

digital

# decsystem10

## users handbook

second edition

introduction

software

timesharing

beginners batch

teco (intro.)

reference

teco

lined

pip

commands

decsystem10 handbook series

# contents

## introduction

|  |     |
|--|-----|
| 1) Introduction to DECsystem-10 Software ..... | 1   |
| 2) Getting Started with Timesharing .....      | 53  |
| 3) Beginners Guide to Multiprogram Batch ..... | 87  |
| 4) Introduction to TECO .....                  | 187 |

## reference

|  |     |
|--|-----|
| 5) TECO, Text Editor and Corrector Program ..... | 221 |
| 6) LINED, Line Editor for Disk Files .....       | 355 |
| 7) PIP, Peripheral Interchange Program .....     | 367 |
| 8) DECsystem-10 Operating System Commands .....  | 429 |

**digital**

# **dec**system10

## **users handbook**

**second edition**

Additional copies of this handbook may be ordered from:  
Program Library, DEC, Maynard, Mass. 01754. Order code: DEC-10-NGZB-D.

**dec**system10 **handbook series**

First Printing November 1971  
Second Printing (Rev.) July 1972

The material in this handbook is for information purposes and is subject to change without notice.

Copyright © 1969, 1970, 1971, 1972 by  
Digital Equipment Corporation

The following are trademarks of Digital Equipment Corporation,  
Maynard, Massachusetts

DEC  
FLIP CHIP  
DIGITAL

PDP  
FOCAL  
COMPUTER LAB

# tab index

introduction

software 

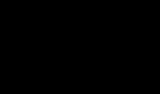
timesharing 

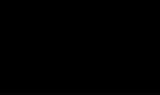
beginners batch 

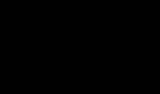
teco (intro.) 

reference

teco 

lined 

pip 

commands 

index 



## NOTICE

For the reader's convenience:

- 1) Consecutive page numbers have been added to the top center of each page in the handbook; these numbers have the form —nn.— (for example —25— ) and are supplied in addition to the standard document numbers printed at the bottom center of each page.
- 2) The appropriate document name has been added to the top outside corner of each page of the handbook.
- 3) A global index comprised of the merged and alphabetized entries of all of the indexes which were previously part of the documents contained by the handbook is supplied at the end of the handbook. The global index replaces the individual document indexes.
- 4) The entries of the global index and the Table of Contents for each document reference the *consecutive* page numbers located at the top center of each page.
- 5) Black locator tabs are printed on the outside edge of the first ten pages of each document in the handbook. A tab locator page on which each set of tabs is identified by the name of the document which they represent is supplied at the front of the handbook.



## FOREWORD

This handbook is an introduction to the DECsystem-10. It is intended to be a guide to using the system and, as such, should be read before advancing to more detailed documentation. The collection of documents in this handbook is taken from the DECsystem-10 SOFTWARE NOTEBOOKS (DEC-10-SYZB-D) and in all cases, the documents are reprinted without change.

The documents in this handbook reflect the following versions of the software:

Monitor 5.05  
TECO version 23  
LINED version 13A  
PIP version 32

Support program version numbers are specified on page 431 of this handbook.

The DECsystem-10 User's Handbook is one in the set of DECsystem-10 handbooks. The other handbooks comprising this series are:

- (1) The COBOL Language Handbook and its supplement,
- (2) The Mathematical Languages Handbook, which includes FORTRAN, BASIC, and ALGOL,
- (3) The Assembly Language Handbook, which includes the System Reference Manual, MACRO, DECsystem-10 Monitor Calls, LOADER, DDT, CREF, FILCOM, FUDGE2, and GLOB.

These handbooks may also be ordered from the Program Library, Digital Equipment Corporation.



introduction



# 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.

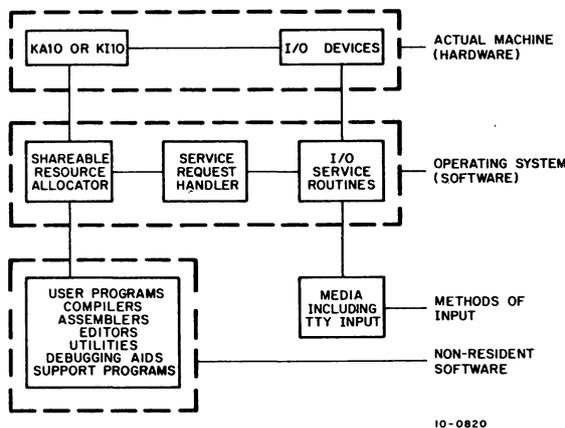


Figure 1-1 DECsystem-10 Components

## 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 five 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. 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 for 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 pre-allocate 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 exchange 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.

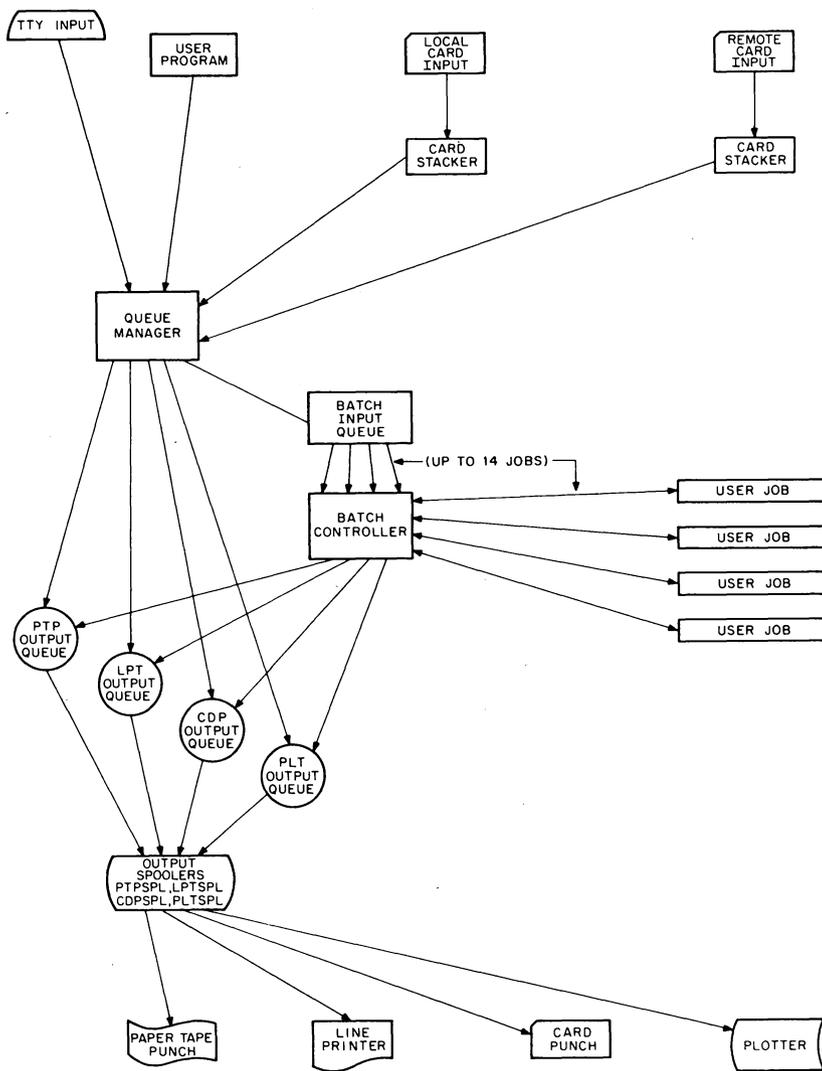
**1.5.1.6 General-Purpose Timesharing** — Timesharing on the DECsystem-10 is general purpose; i.e., the system is designed in such a way that the command language, input/output processing, file structures, and job scheduling are independent of the programming language being used. In addition, standard software interfaces make it easy for the user to develop his own special languages or systems. This 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 Stacker, CDRSTK; the batch controller, BATCON; the queue manager, QMANGR; and the output spoolers, LPTSPL, CDPSP, PTPSPL, and PLTSPL (see Figure 1-2). The *stacker* is responsible for reading the input from the input device and for entering the job into the batch controller's input queue. Although



10-0819

Figure 1-2 Programs in the Batch System

the Stacker is oriented toward card reader input, it allows jobs to be entered from any input device that supports ASCII code. 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 Stacker 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.

After the Stacker 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 Stacker 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 Stacker normally reads from the card reader, but can read from magnetic tape, DECtape, or disk packs 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 bypasses the Stacker 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 this 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 a stacker 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 Stacker. 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 stacker 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 applications, 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 to 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 then links the devices to the interrupt 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** — The DECsystem-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 system's core management facilities provides optimum core usage.

#### 1.5.4 Remote Communications

Until the capability of remote communications was implemented, each remote user of the PDP-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, which 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, they 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 an assembler or a compiler in that a binary version of the program is not produced for storage. In other words, the source text is translated to machine language everytime it is used, allowing for extensive checking of errors during execution.

### 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 programs 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 adaptation 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 commands that are required for solving almost any problem. The BASIC language is divided into two sections: one section of elementary commands that the user must know in order to write simple programs and the second section of advanced techniques for efficient and well-organized 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. When the statements are entered, a run-type command initiates the execution of the program and returns the results almost instantaneously.

The BASIC system has several special features built into its design. For one, 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 renaming the file, or by resequencing the line numbers. The user can combine two files into one and request a listing of all or part of the file on either the line printer or the terminal. Secondly, BASIC allows various peripheral devices to be used for storage or retrieval of programs and data files. The user can input programs or data files from the paper-tape reader on the terminal or output them to the terminal's

paper-tape punch. Also, the data file capability allows a program to read information from or write information to the disk. Thirdly, output to the terminal can be formatted by including tabs, spaces, and columnar headings. Finally, BASIC has an expanded command set that includes commands designed exclusively for matrix computations. Elementary mathematical functions are contained in the command set 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, which when compiled and loaded, 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 the most widely used procedure-oriented programming language. It is designed for solving scientific-type problems and thus is composed of mathematical-like statements constructed in accordance with precisely formulated rules. Therefore, 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 in which scientific problems can be expressed. In addition, FORTRAN is used as the primary data processing language by time-sharing utilities.

Because of this wide market, DECsystem-10 FORTRAN is designed to meet the needs of all users. The FORTRAN system is easy to use in either the time-sharing or batch processing environments. Under time-sharing, 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.

FORTTRAN 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 INTERPRETER

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.

### 2.4 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 the file as he desires, 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; specification; 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.

Commands to TECO 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. One or more characters can be inserted into the editing buffer, deleted from the buffer, searched for, and or typed out with commands from the user at his terminal. 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 preparing 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 TECO or LINED and inputs his material through his terminal. In addition to inputting 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 unchanged text. After the group of modifications have been added to the file, RUNOFF produces a new copy of the file which is properly pagged 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 listing. 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 File Backup

The file backup system enables the user to recover from a system failure or other unintentional destruction of data on the disk by 1) preserving disk files on a storage medium and 2) later retrieving these files and placing them back onto the disk. Two system programs are involved in this storage and retrieval system: the BACKUP program used to save the disk files on the specified storage device, and the RESTORE program used to return these files to the disk. Using the BACKUP program, the user can save individual disk files or the entire disk on magnetic tape, DECTape, or disk. When restoring these saved files to the disk with the RESTORE program, the user can return the entire contents of the storage device to the disk or return only selected portions.

### 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, 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 LOADER

The LOADER provides automatic loading and relocation of binary programs generated by the standard DEC compilers and assemblers, produces an optional storage map, and performs loading and library searching regardless of the input medium. In addition, this program loads and links relocatable binary programs generated by the compilers and assemblers prior to execution and generates a symbol table in core for execution with DDT.

The user specifies in the LOADER command string the device from which the relocatable binary programs are to be loaded and the device on which any storage maps or undefined globals are written. Switches can be included in the command string 1) to specify the types of symbols to be loaded or listed, 2) to indicate that the run time libraries are to be searched for symbol definitions, 3) to load the DDT program, and 4) to clear and restart the LOADER. In addition, special switches allow the user to create CHAIN files—a feature used to segment FORTRAN programs that are too large to be loaded into core as one unit. These CHAIN files consist of complete programs and subroutines that can be read into core and executed as needed.

When the loading process is completed, the loaded program can be written onto an output device with a monitor SAVE command so that it can be executed at a later time without rerunning the LOADER.

### 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.

## 2.6 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-TTYs, a simulated terminal not defined by hardware. All initializations of the pseudo-TTYs 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 determines by appropriate dialogue with the user who he is, whether or not he is currently authorized to use the system, and if so, establishes the user's initial profile, informs him of any messages of the day, and reports any errors detected in his disk files.

### 2.6.4 KJOB-LOGOUT

The system programs KJOB and LOGOUT are used when leaving the DECsystem-10. Their many functions 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.



## CHAPTER 3 THE RESIDENT OPERATING SYSTEM

The resident operating system is made up of a number of separate and somewhat independent parts, or routines (see 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 interrupted 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.

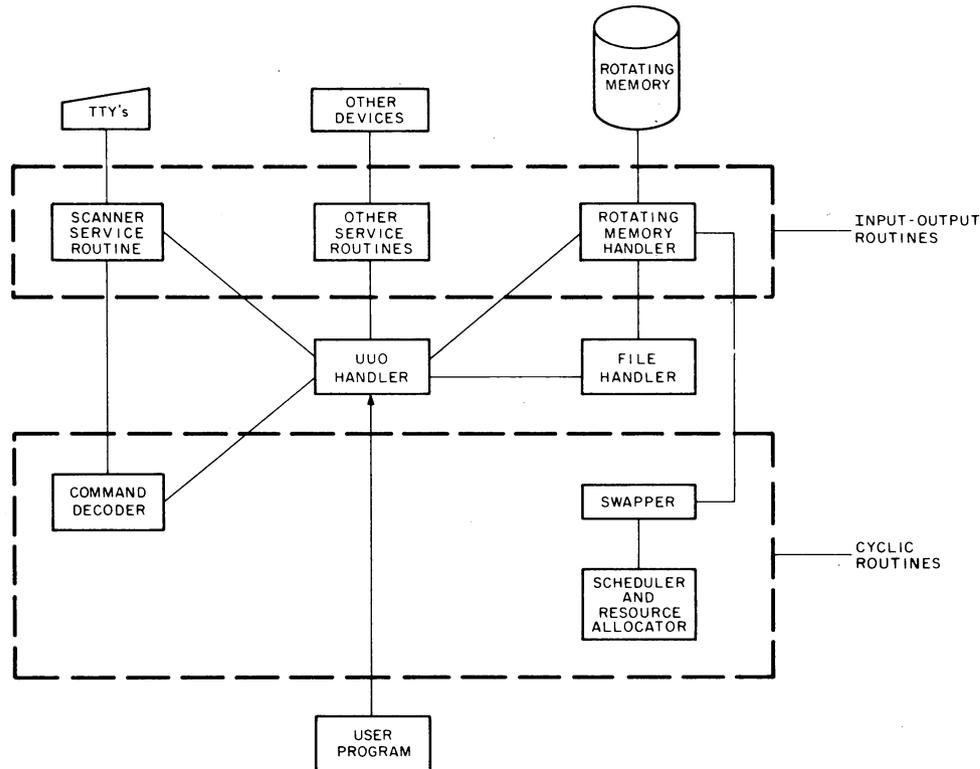
The *UUO handler* is the means by which the user program communicates with the operating system in order to have a service performed. Communication is by way of *programmed operators* (also known as *UUOs*) contained in the user program which, when executed, go

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 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.



10-0821

Figure 3-1 The Resident Operating System

### 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.

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 a 1/2 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 operations.

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 to 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 later 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 waits until the next clock tick.

### 3.4 THE UO HANDLER

The UO handler is responsible for accepting requests for services available through the operating system. These requests are made by the user program via software-implemented instructions known as *programmed operators*, or *UOs*. 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 UUU 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 UUU handler for processing. This routine obtains its arguments from the user program. The core location at which the UUU operation was executed is then remembered. After the UUU request has been processed, control is returned to the user program at the first or second instruction following the UUU. 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 UUU's contained in his program. The actual input/output routines needed are then called by the UUU 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 in 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 command list has been processed. Therefore, during this 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 independent 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 addresses; there are no pointers to areas in other file structures, thereby allowing the file structure to be removed.

A file structure consists of 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 all the names of 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 all the names of files within the 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 batch 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 via a MONGEN question

All disk files are composed of two parts: data and information used to retrieve the 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 leaving the system. Quotas are determined by the individual installations and are, therefore, used to ration disk resources in a predetermined manner.

To a user, a file structure is like a device; i.e., a file structure name or a set of file structure names can be used as the device name in command strings or UUC calls to the operating system. Although 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 reaccess it later. 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.

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 pre-allocated 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.

## CHAPTER 4 GLOSSARY

### **Absolute address**

The address that is permanently assigned to a storage location by the machine designer.

### **Access date**

The date on which a file on disk was last read. If a file has not been read since it was created, the creation date and the access date are the same. The access date is kept in the retrieval information block for the file.

### **Access list**

The table in monitor core that reflects the status of all files open for reading or writing in addition to the status of those files recently closed.

### **Access privileges**

Attributes of a file which specify the class of users allowed to access the file and the type of access which they are allowed.

### **Access time**

The interval between the instant at which data is requested from a storage device or data is requested for a storage device and the instant at which delivery or storage is begun.

### **ACCT. SYS**

The file that contains all project-programmer numbers, passwords, initial profiles, and time of day users are allowed on the system. It does not contain file structure quotas.

### **Accumulator**

The register and associated equipment in the arithmetic unit of the computer in which arithmetical and logical operations are performed.

### **Active search list**

An ordered list of file structures for each job which specifies the order in which the directory is searched. These file structures are the ones listed before the FENCE by

the SETSRC program. Device DSK is defined by this list for each job.

### **Actual transfer**

The hardware operation whereby the channel actually passes data between the memory system and the control. The third step of the transfer operation (verification, search, actual transfer).

### **Address**

- (1) An identification represented by a name, label, or number for a register, a location in storage, or any other data source or destination.
- (2) The part of an instruction that specifies the location of an operand of the instruction.

### **ALCFIL**

A program used for allocating space for a new file or reallocating space for an existing file in one contiguous region on the disk.

### **ALGOTS**

The ALGOL object time system.

### **All CPU job**

A job which the monitor can run on either processor in a dual-processor system depending on the I/O activity and the system load.

### **Arithmetic unit**

The portion of the hardware in which arithmetic and logical operations are performed.

### **Assemble**

To prepare a machine-language program from a symbolic-language program by substituting absolute operation codes for symbolic operation codes and absolute or relocatable addresses for symbolic addresses.

**Assembler**

A program which accepts symbolic code and translates it into machine instruction, item by item.

**Assigning a device**

To allocate an I/O device to the user's job either for the duration of the job or until the user relinquishes it.

**Asynchronous**

- (1) Pertaining to the procedure by which the hardware does not wait for one operation to be completed before starting a second operation.
- (2) Pertaining to the method of data transmission in which each character is sent with its own synchronizing information.

**AUXACC.SYS**

The file that contains the standard list of public file structures for each user and information (such as quotas) for those file structures.

**Bad Allocation Table (BAT) block**

A block written by the MAP program or the monitor on every disk unit. This block enumerates the bad regions of consecutive bad blocks on that unit so that they are not reused. The BAT blocks appear in the HOME.SYS file.

**BADBLK.SYS**

The file that contains all bad blocks. It may be read but not deleted and is useful for testing error recovery.

**Base address**

A given address from which an absolute address is derived by combination with a relative address.

**Batch processing**

The technique of executing a set of computer programs in an unattended mode.

**BATCON**

The Batch controller. This program reads a job's control file, starts the job, and controls the job by passing commands and data to it.

**Block**

A  $2^8$ -word unit of disk storage determined by hardware and software; 128 words are always written, adding zeros as necessary, although less than 128 words can be read.

**BOOTS**

A bootstrap program whose main functions are to load a program into core from a disk SAVE file and to dump core as a SAVE file for later analysis.

**Bootstrap**

A technique or device designed to bring itself into a desired state by means of its own action, e.g., a machine routine whose first instructions are sufficient to bring the rest of itself into the computer from an input device.

**Breakpoint**

A location at which program operation is suspended in order to examine partial results.

**Buffer**

A device or area used temporarily to hold information being transmitted between external and internal storage devices or I/O devices and internal high-speed storage. A buffer is often a special register or a designated area of internal storage.

**Buffer pointer**

A movable position indicator that is positioned between two characters in an editing buffer, before the first character in the buffer, or after the last character in the buffer.

**Byte**

Any contiguous set of bits within a word.

**Calling sequence**

A specified arrangement of instructions and data necessary to pass parameters and control to and from a given subroutine.

**CDRSTK**

The Batch input stacker. CDRSTK reads any sequential input stream, sets up the job's control file and data files, and enters the job into the Batch input queue.

**Central processing unit (CPU)**

The portion of the computer that contains the arithmetic, logical, control circuits, and I/O interface of the basic system.

**Central site**

The location of the central computer. Used in conjunction with remote communications to mean the location of the DECsystem-10 central processor.

**CHAIN**

A program that allows the user to segment FORTRAN programs that are too large to load or fit into available core. It reads successive segments of coding into core and links them to the program already in core.

**Channel**

- (1) A path along which signals can be sent; e.g., data channel, output channel.
- (2) A partially autonomous portion of the PDP-10 which can overlap I/O transmission while computations proceed simultaneously.

**CHECKPOINT**

A program used to maintain the accounting information on the disk.

**Clear**

To erase the contents of a location by replacing the contents with blanks or zeros.

**Cluster**

A single-or multi-block unit of disk storage assignment. It is a parameter of each disk file structure.

**CODE**

A code conversion program that translates files written in binary-coded decimal to ASCII and vice versa.

**COMPIL**

A utility program that allows the user to type a short, concise command string in order to cause a series of operations to be performed. COMPIL deciphers the command and constructs new command strings for the system program that actually processes the command.

**Compressed file pointer**

An 18-bit pointer to the unit within the file structure and to the first super-cluster of the file.

**Concatenation**

The joining of two strings of characters to produce a longer string, often used to create symbols in macro defining.

**Conditional jump**

A jump that occurs if specified criteria are met.

**Context switching**

The saving of sufficient hardware and software information of a process so that it may be continued at a later time, and the restoring of another process.

**Continued directory**

The collection of all directories with a particular name and path on all file structures in the job's search list.

**Continued MFD**

The MFDs on all file structures in the job's search list.

**Continued SFD**

The SFDs on all file structures in the job's search list which have the same name and path.

**Continued UFD**

The UFDs for the same project-programmer number on all file structures in the job's search list.

**Control**

The device which controls the operation of connected units. It can initiate simultaneous positioning commands to some of its units and then perform a data transfer for one of its units.

**Control character**

A character with an ASCII representation of 0-37. It is typed by holding down the CTRL key on the terminal while striking a character key. It can be punched on a card via the multi-punch key.

**Copy**

To transfer a file from one device to another (e.g., with PIP or the FILEX program).

**CORMAX**

The largest contiguous size that an unlocked job can be. This value can range from CORMIN to total user core.

**CORMIN**

The guaranteed amount of contiguous core which a single unlocked job can have. This value can range from 0 to total user core.

**Counter**

A device such as a register or storage location used to represent the number of occurrences of a certain event.

**CPU**

See central processing unit.

**CPU0**

In a dual-processor system, the processor that performs the same activities as the processor in a single processor system, including all I/O operations, command and UUC processing, swapping, and interrupt handling. Also known as the primary processor.

**CPU1**

In a dual-processor system, the processor that operates only in user mode except when it is required to find another job to run or to send APR traps to the user. Also known as the secondary processor.

**CRASH.SAV**

A file written on disk by BOOTS as part of the crash restart procedure. This file is used by FILDDT for system debugging.

**Create**

To open, write, and close a file for the first time. Only one user at a time can create a file with a given name and extension in the same directory or sub-directory of a file structure.

**CREF**

A program which produces a sequence-numbered assembly listing followed by tables showing cross references for all operand-type symbols, all user-defined operators, and/or all op codes and pseudo-op codes.

**Customer**

A Digital customer purchasing a DECSYSTEM-10 as distinguished from a user at a console who may be purchasing time from a customer.

**Cylinder**

The hardware-defined region of consecutive logical disk blocks which can be read or written without repositioning.

**DAEMON**

A program for writing all or parts of a job's core area and associated monitor tables onto disk.

**Data Channel**

The device which passes data between the memory system and the control.

**DATDMP**

A program for dumping the core data base.

**DECTape**

A convenient, pocket-sized reel of random access magnetic tape developed by Digital Equipment Corporation.

**DDT**

The Dynamic Debugging Technique program used for on-line checkout, testing, and program composition of object programs.

**Device routines**

Routines that perform I/O for specific storage devices and translate logical block numbers to physical disk addresses. These routines also handle error recovery and ensure ease of programming through device independence.

**DIRECT**

A program for producing directory listings of disks and DECTapes.

**Directory**

A file which contains the names and the pointers to other files on the device. On disk, a directory is continued across all the file structures in a job's search list. Continued MFDs, UFDs, and SFDs are all directories. The DIRECT monitor command lists a directory.

**Directory device**

A storage retrieval device such as disk or DECTape which contains a file describing the layout of stored data (programs and other files).

**Directory path**

The ordered list of directory names, starting with a UFD name, which uniquely specifies a directory without regard to a file structure. Also known as a path. A file structure name, a path, and a filename and extension are needed to uniquely identify a file in the system.

**Dismounting a file structure**

The process of deleting a file structure from a user's active search list by using the DISMOUNT command. It does not necessarily imply physical removal of the file structure from the system.

**Doorbell**

The device by which processors in a multiprocessing system interrupt each other. This is an optional device.

**Dormant file structure**

A file structure that is physically mounted but has no current users, i.e., the mount count is zero.

**Dormant segment**

A sharable high segment kept on a swapping space, and possible core, which is in no user's addressing space.

**DSK**

The generic device name for disk-like devices. Actual file structure names are defined for each job by the file structure search list.

**DSKLST**

A program which gives statuses and statistics of all user disk files at a given point in time.

**DSKRAT**

A damage assessment program that scans a file structure and reports any inconsistencies detected.

**Dump**

A listing of all variables and their values, or a listing of the values of all locations in core.

**DUMP**

A program that outputs selected portions of a file in one of the various formats that can be specified by the user.

**EDDT**

A version of DDT used for debugging programs, such as the monitor, in executive mode.

**Effective address**

The actual address used, that is, the specified address as modified by any indexing or indirect addressing rules.

**Entry point**

A point in a subroutine to which control is transferred when the subroutine is called.

**Executive mode**

A central processor mode characterized by the lack of memory protection and relocation and by the normal execution of all defined operation codes.

**Extended file**

A file which contains one or more extended RIBs to contain the retrieval pointers.

**Extended RIB**

Additional retrieval information blocks (RIBs) required when the retrieval pointers in a file overflow the prime RIB.

**FAILSAFE**

A utility program used to save the contents of the disk on magnetic tape and later restore the saved contents back onto disk.

**FILDDT**

A version of DDT used for examining and changing a file on disk instead of in core memory. This program is used to examine a monitor for debugging purposes.

**File**

An ordered collection of 36-bit words comprising computer instructions and/or data. A file can be of any length, limited only by the available space on the device and the user's maximum space allotment on that device. A file is uniquely identified in the system by a file structure name or directory name, a directory path, and a filename and extension.

**Filename**

A name of one to six alphanumeric characters chosen by the user to identify a file.

**Filename extension**

One to three alphanumeric characters usually chosen by the program to describe the class of information in a file.

**File specification**

A list of quantities which uniquely identify a named file. A complete file specification consists of: the name of the physical device or file structure on which the file is stored, the name of the file including its extension, the name of the directory in which the file is contained, and the protection code associated with the file. File specifications are ignored for non-file-oriented devices.

**File structure**

The logical arrangement of 128-word blocks on one or more units of the same type to form a collection of named files.

