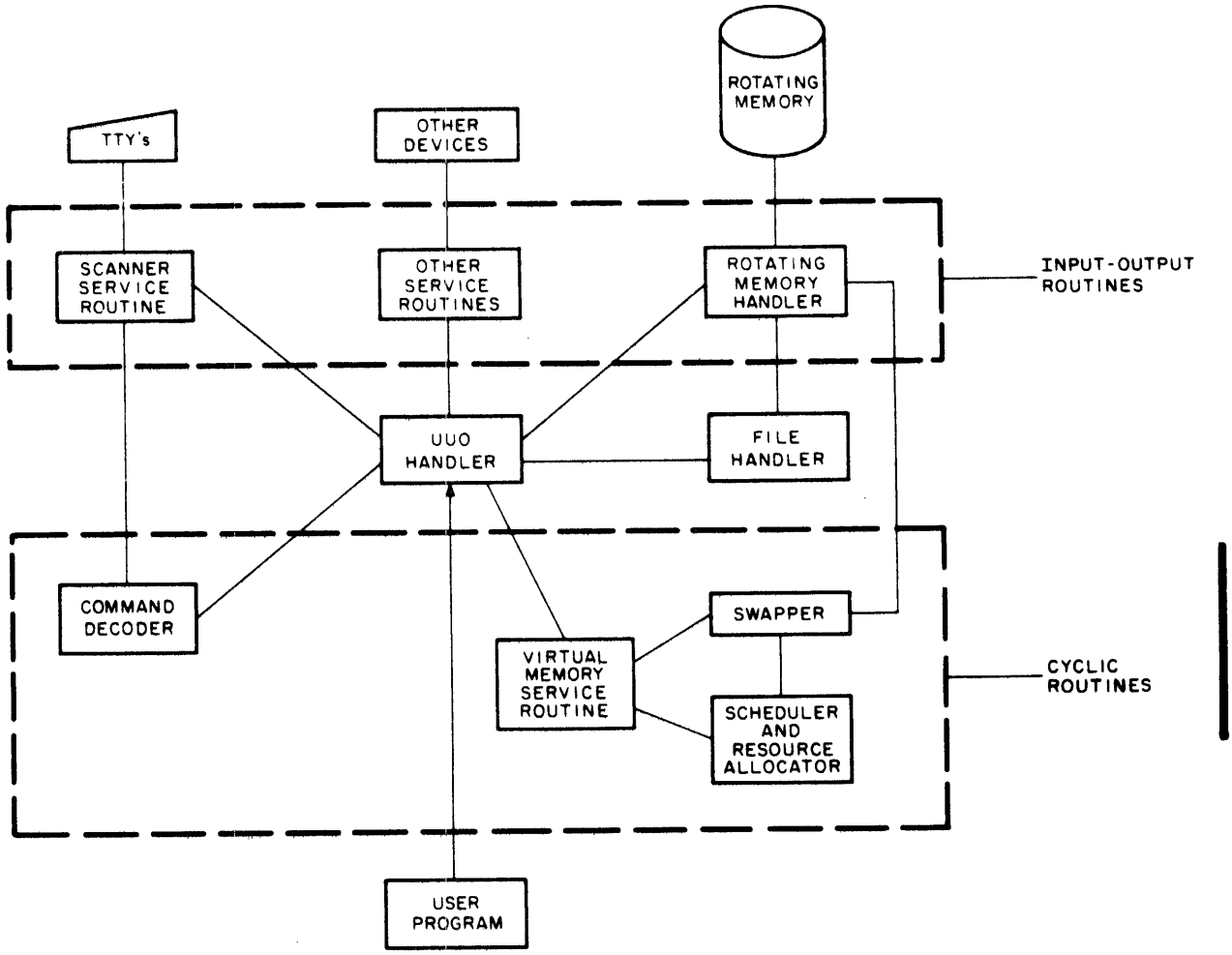

Figure 3-1
The Resident Operating System

The scheduling of jobs into different queues is governed by the system clock. This clock divides the time for the central processor into one-sixtieths of a second. Each job, when it is assigned to run, is given a time slice of ½ second or two seconds, depending on the run queue. When the time slice expires for the job, the clock notifies the central processor and scheduling is performed. The job whose time slice just expired is moved into another run queue, and the scheduler selects the first job in the run queue to run in the next time slice.

Scheduling may be forced before the time slice has expired if the currently running job reaches a point at which it cannot immediately continue. Whenever an operating system routine discovers that it cannot complete a function requested by the job (e.g., it is waiting for I/O to complete or the job needs a device which it currently does not have), it calls the scheduler so that another job can be selected to run. The job that was stopped is then requeued and is scheduled to be run when the function it requested can be completed. For example: when the currently running job begins input from a DECtape, it is placed into the I/O wait queue, and the input is begun. A second job is scheduled to run while the input of the first job proceeds. If the second job then

decides to access a DECtape, it is stopped because the DECtape control is busy, and it is placed in the queue for jobs waiting to access the DECtape control. A third job is set to run. The input operation of the first job finishes, freeing the DECtape control for the second job. The I/O operation of the second job is initiated, and the job is transferred from the device wait queue to the I/O wait queue. The first job is transferred from the I/O wait queue to the highest priority run queue. This permits the first job to preempt the running of the third job. When the time slice of the first job becomes zero, it is moved into the second run queue, and the third job runs again until the second job completes its I/O operation.