**File-structured device**

A device on which data is given names and arranged into files; the device also contains directories of these names.

**File structure owner**

The user whose project-programmer number is associated with the file structure in the administrative file STRLST.SYS. The REACT program is used to enter or delete this project-programmer number or any of the other information that is contained in an STRLST.SYS entry.

**File structure search list**

For each job, a list that specifies the order in which the file structures that user can access are to be searched when device DSK: is specified. Also called a job search list.

**FILEX**

A general file transfer program used to convert between various core image formats and to read and write various DECtape directory formats and standard disk files.

**Flag**

An indicator that signals the occurrence of some condition, such as the end of a word.

**Fragmentation**

The technique used when swapped segments cannot be allocated in one contiguous set of blocks on the swapping space.

**FUDGE2**

A file update generator used to update files containing one or more relocatable binary programs and to manipulate programs within program files.

**Full path name**

The ordered list which uniquely identifies a specific disk file. This list consists of the directory path plus the filename and extension.

**Generic name**

An abbreviation for a physical name. This abbreviation is usually three characters.

**Get**

To transfer a save file from a device into core using a bootstrap program or the monitor.

**GLOB**

A program which reads multiple binary program files and generates an alphabetic cross-referenced list of all the global symbols encountered.

**Global request**

A request to the LOADER to link a global symbol to a program.

**Global symbol**

Any symbol accessible to other programs.

**GRIPE**

A program that reads text from the user and records it in a disk file for later analysis by the operations staff.

**Group**

A contiguous set of disk clusters allocated as a single unit of storage and described by a single retrieval pointer.

**High segment**

The segment of the user's core which generally contains pure code and which can be shared by other jobs; it is usually write-protected.

**Home block**

The block written twice on every unit which identifies the file structure the unit belongs to and its position on the file structure. This block specifies all the parameters of the file structure along with the location of the MFD. The home block appears in the HOME.SYS file.

**HOME.SYS**

The file that contains a number of special blocks for system use. These blocks are the home blocks, the BAT blocks, the ISW blocks, and block zero.

**Idle segment**

A sharable high segment which users in core are not using; however, at least one swapped-out user is using it else it would be a dormant segment.

**Idle time**

The percent of uptime in which no job wanted to run, i.e., all jobs were HALTed or waiting for some external action such as I/O.

**Immediate mode addressing**

The process through which the right half of the word gives the operand and not the address.

**Impure code**

The code which is modified during the course of a run, e.g., data tables.

**Indirect address**

An address in a computer instruction which indicates a location where the address of the referenced operand is to be found.

**INITIA**

A program for performing standard system initialization for a particular terminal. It is used to initiate specific programs, such as the spooling programs, on the designated terminal.

**Initialize**

To set counters, switches, or addresses to zero or other starting values either at the beginning of or at prescribed points in a computer routine.

**Interjob dependency**

The technique by which a Batch job is kept from running until after the running of another job. The first job is dependent on the second job.

**Interleaving**

To increase effective memory speed by configuring the memory addressing so that adjacent addresses reference alternate asynchronous memories.

**Internal symbol**

A symbol which generates a global definition which is used to satisfy all global requests for that symbol.

**Interrupt**

A signal which when activated causes a transfer of control to a specific location in memory thereby breaking the normal operation of the routine being executed. An interrupt is caused by an external event such as a done condition in a peripheral. It is distinguished from a trap which is caused by the execution of a processor instruction.

**ISW block**

A block written by the refresher which contains the bit map for the initial storage allocation table for swapping. Any bad regions are marked as already in use. The ISW block appears in the HOME.SYS file.

**Job**

The entire sequence of steps, from beginning to end, that the user initiates from his interactive terminal or card deck or that the operator initiates from his operator's console.

**Job Data Area**

The first 140 octal locations of a user's core area. This area provides storage for items used by both the monitor and the user program.

**Job search list**

See File Structure Search List.

**Job site**

The location at which jobs are run. Also called program site.

**Job step**

A serial or parallel sequence of processes invoked by a user to perform an operation.

**Jump**

A departure from the normal sequence of executing instructions.

**Label**

A symbolic name used to identify a statement of a program.

**Latency**

- (1) The time from initiation of a transfer operation to the beginning of actual transfer; i.e., verification plus search time.

- (2) The delay while waiting for a rotating memory to reach a given location as desired by the user. The average latency is one half the revolution time.

**LIBOL**

The COBOL object time system.

**Library search mode**

The mode in which a program is loaded only if one or more of its declared entry symbols satisfies an undefined global request. LIB40 is scanned in this mode so as to load only programs that the user needs.

**LIB40**

The standard DEC-supplied library of the FORTRAN object time system and math routines. This library resides on device SYS.

**Line**

A string of characters terminated with a vertical tab, form feed, or line feed. The terminator belongs to the line that it terminates.

**Load**

To produce a core image file from a relocatable binary file (.REL) using the LOADER program. This operation is not to be confused with the GET operation: with the GET operation a core image file has already been produced.

**LOADER**

A program that provides automatic loading and relocation of MACRO, FORTRAN, and COBOL generated binary programs, produces an optional storage map, and performs loading and library searching. Also, the program loads and links relocatable binary programs generated by MACRO, COBOL, and FORTRAN and generates a symbol table in core for execution under DDT.

**Local peripherals**

The I/O devices and other data processing equipment, excluding the central processor, located at the central site.

**Local symbol**

A symbol used only within the program in which it is defined (all non-global symbols). It is not accessible to other programs even though the programs are loaded together.

**Locked job**

A job in core that is never a candidate for swapping or shuffling.

**Logical device name**

An alphanumeric name chosen by the user to represent a physical device. This name can be used synonymously with the physical device name in all references to the device. Logical device names allow device independence in that the most convenient physical device can then be associated with the logical name at run time.

**LOGIN**

The program by which the users gain access to the computing system.

**LOOKFL**

A program for typing the characteristics of a single disk file, such as creation date and number of words written, on the terminal.

**Lost time**

The percent of uptime that the null job was running, but at least on other job wanted to run (was not waiting for a device) but could not because one of the following was true:

- a. the job was being swapped out.
- b. the job was being swapped in.
- c. the job was on disk waiting to be swapped in.
- d. the job was momentarily stopped so devices could become inactive in order to shuffle job in core.

**Low segment**

The segment of core containing the job data area and I/O buffers. This area is unique and accessible to the user and is often used to contain the user's program. If the user is working with a shared program, this area contains data tables, etc.

**MAINT.SYS**

The area of the disk reserved for maintenance use only.

**Macro**

An instruction in a source language which is equivalent to a specified sequence of machine instructions.

**Mask**

- (1) A combination of bits that is used to control the retention or elimination of portions of any word, character, or byte in memory.
- (2) On half-duplex circuits, the characters typed on the terminal to make the password unreadable.

**Master file directory**

The file created at refresh time which contains the name of all user file directories including itself. Referred to as the MFD.

**Master slave system**

A specific multiprocessing system involving two processors where one processor has a more important role than the other.

**Memory protection**

A scheme for preventing access to certain areas of storage for purposes of reading or writing.

**Mnemonic symbol**

A symbolic representation for a computer instruction.

**MONEY**

A program for reading the system's time accounting file and assigning a monetary charge for each user according to the time and resources that he has used on the system.

**MONGEN time**

The time at which the monitor software configuration is being defined or changed. The monitor must then be reloaded with LOADER.

**Monitor**

The collection of programs which schedules and controls the operation of user and system programs, performs overlapped I/O, provides context switching, and allocates resources so that the computer's time is efficiently used.

**Mount Count**

The count of the number of jobs which have a file structure in their active or passive search lists.

**Mounting a device**

To request assignment of an I/O device via the operator.

**Mounting a file structure**

The process of adding a file structure to one's search list. If the file structure is not already defined and mounted, this is requested of the operator.

**Multiprocessing**

Simultaneous execution of two or more computer programs by a computer.

**Multiprocessing system**

A system with two or more central processors sharing some or all of the hardware resources, such as, disks memories, and or monitors.

**Multiprogramming**

A technique that allows scheduling in such a way that more than one job is in an executable state at any one time.

**Named file**

A named ordered collection of 36-bit words (instructions and or data) whose length is not restricted by size or core.

**Nesting**

To include a routine or block of data within a larger routine or block of data.

**Non-directory device**

A device such as a magnetic tape or paper tape which does not contain a file describing the layout of stored data.

**No-op**

An instruction that specifically instructs the computer to do nothing. The next instruction in sequence is then executed.

**Non-sharable segment**

A segment for which each user has his own copy. This segment can be created by a CORE or REMAP UUU or initialized from a file.

**Object time system**

The routines for a particular language which support the compiled code. Usually includes I/O and trap-handling routines.

**Offset**

The number of locations toward zero a program must be moved before it can be executed.

**OMOUNT**

A program for operator interfacing for handling requests concerning removable media.

**ONCE ONLY time**

The time at which the operator can change a number of monitor parameters when the monitor is started up.

**One's complement**

A complement formed by setting each bit in a binary number to the opposite state.

**Operand**

The symbolic addresses of the data to be accessed when an instruction is executed, or the input data or arguments of a pseudo-op or macro instruction.

**Overlay**

The technique of repeatedly using the same blocks of internal storage during different stages of a program. When one routine is no longer needed in storage, another routine can replace all or part of it.

**Pack ID**

A 6-character SIXBIT name or number used to uniquely identify a disk pack.

**Page**

- (1) Any number of lines terminated with a form feed character.
- (2) The smallest allocatable unit of core storage.

**Parity bit**

A binary digit appended to an array of bits to make the sum of all the bits always odd or always even.

**Parity check**

A check that tests whether the number of ones or zeros in an array of binary digits is odd or even.

**Passive search list**

An unordered list of the file structures which have been in the job's active search list and have never been dismounted. Device DSK is not defined by this list.

**Path**

See directory path.

**Peripheral equipment**

Any unit of equipment, distinct from the central processing unit, which can provide the system with outside communication.

**Physical unit name**

The SIXBIT name consisting of 3 to 6 characters that is associated with each unit. Examples: FHA0, FHA1, DPA0, DPA7, LPT, DTA3.

**PIP**

The Peripheral Interchange Program which transfers data files from one standard I/O device to another and performs simple editing and magnetic tape control functions.

**PLEASE**

A program that provides the user with two-way communication with the operator.

**Pointer**

The location containing an address rather than data which is used in indirect addressing.

**Pool**

One or more logically complete file structures that provide file storage for the users and that require no special action on the part of the user.

**Position operation**

The operation of moving the read-write heads of a disk to the proper cylinder prior to a data transfer. This operation requires the control for several microseconds to initiate activity, but does not require the channel or memory system.

**Prime RIB**

The first retrieval information block (RIB) of a file. This block contains all of the user arguments.

**Privileged program**

- (1) Any program running under project number 1, programmer number 2.
- (2) A monitor support program executed by a monitor command and, therefore, has the JACCT (job status bit) set, for example, LOGOUT.

**Priority interrupt**

The interrupt that usurps control of the computer program or system and jumps to an interrupt service routine if its priority is higher than the interrupt currently being serviced, if any.

**Process**

A collection of segments that perform a particular task. A hardware state is associated with a process: a virtual memory, a processor, a stack, etc.

**Program break**

The length of a program; the first location not used by a program (before relocation).

**Program counter (PC)**

A register that, at the beginning of each instruction, normally contains an address one greater than the location of the current instruction.

**Programmed operators**

Instructions which, instead of doing computation, cause a jump into the monitor system or the user area at a predetermined point. The monitor interprets these entries as commands from the user program to perform specified operations.

**Program origin**

The location assigned by the LOADER to relocatable zero of a program.

**Project-programmer number**

Two octal quantities, separated by commas, which, when considered as a unit, identify the user and his file storage area on a file structure.

**Protected location**

A storage location reserved for special purposes in which data cannot be stored without undergoing a screening procedure to establish suitability for storage therein.

**Protection address**

The maximum relative address that the user can reference.

**Pseudo-op**

An operation that is not part of the computer's operation repertoire as realized by hardware; hence, an extension of the set of machine operations. In MACRO, pseudo-ops are directions for assembly operations.

**Public disk pack**

A disk pack belonging to the storage pool and whose storage is available to all users.

**Pure code**

Code which is never modified in the process of execution. Therefore, it is possible to let many users share the same copy of a program.

**Pushdown list**

A list that is constructed and maintained so that the item to be retrieved is the most recently stored item in the list, i.e., last in, first out.

**QMANGR**

The Batch queue manager. QMANGR is called by BATCON to schedule jobs by computing and dynamically revising job priorities.

**Quantum time**

The run time given to each job when it is assigned to run.

**QUE**

The system-wide name defining the location of the spooling and operator work-request queues.

**Queue**

- (1) A list of jobs to be scheduled or run according to system, operator, or user-assigned priorities. Examples: Batch input queue, spooling queues, monitor scheduling queues.

- (2) The system program that allows users to add, delete, list, or modify queue entries in the various system queues.

**QUOLST**

A program that prints the user's quotas for each file structure in his search list and the number of free blocks available in each file structure.

**QUOTA.SYS**

The file that contains a list of users and their quotas for the private file structure on which the file resides.

**Random access**

A process in which the access time is effectively independent of the location of the data.

**REACT**

A program for maintaining administrative control files. It can be used to create, modify, delete or list entries in a file.

**Read**

To open a file for input.

**Record**

A collection of related items of data treated as a unit.

**Reentrant program**

A two-segment program composed of a sharable and non-sharable segment.

**Reformat**

To write new headers on a disk pack using the D50B diagnostic program.

**Refresh**

To remove all files from a file structure and to build the initial set of files based on information in the HOM block.

**Relative address**

The address before hardware or software relocation is added.

**Relocate**

To move a routine from one portion of storage to another and to adjust the necessary address references so that the routine can be executed in its new location.

**Relocation address**

The absolute core address of the first location in the program segment.

**Relocation constant**

The number added by the LOADER to every relocatable reference within a program. The relocation constant is the relocated break of the previous program.

**Remote Batch**

A feature of the computing system that allows data I/O and job control of Batch processing from a distant terminal over a synchronous communication link.

**Remote peripherals**

The I/O devices and other data processing equipment, except the central processor, located at the site of the remote Batch terminal.

**Removing a file structure**

The process of physically removing a file structure from the system. This is requested with the REMOVE switch in the DISMOUNT command string and requires the operator's approval.

**Response time**

The time between the generation of an inquiry and the receipt of an response.

**Return**

- (1) The set of instructions at the end of a subroutine that permits control to return to the proper point in the main program.
- (2) The point in the main program to which control is returned.

**Run**

To transfer a save file from a device into core and to begin execution.

**RUNOFF**

A program that facilitates the preparation of typed or printed manuscripts by performing formatting, case shifting, line justification, page numbering, titling, and indexing.

**SAT.SYS**

The Storage Allocation Table file which contains a bit for each cluster in the file structure. Clusters which are free are indicated by zero and clusters which are bad, allocated and non-existent are indicated by one.

**Save**

To produce a save file from a core image using a bootstrap program or the monitor. This operation is the opposite of the GET operation.

**SCRIPT**

A program that sends predetermined sequences of characters over multiple pseudo-TTYs in order to simulate a load on the system for analysis.

**Search**

The Controller reads sector headers to find the correct sector. The second step in the transfer operation.

**Sector**

A physical portion of a mass storage device.

**Segment**

A logical collection of data, either program data or code, that is the building block of a program. The monitor keeps a segment in core and/or on the swapping device.

**Segment Resident Block**

A block that contains all the information that the monitor requires for a particular segment.

**SETSRC**

A program that allows the user to list or change the search list that is automatically set up for him at job initialization time.

**SFD**

A directory pointed to by a UFD or a higher-level SFD. These directories exist as files under the UFD.

**Sharable segment**

A segment which can be used by several users at a time.

**Shared code**

Pure code residing in the high segment of user's core.

**Single access**

The status of a file structure that allows only one particular job to access the file structure. This job is the one whose project number matches the project number of the owner of the file structure.

**Skip**

An instruction that causes control to bypass one instruction and proceed to the next instruction.

**Spooling**

The technique by which output to slow-speed devices is placed into queues to await transmission; this allows more efficient use of the particular device, core memory, and the central processor unit.

**Static dump**

A dump that is performed at a particular point in time with respect to a machine run, frequently at the end of a run.

**STRNST.SYS**

The administrative file that describes each file structure in the system. This file is used by the MOUNT command only.

**Sub-directory**

A continued SFD.

**Supersede**

To open a file for writing, write the file and close the file when an older copy of the same name already exists. Only one user at a time may supersede a given file at any one time. The older copy of the file is deleted when all users are finished reading it.

**Super-cluster**

A contiguous set of one or more clusters introduced to compress the file pointer for large units into 18 bits. See compressed file pointer.

**Swapping**

The movement by the monitor of user programs between core and secondary storage.

**Swapping class**

The classes of swapping units divided according to speed. Class 0 contains the fastest swapping units.

**Swapping device**

Secondary storage that is suitable for swapping, usually a high-speed disk or drum.

**SWAP.SYS**

The file containing the swapping area on a file structure.

**Symbolic address**

An address used to specify a storage location in the context of a particular program. Symbolic addresses must then be translated into absolute addresses by the assembler.

**Symbol table**

A table which contains all defined symbols and the binary value assigned to each symbol.

**SYS**

A system-wide logical name for the system library. This is the area where the standard programs of the system are maintained.

**SYSDPY**

A variation of the SYSTAT program which runs on a keyboard display at up to 2400 baud.

**SYS search list**

The file structure search list defined at ONCE-ONLY time for device SYS.

**SYSTAT**

A program that displays on the user's terminal the status of the system at any time.

**TECO**

A sophisticated text editor and corrector program that allows simple editing requests, character string searches, complex program editing, command repetition, and text block movement. TECO editing is performed on files recorded in ASCII characters.

**TENDMP**

A utility program used to save and restore core images on DECTape or magnetic tape. It operates only in executive mode.

**Total user core**

The amount of physical core which can be used for locked and unlocked jobs.

**Track**

The portion of a moving storage medium, such as disk, drum, or tape, that is accessible to a given reading head position.

**Transfer operation**

The hardware operation of connecting a channel to a controller and a controller to a unit for passing data between the memory and the unit. The transfer operation involves verification, search, and actual transfer.

**Trap**

An unprogrammed conditional jump to a known location, automatically activated by a side effect of executing a processor instruction. The location from which the jump occurred is then recorded. It is distinguished from an interrupt which is caused by an external event.

**Two's complement**

A number used to represent the negative of a given value. This number is obtained by alternating the bit configuration of each bit in the binary number and adding one to the result.

**UFD**

A file whose entries are the names of files existing in a given project-programmer number area within a file structure.

**UMOUNT**

A program for user interfacing for the handling of requests concerning removable media.

**Unconditional transfer**

An instruction which transfers control to a specified location.

**Unit**

The smallest portion of a device that can be positioned independently from all other units. Several examples of units are: a disk, a disk pack, and a drum.

**Update**

To open a file for reading and writing simultaneously on the same software channel, rewrite one or more blocks in place, and close the file. Only one user at a time may update a given file.

**User**

A person who utilizes the facilities of the DECsystem-10.

**User file directory**

See UFD.

**User I/O mode**

The central processor mode that allows privileged user programs to be run with automatic protection and relocation in effect, as well as the normal execution of all defined operation codes.

**User library**

Any user file containing one or more programs of which some or all can be loaded in library search mode.

**User mode**

A hardware-defined state during which instructions are executed normally except for all I/O and HALT instructions which cause immediate jumps to the monitor. This makes it possible to prevent the user from interfering with other users or with the operation of the monitor. Memory protection and location are in effect so that the user can modify only his area of core.

**User program**

All of the code running under control of the monitor in an addressing space of its own.

**Verification**

The controller reads sector headers to see if the mechanical parts of the system have correctly positioned the arm. The first step in the transfer operation.

**Vestigial job data area**

The first 10 locations of the high segment used to contain data for initializing certain locations in the job data area.

**Virtual core**

The size of the job, both low and high segments.

**Wildcard construction**

A technique used to designate a group of files without enumerating each file separately. The filename, extension, or project-programmer number in a file specification can be replaced totally with an asterisk or partially with a question mark to represent the group of files desired.

**Word**

An ordered set of bits which occupies one storage location and is treated by the computer circuits as a unit. The word length of the DECsystem-10 is 36 bits.

**Zero compression**

The technique of compressing a core image by eliminating consecutive blocks of zeros.

## APPENDIX A

### DECsystem-10 HARDWARE

DECsystem-10 is the name for the family of DEC's large computing systems. Each of the five 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 processors, 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 five systems. The typical configuration of this system has a KA10 central processor, 32 to 64K high-speed ME10 core memories, the RP02G disk system with up to two disk packs, the TM10G magnetic tape system with up to two drives, and low-speed peripheral equipment including a CR10F card reader, an LP10A 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 RM10G swapping drum 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

ME10 core memories, the RP02G disk system with up to four disk packs, the TM10G magnetic tape system, the CR10D card reader, the LP10C line printer, and 32 local lines in either the DC10 or DC68A communications 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 equivalent of DECsystem-1050 with 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 maintain the same interface between the user and the computing system. This approach of co-equal processors gives the user increased computing capacity when processing power is in heavy demand under multi-task loads. In addition to the two KA10 processors, the typical 1055 has 80K of ME10 core memories, with one MX10 memory port multiplexer, one RM10G drum system, one RP03G disk system with up to eight disk packs, one TU40G, 120KC magnetic tape system, one CR10 card reader, the LP10C line printer, and 32 local lines, either a DC10 system or a DC68A system.

#### A.4 DECsystem-1070

The 1070 is a large-scale computing system with more than twice the central processor speed of the DECsystem-1050 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 1070 comprises at least 96K (480K bytes) of ME10 core memory, 690K words (4.1 million characters) of RM10G high-speed drum storage, an RP03G disk system of four disk drives with a total of 41.6 million words (249.6 million characters) of storage, TU40G magnetic tape system with three 120KC drives, a 1200 character-per-minute CR10E card reader, a 1000 line-per-minute LP10C line printer and a communication system capable of 128 lines (either DC10 or DC68A). 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.5 DECsystem-1077

The 1077 is the dual-processor 1070 with fast execution of computing loads because of the second KI10 central processor. In addition, this system typically has 128K (640K bytes) of core memory, 690K words (4.1 million characters) of RM10G drum storage, a RP03G disk system with four disk drives for a total of 41.6 million words (249.6 million characters) of storage, a TU40G magnetic tape system with four 120KC drives, a 1000 line-per-minute LP10C line printer, a 1200 character-per-minute CR10E card reader, and a DC10 or DC68A communication system capable of 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.6 PROCESSOR — KA10

The KA10 arithmetic processor is the processing unit for the three smallest DECsystem-10 machines. Its standard I/O devices are: a. a 300 character-per-second photoelectric paper-tape reader, b. a 50 character-per-second paper-tape punch, c. an operator's console that provides the operator with information and intervention capabilities when desired, and d. a standard Model 35KSR console teleprinter 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 hard-wired. The remainder are programmed operators 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 ns 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 of 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: a. executive mode, which allows all instructions to be executed and suppresses relocation. b. user mode, in which some instructions are not allowed (i.e., I/O instructions) and relocation and protection are in effect, and c. user I/O mode, where all instructions are valid but relocation and protection are still in effect.

#### A.7 PROCESSOR — KI10

The KI10 central processor used with the larger DECsystem-10 machines is nearly 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 the user program to access an effective address space of up to 256K words. This space is segmented into  $512_{10}$  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, which is 16 times larger than the effective address space. (The program's effective address space is 256K (18 bits); the physical address space is 4096K (22 bits)). 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. More core is then able to be addressed when providing a physical address space much larger than the effective address space. This gives programs the ability to access 4 million words.

Eight instructions for double-precision floating-point arithmetic and three instructions for converting between fixed-point and floating-point formats are in the KI10 instruction repertoire. 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, pushdown list overflow conditions, and page failure conditions directly by the execution of programmed trap instructions instead of resorting to a program interrupt system. 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.

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

The KI10 operates in one of two modes, user mode and exec mode. Each of these modes have two submodes: a. public mode and concealed mode in user mode, and b. supervisor mode and kernel mode in exec mode.

User programs operate in user mode. In this mode, the 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 ENTRY 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 proprietary 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 operating system operates in exec mode. The smaller part of the operating system operates in kernel mode and performs both I/O for the system and any functions that effect all users of the system. The larger part of the operating system operates in supervisor mode and performs general management of the system and the functions that effect only one user at a time.

## A.8 CORE MEMORIES

The ME10 core memory contains 16,384 words with a read access time of 600 nanoseconds and a full cycle time of one microsecond. Up to 16 memory modules can be connected to provide 256K of core storage. 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 MD10G mass memory system consists of 64K MD10 core memory and a 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 830 ns, a cycle time of  $1.8\mu s$ , 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 MD10E up to 128K in increments of 32,768 words.

The MD10H mass memory system consists of 128K MD10 core memories and three MD10Es including cables.

## A.9 DRUM SYSTEM

The RM10G drum system consists of a DF10 data channel, a RC10 fixed-head drum control, and a RM10B fixed-head drum. The DF10 controls the transfer of data between a device controller and one port in memory. Up to eight controllers or special devices can be connected to the DF10, providing one data path to core memory. In other words, one device can be transferring data, and other devices on the DF10 must wait until the device has completed the data transfer. The rate of transfer is determined by the speed of the device using the DF10. The RC10 controls up to four RM10B drums. It connects to the processor via the I/O bus for control and status information. Under program control, it establishes a data path between the drum and a core memory port via the

DF10. The RM10B provides 345,000 36-bit words for fast-access storage available for swapping, data storage, and program libraries. It has an average latency time of 8.5 ms and an average transfer time of  $4.5 \mu\text{s}$  per 36-bit word (or about 10.2 ms and  $5.4 \mu\text{s}$  respectively when operating with 50 Hz power). Due to its speed, the drum should be connected to the highest priority memory port via the DF10.

#### A.10 DISK SYSTEMS

The RP02G disk system consists of the DF10 data channel, a RP10 disk control, and two RP02 disk pack drives. The RP10 disk control can provide control of up to eight RP02 disk pack drives. It connects to a DF10 data channel and the I/O bus. 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 ms, which includes 12.5 ms average rotational latency, and the transfer rate is  $15 \mu\text{s}$  per word.

The RP03G disk system includes a DF10 data channel, a RP10C disk control, and four RP03 disk pack drives. The RP10C can control up to eight RP02 or RP03 (or a combination of the two) disk pack drives. The RP03 has a total of 400 cylinders that give twice the storage capacity of the RP02. The average access time is 41.5 ms including the 12.5 ms average rotational latency, and the transfer rate is identical to the RP02.

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  $1.966 \times 10^9$  characters of on-line storage.

#### A.11 MAGNETIC TAPE SYSTEMS

The TD10G DECTape system consists of a TD10 DECTape controller and a TU56 DECTape transport. The TD10 controls either four TU56 dual DECTape transports or 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  $400 \mu\text{s}$ . The TU56 transport reads and writes magnetic tape at 15K characters per second. The tapes are 3-3/4 in. in diameter, 260 ft. long, and 3/4 in. wide. Each tape has a directory providing random access to user files. The tape units are bidirectional and redundantly recorded, resulting in greater maintainability and reliability.

The TM10G 36KC magnetic tape system has a TM10A magnetic tape control and either two TU10 or two TU20 magnetic tape units. The TM10A controls the operation of up to eight tape transports and provides a data path from the tape transport to the central processor via the

I/O bus. The data transfer rate is determined by the speed and density of the drive being controlled. The TU10 magnetic tape unit reads and writes 9-channel (TU10E) or 7-channel (TU10F) industry standard tape at 45 in. per second and a density of 200, 556, and 800 bits per inch (TU10E) or bits per second (TU10F). The TU20A magnetic tape unit reads and writes 9-channel USASI standard magnetic tape at a rate of 45 in. per second and with a density of 800 bits per inch. The TM10 controller assembles four 8-bit characters per 36-bit word transfer. The TU20B magnetic tape unit reads and writes 7-channel industry standard magnetic tape at the rate of 45 in. per second and with densities of 200, 556, and 800 bits per inch. The TM10 controller assembles six 6-bit characters per 36-bit word for transfer.

The TU40G 120KC magnetic tape system includes a DF10 data channel, a TM10B magnetic tape control, and two TU40 magnetic 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 in. 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 in. per second and a density of 200, 556, and 800 bits per inch.

#### A.12 INPUT/OUTPUT DEVICES

##### A.12.1 Card Readers

The card readers offered with the DECSYSTEM-10 have insignificant card wear, high tolerance to damaged cards and are virtually jamproof. 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 CR10D card reader is a table-top model that reads 80-column EIA standard cards at 1000 cards per minute. The capacity of the card hopper is 1000 cards. The card reader controller connects to the BA10 hard copy controller.

The CR10E console model card reader inputs 80-column EIA standard cards at 1200 cards per minute. The maximum number of cards held by the input and output hoppers is 2250 cards. The controller is mounted in the BA10 hard copy controller cabinet.

The CR10F card reader is a table-top model and reads 80-column EIA standard cards at the rate of 300 cards per minute. The hopper of the CR10F holds 600 cards. The controller connects to the BA10 hard copy controller.

#### A.12.2 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. The card punch controller is mounted in the BA10 hard copy controller cabinet.

#### A.12.3 Line Printers

The 64-character LP10A line printer prints 300 lines per minute and 132 columns per line. The printable character set is composed up upper-case characters, numbers, and special characters. The line printer is connected to the I/O bus via a controller mounted in the BA10 hard copy controller.

The 64-character LP10C line printer prints 1000 lines per minute and 132 columns per line. The printable character set is the same as the LP10A character set. The line printer is connected to the I/O bus with the BA10 hard copy controller.

#### A.12.4 Plotters

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

**A.12.4.1 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.12.4.2 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.12.5 BA10 Hard Copy 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.13 TELETYPES AND TERMINALS

The Teletypes and Terminals used on the DC10 and the DC68A are similar except for different cables and interface connectors.

#### A.13.1 Local DC10 Use

The LT33A teleprinter is the 33TS machine (33KSR, friction feed).

The LT33B teleprinter is the 33TY machine (33ASR, sprocket feed, automatic reader control XON/XOFF feature).

The LT35A teleprinter is the VSL312HF machine (35KSR, sprocket feed).

#### A.13.2 Local DC68A Use

The LT33C teleprinter is the 33TS machine (33KSR, friction feed).

The LT33H teleprinter is the 33TY machine (33ASR, sprocket feed, automatic reader control XON/XOFF feature).

The LT35C teleprinter is the VSL312HF machine (35KSR, sprocket feed).

#### A.13.3 CRT displays

The VT06 alphanumeric terminal is a CRT display terminal capable of transmitting data locally or over standard phone lines using data sets conforming to the RS-232-C standard. The VT06 can functionally be interchanged with a teleprinter. In addition, the VT06 can be

used for display-oriented operations by utilizing the cursor-control features. It has 25 lines of 72 characters each and operates asynchronously full-or half-duplex at a variety of baud rates up to 2400 baud, selectable by a switch on the rear panel.

The VT05 alphanumeric terminal is a CRT display terminal capable of full-and half-duplex data transmission at rates up to 300 baud. Alphanumerics can be superimposed over a video image derived from closed circuit TV or video tape.

#### A.14 DATA COMMUNICATIONS 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.14.1 DC10 Data Line Scanner

The DC10 hardwired data line scanner interfaces asynchronous communications lines to the processor via the I/O bus. The DC10A control unit is the basic 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-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 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.

##### A.14.2 DC68A Communication System

The DC68A programmable communications system is built around the 680/I communications version of the PDP-8/I. Characters are assembled via program control, which results in a very low incremental cost per line. The DC68A is optimized for a large number of 110 baud lines, but will operate at speeds up to 300 baud. The PDP-8/I is under monitor control and transfer across the interface occurs on the character-by-character basis. The DC68A provides on-line servicing of up to 63 communication lines. Terminals can be local or remote through data sets. The standard configuration includes one DA10 interface, one PDP-8/I-D computer (4K of memory with MP8/I parity option, and a Model 33ASR teleprinter), one DL8/I serial line adapter, one DC08A serial line multiplexor, and three clocks for line frequency operations at 110, 150, or 300 baud rates. Additional options mentioned in this section are required for implementing a specific number of local or data sets.

The M750 dual serial line adapter implements two full-duplex channels in the basic communication system. One unit is required for every two local or data set lines. The DC08B local line panel accommodates up to 48 local terminals suitable for direct 680/I connection. The DC08F modem interface and control multiplexor accommodates up to 32 dual modem control units to handle up to 64 asynchronous lines. The DC08G dual modem control unit implements two modem control units in the DC08F. It includes 25 ft. cables with modem connector DB-25D.

##### A.14.3 DS10 Synchronous Line Unit

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 an DS10 at up to 9600 baud, or two DS10s at up to 4800 baud each.

##### A.14.4 DC75 Synchronous Communications System

The DC75 synchronous communication 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/DS11 combinations 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.14.5 DC71 Remote Batch Station**

The DC71 remote batch station consists of a PDP-8/I processor, an operator Teletype, a card reader, a line printer, and a synchronous interface. The DC71 connects to the DS10 or the DC75 via a full-duplex synchronous communications link. The remote batch terminal can be either a DC71A or DC71B terminal. The DC71A is configured with a 132-column line printer with a 64-character set. The DC71B is configured with a 96-character set line printer. The DC71D Teletype Concentrator package includes eight lines for connecting to the DC71A or DC71B. Another eight lines can be added by connecting the DC71E to the DC71D. Terminals can be Teletypes, VT06 or VT05 display terminals, or other teletype-compatible terminal interfaces, at speeds up to 2400 baud.



**dec**system10  
GETTING STARTED  
WITH TIMESHARING

1st Printing June 1971

2nd Printing July 1972

Copyright © 1971, 1972 by Digital Equipment Corporation

The material in this manual is for information purposes and is subject to change without notice.

The following are trademarks of Digital Equipment Corporation, Maynard, Massachusetts:

DEC  
FLIP CHIP  
DIGITAL

PDP  
FOCAL  
COMPUTER LAB

CONTENTS

|      |   | <u>Page</u> |
|------|---|-------------|
| 1.0  | Getting on the System                               | 59          |
| 2.0  | Files   | 61          |
| 3.0  | Creating Files                                      | 62          |
| 3.1  | The CREATE Command                                  | 63          |
| 3.2  | The MAKE Command                                    | 64          |
| 4.0  | Editing Files                                       | 64          |
| 4.1  | The EDIT Command                                    | 64          |
| 4.2  | The TECO Command                                    | 65          |
| 5.0  | Manipulating Files                                  | 65          |
| 5.1  | The DIRECT Command                                  | 66          |
| 5.2  | The TYPE Command                                    | 66          |
| 5.3  | The DELETE Command                                  | 67          |
| 5.4  | The RENAME Command                                  | 67          |
| 6.0  | Translating, Loading, Executing, Debugging Programs | 67          |
| 6.1  | The COMPILE Command                                 | 67          |
| 6.2  | The LOAD Command                                    | 68          |
| 6.3  | The EXECUTE Command                                 | 68          |
| 6.4  | The DEBUG Command                                   | 69          |
| 7.0  | Getting Information from the System                 | 70          |
| 7.1  | The PJOB Command                                    | 71          |
| 7.2  | The DAYTIME Command                                 | 71          |
| 7.3  | The TIME Command                                    | 71          |
| 8.0  | Leaving the System                                  | 72          |
| 8.1  | The KJOB Command                                    | 72          |
| 9.0  | How to Live with the Terminal                       | 73          |
| 9.1  | Control -C  | 73          |
| 9.2  | The RETURN Key                                      | 74          |
| 9.3  | The RUBOUT Key                                      | 74          |
| 9.4  | Control -U  | 74          |
| 9.5  | The ALTMODE Key                                     | 75          |
| 9.6  | Control -O  | 75          |
| 10.0 | Peripheral Devices                                  | 75          |
| 11.0 | Commands to Allocate System Resources               | 77          |
| 11.1 | The ASSIGN Command                                  | 77          |

## CONTENTS (Cont)

|      |  | <u>Page</u> |
|------|--|-------------|
| 11.2 | The MOUNT Command                                      | 78          |
| 11.3 | The DEASSIGN Command                                   | 79          |
| 11.4 | The DISMOUNT Command                                   | 79          |
| 11.5 | The REASSIGN Command                                   | 79          |
| 11.6 | The FINISH Command                                     | 80          |
| 11.7 | The CORE Command                                       | 80          |
| 12.0 | Commands to Manipulate Terminals                       | 80          |
| 12.1 | The SEND Command                                       | 80          |
| 12.2 | The DETACH Command                                     | 81          |
| 12.3 | The ATTACH Command                                     | 81          |
| 13.0 | Commands to Request Line Printer Output                | 81          |
| 13.1 | The PRINT Command                                      | 81          |
| 13.2 | The CREF Command                                       | 82          |
| 13.3 | The DIRECT Command                                     | 82          |
| 14.0 | Commands to Manipulate Core Images                     | 83          |
| 14.1 | The SAVE Command                                       | 83          |
| 14.2 | The RUN Command  | 83          |
| 14.3 | The R Command  | 83          |
| 14.4 | The GET Command  | 84          |
| 15.0 | Commands to Start a Program                            | 84          |
| 15.1 | The START Command                                      | 84          |
| 15.2 | The HALT (1C) Command                                  | 84          |
| 15.3 | The CONTINUE Command                                   | 84          |
| 16.0 | Additional Commands to Get Information from the System | 85          |
| 16.1 | The RESOURCES Command                                  | 85          |
| 16.2 | The SYSTAT Command                                     | 85          |

## TABLES

| <u>Table No.</u> | <u>Title</u>       | <u>Page</u> |
|------------------|--------------------|-------------|
| 1                | Peripheral Devices | 75          |

## FOREWORD

Getting Started With Timesharing is a simplified guide intended for the beginning timesharing user of the DECSYSTEM-10. This document presents an overall view of the timesharing use of the System, but does not describe every command available to the user. DECSYSTEM-10 OPERATING SYSTEM COMMANDS (DEC-10-MRDC-D) is the complete reference document for the command repertoire, and it should be referred to for any additional information.



Programs are typed directly into the computer by means of the terminal. By typing in programs, you establish communication with other programs already resident in the computer. The first resident program you communicate with is the monitor, the most important program in the computer. The monitor is the master program that plays an important role in the efficient operation of the computer. Just as the terminal is your link with the computer, the monitor is your link with the programs within the computer.

The monitor has many functions to perform, like keeping a record of what each user is doing and deciding what user should be serviced next and for how long. The one function of the monitor that is of greatest concern at this point is that the monitor retrieves any resident programs that you need. This retrieval happens only if the monitor "understands" what is expected of it. The commands to the monitor which are explained in the following sections are sufficient for the terminal to be the device by which information is inputted into the system and by which the system outputs its results.

See section 9.0 for a discussion on How to Live With the Terminal.

## 1.0 GETTING ON THE SYSTEM

In order to gain access to the timesharing system, you must say hello to the system by "logging in". The first move is to make contact with the computer facility by whatever means the facility has established (e.g., acoustic coupler, telephone, or dataphone). Next, notice the plastic knob (the power switch) on the lower right-hand side of the terminal. This knob has three positions: ON, OFF, and LOCAL (turning clockwise). When the knob is in the LOCAL position, the terminal is like a typewriter; it is not communicating with the system at all. The knob must be turned to the ON position in order to establish communication with the computer. When the terminal is turned ON, type a  $\uparrow$ C (depress the CTRL key and type C). This action establishes communication with the monitor. The monitor

---

We wish to express appreciation to Stanford University for the use of their Stanford A-1 Project User's Manual, Chapter 1, SAILON No. 54, as a guide in writing the material in this section.

signifies its readiness to accept commands by responding with a period (.). All the commands discussed in this document can only be typed to the monitor. They are operative when the monitor has typed a period, signifying that it is waiting for a command.

The first program the monitor should call in for you is the LOGIN program. This is accomplished by typing LOGIN followed by a carriage-return (depress the RETURN key). All commands to the monitor should be terminated with a carriage-return. When the monitor "sees" a carriage-return, it knows that a command has been typed and it begins to execute the command.

In the text, underscoring is used to designate terminal output. A carriage-return is designated by a `)`.

By typing LOGIN, you cause the monitor to read the LOGIN program from the disk into core memory and it is this program that is now in control of your terminal. Before the LOGIN program is called in, the monitor assigns you a job number for system bookkeeping purposes. The system responds with an information message similar to the following.

```
JOB 17   SS0218A   TTY34
#
```

In the first line, the system has assigned your job number (17) and has given both the name of the monitor and its version number and the number of your terminal line. The version number changes whenever a change, or patch, is incorporated into the monitor. In the second line, the number sign (`#`), which is typed out by the LOGIN program, signifies that it wants your identification.

The standard identification code is in the form of project numbers and programmer numbers, but individual installations may have different codes. The numbers, or whatever code each installation uses, are assigned to each user by the installation. The LOGIN program waits for you to type in your project number and your programmer number, separated by a comma and terminated with a carriage-return, following the number sign.

```
JOB 17   SS0218A   TTY34
#27,400 )
```

An alternate method of typing in your project number and programmer number is to type your identification on the same line as the LOGIN command and to follow it with a carriage return. The system responds with the information message, and the LOGIN program does not type out the number sign. For example,

```
.LOGIN 27,400 )
JOB 17   SS0218A   TTY34
```

The LOGIN program needs one more item to complete its analysis of your identification. This it requests in the next line by asking for your password.

```
JOB 17 5S0218A TTY34  
#27,400 )  
PASSWORD: )
```

Type in your password, which is also assigned by the installation, followed by a carriage-return. To maintain password security, the LOGIN program does not print the password.

If the identification typed in matches the identification stored in the accounting file in the monitor, the LOGIN program signifies its acceptance by responding with the time, date, day of the week, the message of the day (if any), and a period.

```
.LOGIN ) .LOGIN 27,400 )  
JOB 17 5S0218A TTY34 JOB 17 5S0218A TTY34  
#27,400 ) PASSWORD: )  
PASSWORD: ) 1050 4-MAY-71 WED  
1050 4-MAY-71 WED  
TYPE SYS:SCHED FOR SYSTEM TYPE SYS:SCHED FOR SYSTEM  
SCHEDULE SCHEDULE  
. .
```

This typeout indicates that the LOGIN program has exited and returned control to the monitor. You have successfully logged in and may now have the monitor call in other programs for you. If the identification typed in does not match the identification in the accounting file, the monitor types out the error message

```
?INVALID ENTRY-TRY AGAIN  
#
```

If this error message occurs, type in the correct project-programmer numbers and password.

## 2.0 FILES

When you want to run a program, first type in the program and decide on a name for it. The program is stored on the disk with the specified name. Then translate the program by calling in a translator and giving it the name of the program you wish to translate.

A program, or data, is stored on the disk in files. If a program is being typed in to a text editor (for example, TECO), the editor is busy accepting the characters being typed in and generating a disk file for them. Then, when the program is to be translated, the translator reads this file just created and generates a relocatable binary file. Since you may have many files and the other users on the computer may have files, there must be a method for keeping all of these files separate. This is accomplished by

giving each user a unique area on the disk. This area is identified by your project and programmer numbers. For example, if your project and programmer numbers are 27,400, you have a disk area by that name. Each file you create goes to your disk area and must be uniquely named.

Files are named with a certain convention, the same as a person is named. The first name, the filename, is the actual name of the file, and the last name, the filename extension, indicates what group the file is associated with. The filename and the filename extension are separated by a period.

File names are from one to six letters or digits. All letters or digits after the sixth are ignored. The filename extension is from one to three letters or digits. It is generally used to indicate the type of information in the file. The following are examples of standard filename extensions.

|      |                                    |
|------|------------------------------------|
| .TMP | Temporary file                     |
| .MAC | Source file in MACRO language      |
| .F4  | Source file in FORTRAN IV language |
| .BAS | Source file in BASIC language      |
| .ALG | Source file in ALGOL language      |
| .CBL | Source file in COBOL language      |
| .REL | Relocatable binary file            |
| .SAV | A saved core image                 |

Since files are identified by the complete name and the project and programmer numbers, two users may use the same filename as long as they have different project and programmer numbers; the files would be distinct and separate. The following are examples of filenames with filename extensions.

|            |                                      |
|------------|--------------------------------------|
| MAIN.F4    | A FORTRAN file named MAIN            |
| SAMPLE.BAS | A BASIC file named SAMPLE            |
| TEST1.TMP  | A temporary file named TEST1         |
| NAME.REL   | A relocatable binary file named NAME |

### 3.0 CREATING FILES

The two commands mentioned in this section use two editors to create a new disk file. One of the editors is LINED, a line-oriented editor, and the other is TECO, the Text Editor and Corrector (refer to the LINED and TECO documents in the DECsystem-10 Software Notebooks). Each command requires a filename as its argument and should have a filename extension. A new file may be created with either of these commands, depending on the editor desired.

### 3.1 The CREATE Command

The CREATE command is used only to create a new disk file. When this command is executed, the monitor calls in LINED to initialize a disk file with the specified name and to accept input from the terminal. At this point, begin to type in your program, line by line. LINED types a line number at the beginning of each line so that later a reference to a given line may be made in order to make corrections. Below is a sample program using the commands discussed so far.

|  |   |
|--|---|
| <pre> †C . LOGIN 27,400 ) JOB 17   SS0218A   TTY34  PASSWORD: )  1050   4-MAY-71   WED TYPE SYS: SCHED FOR SYSTEM SCHEDULE . CREATE MAIN.F4 )  *  I )  00010   TYPE 53 ) 00020  53  FORMAT ( ' THIS IS MY PROGRAM' ) ) 00030   END ) 00040   \$  *  F ) </pre> | <p>Establish communication with the monitor. Type C while depressing the CTRL key.</p> <p>Begin the login procedure and type in your identification.</p> <p>The job number assigned, followed by the monitor name and version and the terminal line number. The LOGIN program requests identification (project number and programmer number) if it was not typed on the same line as the LOGIN command.</p> <p>The LOGIN program requests password. Type it in; it is not printed.</p> <p>If identification matches identification stored in the system, the monitor responds with the time, date, day of the week, message of the day, and a period.</p> <p>A new file on the disk is to be created and called MAIN.F4. The extension .F4 is used because the program is to be a FORTRAN source file. LINED is called in to create the file.</p> <p>Response from LINED signifying it is ready to accept commands.</p> <p>A command to LINED to insert line numbers starting with 10 and incrementing by 10 (refer to the LINED document).</p> <p>Type in your FORTRAN PROGRAM.</p> <p>The <b>( \$ )</b> (altmode) is a command to LINED to end the insert. On the terminal this key is labeled ALT, ESC, or PREFIX.</p> <p>Response from LINED signifying it is ready to accept another command.</p> <p>A command to LINED to end the creation of the file.</p> |
|--|---|

|          |   |
|----------|---|
| <u>*</u> | Response from LINED indicating readiness to accept a command. |
| rC       | Return to the monitor.  |
| <u>.</u> | The monitor now has control of the program.                   |

The three LINED commands (I, \$, E) shown in the examples are fully discussed in the LINED document.

### 3.2 The MAKE Command

This command can also be used to open a new disk file for creation. It differs from the CREATE command in that TECO is used instead of LINED. (TECO is discussed in the DECsystem-10 Software Notebooks). Otherwise, the CREATE and MAKE commands operate in the same manner.

```
.MAKE FILEA.F4 )
*I (Text input) $$
EX$$
EXIT
```

.

The altmode (\$) and the EX command are commands to TECO and are explained in the TECO document.

## 4.0 EDITING FILES

After creating a text file, you may wish to modify, or edit, it. The following two commands cause an existing file to be opened for changes. One command (EDIT) calls in LINED, and the other (TECO) calls in TECO. In general, the editor used to create the file should be used for editing. Each command requires, as its argument, the same filename and filename extension used to create the file.

### 4.1 The EDIT Command

The EDIT command causes LINED to be called in and, as the name implies, signifies that you wish to edit the specified file. LINED responds with an asterisk and waits for input. The file specified must be an already existing sequence-numbered file on the disk. For example, in Paragraph 3.1, the file MAIN.F4 was created. If the command

```
.EDIT MAIM.F4 )
```

is given to edit the file, the computer responds with an error message (assuming that there was no file named MAIM.F4). The command

```
._EDIT MAIN.F4 )
```

causes the right file to be opened for editing.

#### 4.2 The TECO Command

The TECO command is similar to the EDIT command except that it causes the TECO program to open an already existing file on the disk for editing purposes. The command sequence

```
._TECO FILEA.F4 )
_* (editing) $$
*_FX$$
```

causes TECO to open FILEA.F4 for editing and close the file upon completion, creating a backup file out of the original file. Whenever one of the commands used to create or edit a file is executed, this command with its arguments (filename and filename extension) is "remembered" in a temporary file on the disk. Because of this, the file last edited may be recalled for the next edit without having the filename specified again. For example, if the command

```
._CREATE PROG1.MAC )
```

is executed, then you may type the command

```
._EDIT )
```

instead of

```
._EDIT PROG1.MAC )
```

assuming that no other CREATE, TECO, MAKE, or EDIT command that changed the filename was used in-between. As mentioned before, if a command tries to edit a file that has not been created, an error message is given.

#### 5.0 MANIPULATING FILES

You may have many files saved on your disk area. (For discussion on how to save a file on your disk area, refer to Paragraph 14.1.) The list of your files, along with lists of other users' files, is kept on the disk in what are called user directories. Suppose you cannot remember if you have created and saved a particular file. The next command helps in just that type of situation.

### 5.1 The DIRECT Command

The DIRECT command requests from the monitor a listing of the directory of your disk area. The monitor responds by typing on the terminal the names of your files, the length of each file in the number of DECsystem-10 disk blocks written (a block is 128<sub>10</sub> words), and the date on which each file was created. The protection associated with each file is also output. This protection is a code that indicates which users are allowed to access your files. It is automatically assigned when you create the file. Refer to DECsystem-10 Monitor Calls (DEC-10-MRRB-D) for an explanation of file protection.

Names of files not explicitly created by you may show up in the directory. These files were created as intermediate files for storage by programs you may have used. For example, in translating a file, the translator generates a file with the same filename but with a filename extension of .REL. This file contains the relocatable binary translation of the source file. You may also notice filenames with the filename extension of .TMP. This extension signifies a temporary file created and used by different system programs.

### 5.2 The TYPE Command

By listing your directory on the terminal, you know the names of the files on your disk area. But what if you have forgotten the information contained in a particular file? The TYPE command causes the contents of source files specified in your command string to be typed on your terminal. For example, the command

```
_.TYPE MAIN.F4 )
```

causes the file MAIN.F4 to be typed on the terminal. Multiple files separated by commas may be specified in one command string, and only source files, not binary files, may be listed.

This command allows the "asterisk construction" to be used. This means that the filename or the filename extension may be replaced with an asterisk to mean any filename or filename extension. For example, the command

```
_.TYPE FILEB.* )
```

causes all files named FILEB, regardless of filename extensions, to be typed. The command

```
_.TYPE *.MAC )
```

causes all files with the filename extension of .MAC to be typed. The command

```
_.TYPE *.* )
```

causes all files to be typed.

### 5.3 The DELETE Command

Having finished with a file, you may erase it from your disk area with the DELETE command. Multiple files may be deleted in one command string by separating the files with commas. For example,

```
.DELETE LINEAR )
```

and

```
.DELETE CHANGE.F4, SINE.REL )
```

are both legal commands. The asterisk convention discussed in section 5.2 may also be used with the DELETE command.

### 5.4 The RENAME Command

The names of one or more files on your disk area may be changed with the RENAME command. The old filename on the right and the new filename on the left are separated by an equal (=) sign. In renaming more than one file, each pair of filenames (new=old) is separated by commas. For example, the command

```
.RENAME SALES.CBL=GROSS.CBL, FILE2.F4=FILE1.F4 )
```

changes the name of file GROSS.CBL to SALES.CBL and file FILE1.F4 to FILE2.F4. The old filename no longer appears in your directory; instead the new filenames appear containing exactly the same data as in the old files. The asterisk convention may again be used. For example, the command

```
.RENAME *.F4=* )
```

causes all files with no filename extension to have the extension .F4.

## 6.0 TRANSLATING, LOADING, EXECUTING, DEBUGGING PROGRAMS

As this point you know how to get on the system, how to create and edit a source file of a program, and how to list your source file on the terminal. The program has not been executed. This only happens after it has been translated into the binary machine language understandable to the computer and loaded into core memory. More often than not the program must be debugged.

### 6.1 The COMPILE Command

This command has as its argument one or more filenames separated by commas. It causes each command to be processed (translated) if necessary by the appropriate processor (translator). It is considered necessary to process a file if no .REL file of the source file exists, or if the .REL file was created

before the last time the source file was edited. If the .REL file is up-to-date, no translation is done. The appropriate processor is determined by examining the extension of the file. The following shows which processor is used for various extensions.

|                              |                       |
|------------------------------|-----------------------|
| .MAC                         | MACRO assembler       |
| .F4                          | FORTRAN IV compiler   |
| .ALG                         | ALGOL compiler        |
| .CBL                         | COBOL compiler        |
| .REL                         | No processing is done |
| other than above,<br>or null | "Standard processor"  |

The standard processor is used to translate programs with null or nonstandard extensions. The standard processor is FORTRAN at the beginning of the command string, but may be changed by use of various switches (refer to DECsystem-10 Operating System Commands). Although it is not necessary to indicate the extension of a file in the COMPILE command string, the standard processor can be disregarded if all source files are kept with the appropriate extension.

When the appropriate translator has translated the source file, there is a file on your disk area with the filename extension .REL and the same filename as the source file. This file is where the translator stores the results of its translation and is called the relocatable binary of the program. The program is now translated into binary machine language, but is still on the disk. Since the disk is used for storage and not for execution, a copy of the binary program must be loaded into core memory to form a core image. The core memory of the computer is used for execution; it is like a scratch pad. The COMPILE command does not generate a core image, but the following three commands do.

## 6.2 The LOAD Command

The LOAD command performs the same operations as the COMPILE command and in addition causes the LOADER to be run. The LOADER is a program that takes the specified REL files, links them together, and generates a core image. The LOAD command does not cause execution of the program.

## 6.3 The EXECUTE Command

This command performs the functions of the LOAD command and also begins execution of the loaded programs, if no translation or loading errors are detected. The compiled program is now in core memory and running, and what happens next depends on the program. More than likely, the program is not returning the correct answers, and you now enter the magic world of program debugging.

### 6.4 The DEBUG Command

This command prepares for the debugging of a program in addition to performing the functions of the COMPILE and LOAD commands. DDT, the Dynamic Debugging Technique program (refer to the DDT manual, DEC-10-CDDE-D), is loaded into core memory first, followed by the program. Upon completion of loading, DDT is started rather than the program. A command to DDT may then be issued to begin the program execution. This command should be used by the experienced programmer familiar with DDT. The above four commands have extended command forms discussed in DECsystem-10 Operating System Commands.