In addition, data transfers use the scheduler to permit the user to overlap computation with data transmission. In unbuffered data modes, the user supplies an address of a command list containing pointers to locations in his area to and from which data is to be transferred. When the transfer is initiated, the job is scheduled into an I/O wait queue where it remains until the device signals the scheduler that the entire transfer has been completed.

In buffered modes, each buffer contains information to prevent the user and the device from using the same buffer at the same time. If the user requires the buffer currently being used by the device as his next buffer, the user's job is scheduled into an I/O wait queue. When the device finishes using the buffer, the device calls the scheduler to reactivate the job.

3.3 THE SWAPPER

The swapper is responsible for keeping in core the jobs most likely to be runnable. It determines if a job should be in core by scanning the various queues in which a job may be. If the swapper decides that a job should be brought into core, it may have to take another job already in core and transfer it to secondary memory. Therefore, the swapper is not only responsible for bringing a job into core but is also responsible for selecting the job to be swapped out.

A job is swapped into secondary memory for one of two reasons:

1. a job that is more eligible to run needs to be swapped in and there is not enough room in core for both jobs, and

2. the job needs to expand its core size and there is not enough core space to do so.

If the latter case is true, the job must be swapped out and then swapped in later with the new allocation of core.

The swapper checks periodically to see if a job should be swapped in. If there is no such job, then it checks to see if a job is requesting more core. If there is no job wishing to expand its size, then the swapper does nothing further and relinquishes control of the processor until the next clock tick.

3.4 THE UUO HANDLER

The UUO handler is responsible for accepting requests for services available through the operating system. These requests are made by the user program via software-implemented in-

structions known as programmed operators, or UUOs. The various services obtainable by the user program include:

1. communicating with the I/O devices on the computing system, including connecting and responding to any special devices that may be desired on the system for real-time programming.

2. receiving or changing information concerning either the computing system as a whole or the individual program.

3. altering the operation of the computing system as it concerns the user job, such as controlling execution by trapping or suspending, or controlling core memory by locking.

4. communicating and transferring control between user programs.

The UUO handler is the only means by which a user program can give control to the operating system in order to have a service performed. Contained in the user program are operation codes which, when executed, cause the hardware to transfer control to the UUO handler for processing. This routine obtains its arguments from the user program. The core location at which the UUO operation was executed is then remembered. After the UUO request has been processed, control is returned to the user program at the first or second instruction following the UUO. In this way, the software supplements the hardware by providing services that are invoked through the execution of a single core location just as the hardware services are invoked.

3.5 THE INPUT/OUTPUT ROUTINES

I/O programming in the DECsystem-10 is highly convenient for the user because all of the burdensome details of programming are performed by the operating system. The user informs the operating system of his requirements for I/O by means of UUOs contained in his program. The actual input/output routines needed are then called by the UUO handler.

Since the operating system channels communication between the user program and the device, the user does not need to know all the peculiarities of each device on the system. In fact, the user program can be written in a similar manner for all devices. The operating system will ignore, without returning an error message, operations that are not pertinent to the device being used. Thus, a terminal file and a disk file can be processed identically by the user program. In addition, user programs can be written to be independent of any particular device. The operating system allows the user program to specify a logical device name, which can be associated with any physical device at the time when the program is to be executed. Because of this feature, a program that is coded to use a specific device does not need to be rewritten if the device is unavailable. The device can be designated as a logical device name and assigned to an available physical device with one command to the operating system.

Data is transmitted between the device and the user program in one of two methods: unbuffered mode or buffered mode. With unbuffered data modes the user in his program supplies the device with an address, which is the beginning of a command list. Essentially, this command list contains pointers specifying areas to the user's allocated core to or from which data is to be transferred. The user program then waits until the operating system signals that the entire com-

mand list has been processed. Therefore, during the data transfer, the user program is idly waiting for the transfer to be completed.

Data transfers in buffered mode utilize a ring of buffers set up in the user's core area. Buffered transfers allow the user program and the operating system's I/O routines to operate asynchronously. As the user program uses one buffer, the operating system processes another one by filling or emptying it as interrupts occur from the device. To prevent the user program and the operating system from using the same buffer at the same time, each buffer has a use bit that designates who is using the buffer. Buffered data transfers are faster than unbuffered transfers because the user program and the operating system can be working together in processing the data.

Several steps must be followed by the user program in order for the operating system to have the information it needs to control the data transfers. Each step is indicated to the operating system with one programmed operator. In the first step, the specific device to be used in the data transfer must be selected and linked to the user program with one of the software I/O channels available to the user's job (OPEN or INIT programmed operators). This device remains associated with the software I/O channel until it is disassociated from it (via a programmed operator) or a second device is associated with the same channel. In addition to specifying the I/O channel and the device name, the user program can supply an initial file status, which includes the type of data transfer to be used with the device (e.g., ASCII, binary), and the location of the headers to be used in buffered data transfers. The operating system stores information in these headers when the user program executes programmed operators, and the user program obtains from these headers all the information needed to fill or empty buffers.

Another set of programmed operators (INBUF and OUTBUF) establishes the actual buffers to be used for input and output. This procedure is not necessary if the user is satisfied to accept the two buffers automatically set up for him by the operating system.

The next step is to select the file that the user program will be using when reading or writing data. This group of operators (LOOKUP and ENTER) is not required for devices that are not file-structured (e.g., card reader, magnetic tape, paper-tape punch). However, if used, they will be ignored thus allowing file-structured devices to be substituted for non-file structured devices without the user rewriting the program.

The third step is to perform the data transmission between the user program and the file (IN, INPUT, OUT, and OUTPUT). When the data has been transmitted to either the user program on input or the file on output, the file must be closed (CLOSE, fourth step) and the device released from the channel (RELEASE, fifth step). This same sequence of programmed operators is performed for all devices; therefore, the I/O system is truly device dependent because the user program does not have to be changed every time a different device is used.

In addition to reading or writing data to the standard I/O devices, provisions are included in the operating system for using the terminal for I/O during the execution of the user program. This capability is also obtained through programmed operators. As the user program is running, it can pause to accept input from or to type output to the terminal. The operating system does all buffering for the user, thus saving him programming time. This method of terminal I/O provides the user with a convenient way of interacting with his running program.

3.6 FILE HANDLER

The disk file handler manages user and system data; thus , this data can be stored, retrieved, protected, and/or shared among other users of the computing system. All information in the system is stored as named files in a uniform and consistent fashion thus allowing the information to be accessed by name instead of by physical disk addresses. Therefore, to reference a file, the user does not need to know where the file is physically located. A named file is uniquely identified in the system by a filename and extension, an ordered list of directory names (UFDs and SFDs) which identify the owner of the file, and a file structure name which identifies the group of disk units containing the file.

Usually a complete disk system is composed of many disk units of the same and/or different types of disks. Therefore, the disk system consists of one or more file structures - a logical arrangement of files on one or more disk units of the same type. This method of file storage allows the user to designate which disk unit of the file structure he wishes to use when storing his files. Each file structure is logically complete and is the smallest section of file memory that can be removed from the system without disturbing other units in other file structures. All pointers to areas in a file structure are by way of logical block numbers rather than physical disk addreses; there are no pointers to areas in other file structures, thereby allowing the file structure to be removed.

A file structure contains two types of files: the data files that physically contain the stored data or programs, and the directory files that contain pointers to the data files. Included in these directory files are master file directories, user file directories, and sub-file directories. Each file structure has one master file directory (MFD). This directory file is the master list of all the users of the file structure. The entries contained in the MFD are the names of all the user file directories on the file structure. Each user with access to the file structure has a user file directory (UFD) that contains the names of all his files on that file structure; therefore, there are many UFDs on each file structure. As an entry in the user file directory, the user can include another type of directory file, a sub-file directory (SFD). The sub-file directory is similar to the other types of directory files in that it contains as entries the names of all files within that sub-directory. This third level of directory allows groups of files belonging to the same user to be separate from each other. This is useful when organizing a large number of files according to function. In addition, sub-file directories allow non-conflicting simultaneous runs of the same program using the same filenames.

As long as the files are in different sub-file directories, they are unique. Sub-file directories exist as files pointed to by the user file directory, and can be nested to the depth specified by the installation at system generation time.

All disks are composed of two parts: data and information used to retrieve data. The retrieval part of the file contains the pointers to the entire file, and is stored in two distinct locations on the device and accessed separately from the data. System reliability is increased with this method because the probability of destroying the retrieval information is reduced; system performance is improved because the number of positionings needed for random-access methods is reduced. The storing of retrieval information is the same for both sequential and random access files. Thus a file can be created sequentially and later read randomly, or vice versa, without any data conversion.

One section of the retrieval information is used to specify the protection associated with the file. This protection is necessary because disk storage is shared among all users, each of whom may

desire to share files with, or prevent files from being written, read, or deleted by, other users. These protection codes are assigned by the user when the file is created and designate the users who have privileges to access the file. Users are divided into three categories: the user who created the file (the owner of the file), the user on the same project as the owner of the file, and the remaining users of the system. The owner of the file controls the protection of the file; thus, he can indicate who may read, write, or modify his file. It is always possible for the owner to change the protection of his file and when it is changed, the new protection remains until he modifies it again. If a file is created without a protection code, the operating system substitutes an installation-defined standard protection code.

Disk quotas are associated with each user (each project-programmer number) on each file structure in order to limit the amount of information that can be stored in the UFD of a particular file structure. When the user gains access to the computing system, he automatically begins using his logged in quota. This quota is not a guaranteed amount of space, and the user must compete with other users for it. When the user leaves the computing system, he must be within his logged-out quota. This quota is the amount of disk storage space that the user is allowed to maintain when he is not using the system and is enforced by the system program that is used in logging off the system. Quotas are determined by the individual installation and are, therefore, used to ration disk resources in a predetermined manner.

To the user, a file structure is like a device; i.e., a file structure name or set of file structure names can be used as the device name in command strings or UUO calls to the operating system. Although the file structures or the units composing the file structures can be specified by their actual names, most users specify a general, or generic, name (DSK) which will cause the operating system to select the appropriate file structure. The appropriate file structure is determined by a job search list. Each job has its own job search list with the file structure names in the order in which they are to be accessed when the generic name is specified as the device. This search list is established by LOGIN and thus each user has a UFD for his project-programmer number in each file structure in which LOGIN allows him to have files.