The following is an example showing the compilation and execution of a FORTRAN main program and subroutine. The login procedure is not shown.

|  |   |
|--|---|
| <u>._CREATE MAIN.F4 )</u>                              | CREATE a disk file.   |
| <u>*I )</u>  | Command to LINED to begin inserting on line 10, incrementing by 10. |
| <u>00010</u> TYPE 69 )                                 | Statements of the FORTRAN main program.                             |
| <u>00020</u> 69 FORMAT (' THIS IS THE MAIN PROGRAM') ) |   |
| <u>00030</u> CALL SUB1 )                               |   |
| <u>00040</u> END )                                     |   |
| <u>00050</u> \$  | Altmode ends the insert.  |
| <u>*E )</u>  | LINED command to end the edit.                                      |
| <u>*+C</u>   | Return to the monitor.  |
| <u>._CREATE PROG.F4 )</u>                              | Create a disk file for the subroutine.                              |
| <u>*I )</u>  | Begin inserting at line 10 incrementing by 10.                      |
| <u>00010</u> SUBROUTINE SUBR )                         | Statements of the FORTRAN Subroutine.                               |
| <u>00020</u> TYPE 105 )                                |   |
| <u>00030</u> 105 FORMAT (' THIS IS THE SUBROUTINE') )  |   |
| <u>00040</u> RETURN )                                  |   |
| <u>00050</u> \$  | Altmode ends the insert.  |
| <u>*E )</u>  | LINED command to end the edit.                                      |
| <u>*+C</u>   | Return to monitor.  |
| <u>._EXECUTE MAIN.F4,PROG.F4 )</u>                     | Request execution of the programs created.                          |
| <u>FORTRAN: MAIN.F4</u>                                | FORTRAN reports its progress.                                       |
| <u>FORTRAN: PROG.F4</u>                                |   |
| <u>LOADING</u>   |   |

|  |   |
|--|---|
| <u>000001 UNDEFINED GLOBALS</u>                    |   |
| <u>SUB1 000152</u>                                 | There is no subroutine named SUB1.  |
| <u>?</u>   |   |
| <u>LOADER 3K CORE</u>                              | This includes the space for the loader.   |
| <u>?EXECUTION DELETED</u>                          | No execution was done.  |
| <u>EXIT</u>  |   |
| <br>   |   |
| <u>.EDIT )</u>                                     | Ask to edit PROG.F4, filename need not be mentioned since it was the last file named. |
| <u>*P10,20 )</u>                                   | Type lines 00010 and 00020 on the terminal.   |
| <u>00010 SUBROUTINE SUBR</u>                       |   |
| <u>00020 TYPE 105</u>                              |   |
| <u>*I10 )</u>                                      | Insert a new line 10.   |
| <u>00010' SUBROUTINE SUB1 )</u>                    |   |
| <u>00020' S</u>                                    | Terminate the insert.   |
| <u>*E )</u>  | End the edit.   |
| <u>*!C</u>   |   |
| <br>   |   |
| <u>.EXECUTE MAIN.F4,PROG.F4 )</u>                  | Request execution.  |
| <u>FORTTRAN: PROG.F4</u>                           | Only the subroutine is recompiled since only it has been edited.                      |
| <br>   |   |
| <u>LOADING</u>                                     | Both MAIN and PROG are loaded.  |
| <br>   |   |
| <u>LOADER 3K CORE</u>                              |   |
| <br>   |   |
| <u>EXECUTION</u>                                   |   |
| <br>   |   |
| <u>THIS IS THE MAIN PROGRAM</u>                    | Execution begins.   |
| <u>THIS IS THE SUBROUTINE</u>                      |   |
| <u>CPU TIME: 0.03 SEC. ELAPSED TIME: 0.13 SEC.</u> |   |
| <u>NO EXECUTION ERRORS DETECTED</u>                |   |
| <br>   |   |
| <u>EXIT</u>  | Execution ends.   |
| <br>   |   |
| <u>.</u>   |   |

## 7.0 GETTING INFORMATION FROM THE SYSTEM

There are several monitor commands that are used to obtain information from the system. Three commands useful at this point are discussed in this section, and additional commands are discussed in Paragraph 16.0.

### 7.1 The PJOB Command

If you have forgotten the job number assigned to you at LOGIN time, you may use the PJOB command to obtain it. The system responds to this command by typing out your assigned job number. For example,

```
.PJOB )
17
```

### 7.2 The DAYTIME Command

This command gives the date followed by the time of day. The time is presented in the following format:

```
hh:mm:ss
```

where hh represents the hours, mm represents the minutes, and ss represents the seconds. For example,

```
.DAYTIME )
17-MAY-71 14:37:35
```

### 7.3 The TIME Command

The TIME command produces three lines of typeout. The first line is the total running time since the last TIME command was typed. The second line is the total running time since you logged in. The third line is used for accounting purposes. The time is presented in the following format:

```
hh:mm.ss
```

where hh represents the hours, mm the minutes, and ss the seconds to the nearest hundredth. For example,

```
.TIME )
52.45
02:29.95
KILO-CORE-SEC=57
```

In the first two lines, you are told that you have been running 52.45 seconds since the last time you typed the TIME command, and a total of 2 minutes and 29.95 seconds since you logged in. The third line of typeout is used by your installation for accounting and is the integrated product of running time and core size. Refer to DECsystem-10 Operating System Commands.

## 8.0 LEAVING THE SYSTEM

Now that you know how to log into the system and create and run a program, you might be wondering how you leave the system. You have to tell the system you are leaving, and you do this by the KJOB command.

### 8.1 The KJOB Command

The KJOB command is your way of saying goodbye to the system. Many things happen when you type the command. The job number assigned to you is released and your terminal is now free for another user. An automatic TIME command is performed. In addition, if you have any files on your disk area, the monitor responds with

#### CONFIRM:

and you have several options available to you. By typing H and a carriage return after the CONFIRM: message, the monitor lists the options available. For example, the following typeout occurs by responding to the CONFIRM: message with H and a carriage return.

```
IN RESPONSE TO CONFIRM:,TYPE ONE OF: BDFHIKLPQSUWX
B TO PERFORM ALGORITHM TO GET BELOW LOGGED OUT QUOTA
D TO DELETE ALL FILES
(ASKS ARE YOU SURE?, TYPE Y OR CR)
F TO TRY TO LOGOUT FAST BY LEAVING ALL FILES ON DSK
H TO TYPE THIS TEXT
I TO INDIVIDUALLY DETERMINE WHAT TO DO WITH ALL FILES
  AFTER EACH FILE NAME IS TYPED OUT, TYPE ONE OF: EK PQS
  E TO SKIP TO NEXT FILE STRUCTURE AND SAVE THIS FILE IF
  BELOW LOGGED OUT QUOTA ON THIS FILE STRUCTURE
  K TO DELETE THE FILE
  P TO PRESERVE THE FILE
  Q TO REPORT IF STILL OVER LOGGED OUT QUOTA, THEN REPEAT FILE
  S TO SAVE THE FILE WITH PRESENT PROTECTION
K TO DELETE ALL UNPRESERVED FILES
L TO LIST ALL FILES
P TO PRESERVE ALL EXCEPT TEMP FILES
Q TO REPORT IF OVER LOGGED OUT QUOTA
S TO SAVE ALL EXCEPT TEMP FILES
U SAME AS I BUT AUTOMATICALLY PRESERVE FILES ALREADY PRESERVED
W TO LIST FILES WHEN DELETED
X TO SUPPRESS LISTING FILES WHEN DELETED
```

IF A LETTER IS FOLLOWED BY A SPACE AND A LIST OF FILE STRUCTURES  
ONLY THOSE SPECIFIED WILL BE AFFECTED BY THE COMMAND. ALSO  
CONFIRM WILL BE TYPED AGAIN.

NOTE: FILE SIZE IS NO. OF BLOCKS ALLOCATED WHICH MAY BE LARGER THAN THE  
NO. OF BLOCKS WRITTEN (DIRECTORY COMMAND).

A FILE IS PRESERVED IF ITS ACCESS CODE IS GE 100

CONFIRM:

You may now use the options available. If K was used as the option, the following is a sample of what is output to your terminal.

```
JOB 33, USER [27,560] LOGGED OFF TTY34 1317 20-MAY-71  
DELETED ALL 2 FILES (3. DISK BLOCKS)  
RUNTIME 0 MIN, 00.29 SEC
```

Remember that the CONFIRM message is typed only if there are files on your disk area. If there are no files on your disk area, the typeout would look like the following:

```
.KJOB )  
JOB 17, USER [27,3201] LOGGED OFF TTY17 1317 20-MAY-71  
RUNTIME 0 MIN, 00.29 SEC
```

## 9.0 HOW TO LIVE WITH THE TERMINAL

On the terminal, there is a special key marked CTRL called the Control Key. If this key is held down and a character key is depressed, the terminal types what is known as a control character rather than the character printed on the key. In this way, more characters can be used than there are keys on the keyboard. Most of the control characters do not print on the terminal, but cause special functions to occur, as described in the following sections.

There are several other special keys that are recognized by the system. The system constantly monitors the typed characters and, most of the time, sends the characters to the program being executed. The important characters not passed to the program are also explained in the following sections. (Refer to DECsystem-10 Monitor Calls for more explanations of special characters.)

### 9.1 Control - C

Control - C (↑C) interrupts the program that is currently running and takes you back to the monitor. The monitor responds to a control - C by typing a period on your terminal, and you may then type another monitor command. For example, suppose you are running a program in BASIC, and you now decide you want to leave BASIC and run a program in AID. When BASIC requests input from your terminal by typing an asterisk, type control - C to terminate BASIC and return to the monitor. You may now issue a command to the monitor to initialize AID (.R AID). If the program is not requesting input from your terminal (i.e., the program is in the middle of execution) when you type control - C, the program is not stopped immediately. In this case, type control - C twice in a row to stop the execution of the program and return control to the monitor. If you wish to continue at the same place that the program was interrupted, type the monitor command CONTINUE. As an example, suppose you want the computer to add a million numbers and to print the square root of the sum. Since you are charged by the amount of processing time your program uses, you want to make sure your program does

not take an unreasonable amount of processing time to run. Therefore, after the computer has begun execution of your program, type control - C twice to interrupt your program. You are now communicating with the monitor and may issue the monitor command TIME to find out how long your program has been running. If you wish to continue your program, type CONTINUE and the computer begins where it was interrupted.

### 9.2 The RETURN Key

This key causes two operations to be performed: (1) a carriage-return and (2) an automatic line-feed. This means that the typing element returns to the beginning of the line (carriage-return) and that the paper is advanced one line (line-feed). Commands to the monitor are terminated by depressing this key.

### 9.3 The RUBOUT Key

The RUBOUT key permits correction of typing errors. Depressing this key once causes the last character typed to be deleted. Depressing the key n times causes the last n characters typed to be deleted. RUBOUT does not delete characters beyond the previous carriage-return, line-feed, or altmode. Nor does RUBOUT function if the program has already processed the characters you wish to delete.

The monitor types the deleted characters, delimited by backslashes. For example, if you were typing CREATE and go as far as CRAT, you can correct the error by typing two RUBOUTS and then the correct letters. The typeout would be

```
CRAT\TA\EATE
```

Notice that you typed only two RUBOUTS, but \TA\ was printed. This shows the deleted characters, but in reverse order. (Note that when using TECO, deleted characters are not enclosed in backslashes.)

### 9.4 Control - U

Control - U ( $\uparrow$ U) is used if you have completely mistyped the current line and wish to start over again. Once you type a carriage-return, the command is read by the computer, and line-editing features can no longer be used on that line. Control - U causes the deletion of the entire line, back to the last carriage-return, line-feed, or altmode. The system responds with a carriage-return, line-feed so you may start again.

### 9.5 The ALTMODE Key

The ALTMODE key, which is labeled ALTMODE, ESC, or PREFIX, is used as a command terminator for several programs, including TECO and LINED. Since the ALTMODE is a nonprinting character, the terminal prints an ALTMODE as a dollar sign (\$).

### 9.6 Control - O

Control - O (tO) tells the computer to suppress terminal output. For example, if you issue a command to type out 100 lines of text and then decide that you do not want the typeout, type control - O to stop the output. Another command may then be typed as if the typeout had terminated normally.

## 10.0 PERIPHERAL DEVICES

The system controls many peripheral devices, such as terminals, magnetic tape drives, DECtape drives, card readers and punches, line printers, papertape readers and punches, and disks. The monitor is responsible both for allocating these peripheral devices, as well as other system resources (e.g., core memory), and for maintaining a pool of such available resources from which you can draw.

Each device controlled by the system has a physical name associated with it. The physical name is unique. It consists of three letters and zero to three numerals specifying a unit number. The following table lists the physical names associated with various peripheral devices.

Table 1  
Peripheral Devices

| Device            | Physical Name          |
|-------------------|------------------------|
| Terminal          | TTY0, TTY1, ..., TTY77 |
| Console TTY       | CTY                    |
| Paper Tape Reader | PTR                    |
| Paper Tape Punch  | PTP                    |
| Plotter           | PLT                    |
| Line Printer      | LPT                    |
| Card Reader       | CDR                    |
| Card Punch        | CDP                    |
| DECtape           | DTA0, DTA1, ..., DTA7  |
| Magnetic Tape     | MTA0, MTA1, ..., MTA7  |
| Disk              | DSK                    |
| Display           | DIS                    |

You may also give each device a logical device name. The logical device name is an alias, and the device can be referred to either by this alias or by the physical name. The logical name consists of one to six alphanumeric characters of your choice. The reason for logical device names is that in writing a program you may use arbitrarily selected device names (logical device names) that can be assigned to the most convenient physical devices at runtime. However, care should be exercised in assigning logical device names because these names have priority over physical device names. For example, if a DECTape is assigned the logical name DSK, then all of your programs attempting to use the disk via the physical name DSK end up using the DECTape instead. It is wise not to give any device the logical name DSK because certain monitor commands (such as the COMPILE commands) make extensive use of special features that the disk has but other devices do not have. The following examples show the use of logical and physical device names.

|                                  |   |
|----------------------------------|---|
| <code>._ASSIGN DTA ABC )</code>  | Assign a DECTape the logical name ABC.              |
| <code>._ASSIGN MTA1 XYZ )</code> | Assign magnetic tape drive #1 the logical name XYZ. |
| <code>._ASSIGN PTR FOO )</code>  | Assign the papertape reader the logical name FOO.   |

In order to use most peripheral devices, you must assign the desired device to your job. You may assign a device either by a program or from the console. The first kind of assignment occurs when you write a program that uses a particular device. When the program begins using the device, it is assigned to you on a temporary basis and released from you when your program has finished with it. The second kind of assignment occurs when you explicitly assign the device by means of the ASSIGN or MOUNT monitor command. This is a permanent assignment that is in effect until the device is released by a DEASSIGN, DISMOUNT, or FINISH monitor command or by your logging off the system.

When you assign a device to your job, the monitor associates your job number with that device. This means that no other user may use the device while you are using it. The only exception is the disk, which is accessible by all users. When you assign the disk, you are allocated a fraction of the disk, not the entire unit. When you deassign a device or kill your job, the device is returned to the monitor's pool of available resources.

Under normal circumstances, the spooling mechanism built into the system is used to output to slow-speed devices. Spooling is the method by which output to these devices (usually the line printer, card punch, paper tape punch, and plotter) is placed on the disk first and then output to the device at a later time. This method of using a device saves you time because you do not have to wait for the device to be freed if it is being used by another user nor do you have to wait for your files to be output before you

can perform another operation. Once your files have been placed on the disk, you can do another task, such as running a program or leaving the system by killing your job. After you leave the system (KJOB), your files will be output whenever the device you requested to output them is available. The spooling of files to the line printer is described in Paragraph 13.0. Refer to the DECsystem-10 Operating System Commands manual for a discussion of spooling to other devices.

## 11.0 COMMANDS TO ALLOCATE SYSTEM RESOURCES

### 11.1 The ASSIGN Command

The ASSIGN command is used to assign a peripheral device on a permanent basis for the duration of your job or until you explicitly deassign it. This command must have as an argument the legal physical device name (see Table 1) of the device you wish to assign. For example, if you want to assign a DECtape drive to your job, type

```
._ASSIGN DTA n )
```

The monitor responds with the message

```
DTA n ASSIGNED
```

where n is the unit number of the DECtape drive assigned to your job. If all drives are in use, the monitor responds with

```
ASSIGNED TO JOBS N1, N2, ...
```

and you must wait until a drive becomes available. You may assign a specific DECtape drive as follows:

```
._ASSIGN DTA3 )
```

The monitor responds with

```
DTA3 ASSIGNED
```

if the drive is available, or

```
ALREADY ASSIGNED TO JOB n
```

if job n is using DECtape drive #3.

The ASSIGN command may also have, as an optional argument, a logical device name following the physical device name. The logical device name may be used in place of the physical device name in all references to the device. For example, if you want to use DECtape drive #1 and have it named SAMPLE, type the command

```
._ASSIGN DTA1 SAMPLE )
```

If DECtape drive #1 is free, the monitor responds with

```
DTA1 ASSIGNED
```

and stores the logical name you typed. You may then refer to the DECTape by the name SAMPLE until you explicitly release the device, assign the name SAMPLE to another device, or kill your job.

Logical names can be very useful. Suppose you write a program that uses DECTape drive #5 and refers to it by its physical name (DTA5). When you run your program, you find that DECTape drive #3 is the only drive available. Instead of rewriting your program to use DECTape drive #3, type

```
._ASSIGN DTA3 DTA5 )
```

Thereafter, whenever your program refers to DTA5, it is actually referring to DTA3. Since logical device names are strictly your own, they are different from the logical names of other users. The following is an example using physical and logical device names.

|   |   |
|---|---|
| ._ASSIGN DTA NAME )                                       | Assign a DECTape drive the logical name NAME.               |
| <u>DEVICE DTA4 ASSIGNED</u>                               | DECTape drive #4 has been assigned.                         |
| ._ASSIGN DTA LINE )                                       | Find another DECTape drive; assign the logical name LINE.   |
| <u>ASSIGNED TO JOBS N<sub>1</sub>, N<sub>2</sub>, ...</u> | All DECTape drives are in use.                              |
| ._ASSIGN PTP,NAME )                                       | Reserve paper tape punch.                                   |
| <u>%LOGICAL NAME WAS IN USE,</u>                          |   |
| <u>DEVICE PTP ASSIGNED</u>                                | Paper tape punch is assigned and NAME now refers to PTP.    |
| ._ASSIGN DTA3 LINE )                                      | Request DECTape drive #3 and give it the logical name LINE. |
| <u>ALREADY ASSIGNED TO JOB7</u>                           | Another user (job 7) has DTA3.                              |

## 11.2 The MOUNT Command

The MOUNT command is similar to the ASSIGN command in that it is used to assign a peripheral device to your job. However, unlike the ASSIGN command, it requests operator intervention. This is useful for users who cannot place their devices on the computer because they are too far away. These users are called remote users because they are connected to the computer via communications lines. For example, if you have DECTapes at the location of the computer (commonly called the central site) but are using the computer remotely, you can use the MOUNT command to assign a DECTape drive and to have the operator place the DECTape on the drive.

This command must have as an argument the legal physical device name (see Table 1) of the device you wish to assign and may have a logical device name. These arguments are the same as in the ASSIGN command. In addition, switches can be used to specify items to be considered by the operator. Only the following three switches are applicable in this manual; the remainder are described in

### DECsystem-10 Operating System Commands

|                  |  |
|------------------|--|
| /RONLY or /WLOCK | Specifies that the volume is read only and that it cannot be written on. |
|------------------|--|

|                        |   |
|------------------------|---|
| <code>/VID:name</code> | Specifies the name used to identify the volume (storage medium) to the operator. The name can be in one of two forms:<br>1) any string of 25 characters or less containing only letters, digits, periods, and hyphens or 2) any string of 25 characters or less enclosed in single quotes. The string cannot contain break characters or single quotes. |
| <code>/WENABL</code>   | Specifies that the volume is enabled for writing. This condition is assumed if no switches appear in the MOUNT command string.  |

### 11.3 The DEASSIGN Command

The DEASSIGN command is used to release one or more devices currently associated with your job. This command may have as an argument a physical or logical device name. If an argument is given, the specified devices are released. If an argument is not specified, all devices assigned to your job are released. When devices are released, they are returned to the monitor's pool of available resources for use by other users. The DEASSIGN command does not affect any temporary assignments your job may have for devices.

### 11.4 The DISMOUNT Command

The DISMOUNT command is similar to the DEASSIGN command because it is used to return devices to the monitor. In addition, it notifies the operator to remove the volume (storage medium) from the device (i.e., DECtape from a DECtape drive, cards from a card reader, and so forth). This command takes a physical device name as an argument. The device must have been previously assigned with the ASSIGN or MOUNT command. The switch /REMOVE follows the device name in order to tell the operator to physically remove the volume from the device. For example,

```
•DISMOUNT DTA4:/REMOVE )
```

notifies the operator to deassign DTA4 and remove the tape from the drive.

### 11.5 The REASSIGN Command

The REASSIGN command allows you to give a device assigned to you to another user without having the device returned to the monitor's pool of available resources. Two arguments are required with this command: the name of the device being reassigned and the job number of the user who is receiving the device. For example, suppose you have finished with DECtape drive #6 and the person who is job 10 wants it. Type the command

```
•REASSIGN DTA6 10 )
```

This deassigns DECtape drive #6 from your job and assigns it to job 10, just as if you had typed

```
•DEASSIGN DTA6 )
```

and job 10 had typed

```
._ASSIGN DTA6 )
```

immediately thereafter. All devices except the job's terminal can be reassigned.

### 11.6 The FINISH Command

The FINISH command is used to prematurely terminate a program that is being executed while preserving as much output as possible. If this command is not used, part or all of the output file may be lost. The FINISH command may be followed by a physical or logical device name, in which case any input or output currently in progress in relation to that device is terminated. If no device is specified, input or output is terminated on all devices assigned to your job. The monitor responds to this command by terminating output, closing the file, and releasing the device for use by others.

This command could be used if you were generating an assembly listing of a program on your disk area and decided that you wanted only the first part of the listing, not the entire listing. Type

```
↑C
._FINISH DSK )
```

and the monitor completes the writing of your listing and releases the disk.

### 11.7 The CORE Command

The CORE command allows you to modify the amount of core assigned to your job. The command is followed by a decimal number representing the total number of 1K blocks (1024 word blocks) that you want the program to have from this point on. For example, if you want the program to have 8K blocks of core, type

```
._CORE 8 )
```

and the monitor gives the program 8K blocks, if available. If you request additional core and there is none available, the monitor responds with an error message. If the CORE command is followed by the decimal number 0, your program disappears from core because you are requesting 0K blocks of core.

If the decimal number following the command is omitted, the monitor types out (1) the total number of 1K blocks you have, (2) the maximum you can request, and (3) the amount of core not assigned to any user.

## 12.0 COMMANDS TO MANIPULATE TERMINALS

### 12.1 The SEND Command

The SEND command allows you to send a line of text to another terminal in the system. The command is typed followed by the number of the terminal to which you are sending the message followed by the message and a carriage return. This message is printed on the receiving terminal and is preceded by

the number of your terminal. If the receiver of the message is busy, that is, his terminal is not communicating with the monitor, you receive the message BUSY and your message is not sent. If you are sending a message to an operator, the receiving terminal is never busy.

### 12.2 The DETACH Command

The DETACH command causes your terminal to be disconnected from your program and released to control another job. This means that, while your program is disconnected, you may log in again, receive a new job number, and do something else. The job that was disassociated from your terminal is said to be a detached job. This means that it is not under control of any user's console. If your detached job attempts to type something to the terminal, it is stopped, for there is no terminal attached to it.

### 12.3 The ATTACH Command

The ATTACH command allows you to attach a console to a detached job. You must specify the number of the job to which you wish to attach. If you are the owner of the detached job, your console is immediately detached from your current job and attached to your detached job. After this command is executed, the console is in communication with the monitor. If the job you just attached to happens to be running, type CONTINUE without affecting the status of the job.

If you are not the owner of the detached job, you must also specify the project-programmer number of the owner. The project-programmer number must be enclosed in square brackets (e.g., [27,400]) for this command to work. If the job whose job number you typed is already attached to a terminal, you cannot attach and the monitor responds with

?TTYn ALREADY ATTACHED

where n is the number of the terminal attached to the job. Observe that only one terminal can be attached to a job at any time.

## 13.0 COMMANDS TO REQUEST LINE PRINTER OUTPUT

In Paragraph 5.2, the TYPE command for listing source files on your terminal was discussed. In addition, there are three commands that may be used to list files on the line printer via the spooling mechanism.

### 13.1 The PRINT Command

The PRINT command is used to list disk files on the line printer via the spooling mechanism. This command takes a filename, or many filenames separated by commas, as an argument. Switches can also be used with the PRINT command. Although many switches are available, only a few pertinent ones are mentioned below. The remainder are discussed in DECsystem-10 Operating System Commands.

/COPIES:n

Specifies the number of copies that you want of the file. This number must be less than 64. If this switch is not given, one copy is produced.

|  |  |
|--|--|
| <code>/LIMIT:n</code>  | Specifies the maximum number of pages you want printed. If this switch is not given, the maximum number is 200 pages.                              |
| <code>/SPACING:DOUBLE</code><br><code>/SPACING:SINGLE</code><br><code>/SPACING:TRIPLE</code> | Specifies that the output will be double, single, or triple spaced. If the <code>/SPACING</code> switch is not given, the output is single-spaced. |

All files remain in your disk area except for temporary files; these files are deleted after they are printed.

### 13.2 The CREF Command

The CREF command is used to list a certain type of file called a cross-reference file. This command is an invaluable aid in program debugging. If a `COMPILE`, `LOAD EXECUTE`, or `DEBUG` command string (refer to Paragraph 6.0) has a `/CREF` switch, the command string generates an expanded listing that includes (1) the original code as it appears in the file, (2) the octal values the code represents, (3) the relative locations into which the octal values go, (4) a list of all the symbols your program uses, and (5) the numbers of the lines on which each symbol appears. This is called a cross-reference listing. To print this listing file, you must call in a special cross-reference lister with the CREF command. All the cross-reference listing files you have generated since the last CREF command are printed on the line printer. The file containing the names of the cross-reference listing files is then deleted so that subsequent CREF commands will not list them again.

### 13.3 The DIRECT Command

When a `DTAn:` argument is specified with the `DIRECT` command, the directory of DECTape n is typed on the terminal. (Refer to Paragraph 5.1 for a discussion of the `DIRECT` command when no argument is specified.) For example, the command

```
._DIRECTORY DTA2:.)
```

types the directory of DECTape drive #2 on the terminal.

Besides having optional device arguments, this command has several switch options. One switch option is `/F`. Including `/F` in the command string causes the short form of the directory to be listed on the terminal. The short form of the directory consists of the names of your files. (The long form of the directory also lists the creation dates and lengths of each file.) Another switch option is `/L`. Including `/L` in the command string causes the output of the directory to go to the line printer rather than to the terminal. For example, the command

```
._DIRECTORY /L )
```

lists your directory of your disk area on the line printer. The line printer is assigned to you on a temporary basis and is released when the output is finished.

## 14.0 COMMANDS TO MANIPULATE CORE IMAGES

By using one of the following commands, you can load core image files (refer to Paragraph 6.1 for the definition of a core image file) from disk, DECtapes, and magnetic tapes into core and then later save the core images. These files can be retrieved and controlled from the user's console. Files on disk and DECtape are called by filename, and if you have any files on magnetic tape, you must position the tape to the beginning of the file.

### 14.1 The SAVE Command

The SAVE command causes your current core image to be saved on the specified device with the specified filename. This command must be followed by several arguments. First, you must tell the monitor the device on which you want to save the core image. A colon must follow the device name. Second, you must give a name to the core image file. If the filename extension is not specified, the monitor designates one. You may specify the amount of core in which you want your file saved by specifying a decimal number to represent the number of 1K blocks. For example, if you want to save your core image on DECtape drive #2, give it the name SALES, and allow 12K of core for storage, type

```
._SAVE DTA2: SALES 12 )
```

A file called SALES is created and your core image is stored in it. If you list your DECtape directory, the length of the file is slightly over 12,000 words. After you use this command, you cannot continue executing the program. The program can be restarted only from the beginning.

### 14.2 The RUN Command

The RUN command allows you to run programs you previously saved on the disk, DECtape, or magnetic tape. This command reads the core image file from a storage device and starts its execution. You must specify the device containing the core image file and the name of that file. The file must have been saved previously with a SAVE command. If the file is not a saved program, the monitor responds with an error message. If the core image file you want to execute is on another user's disk area, you must specify his project-programmer number, enclosed in square brackets. Again, you may specify the amount of core to be assigned to the program if different from the minimum core needed to load the program or from the core argument of the SAVE command.