File writing on the disk can be defined by one of three methods: creating, superseding, and updating. The user is creating a file if no other file of the same name exists in the user's directory on the indicated file structure. If another file with the same name already exists in the directory, the user is superseding, or replacing, the old file with the new file. Other users sharing the old file at the time it is being superseded continue using the old file and are not affected until they finish using the file and then try to access it again. At that time, they read the new file. When a user updates a file, he modifies selected parts of the file without creating an entirely new version. This method eliminates the need to recopy a file when making only a small number of changes. If other users try to access a file while it is being updated, they receive an error indication issued by the system.

File storage is dynamically allocated by the file handler during program operations, so the user does not need to give initial estimates of file length or the number of files. Files can be any length, and each user may have as many files as he wishes, as long as disk space is available and the user has not exceeded his logged-in quota. This feature is extremely useful during program development or debugging when the final size of the file is still unknown. However, for efficient random access, a user can reserve a contiguous area on the disk if he desires. When he has completed processing, he can keep his preallocated file space for future use or return it so that other users can have access to it.

3.7 SUMMARY

In summary, the resident operating system supervises user jobs and provides various services to these jobs. It acts as an operator by performing specific functions in response to specific events which occur within the system. Many functions are performed in accordance with a periodic event, the system clock interrupt. Other functions are responded to in accordance with the action of the user program.

APPENDIX A

DECSYSTEM-10 HARDWARE

DECsystem-10 is the name for the family of DEC's large computing systems. Each of the six systems in the DECsystem-10 range is centered around one or two PDP-10 central processors. The systems are distinguished from each other by their range of performance, which is achieved by adding more hardware. The additional hardware that increases performance in the expansion from a small to a larger system includes: swapping devices, central processor, core memories, and peripheral equipment, including data communications systems. The systems have no fixed hardware boundary because an individual system can be expanded to any size. No software changes are required in expanding an individual system; all configurations of the DECsystem-10 use the same operating system for all applications.

A.1 DECSYSTEM-1040

The 1040 is the smallest of the six systems. The typical configuration of this system has a KA10 central processor, 64K high-speed MF10 core memories, the RP02 disk system with up to two disk packs, the TU10A magnetic tape system with up to two drives, and low-speed peripheral equipment including a CR10-F card reader, an LSP10 line printer, and local DC10 lines. This is an excellent system for the scientific research lab where multiple real-time tasks and general computing are required, and also for small colleges where there is a need for handling administrative, student, and faculty workloads simultaneously. The system is easily expandable with most equipment on the DECsystem-10 Equipment List.

A.2 DECSYSTEM-1050

The 1050 is a full capability, medium power system. The addition of a high-speed RH10 swapping system substantially increases the number of simultaneous users on the system. Other components of this system include: the KA10 central processor, 64 to 96K high-speed MF10 core memories, the RP03 disk system with up to four disk packs, the TU10A magnetic tape system, the CR10-D card reader, the LP10-F line printer, and 32 local lines in the DC10 communication system. The 1050 is well suited for the educational and scientific environments because it has the capability of running ALGOL, BASIC, COBOL, and FORTRAN compilers concurrently on a configuration that is economically priced and easy to learn and use. Business data processing areas find that with the 1050, COBOL program preparation is enhanced by interactive editing and debugging via local or remote terminals.

A.3 DECSYSTEM-1055

The 1055 is the dual processor 1050; it provides fast execution of compute-bound jobs because of the addition of the second processor.This system has two parallel KA10 processors connected with one operating system in order to double the computing power of the 1050 and at the same time to maintan the same interface between the user and the computing system.This approach of co-equal processors gives the user increased computing capacity when processng power is in

heavy demand under multi-task loads. In addition to the two KA10 processors, the typical 1055 has 96K words of MF10 core memories, with one MX10 memory port multiplexer, one RH10 swapping system, one RP03 disk system with up to eight disk packs, one TU40, 120KC magnetic tape system, one CR10-D card reader, the LP10-F line printer, and 32 local lines in the DC10 communication system.

A.4 DECSYSTEM-1060

The 1060 computing system is designed for the smaller installation whose requirements are not met by the DECsystem-1040. The 1060 provides almost double the central processor speed of the 1040 because of the KI10 central processor. This processor has hardware memory paging, double-precision floating-point arithmetic, instruction lookahead and virtual memory capability. In addition to the KI10 processor, the typical 1060 has at least 64K words of MF10 core memory, two drives on the RP03 disk system, the TU40 magnetic tape system with one drive, the CR10-E card reader, the LP10-F line printer and 16 lines in the DC76 communication system.

A.5 DECSYSTEM-1070

The 1070 is a large-scale computing system with almost twice the central processor speed of the DECsystem-1050 because of the KI10 central processor. In addition to the KI10 processor, the typical 1070 comprises at least 96K words of MF10 core memory, the RH10 swapping system with two drives, an RP03 disk system of four disk drives with a total of 41.6 million words (249.6 million characters) of storage, a TU40 magnetic tape system with three 120KC drives, a 1200 character-per-minute CR10-E card reader, a 1250 line-per-minute LP10-F line printer and a DC76 communication system capable of supporting 128 lines. With the increased memory size, the high performance peripheral systems, and the large file system, the 1070 is configured for maximum support of remote batch capabilities through the synchronous communication equipment. Multiple remote stations have simultaneous access to the DECsystem-1070, with each capable of concentrating up to 16 terminals to its computer.