### 14.3 The R Command

The R command is a special form of the RUN command. This command runs programs that are part of the system, rather than programs that are your own. The R command is the usual way to run a system program that does not have a direct monitor command associated with it. For example, the only way to run BASIC and AID is by the commands

```
._R BASIC )
```

and

```
._R AID )
```

A device name or a project programmer number may not be specified for this command.

#### 14.4 The GET Command

The GET command is the same as the RUN command except that it does not start the program; it merely generates a core image and exits. The monitor types

```
._JOB SETUP
```

and is ready to accept another command.

### 15.0 COMMANDS TO START A PROGRAM

#### 15.1 The START Command

The START command begins execution of the program at its starting address, the location specified within the file, and is valid only if you have a core image. This command allows you to specify another starting address by typing the octal address after the command. Normally, to start a program, type

```
._START )
```

but to start a program at the specified octal location 347, type

```
._START 347 )
```

A GET command followed by a START command is equivalent to a RUN command.

#### 15.2 The HALT ( ↑ C) Command

Typing ↑C stops your program and takes you back to the monitor. The program "remembers" at what point it was interrupted so that it may subsequently be continued. After typing ↑C, you may type any commands that do not affect the status of your program (e.g., PJOB, DAYTIME, RESOURCES) and still be able to continue the execution of the program with a CONTINUE command. However, continuing is impossible if you issue any command that runs a new program, such as a RUN or R command.

#### 15.3 The CONTINUE Command

If you stop your program by a HALT (↑C) command, you may resume execution from the point at which it was interrupted by typing the CONTINUE command. You may continue the program only if you exit by typing control -C. If the program exited on an error condition of some sort, the monitor does not let you continue. It types

```
._CAN'T CONTINUE
```

if you try. However, you may continue your program if it has halted and given the typeout

HALT AT USER n

### 16.0 ADDITIONAL COMMANDS TO GET INFORMATION FROM THE SYSTEM

#### 16.1 The RESOURCES Command

The RESOURCES command types out a list of all the available devices (except terminals) on your terminal. For example,

```
.RESOURCES )  
PTY1,CDR,PTR,MTA1,CDP,PLT
```

At the time of this command, there were six devices available.

#### 16.2 The SYSTAT Command

The SYSTAT command produces a summary of the current status of the system and may be typed without logging in. Included in the summary is a list of the jobs currently logged in, along with their project-programmer numbers, program names being run, and runtime. The following typeout is a partial example of SYSTAT output. More information is contained in this program and can be obtained by running SYSTAT.

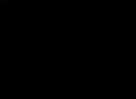
STATUS OF 5S0224D SYSTEM #2 AT 1:34:02 P.M. ON 11-MAY-71

UPTIME 5:10:56, 24% NULL TIME = 19% IDLE + 5% LOST  
22 JOBS IN USE OUT OF 37. 22 LOGGED IN, 1 DETACHED

| JOB | WHO    | LINE# | WHAT   | SIZE(K) | STATE | RUN TIME |
|-----|--------|-------|--------|---------|-------|----------|
| 1   | [OPR]  | P0    | OMOUNT | 2+4     | SL SW | 21       |
| 2   | [OPR]  | P1    | OMOUNT | 2+4     | SL SW | 22       |
| 3   | [OPR]  | P2    | CDRSTK | 2       | SL SW | 1:01     |
| 4   | [OPR]  | P3    | BATCON | 4+4     | SL SW | 47       |
| 5   | [OPR]  | P4    | LPTSPL | 3+4     | CB SW | 5:39     |
| 6   | [OPR]  | P5    | PTSPPL | 2+3     | SL SW | 41       |
| 7   | [OPR]  | P6    | CHKPNT | 2       | SL SW | 5        |
| 8   | [OPR]  | P7    | MSCOPE | 1+SPY   | SL    | 58:15 &  |
| 9   | [OPR]  | P10   | TYLOST | 2+5     | SL SW | 3        |
| 10  | 10,16  | 23    | DIRECT | 1+3     | +C SW | 2:31     |
| 11  | [OPR]  | 12    | SYSDPY | 3+SPY   | RN    | 45:09    |
| 12  | **,**  | DET   | DAEMON | 7+SPY   | SL SW | 1        |
| 13  | [OPR]  | CTY   | OPSER  | 1+2     | SL SW | 25       |
| 14  | 20,574 | 1     | DIRECT | 1+3     | TI SW | 13       |
| 15  | 40,65  | 21    | TECO   | 2+3     | TI SW | 20       |
| 16  | 10,566 | 3     | BATCON | 0+4     | CB SW | 4:17     |
| 17  | 11,131 | 11    | DIRECT | 1+3     | +C SW | 4        |
| 18  | 10,77  | 20    | MONLOD | 12+2    | RN SW | 4:59     |
| 19  | [OPR]  | 2     | FAILSA | 10      | WS    | 16       |
| 20  | 10,63  | 0     | FD5224 | 23      | TI SW | 3        |
| 21  | [SELF] | 26    | SYSTAT | 4+SPY   | RN    | 4        |
| 22  | 10,34  | 24    | KJOB   | 6+4     | RN    | 3        |

& MEANS LOCKED IN CORE  
PNN CORRESPONDS TO TTY42+NN





**dec**system10  
BEGINNER'S GUIDE TO  
MULTIPROGRAM BATCH

Copyright © 1972 by Digital Equipment Corporation

The material in this manual is for informational purposes and is subject to change without notice.

The following are trademarks of Digital Equipment Corporation, Maynard, Massachusetts:

|           |              |
|-----------|--------------|
| DEC       | PDP          |
| FLIP CHIP | FOCAL        |
| DIGITAL   | COMPUTER LAB |

## FOREWORD

The Beginner's Guide to Multiprogram Batch has been written for the inexperienced or casual user who has little knowledge of programming techniques and who requires only a rudimentary knowledge of Batch operations.

### HOW TO USE THIS MANUAL

For those users whose mode of input is cards, the following chapters or sections of chapters should be read.

- Chapter 1    Introduction
- Chapter 2    Entering a Job to Batch from Cards
- Chapter 4    Interpreting Your Printed Output
- Chapter 5, Section 5.2    Using Cards to Enter Jobs

According to the language in which his program is written, the user should pay particular attention to the following sections.

- FORTRAN    -    Section 2.2.3 Card Deck to Run FORTRAN Programs
- ALGOL        -    Section 2.2.1 Card Deck to Run ALGOL Programs
- COBOL        -    Section 2.2.2 Card Deck to Run COBOL Programs
- MACRO        -    Section 2.2.4 Card Deck to Run MACRO Programs
- BASIC        -    Section 2.3.1 Card Decks for Programs That Do Not Have Special Control Cards

For users who input their jobs through interactive terminals, the following chapters or sections of chapters are recommended.

- Chapter 1    Introduction
- Chapter 3    Entering a Job to Batch from a Terminal
- Chapter 4    Interpreting Your Printed Output
- Chapter 5, Section 5.1    Using the Terminal to Enter Jobs

## REFERENCES

Not all of the commands and cards for Batch are described in this manual. Those users who wish to know more about Multiprogram Batch can refer to Chapter 3 in the DECsystem-10 Operating System Commands manual. Also in that manual, the SUBMIT command is described in Chapter 2.

An elementary description of the basic monitor commands can be found in the document Getting Started with Timesharing. The DECsystem-10 Operating System Commands manual contains the descriptions of all the monitor commands available to the user.

Error messages from the system programs supplied by DEC that are invoked by the user's job are explained in the applicable manuals. For example, if a user's FORTRAN program fails to compile successfully, the error messages he receives from the FORTRAN compiler can be found in Chapter 11 of the FORTRAN IV Programmer's Reference Manual in the DECsystem-10 Mathematical Languages Handbook.

### CONVENTIONS USED IN THIS MANUAL

The following is a list of symbols and conventions used in this manual.

|                |  |
|----------------|--|
| dd-mmm-yy hhmm | A set of numbers or numbers and a word that indicates the date and time, e.g., 15-5-72 1415 or 15-MAY-72 1415 means 2:15 PM on May 15, 1972.   |
| filename.ext   | The name and extension that can be put on a file. The name can be 1 to 6 characters in length and the extension can be 1 to 4 characters in length. The first character of the extension must always be a period. The extension is optional. Refer to the glossary for definitions of filename and filename extension. |
| hh:mm:ss       | A set of numbers representing time in the form hours:minutes:seconds. Leading zeros can be omitted, but colons must be present between two numbers. For example, 5:35:20 means five hours, 35 minutes, and 20 seconds.   |
| jobname        | The name that is assigned to a job. It can contain up to 6 characters. Refer to the glossary for the definition of a job.  |
| [proj,prog]    | The user number assigned to each user, commonly called a project-programmer number. It must be enclosed in square brackets. The two numbers that make up the project-programmer number must be separated by a comma or a slash. Refer to the glossary for the definition of a project-programmer number.               |
| n              | A number that specifies either a required number or an amount of things such as cards or line-printer pages. This number can contain as many digits as are necessary to specify the amount required, e.g., 5, 25, 125, etc.  |
| t              | A number representing an amount of time usually in minutes. This number can contain as many digits as are necessary to specify the amount of time required, e.g., 5, 25, 125, etc.   |

## GLOSSARY

| Term              | Definition  |
|-------------------|---|
| ALGOL             | ALGO <sup>r</sup> ithmic Language. A scientific oriented language that contains a complete syntax for describing computational algorithms.  |
| Alphanumeric      | The characters which include the letter of the alphabet (A through Z), the numerals (0 through 9), and letters of the other special symbols such as -, /, *, \$, ., (, ), +.  |
| ASCII Code        | American Standard Code for Information Interchange. A 7-bit code in which information is recorded.  |
| Assemble          | To prepare a machine-language program from a symbolic-language program by substituting absolute operation codes for symbolic operation codes and absolute or relocatable addresses for symbolic addresses.  |
| Assembler         | A program which accepts symbolic code and translates it into machine instruction, item by item. The assembler on the DECsystem-10 is called the MACRO assembler.  |
| Assembly Language | The machine-oriented symbolic programming language belonging to an assembly system. The assembly language for the DEC-system-10 is MACRO.   |
| Assembly Listing  | A printed list which is the byproduct of an assembly run. It lists in logical-instruction sequence all details of a routine showing the coded and symbolic notation next to the actual assigned notations established by the assembly procedure.  |
| BASIC             | Beginner's All-purpose Symbolic Instruction Code. A time-sharing computer programming language that is used for direct communication between teletype units and remotely located computer centers. The language is similar to FORTRAN II and was developed by Dartmouth College.                              |
| Batch processing  | The technique of executing a set of computer programs in an unattended mode.  |
| Card              | A punch card with 80 vertical columns representing 80 characters. Each column is divided into two sections one with character positions labeled zero through nine, and the other labeled eleven (11) and twelve (12). The 11 and 12 positions are also referred to as the X and Y zone punches, respectively. |

GLOSSARY (Cont)

| Term                          | Definition   |
|-------------------------------|--|
| Card Column                   | One of the vertical lines of punching positions on a punched card.   |
| Card Field                    | A fixed number of consecutive card columns assigned to a unit of information.  |
| Card Row                      | One of the horizontal lines of punching positions on a punched card.   |
| Central processing unit (CPU) | The portion of the computer that contains the arithmetic, logical, control circuits, and I/O interface of the basic system.  |
| Central Site                  | The location of the central computer. Used in conjunction with remote communications to mean the location of the DECsystem-10 central processor.   |
| Character                     | One symbol of a set of elementary symbols such as those corresponding to the keys on a typewriter. The symbols usually include the decimal digits 0 through 9, the letters A through Z, punctuation marks, operation symbols, and any other special symbols which a computer may read, store, or write.    |
| COBOL                         | COmmon Business Oriented Language. An automatic programming language used in programming data processing applications.   |
| Command                       | The part of an instruction that causes the computer to execute a specified operation.  |
| Compile                       | To produce a machine or intermediate language routine from a routine written in a high level source language.  |
| Compiler                      | A programming system which translates a high level source language into a language suitable for a particular machine. A compiler is a translator that converts a source language program into intermediate or machine language. Some compilers used on the DECsystem-10 are: ALGOL, BASIC, COBOL, FORTRAN. |
| Computer                      | A device with self-contained memory capable of accepting information, processing the information, and outputting results.  |
| Computer Operator             | A person who manipulates the controls of a computer and performs all operational functions that are required in a computing system, such as, loading a tape transport, placing cards in the input hopper, removing printouts from the printer rack, and so forth.  |
| Continuation Card             | A punched card which contains information that was started on a previous punched card.   |
| Control File                  | The file made by the user that directs Batch in the processing of his job.   |

## GLOSSARY (Cont)

| Term                    | Definition   |
|-------------------------|--|
| Core Storage            | A storage device normally used for main memory in a computer.  |
| CPU                     | See central processing unit.   |
| Cross Reference Listing | A printed listing that identifies all references of an assembled program to a specific label. This listing is provided immediately after a source program has been assembled.  |
| Data                    | A general term used to denote any or all facts, numbers, letters, and symbols, or facts that refer to or describe an object, idea, condition, situation, or other factors. It represents basic elements of information which can be processed or produced by a computer. |
| Debug                   | To locate and correct any mistakes in a computer program.  |
| Disk                    | A form of mass storage device in which information is stored in named files.   |
| Dump                    | A listing of all variables and their values, or a listing of the values of all locations in core.  |
| Execute                 | To interpret an instruction and perform the indicated operation(s).  |
| Extension               | See filename extension.  |
| File                    | An ordered collection of 36-bit words comprising computer instructions and/or data. A file can be of any length, limited only by the available space on the device and the user's maximum space allotment on that device.  |
| Filename                | A name of one to six alphanumeric characters chosen by the user to identify a file.  |
| Filename extension      | One to four alphanumeric characters usually chosen to describe the class of information in a file. The first character of the extension must always be a period.   |
| FORTRAN                 | FORmula TRANslator. A procedure oriented programming language that was designed for solving scientific type problems. The language is widely used in many areas of engineering, mathematics, physics, chemistry, biology, psychology, industry, military, and business.  |
| Job                     | The entire sequence of steps, from beginning to end, that the user initiates from his interactive terminal or card deck or that the operator initiates from his operator's console.  |

GLOSSARY (Cont)

| Term                      | Definition  |
|---------------------------|---|
| Jobstep                   | A serial or parallel sequence of processes invoked by a user to perform an operation.   |
| K                         | A symbol used to represent a thousand; for example, 32K is equivalent to 32,000.  |
| Label                     | A symbolic name used to identify a statement in the control file.   |
| Log File                  | A file into which Batch writes a record of a user's entire job. This file is printed as the final step in Batch's processing of a job.  |
| Monitor                   | The collection of programs which schedules and controls the operation of user and system programs.  |
| Monitor Command           | An instruction to the monitor to perform an operation.  |
| Mounting a device         | To request assignment of an I/O device via the operator.  |
| Multiprogramming          | A technique that allows scheduling in such a way that more than one job is in an executable state at any one time.  |
| Object Program            | The program which is the output of compilation or assembly. Often the object program is a machine language program ready for execution.   |
| Password                  | The word assigned to a user that, along with his user number (project-programmer number), identifies him uniquely to the system.  |
| Peripheral devices        | Any unit of equipment, distinct from the central processing unit, which can provide the system with outside communication.  |
| Project-programmer number | Two numbers separated by a comma, which, when considered as a unit, identify the user and his file storage area.  |
| Program                   | The complete plan for the solution of a problem, more specifically the complete sequence of machine instructions and routines necessary to solve a problem.   |
| Programming               | The science of translating a problem from its physical environment to a language that a computer can understand and obey. The process of planning the procedure for solving a problem. This may involve among other things the analysis of the problem, preparation of a flowchart, coding of the problem, establishing input-output formats, establishing testing and checkout procedures, allocation of storage, preparation of documentation, and supervision of the running of the program on a computer. |

## GLOSSARY (Cont)

| Term            | Definition   |
|-----------------|--|
| Queue           | A list of jobs to be scheduled or run according to system, operator, or user-assigned priorities. For example, the Batch input queue.  |
| Software        | The totality of programs and routines used to expend the capabilities of computers, such as compilers, assemblers, operational programs, service routines, utility routines, and subroutines.  |
| Source Deck     | A card deck comprising a computer program, in symbolic language.   |
| Source Language | The original form in which a program is prepared prior to processing by the computer.  |
| Source Program  | A computer program written in a language designed for ease of expression of a class of problems or procedures, by humans. A translator (assembler, compiler, or interpreter) is used to perform the mechanics of translating the source program into a machine language program that can be run on a computer. |
| Terminal        | A keyboard unit that is often used to enter information into a computer and to accept output from a computer. It is often used as a time-sharing terminal on a remotely located computer center.   |

CONTENTS

|           | Page   |
|-----------|--|
| CHAPTER 1 | INTRODUCTION   |
| 1.1       | What is Multiprogram Batch 99                                      |
| 1.2       | How to Use Batch 99  |
| 1.2.1     | Setting Up Your Job 99   |
| 1.2.2     | Running Your Job 100   |
| 1.2.3     | Receiving Your Output 100  |
| 1.2.4     | Recovering from Errors 100   |
| 1.3       | Summary 100  |
| CHAPTER 2 | ENTERING A JOB TO BATCH FROM CARDS                                 |
| 2.1       | Format of the Cards in Your Deck 103                               |
| 2.2       | Setting up Your Card Deck 104                                      |
| 2.2.1     | Card Deck to Run ALGOL Programs 105                                |
| 2.2.2     | Card Deck to Run COBOL Programs 106                                |
| 2.2.3     | Card Deck to Run FORTRAN Programs 106                              |
| 2.2.4     | Card Deck to Run MACRO Programs 107                                |
| 2.3       | Putting Commands into the Control File from Cards 108              |
| 2.3.1     | Card Decks for Programs that do not have Special Control Cards 109 |
| 2.4       | Control Cards for Batch (in Alphabetical Order) 111                |
| 2.4.1     | The \$ALGOL Card   |
| 2.4.2     | The \$COBOL Card 113   |
| 2.4.3     | The \$DATA Card 115  |
| 2.4.4     | The \$DECK Card 119  |
| 2.4.5     | The End-of-File Card 121   |
| 2.4.6     | The \$EOD Card 121   |
| 2.4.7     | The \$ERROR Card 122   |
| 2.4.8     | The \$FORTRAN Card 123   |
| 2.4.9     | The \$JOB Card 125   |
| 2.4.10    | The \$MACRO Card 127   |
| 2.4.11    | The \$NOERROR Card 129   |
| 2.4.12    | The \$PASSWORD Card 130  |
| 2.4.13    | The \$SEQUENCE Card 131  |
| 2.5       | Specifying Error Recovery in the Control File 131                  |
| CHAPTER 3 | ENTERING A JOB TO BATCH FROM A TERMINAL                            |
| 3.1       | Creating the Control File 137                                      |
| 3.1.1     | Format of Lines in the Control File 138                            |
| 3.2       | Submitting the Job to Batch 139                                    |
| 3.2.1     | Queue Operation Switches 140                                       |
| 3.2.2     | General Switches 141   |

## CONTENTS (Cont)

|           | Page  |     |
|-----------|---|-----|
| 3.2.3     | File-Control Switches                         | 143 |
| 3.2.4     | Examples of Submitting Jobs                   | 144 |
| 3.3       | Batch Commands (in Alphabetic Order)          | 146 |
| 3.3.1     | The .BACKTO Command                           | 146 |
| 3.3.2     | The .ERROR Command                            | 146 |
| 3.3.3     | The .GOTO Command                             | 147 |
| 3.3.4     | The .IF Command                               | 148 |
| 3.3.5     | The .NOERROR Command                          | 149 |
| 3.4       | Specifying Error Recovery in the Control File | 150 |
| CHAPTER 4 | INTERPRETING YOUR PRINTED OUTPUT              |     |
| 4.1       | Output from Your Job                          | 153 |
| 4.2       | Batch Output                                  | 153 |
| 4.3       | Other Printed Output                          | 154 |
| 4.4       | Sample Batch Output                           | 154 |
| 4.4.1     | Sample Output from a Job on Cards             | 154 |
| 4.4.2     | Sample Output from a Job from a Terminal      | 157 |
| 4.4.3     | Sample Dump                                   | 160 |
| CHAPTER 5 | PERFORMING COMMON TASKS WITH BATCH            | 169 |
| 5.1       | Using the Terminal to Enter Jobs              | 169 |
| 5.2       | Using Cards to Enter Jobs                     | 176 |

# CHAPTER 1

## INTRODUCTION

### 1.1 WHAT IS MULTIPROGRAM BATCH

Multiprogram Batch is a group of programs that allow you to submit a job to the DECsystem-10 on a leave-it basis. That is, you give the job to an operator (if on cards) or submit it directly to the computer (if from a timesharing terminal) so that you can do something else while your job is running. A job is any combination of programs, their associated data, and commands necessary to control the programs.

Some of the jobs that are commonly processed under Batch are those that:

1. Are frequently run for production,
2. Are large and long running,
3. Require large amounts of data, or
4. Need no actions by you when they are running.

### 1.2 HOW TO USE BATCH

Batch allows you to submit your job to the computer through either an operator or a timesharing terminal, and receive your output from the operator when the job has finished. Output is never returned at a timesharing terminal even if your job is entered from one; instead, it is sent to a peripheral device (normally the line printer) at the computer site and returned to you in the manner designated by the installation manager.

#### 1.2.1 Setting Up Your Job

You must make up a control file to use Batch. A control file is a list of commands for the monitor, system programs, or Batch itself that tells Batch what steps to follow to process your job and the order in which to process them. When you enter your job on cards, you can take advantage of the special control cards that cause Batch to insert commands into the control file for you. When you enter your job from a timesharing terminal, you must put all the commands for your job into the control file yourself. The steps that you must take to create a control file from cards are described in Chapter 2. Creating a control file from a timesharing terminal is described in Chapter 3.

### 1.2.2 Running Your Job

After you submit the job, it waits in a queue with other jobs until Batch schedules it to run under guidelines established by the installation manager. Some factors that affect how long your job waits in the queue are its size, the amount of core it needs, the amount of time that it will take to run it, and whether or not you have specified a certain deadline when you want it run. When the job is started, Batch reads the control file and performs the actions necessary to run the job. For example, Batch passes monitor commands to the monitor which performs the actions called for and passes commands to system programs so that their processing can be performed.

As each step in the control file is performed, Batch records it in a log file. For example, if a monitor command such as COMPILE is processed, Batch passes it to the monitor and writes it in the log file. The monitor response is also written in the log file. Any response from your job that would be written on the terminal during timesharing is written in the log file by Batch.

### 1.2.3 Receiving Your Output

When the job is completed successfully and output has been sent to all devices, Batch terminates the log file and has it printed. The output from your job and the log file are then returned to you. Output from your job can be in the form of line-printer listings, punched cards, punched paper tape, plots, DECTape, or magnetic tape. If the output is to a DECTape or magnetic tape, you must include commands in your job to mount these tapes so that your output can be written on them. This is also true if you have input to any of the programs in your job written on tape. If your output is to cards, paper tape, the plotter, or the line printer, you must specify to Batch the approximate amount of cards, paper tape, plotter time, or pages that you require. These restrictions are to help Batch restrain runaway programs. An example of using the MOUNT command in the control file to request mounting of tapes is shown in Chapter 5. The way that you specify the amounts of paper, cards, etc. is described in Chapter 2, "The \$JOB Card" and in Chapter 3, "Submitting Your Job."

### 1.2.4 Recovering from Errors

If an error occurs in your job, either from an error in your program or from an erroneous command in the control file, Batch writes the error message in the log file and usually terminates the job. In addition, if the error occurred in your program, Batch causes a dump to be taken of your area of core. You can, however, put commands in the control file so that Batch can help you recover from errors in your job and continue running. Error recovery from a card job is described in Chapter 2; from a job entered from a terminal, in Chapter 3. Dumps, along with other printed output from a Batch job, are described in Chapter 4.

## 1.3 SUMMARY

In summary, the steps that you must perform to enter a job to the computer through Batch are as follows:

1. Create a control file either from cards (refer to Chapter 2) or from a terminal (refer to Chapter 3).
2. Submit the job to Batch, either to the operator for a card job (Chapter 2) or directly to Batch for a terminal (Chapter 3).
3. Pick up your output and interpret it (refer to Chapter 4).

Some sample jobs that are run through Batch from cards and from a terminal are shown in Chapter 5.



## CHAPTER 2 ENTERING A JOB TO BATCH FROM CARDS

Batch runs your job by reading a control file that contains commands to the monitor, system programs, or Batch itself. You have to make up the control file, but Batch provides you with special control cards to help you make up control files for simple jobs. These control cards make it easy for you to submit your programs to the computer and to create your control file to run these programs. Most of these control cards cause Batch to insert commands into the control file and/or copy programs and data into disk files. Some are used to show the beginning of your job and to identify it; and one is used to indicate the end of it. Batch control cards are also available to help you recover from errors that may occur while your job is running. The following is a list of the control cards which are described in greater detail in Section 2.4.

|             |                |
|-------------|----------------|
| \$SEQUENCE  | Section 2.4.12 |
| \$JOB       | Section 2.4.9  |
| \$PASSWORD  | Section 2.4.12 |
| \$ALGOL     | Section 2.4.1  |
| \$COBOL     | Section 2.4.2  |
| \$FORTRAN   | Section 2.4.8  |
| \$MACRO     | Section 2.4.10 |
| \$DECK      | Section 2.4.4  |
| \$DATA      | Section 2.4.3  |
| \$EOD       | Section 2.4.6  |
| \$ERROR     | Section 2.4.7  |
| \$NOERROR   | Section 2.4.11 |
| end-of-file | Section 2.4.5  |

### 2.1 FORMAT OF THE CARDS IN YOUR DECK

The card decks that you input to Batch can contain any combination of Batch control cards; commands to the monitor, system programs, and Batch itself; programs and data that will be copied into separate disk files; and data that will be copied into the control file for your program to read.

The Batch control cards must contain a dollar sign (\$) in column 1 and a command that starts in column 2. The command must be followed by at least one space, which can then be followed by the other information on the card. The end-of-file is the only exception to this format; it is identified by special punches in columns 1 and 80. Refer to the individual description of each card for any special format requirements.

If you include a card with a monitor command, you must place a period in column 1 and follow it immediately with the command. Any information that follows the command is in the format that is shown for the command in the DECsystem-10 Operating System Commands manual.

To include a command to a system program on a card, you must punch an asterisk (\*) in column 1 and punch the command string immediately following the asterisk. Refer to the manual for the system program that you wish to use.

Batch commands are punched like monitor commands; that is, a period is punched in column 1 and the command immediately follows the period.

The card format for your program depends on the language in which you have written the program; refer to the reference manual for the programming language that you are using for the format of each line of your program. The same is true for your data. The format that is required for the data by the programming language that you are using is described in the language reference manual.

If you want to include data for your program in the control file, you punch it as you would data that is read from a separate file. The only restriction on data in the control file is that alphabetic data that is preceded by a dollar sign (\$) must be preceded by an additional dollar sign so that Batch will not interpret such data as its own control commands.

If you put any special characters other than those described above in the first column of a card, you may get unexpected results because Batch interprets other special characters in special ways. If you want to know about other special characters, refer to the DECsystem-10 Operation System Commands manual, Chapter 3.

If you have more information than will fit on one card, you can continue on the next card by placing a hyphen (-) as the last nonspace character on the card to be continued and the rest of the information on the next card.

Comments can also be included either as separate cards or on cards containing other information.

To include a comment on a separate card, you must punch a semicolon (;) in column 1 and follow it immediately with the comment. To add a comment to a card, you must precede the comment with a semicolon (;) after all the information that you need has been put on the card. Comments that are on separate cards will normally be copied by Batch into your control file and later copied into your log file.