A.6 DECSYSTEM-1077

The 1077 is the dual-processor 1070 with fast execution of computing loads because of the second KI10 central processor. From any terminal it appears to the user as though there is one larger system with all resources shared among all users. This system typically has at least 128K of MF10 core memory, four drives on the RH10 swapping system, the RP03 disk system with four disk drives providing a total of 41.6 million words (249.6 million characters) of storage, a TU40 magnetic tape system with four 120KC drives, a 1250 line-per-minute LP10-F line printer, a 1200 card-per-minute CR10-E card reader, and a DC76 communication system capable of supporting 128 lines. In expanding to the 1077 from a smaller system, the user notices increased computing power, but he does not need to change his programs or learn a new command language or operating system.

A.7 PROCESSOR - KA10

The KA10 arithmetic processor is the processing unit for the three smallest DECsystem-10 machines. Its standard I/O devices are:

1. a 300 character-per-second photoelectric paper-tape reader
2. a 50 character-per-second paper-tape punch
3. an operator's console that provides the operator with information and intervention capabilities when desired, and
4. a standard Model 35KSR console teletypewriter operating at 10 characters-per-second (considered as part of the operator's console).

The 36-bit instruction word format of the KA10 provides 512 operation codes, of which 366 are hardwired. The remainder are programmed operators UUOs or are reserved for future use.

The fast registers, KM10, are sixteen 36-bit integrated circuit registers, used as multiple accumulators, index registers, or memory locations. These registers have an access time of 200 nanoseconds and when used as memory locations can double the execution speed of a program. The dual memory protection and relocation registers, KT10A, allow the software to define two areas for each user and to protect the remaining core from these users.

The priority interrupt system of the central processor has seven levels of interrupts for the devices attached to the I/O bus.The entire priority interrupt system is programmable. With software, any number of devices can be attached to any level. Individual levels or the entire priority interrupt system can be deactivated and later reactivated, and interrupts can be requested on any level. With the executive control logic, the KA10 operates in one of three modes:

1. executive mode, which allows all instructions to be executed and suppresses relocation;
2. user mode, in which some instructions are not allowed (i.e., I/O instructions) and relocation and protection are in effect; and
3. user I/O mode, where all instructions are valid but relocation and protection are still in effect.

A.8 PROCESSOR - KI10

The KI10 central processor used with the larger DECSystem-10 machines is almost twice as fast as the KA10 processor. This increase in speed results from the use of different architecture, faster circuits, a more complex adder, improved algorithms, and lookahead instruction logic, which obtains the next instruction during the execution of the current instruction.

Core memory is managed by the paging system of the KI10. This system allows a user program to access an effective address space of up to 256K words. This space is segmented into pages of 512(10) contiguous words each. These pages do not have to be contiguous in the physical core memory.

The KI10 processor provides memory address mapping from a program's effective address space to the physical address space by substitution of the most significant bits of the effective address. This mapping provides access to the entire physical memory space. Memory mapping takes place using a page table as follows: the most significant nine bits of the effective address, the page number, is used as an index into the appropriate page table. The effective page number is then replaced by the information located in the page table entry. This information is a physical page number of 13 bits. These 13 bits are concatenated with the least significant 9 bits of the effective address, the word address within the page, in order to form the 22-bit physical address.

The KI10 processor can be supplied with the VMSER option; a minimum of 128K is recommended for the VMSER option. VMSER allows for virtual memory utilizing a temporary storage device, in the form of a disk or drum.Virtual memory permits a user program to execute with an address space greater than the physical memory actually allocated to that program. When a page is not present in core memory, it is kept on a secondary (temporary) storage device. A page fault handler swaps pages between core and virtual memory whenever a program requires access to a page not in core.

When executing a user program under a virtual memory system, the entire address space referenced by the program need not be in core. The page fault handler determines which pages (and how many pages) to place in core while a program is executing.

The use of the virtual memory capability is a privilege that the system administrator can limit to specific users. The administrator can also establish, for each user of the system, a limit on the amount of physical core and virtual memory utilized during execution.

The virtual memory facility monitors both the paging rates for each virtual memory user and a system-wide rate for all virtual memory users. Whenever these paging rates exceed the values established by the system administrator, the priorities of the appropriate virtual memory jobs are lowered until the rates return to acceptable values.

Eight instructions for double-precision floating-point arithmetic and three instructions for converting between fixed-point and floating-point formats are in the KI10 instructionrepertoire. The double-precision word format gives precision of 1 part in $4.6 \times 10_{18}$ and an exponent to the power of 256.

The KI10 processor provides measures for handling arithmetic overflow and underflow conditions, push-down list overflow conditions, and page failure conditions directly by the execution of programmed trap instructions. The trap instruction is executed in the same address space as the instruction that caused the trap. Therefore, user programs can handle their own traps by directing the monitor to place a jump to a user routine in the trap location. User programs may also initiate the Software Interrupt System which will transfer control to a user interrupt servicing routine whenever an interrupt condition is detected.

The maximum uninterruptable interval on the priority interrupt system is 10 microseconds. The I/O bus cycle time of the KI10 processor is 2.7 microseconds. Interrupt response is enhanced by the four blocks of general-purpose registers. Each block contains 16 registers. The existence of several blocks of registers facilitates both rapid context switching between programs and interrupt handling.

The KI10 operates in one of two modes, user mode and executive mode. Each of these modes has two submodes. User mode is subdivided into public mode and concealed mode, and executive into supervisor mode and kernel mode.

User programs operate in user mode. In this mode, a program can access up to 256K words. All instructions are legal except those that interfere with other users or the integrity of the system. A program in public mode can transfer to a program in concealed mode only by transferring to locations that have PORTAL instructions. A program in concealed mode can read, write (if allowed), execute, and transfer to any location designated as public. Concealed mode allows the loading of propriety software with a user program and data, but prevents the user program from changing or copying the software. This provides direct interaction between the user and the proprietary software with virtually no overhead.

The supervisor and kernal submodes are similar but not identical to the public and concealed modes. Supervisor mode programs can access but cannot alter areas designated as concealed. In kernal mode all of memory is accessible and can be altered. Programs operating in kernal mode can address portions of memory directly, without paging. The operating system executes in kernal mode.

The DECsystem-10 provides a facility for setting address breaks. Occassionally, programs exhibit behavior for which the normal debugging aids (DDT, FORDDT, COBDDT, etc.) are not appropriate. In such cases, it is valuable to have hardware assistance in debugging. In the KI10, this assistance takes the form of providing the user with the ability to specify an address break and break conditions. The computer will check the location being referenced and the type of reference (READ, WRITE or EXECUTE) for each memory access. For example, in a program where a word of data is being altered inexplicably, a break on WRITE is particulary helpful in debugging, without appreciably degrading the execution speed of the program.

A.9 CORE MEMORIES

The MF10 core memory contains 32,768 words with a read access time of 600 nanoseconds and a full cycle time of 950 nanoseconds. Up to 16 memory modules can be connected to provide 1,048,576 words of core storage (on a KI10). Each module can contain up to four ports. This memory features both two- and four-way interleaving with switches on each memory module. It is specifically built for the KI10 processor in that it can recognize the 22-bit address space. It also takes advantage of the overlap memory control of the KI10, which results in a 20% increase in speed.

The MD10 mass memory system consists of 64K MD10 core memory and an MD10E including cables. The basic unit of the MD10 memory has 65,536 words of storage at 36 bits per word. The unit has an access time of 800 nanoseconds, a cycle time of 1.8 microseconds, and two- or four-way interleaving between cabinets. This memory is equipped with four access ports for connection to the processor and data channels. The MD10E core memory expansion module expands the MD10 up to 128K words in increments of 32K words. Thus, a user has the option of implementing his system with 64, 96, or 128K blocks up to a maximum directly addressable memory capacity of 256K words.

A.10 FIXED-HEAD DISK SYSTEM

The RH10 is a high-speed, fixed-head disk system. Control for this swapping disk is provided by a controller which is interfaced to both the I/O bus and to a DF10 data channel. The DF10 interfaces directly to one port in each of the DECsystem-10 memory modules. The data channel also allows data transfers between the RH10 system and core memory to take place simultaneously with central processor computation, provided that the channel and processor are not accessing the same memory module. A single controller can handle up to eight drives providing a total swapping capacity of two million 36-bit words. The RH10 has zero positioning time, an average latency time of 8.5 milliseconds and an average transfer rate of 4 microseconds per 36-bit word. With these capabilities, the RH10 system can swap a typical 4K user job in or out of core memory in as little as 16.7 milliseconds.

A.11 DISK SYSTEMS

The RP02 disk drive provides storage for up to 5,120,000 36-bit words on interchangeable disk packs. The average access time is 47.5 milliseconds, which includes 12.5 milliseconds average rotational latency, and the transfer rate is 15 microseconds per word.

The RP03 has a total of 400 cylinders per disk pack that give twice the storage capacity of the RP02. The average access time is 41.5 milliseconds including 12.5 milliseconds average rotational latency, and the transfer rate is identical to the RP02.

The RP10C disk drive controller operates with any combination of the two types of disk drives, the RP02 and the RP03. The RP10C provides control for as many as eight disk drives: RP02's, RP03's, or a combination of RP02's and RP03's. Through the DF10 data channel, both drives transfer data directly to and from memory, allowing simultaneous I/O and computational operations.

The maximum disk system storage capacity is: up to four controllers, each with eight drives, giving a total of 327,680,000 words, or in excess of 1.9 billion characters of on-line storage.

A.12 MAGNETIC TAPE SYSTEMS

The TD10G DECtape system consists of a TD10 DECtape controller and a TU56 DECtape transport. The TD10 controls either a maximum of four TU56 dual DECtape transports or up to eight TU55 DECtape transports. Data is transferred between the TD10 and the central processor over the I/O bus at the average rate of one 36-bit word every 400 microseconds. The TU56 transport reads and writes magnetic tape at 925 36-bit words per second. The tapes are 3.9 inches in diameter, 260 feet long, and ¾ inch wide. Each tape has a directory providing random access to user files. The tape units are bi-directional and redundantly recorded, resulting in greater maintainability and reliability.

The TU40 120KC magnetic tape system includes a DF10 channel, a TM10B magnetic tape control, and eight TU40 magentic tape units. The DF10 controls the transfer of data between eight device controllers and one port of core memory. The TM10B controls up to eight tape transports. This control uses the I/O bus to receive information from and to provide status to the processor. It establishes a data path from the tape transport to core memory via the DF10. The transfer rate of the control is determined by the speed and density of the tape transport performing the transfer. The TU40 reads and writes 9-channel USASI standard magnetic tape at 150 inches per second and a density of 200, 556, and 800 bits per inch. The TU41 reads and writes 7-channel industry-standard tape at 150 inches per second and a density of 200, 556, and 800 bits per inch.