## 2.2 SETTING UP YOUR CARD DECK

Since the most common tasks performed in a job are compilation and execution of one or more programs, simple control cards are available that will cause Batch to insert commands into the control file for these tasks. However, a Batch job can do anything a timesharing job can do and if you wish to perform more complicated tasks, you will have to include monitor commands in your deck to direct

Batch to execute your tasks. The way in which you include monitor commands and also commands to other system programs is described in Section 2.3.

The control cards that you can use to compile and execute programs written in ALGOL, COBOL, FORTRAN, and MACRO are shown in sections 2.2.1, 2.2.2, 2.2.3, and 2.2.4. Certain control cards are always required in a Batch job. Others are required only at some installations. The \$JOB card and the end-of-file card are always required. The \$SEQUENCE and \$PASSWORD cards may be required, depending on the installation.

If the \$SEQUENCE card is required, it must be the first card in the deck. The \$JOB card must always be either the second card in the deck if the \$SEQUENCE card is required, or the first card in the deck if the \$SEQUENCE card is not required. If it is required, the \$PASSWORD card must immediately follow the \$JOB card. It will be assumed in this manual that the \$SEQUENCE and the \$PASSWORD cards are required. The end-of-file card must be the last card in the deck to indicate to Batch that it has read the end of your job. This end-of-file card is only used to end your entire job, not to end individual files in your job.

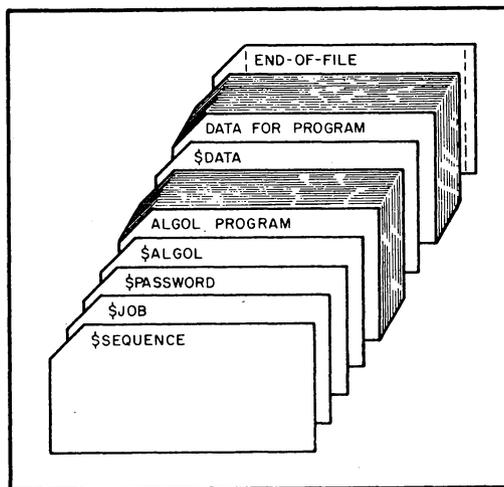
The cards that come between the first and last cards constitute your job. Setting up decks for specific languages is shown in the sections that follow.

### 2.2.1 Card Deck to Run ALGOL Programs

To run ALGOL programs, you use the \$ALGOL and \$DATA cards. You put a \$ALGOL card in front of your ALGOL program to make Batch copy your program into a disk file and insert a COMPILE command into your control file. The \$ALGOL card is described in detail in Section 2.4.1.

You put a \$DATA card in front of the data that goes with the program to make Batch copy your data into another disk file and insert an EXECUTE command into your control file. The \$DATA card is described in Section 2.4.3.

Thus, to compile and execute an ALGOL program, your card deck would appear as follows.



10-0915

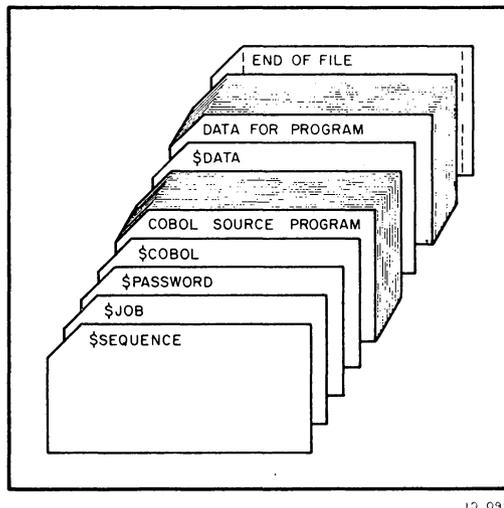
Refer to the description of each card for the information that goes on it. The way that you tell your program how to find its data is described in Section 2.4.3.1.

2.2.2 Card Deck to Run COBOL Programs

To run COBOL programs, you can use the \$COBOL card and the \$DATA card. You put a \$COBOL card in front of your COBOL program to make Batch copy your program into a disk file and insert a COMPILE command into your control file. The \$COBOL card is described in detail in Section 2.4.2.

You put a \$DATA card in front of the data that goes with your program to make Batch copy your data into another disk file and insert an EXECUTE command into your control file. The \$DATA card is described in Section 2.4.3.

Thus, to compile and execute one COBOL program, your card deck would appear as follows.



10 0916

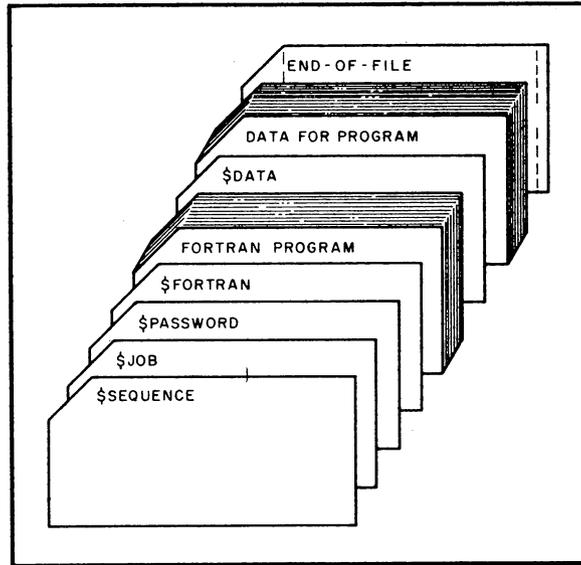
Refer to the description of each card for the information that goes on it. The way that you tell your program how to find its data is described in Section 2.4.3.1

2.2.3 Card Deck to Run FORTRAN Programs

To run FORTRAN programs, you can use the \$FORTRAN and \$DATA cards. You put a \$FORTRAN card in front of your FORTRAN program to make Batch copy your program into a disk file and insert a COMPILE command into your control file. The \$FORTRAN card is described in detail in Section 2.4.8.

You put a \$DATA card in front of the data that goes with your program to make Batch copy your data into another disk file and insert an EXECUTE command into your control file. The \$DATA card is described in Section 2.4.3.

Thus, to compile and execute one FORTRAN program, your card deck would appear as follows.



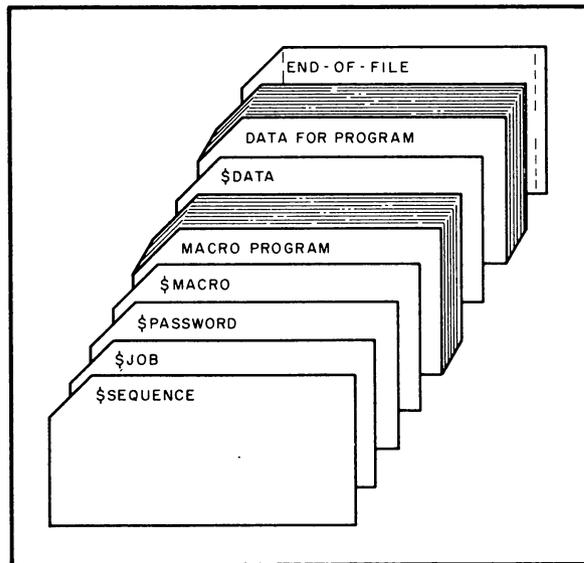
10-0917

Refer to the description of each card for the information that goes on it. The way that you tell your program how to find its data is described in Section 2.4.3.1.

#### 2.2.4 Card Deck to Run MACRO Programs

To run MACRO programs, you can use the \$MACRO and \$DATA cards. You put a \$MACRO card in front of your MACRO program to make Batch copy your program into a disk file and insert a COMPILE command into your control file. The \$MACRO card is described in detail in Section 2.4.10.

You put a \$DATA card in front of the data that goes with your program to make Batch copy your data into another disk file and insert an EXECUTE command into your control file. The \$DATA card is described in Section 2.4.3. Thus, to assemble and execute one MACRO program, your card deck would appear as follows.

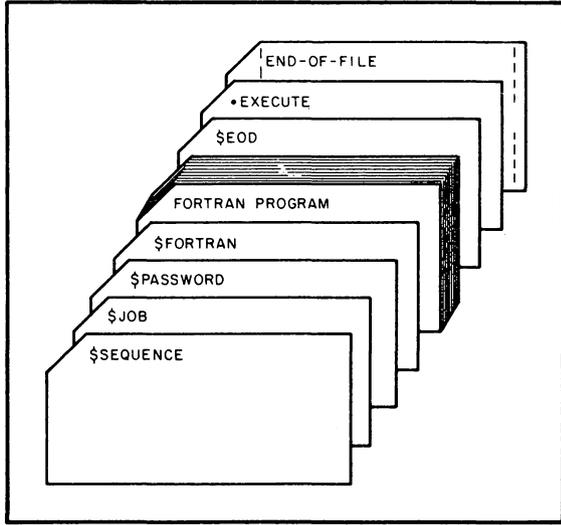


10-0918

Refer to the description of each card for the information that goes on it.

### 2.3 PUTTING COMMANDS INTO THE CONTROL FILE FROM CARDS

Batch puts commands into the control file for you when you use certain control cards. However, only a small number of kinds of commands can be put in the control file in this way. If you wish to perform operations in addition to compilation and execution, you must include commands in your card deck so that Batch will copy them into your control file. Where you put these commands in your card deck determines their positions in the control file. Batch reads your card deck in sequential order, copying commands into the control file as they, or the special control cards, are read. However, Batch, when it reads a control card that tells it to copy a program or data into a disk file, copies every card that follows such a control card until it meets another control card. To ensure that your commands are not copied into a file with programs or data, you must place a special control card, the \$EOD card, at the end of a program deck if you wish to follow the program with a command. For example, if you have a FORTRAN program that creates its own data and does not need to use a \$DATA card, you could include the following cards in your deck.



10-0919

(command to load and execute the program)  
 (to tell Batch to stop copying into the program file)

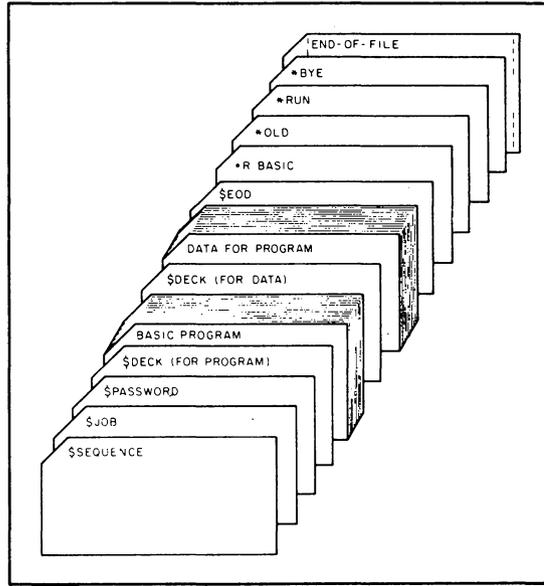
The only commands that you cannot use in a Batch job are CSTART, CCONT, ATTACH, DETACH, and SEND. Batch will ignore these commands when it reads them in the control file. Also, you cannot use the LOGIN command in your Batch job because you will get an error that will terminate your job. Batch logs your job in according to your \$JOB and \$PASSWORD cards.

### 2.3.1 Card Decks for Programs That Do Not Have Special Control Cards

By combining monitor commands with control cards such as \$DECK and \$EOD, in addition to the required control cards, you can process any program that does not have special control cards for it. You put a \$DECK card in front of a program, data, or any other group of cards to make Batch copy the cards that follow the \$DECK card into a disk file. However, Batch does not put a command into the control file when it reads a \$DECK card. The \$DECK card is described in detail in Section 2.4.4.

For example, a BASIC program does not have a specific control card. To run a BASIC program under Batch from cards, you can combine the \$DECK card and the \$EOD card with monitor commands. You also use a \$DECK card to copy the data for a BASIC program because the \$DATA card puts an EXECUTE command into the control file and BASIC does not use the EXECUTE command to run.

The following example shows a card deck that enters a BASIC program for running under Batch.

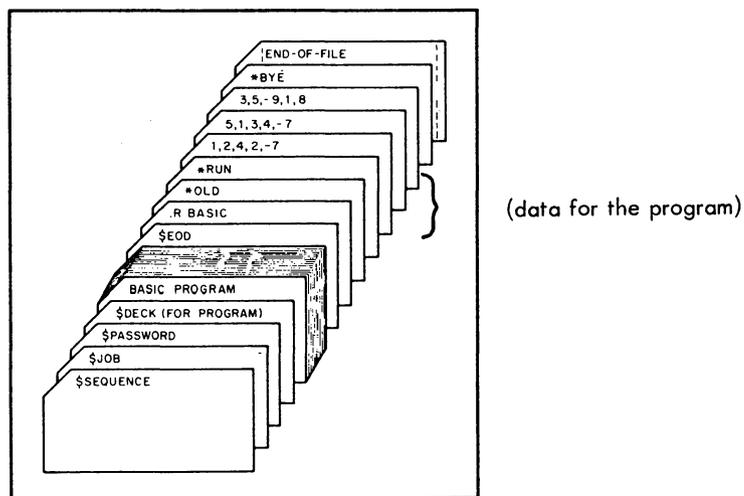


The BASIC program contains statements that read data from a disk file. You answer OLD to the BASIC question

NEW OR OLD --

because the file is on disk and can be retrieved by BASIC.

If your BASIC program reads data that is to be input by you during the running of the program, you enter the data in the control file so that it will be passed to your program by Batch. This is shown in the following example.



You can use the same technique to enter programs written in any language that does not have a specific control card, provided that your installation supports the language. Also, you can run system programs under Batch using the same technique.

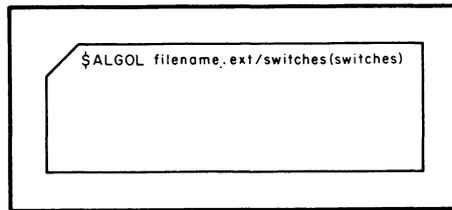
### 2.4 CONTROL CARDS FOR BATCH (IN ALPHABETICAL ORDER)

The special control cards for Batch are described below in detail. Only the control cards that are pertinent to this manual are discussed. Refer to DECsystem-10 Operating System Command (DEC-10-MRDC-D) for the remaining cards. The same is true for some of the switches that can be included on each card. If a switch is not described in this manual, it can be found in the DECsystem-10 Operating System Commands manual.

#### 2.4.1 The \$ALGOL Card

You put a \$ALGOL card in front of your ALGOL program to make Batch copy your ALGOL program into a disk file and insert a COMPILE command into your control file. Thus, when Batch runs your job, your ALGOL program will be compiled. You can put some optional information on the \$ALGOL card to tell Batch more about your program or the cards that your program is punched on.

The \$ALGOL card has the form:



10-0902

- filename.ext specifies the optional filename and extension that you can tell Batch to put on the file that it creates for your program. If you omit the filename and extension, Batch will create a unique name for your file and add the extension .ALG to it.
- /switches are switches to Batch to tell it how to read your program and whether or not to request a compilation listing when the program is compiled. The switches can be put on the card in any order and are described below.
- (switches) are switches that Batch passes to the ALGOL compiler when it puts the COMPILE command in the control file. These switches must be enclosed in parentheses, must not be preceded by slashes, and may or may not be separated by commas. The switches for the ALGOL compiler are described in Section 18.1 in Chapter 18 of the DECsystem-10 ALGOL Programmer's Reference Manual (DEC-10-KAZB-D).

`/WIDTH:n Switch`

Normally, Batch reads up to 80 columns on every card of the ALGOL program. You can make Batch stop reading at a specific column by means of the `/WIDTH` switch, in which you indicate the number of a column at which to stop. Thus, if you have no useful information in the last 10 columns of each card of your program, you can tell Batch to read only up to column 70 by specifying

```
/WIDTH:70
```

on the `$ALGOL` card.

`/NOLIST Switch`

Normally, the `$ALGOL` card tells Batch to ask the compiler to generate a compilation listing of your ALGOL program. The listing is then printed as part of your job's output. If you don't want this listing, you can include the `/NOLIST` switch on the ALGOL card to stop generation of the listing.

`/SUPPRESS:OFF Switch`

When Batch reads the cards of your ALGOL program, it normally does not copy any trailing spaces into the disk file to save space on the disk. If you want Batch to copy everything on the cards up to column 80 or any column that you may specify in the `/WIDTH` switch, you must include the `/SUPPRESS:OFF` switch on the `$ALGOL` card.

## Examples

The simplest form of the `$ALGOL` card is shown in the following example.

```
$ALGOL
```

This card causes Batch to copy your program into a file to which Batch gives a unique name and the extension `.ALG`. The cards in the program are read up to column 80 with trailing spaces suppressed. A listing file is produced when the program is compiled. This listing is written as part of the job's output. No compiler switches are passed to ALGOL.

The following is an example of a `$ALGOL` card with switches.

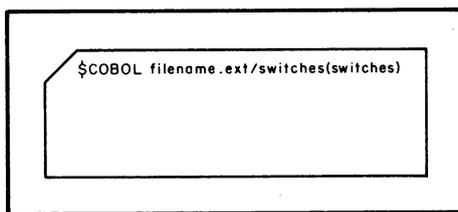
```
$ALGOL MYPROG.ALG /WIDTH:72 /NOLIST (1000D,N,Q)
```

With this card, your ALGOL program is copied into a file named `MYPROG.ALG` and a `COMPILE` command is entered into the control file. The cards in the program are read up to column 72 and trailing spaces up to column 72 are not copied into the file. When the program is compiled, no listing is produced, and the compiler reads and acts upon the switches `1000D`, `N`, and `Q` given to it by Batch.

### 2.4.2 The \$COBOL Card

You place the \$COBOL card in front of your COBOL program to make Batch copy your COBOL program into a disk file and insert a COMPILE command into your control file. Thus, when Batch runs your job, your COBOL program will be compiled. You can put some optional information on the \$COBOL card to tell Batch more about your program or the cards that your program is punched on.

The \$COBOL card has the form:



- filename.ext specifies the optional filename and extension that you can tell Batch to put on the file that it creates for your program. If you omit the filename and extension, Batch will create a unique name for your file and add the extension .CBL to it.
- /switches are switches to Batch to tell it how to read your program. The switches are described below.
- (swiches) are switches that Batch passes to the COBOL compiler when it puts the COMPILE command in the control file. These switches must be enclosed in parentheses, must not be preceded by slashes, and may or may not be separated by commas. The switches for the COBOL compiler are described in Table D-3 in Appendix D of the DECsystem-10 COBOL Programmer's Reference Manual (DEC-10-KCTC-D).

#### /WIDTH:n Switch

Normally, Batch reads up to 80 columns on every card of the COBOL program. You can make Batch stop reading at a specific column by means of the /WIDTH switch, in which you indicate the number of a column at which to stop. Thus, if you have no useful information in the last 10 columns of each card of your program, you can tell Batch to read only up to column 70 by specifying

/WIDTH:70

on the \$COBOL card.

**/SUPPRESS:OFF Switch**

When Batch reads the cards of your COBOL program, it normally does not copy any trailing spaces into the disk file to save space on the disk. If you want Batch to copy everything on the card up to column 80 or any column that you may specify in the /WIDTH switch, you must include the /SUPPRESS:OFF switch on the \$COBOL card.

**/CREF Switch**

If you want a cross-reference listing of your COBOL program, you can include the /CREF switch on the \$COBOL card to tell Batch to ask the COBOL compiler to produce a cross-reference listing when it compiles your program. This listing is printed as part of your job's output. You do not have to include a command to run the CREF program to get this listing, Batch will do it for you.

**/SEQUENCE Switch**

The COBOL compiler assumes that your COBOL program is in standard DECsystem-10 format. The /SEQUENCE switch, which Batch passes to the compiler, makes the compiler recognize that your program is in conventional (i.e., industry-wide) format. A program in conventional format has sequence numbers in columns 1 through 6 and comments that begin in column 73. When the /SEQUENCE switch is specified, the width of the card is assumed to be 72, not 80 columns. The following example shows programs in conventional and standard formats.

IF YOUR PROGRAM LOOKS LIKE:

```

1           8                               73
000010 IDENTIFICATION DIVISION..... MYPROG
000020 PROGRAM-ID. MYPROG..... MYPROG
000030 AUTHOR. ABB..... MYPROG
      .
      .
      .
    
```

YOU SHOULD:

Include the /SEQUENCE switch because your program is in conventional format.

IF YOUR PROGRAM LOOKS LIKE:

```

1
IDENTIFICATION DIVISION.....
PROGRAM-ID. MYPROG.....
AUTHOR. ABB.....
.
.
.
    
```

YOU SHOULD:

Omit the /SEQUENCE switch because your program is in DECsystem-10 standard format.

**Examples**

The simplest form of the \$COBOL card is:

**\$COBOL**

This card tells Batch to copy your program into a file and assign a unique name and the extension .CBL. All 80 columns of the cards are read, trailing spaces are not copied, and the compiler is told that the program is in standard format. No switches are passed to the compiler, and a listing file is produced when the job is run. The listing is printed as part of the job's output.

The following is an example of a \$COBOL card with switches.

\$COBOL MYPROG.CBL /SEQUENCE (N,P)

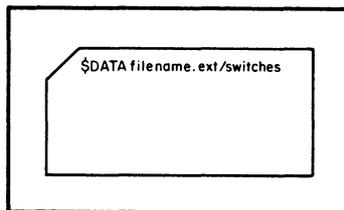
With this card, your COBOL program is copied into a disk file named MYPROG.CBL and a COMPILÉ command is inserted into the control file. The cards are read only up to column 72 and trailing spaces up to column 72 are not copied into your file. Batch passes the N and P switches to the compiler, and tells the compiler to accept the program in conventional format. A listing file is produced when the program is compiled. This listing is printed as part of the job's output.

### 2.4.3 The \$DATA Card

You put a \$DATA card in front of the data for your program to make Batch copy it into a disk file and to insert an EXECUTE command into your control file. Within the EXECUTE command, Batch requests a loader map for you. When your job is run, any programs that were entered with \$ALGOL, \$FORTRAN, or \$MACRO cards that came before the \$DATA card are executed. Every time that Batch reads one of the \$language cards, it adds it to a list that it keeps. When it then reads a \$DATA card, each program in Batch's list is put into the EXECUTE command string that the \$DATA card puts into the control file. After the \$DATA card is read by Batch and the EXECUTE command is put into the control file with the names of the programs that preceded the \$DATA card, Batch clears its list so that it can start a new list for programs entered later. If you have more than one set of data for a program or programs, you can precede each set with a \$DATA card to put two EXECUTE commands into the control file to run your program or programs twice. An EXECUTE command following another EXECUTE command in the control file without intervening \$language cards causes the programs executed by the first EXECUTE command to be loaded and executed again.

If your data is included in the program so that you do not have cards with data on them, you can still use the \$DATA card to insert an EXECUTE command into the control file.

The form of the \$DATA card is:



10-0904

filename.ext

specifies the optional filename and extension that you can tell Batch to put on the file that it creates for your data. If you omit the filename and extension, Batch will create a unique name for your file and add the extension .CDR to it.

`/switches`

are switches to Batch to tell it how to read the cards of your data. The switches are described below.

`/WIDTH:n Switch`

Normally, Batch reads up to 80 columns on every card of your data. You can make Batch stop reading at a specific column by means of the `/WIDTH` switch, in which you indicate the number of a column at which to stop. Thus, if you have information in the last 10 columns of each card of your data, you can tell Batch to read only up to column 70 by specifying

`/WIDTH:70`

on the `$DATA` card.

`/SUPPRESS:OFF Switch`

When Batch reads the cards of your data, it normally does not copy any trailing spaces into the disk file to save space on the disk. If you want Batch to copy everything on the cards up to column 80 or any column that you specify in the `/WIDTH` switch, you must include the `/SUPPRESS:OFF` switch on the `$DATA` card.

Examples

The simplest form of the `$DATA` card is:

`$DATA`

This card causes Batch to copy your data into a file and to assign a unique name and the extension `.CDR` to it. All 80 columns of the cards are read and trailing spaces are not copied into the file.

The following example shows a `$DATA` card with switches.

`$DATA MYDAT.DAT /WIDTH:72 SUPPRESS:OFF`

The data that follows this card is copied into a file named `MYDAT.DAT` and an `EXECUTE` command is inserted into the control file. When Batch reads the cards of the data, it reads only up to column 72 and copies trailing spaces into the data file.

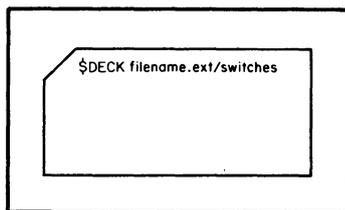
2.4.3.1 Naming Data Files on the `$DATA` Card - If you want to name your data file on the `$DATA` card rather than letting Batch name it for you, you must, in your program, assign that file to disk as shown in the following examples.







The form of the \$DECK card is:



filename.ext

specifies the optional filename and extension that you can tell Batch to put on the file that it creates for your program or data. If you omit the filename and extension, Batch will create a unique name for your file.

/switches

are switches to Batch to tell it how to read the cards in your deck. The switches are described below.

#### /WIDTH:n Switch

Normally, Batch reads up to 80 columns on every card in your deck. You can make Batch stop reading at a specific column by means of the /WIDTH switch, in which you indicate the number of a column at which to stop. Thus, if you have information in the last 10 columns of each card in your deck, you can tell Batch to read only up to column 70 by specifying

/WIDTH:70

on the \$DECK card.

#### /SUPPRESS:OFF Switch

When Batch reads the cards in your deck, it normally does not copy any trailing spaces into the disk file to save space on the disk. If you want Batch to copy everything on the cards up to column 80 or any column that you specify in the /WIDTH switch, you must include the /SUPPRESS:OFF switch on the \$DECK card.

#### Examples

The simplest form of the \$DECK card is:

\$DECK

This card causes Batch to copy your deck into a disk file and to assign a unique name to it. All 80 columns of the cards are read and trailing spaces are not copied into the file.

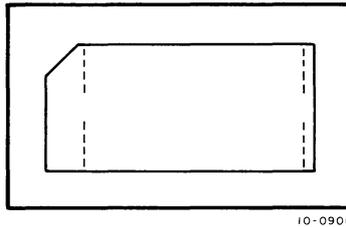
The following shows an example of a \$DECK card.

```
$DECK MYDECK.CDS /WIDTH:50 /SUPPRESS:OFF
```

The deck that follows this card is copied into a disk file named MYDECK.CDS. When Batch reads the cards in the deck, it reads up to column 50 and copies trailing spaces into the file.

#### 2.4.5 The End-of-File Card

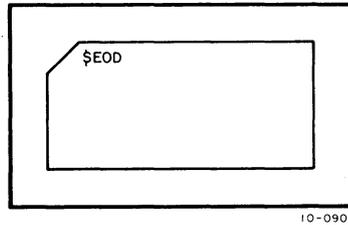
You must put the end-of-file card at the end of the deck containing your complete job to tell Batch that it has reached the end of your job. Unlike the other Batch control cards, the end-of-file card does not have a dollar sign (\$) and a command on it. It contains special punches that are recognized by Batch as the end-of-file. These punches must be in rows 6,7,8, and 9 of column 1. So that the end-of-file card can be recognized in any orientation (e.g., upside down), you should punch rows 12, 11,0,1,6,7,8, and 9 and leave rows 2,3,4, and 5 blank in both columns 1 and 80. If you omit the end-of-file card, an error message will be issued unless the installation makes the operator put the card on any deck that does not have one. However, your job will still be scheduled. The form of the end-of-file card is shown below.



#### 2.4.6 The \$EOD Card

You put a \$EOD card at the end of the cards being copied into a file due to a \$DECK, \$DATA, or \$language card. This card tells Batch to stop copying cards into the file. If another Batch control card follows the cards being copied, you don't need the \$EOD card because Batch stops copying cards into a file when it reads a Batch control card. The only time that the \$EOD card is necessary is when you wish to follow the cards being copied into a file by a card other than a control card, e.g., a card containing a command. Refer to Section 2.3 for a description of including commands in your deck.