The TU10A magnetic tape system has a transfer rate of 36K characters per second at 800 bits per inch. The TU10A may be combined with other DEC drives on the same TU10B controller. Up to eight TU10A transports may be interfaced through the TM10 control unit. The TU10A is capable of handling both 7 and 9 channel tape at recording densities of 200, 556, and 800 bits per inch.

A.13 INPUT/OUTPUT DEVICES

A.13.1 Card Readers

The card readers offered with the DECsystem-10 have high tolerance to damaged cards, are virtually jamproof and minimize card wear. The life of an individual card has been proven to be in excess of 1000 passes. These readers are designed to permit the operator to load and unload cards while the reader is operating. The CR10 card readers accept 80 column EIA/ANSI standard cards.

The CR10-D card reader is a table-top model that reads at the rate of 1000 cards per minute. The capacity of the card hopper is 1000 cards. The card reader controller connects to the BA10 hardcopy controller.

The CR10-E console model card reader reads at 1200 cards per minute. The maximum number of cards held by the input and output hoppers is 2300 cards each. The controller is mounted in the BA10 hardcopy controller cabinet.

The CR10-F card reader is a table-top model and reads at a rate of 300 cards per minute. The maximum capacities of the input and output hoppers are 600 cards each. The controller connects to the BA10 hardcopy controller.

A.13.2 Card Punch

The CP10A card punch punches cards at the rate of either 200 cards per minute when punching all 80 columns or 365 cards per minute when punching only the first 16 columns. The card hopper and stacker capacities are 1000 cards each. The card punch controller is mounted in the BA10 hardcopy controller cabinet.

A.13.3 Line Printers

The 64-character LP10-F line printer prints 1250 lines per minute and 132 columns per line. The printable character set is composed of upper-case alphabetic characters, digits, and special characters. The line printer is connected to the I/O bus via a controller mounted in the BA10 hardcopy controller.

The 96-character LP10-H line printer prints 925 lines per minute and 132 columns per line. The printable character set includes that provided on the LP10-F as well as lower-case alphabetic characters and additional special characters. The line printer is connected to the I/O bus with the BA10 hardcopy controller.

The 64-character LSP10 line printer prints 245 lines per minute and 132 columns per line. The interface logic allows direct connection of the LS10 printer to the I/O bus, thus performing essentially the same functions as does the BA10 for the LP10 series line printers.

A.13.4 PLOTTERS

The XY plotter control is the interface for CalComp 500 and 600 series digital incremental plotters. It is normally mounted in the BA10 hardcopy controller, but can be mounted in the TD10 DECtape controller cabinet.

A.13.4.1 XY10A CalComp Plotter Model 565 - The XY10A plotter is interfaced to the I/O bus via a controller mounted in the BA10. This plotter has the following specifications:

Step Size	Steps Minute	Width of Paper
0.01 in.	18,000	12 in.
0.005 in.	18,000	12 in.
0.1 mm.	18,000	12 in.

A.13.4.2 XY10B CalComp Plotter Model 563 - The XY10B plotter is interfaced to the I/O bus via a controller mounted in the BA10. The plotter has the following specifications:

Step Size	Steps Minute	Width of Paper
0.01 in.	12,000	31 in.
0.005 in.	18,000	31 in.
0.1 mm.	18,000	31 in.

A.13.5 BA10 Hardcopy Control

The BA10 control cabinet contains the controllers for the card readers, card punch, line printers, and plotters. It has the power supplies and fans necessary to support the controllers.

A.14 TELETYPEWRITERS AND TERMINALS

The teletypewriters and terminals used on the DC10 and the DC76 are similar except for different cables and interface connectors.

A.14.1 Hardcopy Terminals

The LA30 DECwriter is a 30 character per second teletypewriter. The unit provides both a hardcopy original and one copy on a standard, 9-7/8 inch wide, tractor-driven, continuous form.

The LT33A and LT33C teletypewriters are the 33TS machines (KSR33, friction feed). They are used with the DC10 and DC76 respectively.

The LT33B and LT33H teletypewriters are the 33TY machines (ASR33, sprocket feed, automatic reader control XON/XOFF feature). They are used with the DC10 and DC76 respectively.

The LT35A and LT35C teletypewriters are the USL312HF machines (KSR35, sprocket feed). They are used with the DC10 and DC76 respectively.

A.14.2 CRT Display Terminals

The VT05 alphanumeric terminal is a CRT display terminal capable of full- and half-duplex transmission at rates up to 2400 baud. The VT05 can be interfaced directly with all widely used computer systems as well as serve as a remote terminal. Alphanumerics can be superimposed over a video image derived from closed circuit TV or video tape. In addition, the VT05 features direct cursor addressing. This provides the user with the capability to place the cursor at any position on the screen.

The GT40 is a computer level graphics display system consisting of a graphical display system and a general-purpose mini-computer. The GT40 can display information as straight lines, vectors, single random position points or characters. The characters available consist of the 96 ASCII characters and 31 special characters which include some Greek letters, and architectural and mathematical symbols. Seven programming modes are available including character mode, long vector, short vector, point mode, relative point mode and X and Y graph-plot mode. Hardware drawn vectors employ a constant velocity technique and have four line types. A solid-state light pen is provided to facilitate interaction with the system. In addition, when not engaged in graphics tasks, the GT40 can serve as a general-purpose computer which can operate as a standalone system or initiate dialogue with a central computer as part of a computer network.

A.15 DATA COMMUNICATION SYSTEMS

The data communication equipment includes two systems for asynchronous communications (hardwired and programmable), two systems for synchronous communications (low capacity and high capacity) and a remote batch terminal.

A.15.1 Asynchronous Communications

The DC10 hardwired data line scanner interfaces asynchronous communications lines to the processor via the I/O bus. The DC10A control unit is thebasic unit and contains the I/O interface and control logic. This unit provides on-line servicing of up to 64 local communication lines. It accommodates any device that uses eight-level or five-level serial teletype code. Standard system software supports interactive ASCII terminals at speeds up to 2400 baud. For some special communication applications, the DC10 can operate at higher speeds. Full duplex with local copy and half-duplex data modes are available on each line serviced.

The DC10B is an eight-line group unit connected to the DC10A and provides an interface for up to eight local lines. It can be used in full-duplex or half-duplex mode with local copy operation. To provide carrier detection or data set status control, the DC10E is required.

The DC10C eight-line telegraph relay assembly provides an interface to long distance telegraph lines using full- or half-duplex facilities.

The DC10D telegraph power supply is the standard line voltage supply used with DC10C (120 Vdc at 2A).

The DC10E data set control provides expanded processor control of eight data sets in the DC10 system.

The DC76 Asynchronous Communications Interface consists of from one to four PDP-11 processors connected to the DECsystem-10 via a D10 data link. Each PDP-11 is equipped with from one to eight DH11 asynchronous multiplexers. Up to 16 asynchronous terminals can be connected to each DH11 so that each of the PDP-11's can support up to 128 terminals including IBM 2741-type terminals. The maximum number of lines that can be connected to a single DC76 system is 512. Full modem control and automatic baud-rate detection are provided.

The DC44 TYPESET interface is similar to the DC76 Communication System. The DC44 provides interface facilities to wire-service lines, PR68E paper-tape readers, PP67 paper-tape punches and on-line photo composition devices. It consists of a PDP-11 processor connected to the DECSystem-10 via a DL10 data link and other equipment dependent on the usage of the DC44. Wire-service lines are connected to a DH11 asynchronous multiplexer. In addition the DC44 can handle up to eight PA611R paper-tape reader controllers (each controller can accommodate two reader units), up to eight PA611P paper-tape punch controllers (each of which can accommodate two punch units), and up to six LPC-11 TYPESET interfaces. Each DC44 is also equipped with a controlling terminal.

A.15.2 Synchronous Communications

The DS10 synchronous line unit is an interface between the DECsystem-10 I/O bus and one full- or half-duplex voice grade synchronous modem to a remote batch terminal, high-speed display, remote job entry station, or another computer. The synchronous modem meets EIA RS-232B or C standards, such as the Bell System 201B, System software supports full-duplex operation of a DS10 at up to 9600 baud, or two DS10s at up to 4800 baud each.

The DC75 synchronous communications system is a PDP-11 based front-end system designed to efficiently handle multiple synchronous lines. The basic DC75 system includes a DL10 interface, one PDP-11/20, and a DS11 synchronous modem interface implemented for eight lines.

The DL10 is a direct memory interface between the DECsystem-10 memory and the PDP-11 communications processor. Each DL10 can connect up to four PDP-11s.

A basic DC75 system can handle eight full-duplex lines at speeds up to 4800 baud each, or four lines at 9600 baud. It can be expanded to handle 16 lines at 2400 baud by implementing additional DS11 line capability.

For applications requiring additional line capability at 4800 baud or 9600 baud, up to three additional PDP-11/DS11combinations can be added to the DL10 interface unit. Each additional PDP-11/DS11 combination provides a line throughput capability equal to the initial system.

For special applications, the DC75 can be programmed to handle a mix of line speeds, character sizes, and message formats. The DS11 modem interface hardware has provision for 6-, 8-, or 12-bit character sizes, and these characters can be efficiently packed into DECsystem-10 memory by the DL10.

A.15.3 DC72 Remote Station

A DC72 remote station includes a PDP-8e computer with at least 8192 12-bit words of core memory, a ROM, a line frequency clock, and a 300 card-per-minute card reader. Optionally, the Remote Station will be designated as either a DC72A, B, or C. The DC72A also includes an LS8e light-duty 64-character line printer capable of printing up to 165 characters per second. The DC72B includes an LE-8 64-character line printer capable of printing up to 245 lines per minute. The DC72C includes an LE-8 96-character line printer capable of printing up to 173 lines per minute. The DC72C inculdes an LE-8 96 character line printercapable of printing up to 173 lines per minute. The DC72L is used to provide interfacing capabilities for up to eight teletype-compatibleterminals. Up to two DC72L's may be installed on each DC72.

INDEX

READER'S COMMENTS

DECsystem-10 INTRODUCTION TO SOFTWARE DEC-10-MZDC-D

NOTE: **This form is for document comments only. Problems with software should be reported on a Software Problem Report (SPR) form**

Did you find errors in this manual? If so, specify by page.

_____

_____

_____

_____

Did you find this manual understandable, usable, and well-organized?
Please make suggestions for improvement.

_____

_____

_____

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

If you do not require a written reply, please check here. ☐

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE PUBLICATIONS
200 FOREST STREET MR1-2/E37
MARLBOROUGH, MASSACHUSETTS 01752

Do Not Tear - Fold Here and Tape

Cut Along Dotted Line