The \$EOD card has the form:



An example of using the \$EOD card is shown below where the user wishes to load the COBOL debugging program COBDDT with his program.

```

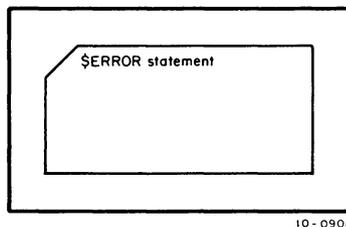
$COBOL MYPROG.CBL
:
$EOD
.LOAD %S MYPROG.CBL, SYS:COBDDT
.START MYPROG
    
```

If the \$EOD card had not been included in the above example, the .LOAD and .START commands would have been copied into the file with the COBOL program, rather than being copied into the control file.

#### 2.4.7 The \$ERROR Card

You can use the \$ERROR card and the \$NOERROR card (described later in this chapter) to specify error recovery in the control file. When Batch reads the \$ERROR card, it inserts a special Batch command into the control file, the .IF (ERROR) command. This command will later tell Batch what to do when an error occurs when your job is being processed. How to perform error recovery is described in Section 2.5.

The \$ERROR card has the form:



statement

is a command to the monitor, to a system program or a special Batch command such as .GOTO or .BACKTO.

Batch enters an .IF (ERROR) command into the control file when it reads the \$ERROR card, and includes the statement from the \$ERROR card in the .IF (ERROR) command in the form:

```

.IF (ERROR) statement
    
```

The Batch commands .GOTO and .BACKTO have the forms:

.GOTO statement label  
.BACKTO statement label

statement label

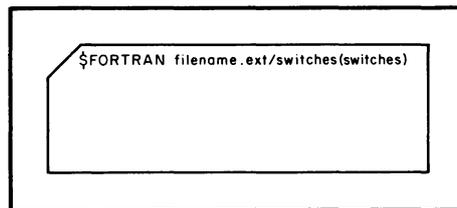
is the label of a line in the control file. The label can contain from 1 to 6 alphabetic characters and must be followed by a double colon (::) when it is labelling a line.

The .GOTO command tells Batch to search forward in the control file on disk until it finds the line containing the label. The .BACKTO command tells Batch to search back in the control file on disk to find the line containing the label. You must supply the labelled line and any related lines for which Batch will search. Include these lines in your card deck where you want them to be copied into the control file. If Batch cannot find a labelled line that is searching for as a result of a .GOTO or a .BACKTO statement, it terminates your job.

#### 2.4.8 The \$FORTRAN Card

You place the \$FORTRAN card in front of your FORTRAN program to make Batch copy your program into a disk file and insert a COMPILE command into your control file. Thus, when Batch runs your job, your FORTRAN program will be compiled. You can put some optional information on the \$FORTRAN card to tell Batch more about your program or the cards that your program is punched on.

The \$FORTRAN card has the form:



10-0909

filename.ext

specifies the optional filename and extension that you can tell Batch to put on the file that it creates for your program. If you omit the filename and extension, Batch will create a unique name for your file and add the extension .F4 to it.

/switches

are switches to Batch to tell it how to read your program. The switches are described below.

(switches)

are switches that Batch passes to the FORTRAN compiler when it puts the COMPILE command in the control file. These switches must be enclosed in parentheses, must not be preceded by slashes, and may or may not be separated by commas. The switches for the FORTRAN compiler are described in Table 11-1 in Chapter 11 of the DECsystem-10 FORTRAN IV Programmer's Reference Manual (DEC-10-AFDO-D).

/WIDTH:n Switch

Normally, Batch reads up to 72 columns on every card of the FORTRAN program. You can make Batch stop reading at a specific column by means of the /WIDTH switch, in which you include the number of the column at which to stop. The FORTRAN compiler only reads FORTRAN statements up to column 72, even if you tell Batch to read up to column 80. But, if you wish to have MPB read only up to column 60, you can specify

/WIDTH:60

on the \$FORTRAN card.

/SUPPRESS:OFF Switch

When Batch reads the cards of your FORTRAN program, it normally does not copy any trailing spaces into the disk file to save space on the disk. If you want Batch to copy everything on the card up to column 72 or any column that you specify in the /WIDTH switch, you must include the /SUPPRESS:OFF switch on the \$FORTRAN card.

/CREF Switch

If you want a cross-reference listing of your FORTRAN program, you can include the /CREF switch on the \$FORTRAN card to tell Batch to ask the FORTRAN compiler to produce a cross-reference listing when it compiles your program. This listing is printed as part of your job's output. You do not have to include a command to run the CREF program to get this listing, Batch will do it for you.

/NOLIST Switch

Normally, the \$FORTRAN card tells Batch to ask the compiler to generate a compilation listing of your FORTRAN program. The listing is then printed as part of your job's output. If you don't want this listing, you can include the /NOLIST switch on the \$FORTRAN card to stop generation of the listing.

## Examples

The simplest form of the \$FORTRAN card is:

\$FORTRAN

This card tells Batch to copy your program into a disk file and assign a unique name and the extension .F4. The first 72 columns of the cards are read, trailing spaces are not copied, and a listing file is produced when the job is run. No switches are passed to the compiler. The listing is printed as part of the job's output.

The following is an example of a \$FORTRAN card with switches.

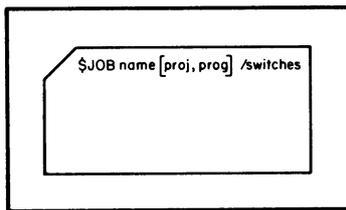
```
$FORTRAN MYPROG.F4 /CREF /NOLIST/SUPPRESS:OFF (I,M)
```

With this card, your FORTRAN program is copied into a disk file named MYPROG.F4 and a COMPILE command is inserted into the control file. The cards are read only up to column 72 and trailing spaces up to column 72 are copied into your file. A cross-reference listing of your program will be generated, but a compilation listing will not. Batch passes the I and M switches to the compiler.

### 2.4.9 The \$JOB Card

You must include the \$JOB card as the first card in your deck or as the second card following the \$SEQUENCE card, which is described later in this chapter. The \$JOB card tells Batch whose job that it is processing and, optionally, the name of the job, and any constraints that you want to place on the job. When Batch reads the \$JOB card and the \$PASSWORD card, if it is required, it creates the control file and begins the log file for your job. Batch then places commands into the control file that are taken from the cards that follow the \$JOB card.

The \$JOB card has the form:



- name
is the optional name that you can give to the job. If you omit the name, Batch will create a unique name for your job. The name of the job is that which Batch gives to your control file and log file. To the job name, Batch adds the extension .CTL for the control file. It adds the extension .LOG to the name for the log file.
- [proj, prog]
is your project-programmer number, i.e., the number that you were assigned by the installation to allow you to gain access to the DECsystem-10. Normally, the project-programmer number is two numbers separated by a comma and enclosed in square brackets.

/switches

are switches to Batch to tell it the constraints that you have placed on your job. They are described below.

/AFTER:dd-mmm-yy hhmm Switch

If you don't want Batch to run your job until after a certain time on a certain day, you can include the /AFTER switch on your \$JOB card. The date and time are specified in the form dd-mmm-yy hhmm (e.g., 20-MAY-72 0215). If this switch is not included, Batch runs your job at the time that it would normally schedule such a job, based on its size, the amounts of core and time required, and other parameters.

/AFTER:++ Switch

If you don't want Batch to run your job until after a certain number of minutes have elapsed since the job was entered, include this form of the /AFTER switch on the \$JOB card. The number of minutes that the job must wait after it has been entered is specified in the form ++ (e.g., +15). If this switch is not included, Batch will schedule the job as it normally does.

## NOTE

If any of the programs in your job have output to slow-speed devices such as the card punch, the paper-tape punch, the line printer, and the plotter, do not include an ASSIGN command in your job. Batch will take care of this output for you as long as you specify the switches for these devices, which are described below.

/CARDS:nK Switch

If any program in your job has punched card output, you must include the /CARDS switch on the \$JOB card to specify the approximate number of cards that your job will punch. Up to a maximum of 10,000 cards can be specified in the form nK or n (e.g., 5K or 5 specifies 5,000 cards). If you do not specify the /CARDS switch, no cards will be punched, even if you want them. If you do not specify enough cards, the remaining output over the number of cards specified will be lost without notification to you.

/CORE:nK Switch

You can specify the amount of core in which the programs in your job will run by means of the /CORE switch. You specify the amount of core in the form n or nK (e.g., 25 or 25K). You should try to estimate as closely as possible the amount of core that your job will need. If you don't specify enough, your job can't run. If you don't specify the amount of core that your job will need, Batch will assume 25K or an amount set by the installation.

/FEET:n Switch

If any program in your job has punched paper-tape output, you must include the /FEET switch on the \$JOB card to specify the approximate number of feet of paper tape that your job will punch. You

specify the number of feet in the form n (e.g., 50). If you do not specify the /FEET switch, no paper tape will be punched, even if you want it. If you do not specify enough paper tape, the output that remains over the number of feet that you specify will be lost and the message ?OUTPUT FORMS LIMIT EXCEEDED will be punched in block letters in the tape.

/PAGES:n Switch

Normally, Batch allows your job to print up to 100 pages. Included in this number are the log file and any compilation listings that you may request. If you need more than 100 pages for your job, you must include the /PAGES switch on the \$JOB card to indicate the approximate number of pages that your job will print. If your output exceeds either the maximum that Batch allows or the number that you specified in the /PAGES switch, the excess output will not be printed and the message ?OUTPUT FORMS LIMIT EXCEEDED will be written in the log file. However, even if you exceed the maximum, the first 10 pages of the log file will be printed.

/TIME:hh:mm:ss Switch

Normally, Batch allows your job to use up to 5 minutes of central processor time. Central processor (CPU) time is the amount of time that your job runs in core, not the amount of time that it takes Batch to process your job. If you need more than 5 minutes of CPU time, you must include the /TIME switch on the \$JOB card to indicate the approximate amount of time that you will need. If you don't specify enough time, Batch will terminate your job when the time is up. However, if you specify a large amount of time, Batch may hold your job in the queue until it can schedule a large amount of time for it.

The value in the /TIME switch is given in the form hh:mm:ss (hours:minutes:seconds). However, if you specify only one number, Batch assumes that you mean seconds. Two numbers separated by a colon (:) is assumed to mean minutes and seconds. Only when you specify all three numbers, separated by colons, does Batch assume that you mean hours, minutes, and seconds. For example:

|               |                               |
|---------------|-------------------------------|
| /TIME:25      | means 25 seconds              |
| /TIME:1:25    | means 1 minute and 25 seconds |
| /TIME:1:25:00 | means 1 hour and 25 minutes   |

/TPLOT:t Switch

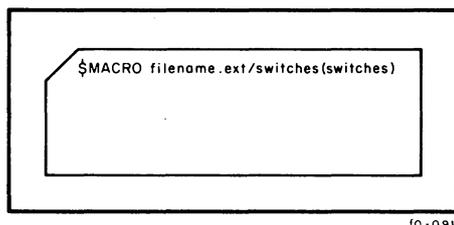
If you have any programs in your job that do output to the plotter, you must include the /TPLOT switch on the \$JOB card so that your output will be plotted. If the /TPLOT switch is not included, no output will be plotted. If enough minutes (specified in the form t) are not specified, any plotter output left after the time has expired will be lost without notification to you.

2.4.10 The \$MACRO Card

You place a \$MACRO card in front of your MACRO program to make Batch copy your program into a disk file and insert a COMPILE command into your control file. Thus, when Batch runs your job,

your MACRO program will be assembled. You can put some optional information on the \$MACRO card to tell Batch more about your program or the cards that your program is punched on.

The \$MACRO card has the form:



- |              |  |
|--------------|--|
| filename.ext | specifies the optional filename and extension that you can tell Batch to put on the file that it creates for your program. If you omit the filename and extension, Batch will create a unique name for your file and add the extension .MAC to it.   |
| /switches    | are switches to Batch to tell it how to read your program and the kind of listings that you want. The switches are described below.  |
| (switches)   | are switches that Batch passes to the MACRO assembler when it puts the COMPILE command in the control file. The switches must be enclosed in parentheses, must not be preceded by slashes, and may or may not be separated by commas. The switches for the MACRO assembler are described in Table H-1 in Appendix H of the <u>DECsystem-10 MACRO-10 Assembler Programmer's Reference Manual</u> (DEC-10-AMZB-D). |

#### /WIDTH:n Switch

Normally, Batch reads up to 80 columns on every card of your MACRO program. You can make Batch stop reading at a specific column by means of the /WIDTH switch, in which you include the number of the column at which to stop. Thus, if you wish to have Batch read only up to column 70, you can specify

/WIDTH:70

on the \$MACRO card.

#### /SUPPRESS:OFF Switch

When Batch reads the cards of your MACRO program, it normally does not copy any trailing spaces into the disk file to save space on the disk. If you want Batch to copy everything on the cards up to column 80 or any column that you specify in the /WIDTH switch, you must include the /SUPPRESS:OFF switch on the \$MACRO card.

### /CREF Switch

If you want a cross reference listing of your MACRO program, you can include the /CREF switch on the \$MACRO card to tell Batch to ask the MACRO assembler to produce a cross-reference listing when it assembles your program. This listing is printed as part of your job's output. You do not have to include a command to run the CREF program to get this listing, Batch will do it for you.

### /NOLIST Switch

Normally, the \$MACRO card tells Batch to ask the assembler to generate an assembly listing of your MACRO program. The listing is then printed as part of your job's output. If you don't want this listing, you can include the /NOLIST switch on the \$MACRO card to stop generation of the listing.

### Examples

The simplest form of the \$MACRO card is:

```
$MACRO
```

This card tells Batch to copy your program into a disk file and assign a unique name and the extension .MAC to it. All 80 columns of the cards are read, trailing spaces are not copied, and a listing file is produced when the job is run. The listing is printed as part of the job's output. No switches are passed to the assembler.

The following is an example of a \$MACRO card with switches.

```
$MACRO MYPROG.MAC /SUPPRESS:OFF /WIDTH:72 (P,Q,X)
```

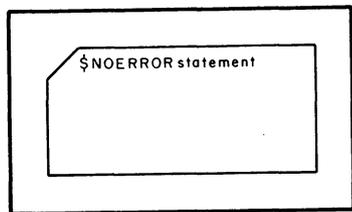
With this card, your MACRO program is copied into a disk file named MYPROG.MAC and a COMPILE command is inserted into the control file. The cards are read only up to column 72 and trailing spaces are copied into your file. An assembly listing is generated, and Batch passes the P,Q, and X switches to the assembler.

#### 2.4.11 The \$NOERROR Card

You can use the \$NOERROR card and the \$ERROR card (described in Section 2.3.7) to specify error recovery in the control file.

When Batch reads the \$NOERROR card, it inserts a special Batch command into the control file, the .IF (NOERROR) command. This command tells Batch what to do when an error occurs when your job is being processed. How to perform error recovery is described in Section 2.5.

The \$NOERROR card has the form:



10-0912

statement

is a command to the monitor or a special Batch command such as .GOTO or .BACKTO.

Batch enters an .IF (NOERROR) command into the control file when it reads the \$NOERROR card, and includes the statement from the \$NOERROR card in the .IF (NOERROR) command in the form:

.IF (NOERROR) statement

The Batch commands .GOTO and .BACKTO have the forms:

.GOTO statement label

.BACKTO statement label

statement label

is the label of a line in the control file. The label can contain from 1 to 6 alphabetic characters and must be followed by a double colon (::) when it is labelling a line.

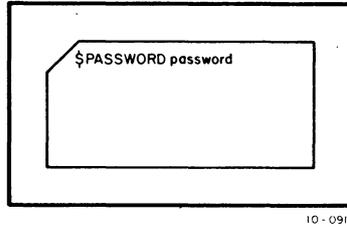
The .GOTO command tells Batch to search forward in the control file until it finds the line containing the label. The .BACKTO command tells Batch to search back in the control file to find the line containing the label. You must supply the labelled line and any related lines for which Batch will search. Include these lines in your card deck where you want them to be copied into the control file. If Batch cannot find a labelled line that is searching for as a result of a .GOTO or a .BACKTO statement, it terminates your job.

#### 2.4.12 The \$PASSWORD Card

You put the password that has been assigned to you on the \$PASSWORD card to tell Batch that you are an authorized user of the system.

In conjunction with the \$JOB card, the \$PASSWORD card identifies you to Batch and tells Batch to create the control file and log file for your job. If you put a password on the \$PASSWORD card that does not match the password stored in the system for you, Batch will not create any files and will terminate your job. Some installations may not require the \$PASSWORD card; if it is required at your installation, you must put it immediately after the \$JOB card.

The \$PASSWORD card has the form:



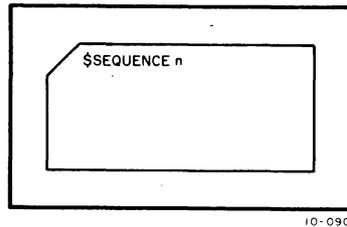
password

is a 1 to 6 character password that is stored in the system to identify you.

#### 2.4.13 The \$SEQUENCE Card

You use the \$SEQUENCE card to specify a unique sequence number for your job. This card may or may not be required by the installation or may be supplied by the personnel at the installation. If the card is required, you must include it as the first card in the deck containing your job.

The form of the \$SEQUENCE card is:

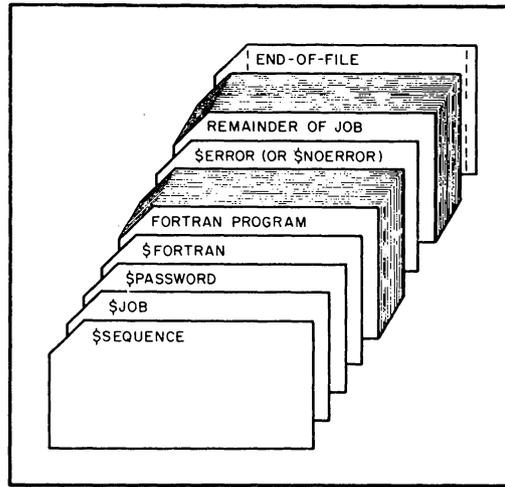


n

is the unique sequence number assigned to your job.

### 2.5 SPECIFYING ERROR RECOVERY IN THE CONTROL FILE

Normally, when an error occurs in your job, Batch terminates the job and, if the error occurred when one of your programs was running, causes a dump of your core area. The dump is printed with your output and log file. You can specify recovery from errors in the control file by means of the \$ERROR and \$NOERROR cards, described in Sections 2.4.7 and 2.4.11. You must include one of these cards at the point in the control file that an error may occur. When an error occurs, Batch examines the next monitor-level line (i.e., not a line that contains data or a command string to a system program) to find an .IF (ERROR) statement to tell it what to do about the error. If an error does not occur and an .IF (ERROR) statement is present, the .IF (ERROR) statement is not executed. Thus, if you have a program that you are not sure is error-free, you can include a \$ERROR or \$NOERROR card to tell Batch what to do if an error occurs, as shown in the following example.



10-09'4

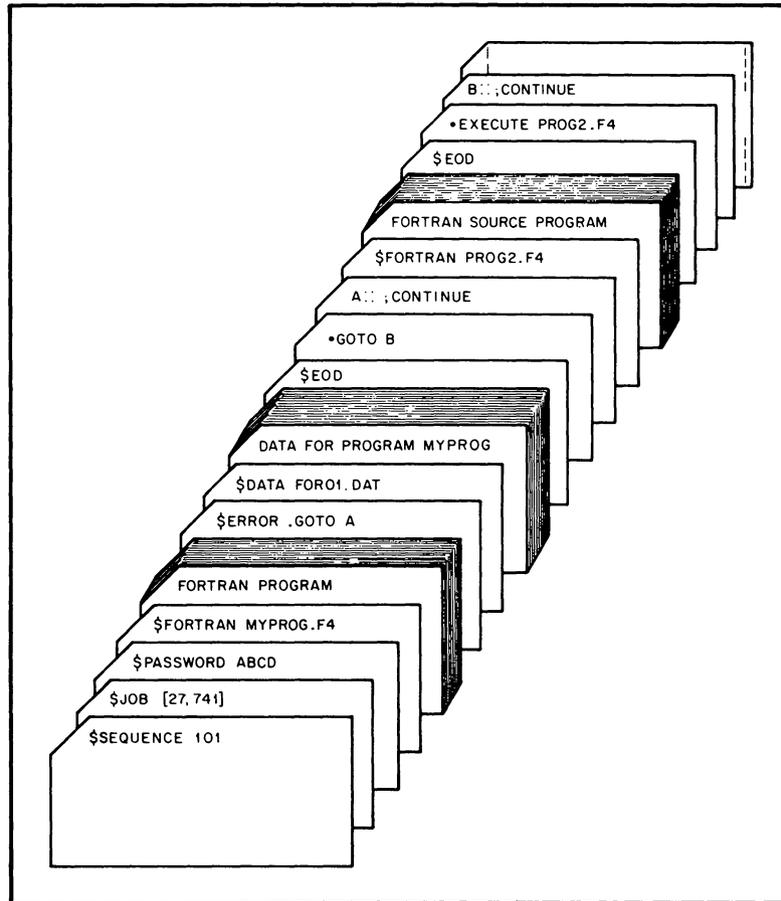
The above cards would cause Batch to make the following entries in the control file.

```
.COMPILE ...
  .IF (ERROR) statement
  .
  .
  .
```

On either the \$ERROR or \$NOERROR card, you must include a statement that tells Batch what to do. You can use any monitor command, a command to a program, or one of the special Batch commands. The .GOTO and .BACKTO commands are two Batch commands for this purpose. Refer to Section 2.4.7 for descriptions of these commands. Be sure, if you use .GOTO or .BACKTO on your \$ERROR or \$NOERROR card, that you supply a line for the control file that has the label that you specified in the .GOTO or .BACKTO commands.

Two sample jobs are shown below. The first shows using \$ERROR and the .GOTO command to specify error recovery. The second example shows the use of the \$NOERROR card and the .GOTO command.

If you have a program that you are not sure will compile without errors, you can include another version of the same program in your job (that hopefully will compile) and tell Batch to compile the second program if the first has an error. The cards to enter this job are shown below.



10-0922

These cards set up the following control file for you.

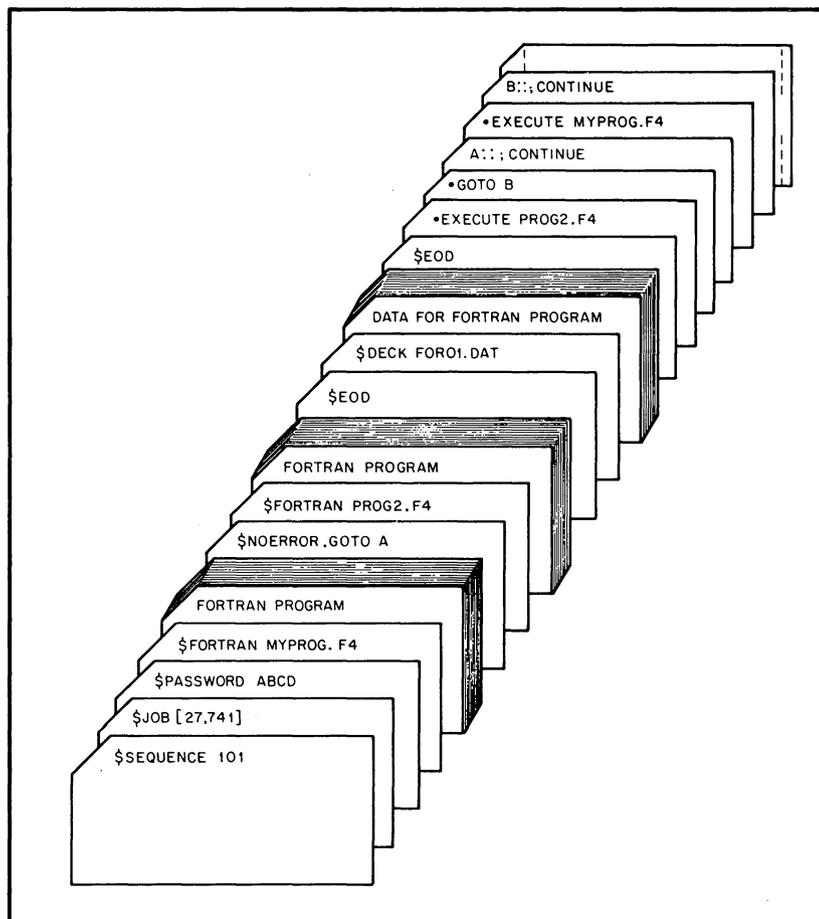
```
.COMPILE /COMPILE MYPROG.F4 /LIST
.IF (ERROR) .GOTO A
.EXECUTE MYPROG.REL /MAP:MAP.LST
.GOTO B
A::;CONTINUE
.COMPILE /COMPILE PROG2.F4 /LIST
.EXECUTE PROG2.F4
B::;CONTINUE
```

The \$FORTRAN card told Batch to copy the program MYPROG.F4 into a disk file and to insert a COMPILE command into the control file. The \$ERROR card told Batch to insert .IF (ERROR) .GOTO A into the control file. The data was copied into a disk file and an EXECUTE command was put into the control file because of the \$DATA card. The \$EOD card told Batch to stop copying cards into the data file, so Batch put the next two lines into the control file. The second \$FORTRAN card told Batch to copy the program PROG2.F4 into a disk file and put a COMPILE command into the control file. Another \$EOD card told Batch to stop copying into the program file, so Batch put the next two lines into the control file. An EXECUTE command was used instead of a

\$DATA card because the data was already in a file on disk, although the \$DATA card does not have to have data with it to put an EXECUTE command in the control file.

When the job is started, Batch reads the control file and passes commands to the monitor. If an error occurs in the compilation of the first program, Batch finds the .IF statement and executes the .GOTO command contained in it. The command tells Batch to look for the line labelled A, which contains a comment, so Batch goes on to the next line. The second program is compiled and then executed with the data. The next line is a comment, so Batch continues to the end of the control file. If an error does not occur in the first program, Batch skips the .IF statement, executes the program with the data, skips the unnecessary error procedures, and continues to the end of the control file.

A variation of the above procedure is shown below using the \$NOERROR card and the .GOTO command. The difference is that Batch skips the .IF statement if an error occurs, and performs it if an error does not occur.



10-0923

Batch reads the cards and puts the following commands into the control file.

```
.COMPILE /COMPILE MYPROG.F4 /LIST
.IF (NOERROR) .GOTO A
.COMPILE /COMPILE PROG2.F4 /LIST
.EXECUTE PROG2.F4
.GOTO B
A:: ;CONTINUE
.EXECUTE MYPROG.F4
B:: ;CONTINUE
```

The \$FORTRAN card tells Batch to copy the FORTRAN program into a file named MYPROG.F4, and to insert a COMPILE command into the control file. The \$ERROR card tells Batch to insert an .IF command into the control file. The second \$FORTRAN card tells Batch to copy the second program into a disk file named PROG2.F4 and to insert another COMPILE command into the control file. Instead of a \$DATA card, a \$DECK card is used to tell Batch to copy the data into a disk file named FOR01.DAT. The \$DATA card is not used here because it would have the names of both programs in its list for the EXECUTE command generation, which would cause an error when the job is run. To tell Batch to stop copying cards into the data file, the \$EOD card comes next. Thus, Batch copies the next five lines into the control file.

When the job is run, Batch passes the COMPILE command to the monitor to compile the first program. If an error does not occur, the .IF command is read and the .GOTO command is executed. Batch skips to the line labelled A, which is a comment, and continues reading the control file. The program MYPROG.F4 is executed with the data and the end of the job is reached. If an error occurs, Batch skips the .IF statement and continues reading the control file. PROG2.F4 is compiled and then executed with the data. Batch is then told to go to the line labelled B, which is a comment line. The end of the job follows.

The examples shown above illustrate only two ways that you can specify error recovery in the control file. You can also use the .BACKTO command or any monitor command that you choose to help you recover from errors in your job.

You do not have to attempt to recover from errors while your job is running. You can correct your errors according to the error messages in the log file when your job is returned to you, and then run your job again. Batch will also print a dump of your core area if an error occurs while your job is running and you have not specified error recovery. If you can read dumps, this can also aid you to correct your errors. The log file and dumps are described in Chapter 4.

BEGINNER'S BATCH

- 136 -

## CHAPTER 3 ENTERING A JOB TO BATCH FROM A TERMINAL

When you enter a job to Batch from a timesharing terminal, you must create a control file that Batch can use to run your job. The control file contains all the commands that you would use to run your job if you were running under timesharing. For example, if you wanted to compile and execute a program called MYPROG.CBL, the typeout would appear as follows.

|                     |   |                    |
|---------------------|---|--------------------|
| .COMPILE MYPROG.CBL | } | (Your request)     |
| COBOL: MYPROG       |   | The system's reply |
| EXIT                | } | (Your request)     |
| .EXECUTE MYPROG.CBL |   | The system's reply |
| LOADING             | } |                    |
| LOADER 1K CORE      |   |                    |
| EXECUTION           |   |                    |
| EXIT                |   |                    |

The control file to tell Batch to run the same job appears as the following.

```
.COMPILE MYPROG.CBL
.EXECUTE MYPROG.CBL
```

When the job is run, the commands are passed to the monitor to be executed. The commands and their replies from the monitor are written in the log file so that the entire dialog shown above appears in the log file.

To create a control file and submit it to Batch from a terminal, you must perform the following steps.

1. LOGIN to the system as a timesharing user.
2. Write a control file using an editor such as TECO or LINED.
3. When you finish the control file, close and save it on disk.
4. Submit the job to Batch using the monitor command SUBMIT or QUEUE INP:.

You can then wait for your output to be returned at the designated place.

### 3.1 CREATING THE CONTROL FILE

After you have logged into the system as you normally would to start a timesharing job, you must run an editor so that you can create your control file.

The control file can contain monitor commands, system program commands, data that would normally be entered from a terminal, and special Batch commands. The Batch commands are described in

Section 3.3. What you write in the control file depends on what you wish your job to accomplish.

An example of a job that you can enter to Batch from a terminal is as follows.

1. Compile a program that is on disk.
2. Load and execute the program with data from another disk file.
3. Print the output on the line printer.
4. Write the output into a disk file also.
4. Compile a second program.
6. Load and execute the second program with the data output from the first program.
7. Print the output from the second program.

The control file that you would write for the above job is as follows.

```
.COMPILE MYPROG.F4/COMPILE
.EXECUTE MYPROG.F4
.COMPILE PROG2.F4/COMPILE
.EXECUTE PROG2.F4
```

You include statements in your programs to read the data from the disk files and write the output to the printer and the disk. The output to the line printer is written with your log file as part of the total output of your job.

If an error occurs in your job, Batch will not continue, but will terminate the job and, if the error occurs while one of your programs is running, cause a dump to be taken of your core area. The dump is then printed with your job's output. To avoid having your job terminated because an error occurs, you can specify error recovery in the control file using the special Batch commands. Error recovery is described in Section 3.4.

Any monitor command that you can use in a timesharing job can be used in a Batch job with the following exceptions. The ATTACH, DETACH, CCONT, CSTART, and SEND commands have no meaning in a Batch job. If you include one of these commands in your job, Batch will write the command and an error label BAERR into your log file, will not process the command, and will then continue the job from that point. Do not include a LOGIN command in your control file because Batch logs the job for you. If you put in a LOGIN command, your job will be terminated.

### 3.1.1 Format of Lines in the Control File

Since you can put monitor, system program, and Batch commands, as well as data into the control file, you have to tell Batch what kind of line it is reading. The format of each of these lines is described below. Each line normally begins in column 1, but Batch always starts reading at the first nontab or nonblank character, regardless of the column in which it appears.

To include a monitor or Batch command, you must put a period (.) in the first column and follow it immediately with the command. Any information that follows a monitor command is in the format

shown for the command in Chapter 3 of the DECsystem-10 Operating System Commands manual. Any information that follows a Batch command is in the format shown in Section 3.3 in this chapter.

If you include a command string to a system program, you must place an asterisk (\*) in column 1 and follow it immediately with the command string. For the format of command strings, refer to the manual for the specific system program that you wish to use.

If you want to include a command to a system program that does not accept carriage return as the end of the line (e.g., TECO and DDT), you must substitute an equal sign (=) for the asterisk so that Batch will suppress the carriage return at the end of the line.

To include data for your program in the control file, write it as you would data that is read from a separate file. The only restriction on data in the control file is that alphabetic data that is preceded by a dollar sign (\$) must be preceded by an additional dollar sign so that Batch will not mistake it for a comment.

If you put any special characters other than those described above in the first column of the line, you may get unexpected results because Batch interprets other special characters in special ways. If you want to know about other special characters, refer to Chapter 3 of the DECsystem-10 Operating System Commands manual.

If you have more information than will fit on one line, you can continue on the next line by placing a hyphen (-) as the last nonspace character on the line to be continued and the rest of the information on the next line.

Comments can also be included either as separate lines in the control file or on lines containing other information. To include a comment on a separate line, you must put a semicolon(;) in column 1 and follow it immediately with the comment. To add a comment to a line, you must precede the comment with a semicolon (;) after all the information that you need has been put on the line.

## 3.2 SUBMITTING THE JOB TO BATCH

After you have created the control file and saved it on disk, you must enter it into the Batch queue so that it can be run. All programs and data that are to be processed when the job is run must be made up in advance or be generated during the running of the job. You can have them on any medium but, if they are on devices other than disk, you must include commands in your control file to have the operator mount the devices on which your programs and data reside. It is recommended that your programs and as much of your data as is possible reside on disk. An example of including MOUNT commands in the control file to mount tapes is shown in Chapter 5.

You enter your job into Batch's queue by means of the SUBMIT or QUEUE INP: monitor command. These commands have the forms:

```
SUBMIT jobname = control filename.ext, log filename.ext /switches
QUEUE INP:jobname = control filename.ext, log filename.ext / switches
```

|                      |   |
|----------------------|---|
| jobname              | is the name that you give to your job. If this name is omitted, Batch uses the name of the control file.  |
| control filename.ext | is the name that you have given to the control file that you created. You can add an extension, but if you don't, Batch will assume an extension of .CTL. |
| log filename.ext     | is the name that Batch will give the log file when it is created. You can add an extension, but if you don't, Batch will assume an extension of .LOG.     |

You must specify the name of the control file. If the name of the log file is omitted, its name will be taken from the name of the control file.

|           |  |
|-----------|--|
| /switches | are switches to Batch to tell it how to process your job and what your output will look like. Most switches can appear anywhere in the command string; however, a few must be placed after the files to which they pertain. The various kinds of switches are described below. |
|-----------|--|

Three kinds of switches are available to you to use in the SUBMIT and QUEUE INP: commands. The switches are: queue operation, general, and file control. Each category of switch and the switches in each category are described in the following sections.

### 3.2.1 Queue Operation Switches

Queue operation switches describe the actions that you want Batch to perform with your job. Only one of this type of switch can be placed in the command string, and it can appear anywhere in the command string.

#### /CREATE Switch

With the /CREATE switch, you tell Batch that you are entering a job into its queue. The job will then wait in the queue until Batch is ready to process it. If you omit a queue operation switch from the SUBMIT command string, Batch will assume the /CREATE switch, i.e., it will assume that you are entering a job. An example of this switch follows.

```
SUBMIT MYJOB = MYFILE,CTL, MYLOG.LOG /CREATE
```

#### /KILL Switch

You put the /KILL switch in a SUBMIT command to tell Batch that you want to delete a job that you previously entered into its queue. For example, if you submit a job and discover that you left a command out of the control file, you could delete the queue entry by issuing another SUBMIT command for that job with a /KILL switch in it. After you have corrected the control file, you could resubmit the job to Batch. However, if Batch has already started to run your job, it will ignore

your request to delete the job and issue the message %QUEUE REQUEST INP:jobname [proj, prog] INTERLOCKED IN QUEUE MANAGER. When you use the /KILL switch, you must specify the job name in the SUBMIT command or you will kill all the jobs that you may have in the Batch input queue.

#### /MODIFY Switch

If you want to change any switch value that you have previously entered in a SUBMIT command, you can include the /MODIFY in a new SUBMIT command to tell Batch which switch value that you want to change. You can change any switch value that can be entered in a SUBMIT command. The switch value that you want changed is written immediately after the /MODIFY switch. For example, to change the number of pages in a /PAGE switch (described below), you could issue the following command:

```
SUBMIT MYJOB = /MODIFY/PAGE:500
```

The value specified in the /PAGE switch that follows the /MODIFY switch replaces the previous value. If Batch has already started the job in which you wish to change a switch, the /MODIFY switch will be ignored, and Batch will issue the message %QUEUE REQUEST INP:jobname [proj, prog] INTERLOCKED IN QUEUE MANAGER.

### 3.2.2 General Switches

You use the general switches to define limits for your job. Such limits as core, pages of output, and the time that your job will run can be specified as general switches. Each general switch can be specified only once in a SUBMIT command, although each can be modified in subsequent SUBMIT commands by means of the /MODIFY switch. You can put a general switch anywhere in the command string because it affects the entire job, not just one file in the job.

#### /AFTER:t Switch

If you don't want Batch to run your job until after a certain time or until after a certain number of minutes have elapsed since the job was entered, you can include the /AFTER switch in the SUBMIT command string. The time is specified in the form hhmm (e.g., 1215) and the number of minutes that the job must wait is specified in the form ++ (e.g., +15). If you omit the switch, or the colon and the value in the switch, Batch will schedule your job as it normally would.

#### NOTE

If any of the programs in your job have output to slow-speed devices such as the card punch, the paper-tape punch, the line printer, and the plotter, do not include an ASSIGN command in your job. Batch will take care of this output for you as long as you specify the switches for these devices, which are described below.

### `/CARDS:n Switch`

If any program in your job has punched card output, you must include the `/CARDS` switch in the SUBMIT command to specify the approximate number of cards that your job will punch. The number of cards is specified in the form `n` (e.g.; 1000). If you do not specify the `/CARDS` switch, no cards will be punched, even if you want them. If you specify the switch without the colon and a value, up to 2000 cards can be punched by your job. If you do not specify enough cards, the output that remains after the limit is reached will be lost without notification to you.

### `/CORE:n Switch`

You can specify the maximum amount of core in which the programs in your job will run by means of the `/CORE` switch. You specify the amount of core in the form `n` (e.g., 25) which indicates decimal thousands. You should try to estimate as closely as possible the amount of core that your job will need. If you don't specify enough, your job can't run to completion. If you omit the switch, Batch will assume 25K of core or an amount set by the installation. If you specify the switch without the colon and a value, Batch will assume 40K of core or an amount set by the installation.

### `/FEET:n Switch`

If any program in your job has punched-paper-tape output, you must include the `/FEET` switch in the SUBMIT command to specify the approximate number of feet of paper tape that your job will punch. You specify the number of feet in the form `n` (e.g., 50). If you do not specify the `/FEET` switch, no paper tape will be punched, even if you want it. If you specify the `/FEET` switch without the colon and a value, Batch will assume the number of feet equal to 10 times the number of disk blocks that your paper tape output would occupy plus 20. If you do not specify enough paper tape, the output that remains after the limit is exceeded will be lost and the message `?OUTPUT FORMS LIMIT EXCEEDED` will be punched into the tape in block letters.

### `/PAGE:n Switch`

Normally, Batch allows your job to print up to 200 pages. Included in this number are the log file and any listings that you may request. If you need more than 200 pages for your job, you must include the `/PAGES` switch in the SUBMIT command to indicate the approximate number of pages that your job will print. If you include the switch without the colon and a value, Batch will assume that you will print up to 2000 pages. If your output exceeds either the maximum that Batch allows or the number that you specified in the `/PAGE` switch, the excess output will be lost and the message `?OUTPUT FORMS LIMIT EXCEEDED` will be printed. However, even if you exceed the maximum, the first 10 pages of the log file will be printed.

### `/TIME: hh:mm:ss Switch`

Normally, Batch allows your job to use up to 5 minutes of central processor time. Central processor (CPU) time is the amount of time that your job runs in core, not the amount of time that it takes

Batch to process your job. If you need more than 5 minutes of CPU time, you must include the /TIME switch in the SUBMIT command to indicate the approximate amount of time that you will need. If you specify the switch without the colon and a value, Batch will assume that you need 1 hour of CPU time. If you don't specify enough time, Batch will terminate your job when the time is up.

The value in the /TIME switch is given in the form hh:mm:ss (hours:minutes:seconds). However, if you specify only one number, Batch assumes that you mean seconds. Two numbers separated by a colon is assumed to mean minutes and seconds. Only when you specify all three numbers, separated by colons, does Batch assume that you mean hours, minutes, and seconds. For example:

|               |                               |
|---------------|-------------------------------|
| /TIME:25      | means 25 seconds              |
| /TIME:1:25    | means 1 minute and 25 seconds |
| /TIME:1:25:00 | means 1 hour and 25 minutes.  |

#### /TPLOT:t Switch

If you have any programs in your job that do output to the plotter, you must include the /TPLOT switch in the SUBMIT command so that your output will be plotted. If the /TPLOT switch is not included, no output will be plotted. If you specify the switch without the number of minutes (specified in the form t), Batch will allow output equal to 10 minutes of plotter time. If enough minutes are not specified, any plotter output left after the time has expired will be lost without notification to you.

### 3.2.3 File-Control Switches

File-control switches allow you to specify parameters for individual files in the SUBMIT command. The control file can receive a special parameter, while the log file does not, and vice versa. If you place a file-control switch before the two filenames in the SUBMIT command, the switch applies to both files in the request. If you place the switch after one of the files in the command, it refers to only that file.

#### /DISPOSE Switch

The /DISPOSE switch can have one of three values:

```

/DISPOSE:DELETE
/DISPOSE:PRESERVE
/DISPOSE:RENAME

```

/DISPOSE:DELETE allows you to specify that either the control file or the log file (or both) should be deleted after the job is run. The log file is deleted from your disk area only after it has been printed.

/DISPOSE:PRESERVE allows you to specify that one or both of your files should be left in your disk area after the job is finished and all output printed.

`/DISPOSE:RENAME` tells Batch that you want the specified file to be taken from your disk area immediately and put in Batch's disk area. In the case of the log file, `/DISPOSE:RENAME` only works for a log file that already exists on your disk area. Do not use `/DISPOSE:RENAME` for a log file that does not yet exist. After the job has been run and the output has been printed, the file that was renamed is deleted from Batch's disk area.

If you omit the `/DISPOSE` switch, Batch assumes `/DISPOSE:PRESERVE`. That is, both the control file and the log file are saved in your disk area. If you plan to use the control file again, then it is best to omit the `/DISPOSE` switch for the control file. If you don't want to keep the control file because you don't have enough room in your disk area, specify either `/DISPOSE:DELETE` or `/DISPOSE:RENAME`. `/DISPOSE:DELETE` will cause the control file to stay in your disk area until after the job is finished and then be deleted. `/DISPOSE:RENAME` will cause Batch to immediately move your control file to its own disk area where it will stay until the job is finished, at which time it will be deleted. You should use `/DISPOSE:RENAME` when you will be over your logged-out quota if the control file remains in your disk area when you log off the system.

Unless you have some use for the copy of the log file that will remain in your disk area even after it has been printed, use the `/DISPOSE:DELETE` switch to have the log file deleted after it is printed. If you do not delete the log file and you run the job again using the same log filename, your new log file will be appended to the old log file and they will both be printed as part of the new job.

The switches, and the assumptions made if they or their values are omitted, are all subject to change by each installation. Check with the installation where you run your jobs to find out what differences exist between the values described here and those at the installation. Additional switches are available for use with the `SUBMIT` command. For information about these switches, refer to the `SUBMIT` command in Chapter 2 of the DECsystem-10 Operating System Commands manual (DEC-10-MRDC-D). You can obtain further information about Batch in Chapter 3 of the aforementioned manual.

#### 3.2.4 Examples of Submitting Jobs

The following are sample jobs that are entered to Batch by means of the `SUBMIT` command. The jobs are shown in the following order.

1. Creating the control file.
2. Submitting the job to Batch using the `SUBMIT` command.

```
.COMPILE MYPROG.F4 /LIST/COMPILE  
.EXECUTE MYPROG.F4
```

After the control file to compile and execute the FORTRAN program has been written and saved, you must submit the job to Batch.

```
SUBMIT MYFILE
```

When Batch reads this SUBMIT command, it assumes the following:

1. The control filename and extension are MYFILE.CTL.
2. The name of the job is MYFILE.
3. The log file will be named MYFILE.LOG.
4. Both the control file and the log file will be saved in your disk area (/DISPOSE:PRESERVE).
5. An entry is being created in Batch's queue (/CREATE).
6. No cards will be punched by the job (/CARDS:0).
7. The maximum amount of core to be used to run the job is 25K (/CORE:25).
8. No paper tape will be punched (/FEET:0).
9. 200 is the maximum number of pages to be printed (/PAGE:200).
10. The maximum amount of CPU time is 5 minutes (/TIME:5:00).
11. No plotter time will be used (/TPLOT:0).

The next example shows the control file that was created at the beginning of this chapter being submitted to Batch.

```
.COMPILE MYPROG,F4/COMPILE
.EXECUTE MYFILE,F4
.COMPILE PROG2,F4/COMPILE
.EXECUTE PROG2,F4
```

After you have saved the control file, you must submit the job to Batch.

```
SUBMIT = MYFILE,MYFILE.LOG/DISPOSE:DELETE/TIME:20:00/CARDS:500
```

When Batch reads this request, it assumes the following:

1. The name of the job is MYFILE.
2. The name of the control file is MYFILE.CTL.
3. The log file will be named MYFILE.LOG.
4. An entry is being created in Batch's queue (/CREATE).
5. The log file will be deleted after it is printed (/DISPOSE:DELETE).
6. The control file will be saved in your disk area (/DISPOSE:PRESERVE).
7. A maximum of 500 cards can be punched by the job (/CARDS:500).
8. The maximum amount of core that can be used is 25K (CORE:25).
9. No paper tape will be punched by the job (/FEET:0).
10. 20 is the maximum number of pages that can be printed (/PAGE:20).
11. The maximum amount of CPU time that the job can use is 20 minutes (/TIME:20:00).
12. No plotter time will be used (/TPLOT:0).

If you made an error in the SUBMIT command when you submitted either of these jobs, Batch will type an error message on your terminal to explain your error so that you can correct it.



If you do not specify a character in the .ERROR command, Batch uses the standard error character, the question mark. When a line that is preceded by the character that you specify in the .ERROR command is passed to Batch from the monitor, a system program or is issued by Batch itself, Batch treats the line as a fatal error and terminates the job, exactly as it would if the line were preceded by a question mark. Any messages preceded by other characters will not be recognized by Batch as errors. The only exception is the ?TIME LIMIT EXCEEDED message. No matter what character you specify as the beginning of an error, Batch will recognize this message and terminate your job.

If you do not include the .ERROR command in your control file, Batch will recognize the question mark as the beginning character of a fatal error message, unless you include the .NOERROR command in your control file to cause Batch to ignore fatal errors (refer to Section 3.3.5).

An example of the .ERROR command follows.

```

      .
      .
      .
      .ERROR %
      .
      .
      .
      .ERROR
  
```

In this example, you specify in the middle of the control file that you want Batch to recognize the percent sign (%) as the beginning character of a fatal error from that point in the control file. Further on in the control file, you tell Batch to go back to recognizing the question mark as the beginning of a fatal error message.

### 3.3.3 The .GOTO Command

You can include the .GOTO command in your control file to direct Batch to skip over lines in the control file to find a specific line. The .GOTO command has the form:

```
.GOTO label
```

label

is a 1- to 6-character alphanumeric label for a statement. It must be followed by a double colon (::) when it labels a statement to show that it is a label.

When Batch encounters a .GOTO command in the control file, it searches forward in the control file to find the label specified in the .GOTO command. Batch then resumes processing of the control file at the line with the specified label. If Batch cannot find the labelled line, it terminates your job.

If you do not include a .GOTO command in the control file, Batch reads the control file sequentially from the first statement to the last, unless you include a .BACKTO statement (refer to Section 3.3.1).

An example of the .GOTO command follows.

```

      .
      .
      .GOTO ABC
      .
      .
      ABC;; ,DIRECT
  
```

You can use the .GOTO command as the statement in an .IF command (refer to Section 3.3.4) to aid you in error processing. For example:

```

      .
      .
      .IF (ERROR) ,GOTO ABC
      .
      .
      ABC;; ,TYPE MYPROG
  
```

### 3.3.4 The .IF Command

You can include the .IF command in your control file to specify an error recovery procedure to Batch or to specify normal processing if an error does not occur. The .IF statement has the forms:

```

      .IF (ERROR) statement
      .IF (NOERROR) statement
      statement
  
```

is a command to the monitor, to a program, or to Batch.

The .IF command can be used in two ways as shown in its two forms. You can include the .IF (ERROR) command in your control file at the place where you may have an error. The .IF (ERROR) command must be the next monitor-level line (as opposed to a line in your program or a line of data) in your control file after an error occurs so that Batch will not terminate your job. In the .IF (ERROR) command, you direct Batch to either go back or forward in your control file to find a line that will perform some task for you, or direct Batch to perform a task for you at that point in your control file, or to direct the monitor or any other program to perform some task for you.

You can use the .IF (NOERROR) command also to direct Batch or the monitor to perform tasks for you when an error does not occur at the point in your control file where you place the .IF (NOERROR) command. Thus, if you expect that an error will occur in your program, you can include an .IF (NOERROR) command to direct Batch in case the error does not occur, and then put the error processing lines immediately following the command. Refer to Section 3.4 for an example of using .IF (NOERROR) and .IF (ERROR).

If an error occurs and Batch does not find an .IF command as the next monitor-level line in the control file, Batch writes an error message in the log file and terminates the job. If one of your



### 3.4 SPECIFYING ERROR RECOVERY IN THE CONTROL FILE

If you don't specify error recovery when an error occurs in your job, Batch terminates the job and, if the error occurs when one of your programs is running, causes a dump of your core area. You can specify error recovery in the control file by means of the Batch commands, especially the .IF command. You must include the .IF command at the point between programs in the control file that an error may occur. When an error occurs, Batch examines the next monitor-level line (i.e., not a line that contains data or a command string to a system program) to find an .IF command to tell it what to do with the error. If an error does not occur and an .IF (ERROR) command is present, the .IF (ERROR) command is not executed. Similarly, if an error does not occur and you have included an .IF (NOERROR) command, the .IF command is processed. Thus, if you have a program that you are not sure is error-free, you can include an .IF command to tell Batch what to do if an error occurs, as shown in the following example.

```
.COMPILE MYPROG.F4
.IF (ERROR) STATEMENT
:
:
:
```

In either the .IF (ERROR) or the .IF (NOERROR) command, you must include a statement that tells Batch what to do. You can use any monitor command or one of the Batch commands. The .GOTO and .BACKTO commands are commonly used for this purpose. Refer to Sections 3.3.1 and 3.3.3 for descriptions of these commands. Be sure, if you use .GOTO or .BACKTO in the .IF command, that you supply a line in the control file that has the label that you specified in the .GOTO or .BACKTO command.

Two sample jobs are shown below. The first shows the .IF (ERROR) command and the .GOTO command to specify error recovery. The second example shows the use of the .IF (NOERROR) and .GOTO commands.

If you have a program that you are not sure will compile without errors, you can include another version of the same program in your job (that hopefully will compile) and tell Batch to compile the second program if the first has an error. You write the control file as follows.

```
.COMPILE /COMPILE MYPROG.F4 /LIST
.IF (ERROR) ,GOTO A
.EXECUTE MYPROG.F4
.GOTO B
A:: !CONTINUE
.COMPILE /COMPILE PROG2.F4 /LIST
.EXECUTE PROG2.F4
B:: !CONTINUE
```

When the job is run, Batch reads the control file and passes commands to the monitor. If an error occurs in the compilation of the first program, Batch finds the .IF (ERROR) command and executes the .GOTO command contained in it. The command tells Batch to look for the line labelled A, which contains a comment, so Batch continues to the end of the control file. If an error does not

occur in the first program, Batch skips the .IF (ERROR) command, executes the program with its data, skips the unnecessary error procedures, and continues to the end of the control file.

A variation of the above procedure is shown below using the .IF (NOERROR) command and the .GOTO command. The difference is that Batch skips the .IF (NOERROR) command if an error occurs, and performs it if an error does not occur. The following is the control file that you would create.

```
.COMPILE /COMPILE MYPROG.F4 /LIST
.IF (NOERROR) ,GOTO A
.COMPILE /COMPILE PROG2.F4 /LIST
.EXECUTE PROG2.F4
.GOTO B
A!! ICONTINUE
.EXECUTE MYPROG.F4
B!! ICONTINUE
```

When the job is run, Batch passes the COMPILE command to the monitor to compile the first program. If an error does not occur, the .IF (NOERROR) command and the .GOTO command are executed. Batch skips to the line labelled A, which is a comment, and continues reading the control file. The program MYPROG.F4 is executed with its data and the end of the job is reached. If an error occurs, Batch skips the .IF (NOERROR) command and continues reading the control file. PROG2.F4 is compiled and then executed with the same data that the first program would have used. Batch is then told to go to the line labelled B, which is a comment line. The end of the job follows.

The examples shown above illustrate only two ways that you can specify error recovery in the control file. You can also use the other Batch commands, or any monitor command that you choose to help you recover from errors in your job.

You do not have to attempt to recover from errors while your job is running. You can correct your errors according to the error messages in the log file when your job is returned to you, and then run your job again. Batch will also print a dump of your core area if an error occurs while your job is running and you have not specified error recovery. If you can read dumps, this can also aid you to correct your errors. The log file and dumps are described in Chapter 4.



## CHAPTER 4

# INTERPRETING YOUR PRINTED OUTPUT

You can receive three kinds of printed output from your Batch job:

1. Output that you request, i.e., the results of your job.
2. Output from Batch, i.e., the log file.
3. Output that is the result of actions by your job or by Batch, the monitor, or system programs. Examples of this output are compilation listings, cross-reference listings, error messages, and core dumps requested by Batch.

### 4.1 OUTPUT FROM YOUR JOB

Although this chapter deals mainly with printed output, you can have output to any device that the installation supports, as long as the installation allows you to use these devices. If your output is directed to the line printer, it will be printed separate from the log file. The printed output from each program will be preceded by two pages containing your name and project-programmer number and other pertinent information. Following these pages are two header pages containing the name of your output file in block letters. The output follows these header pages. A trailer page follows your output. This page contains the same information that is on the first two pages. The header and trailer pages also include three rows of numbers (read vertically from 001 to 132).

If your output is that which would normally be sent to the terminal, it will be printed in the log file. In the sample output shown in Section 4.4, the output from the program is included in the log file because it is directed to the terminal rather than the line printer.

### 4.2 BATCH OUTPUT

The output from Batch consists of a log file that contains all the statements in the control file, commands sent to the monitor from Batch for you, and the replies to the commands from the monitor and system programs like the compilers. Any error message sent from the monitor or a system program, or from Batch itself, is also written in the log file. Refer to the DECsystem-10 Operating System Commands manual (DEC-10-MRDC-D) for a list of the error messages from the monitor. The messages from each system program are listed in the applicable manuals.

You can ignore most of the information in the log file because it is system information and need not concern you. If you wish, you can keep it for reference by system programmers if unexpected results occur in your job.

### 4.3 OTHER PRINTED OUTPUT

Other output that you can get as a result of your job includes compiler and cross-reference listings, loader maps for programs that were successfully loaded, and dumps that you can request or that Batch gives to you when an error occurs in your program.

The compiler and cross-reference listings are those listings generated by the compiler if you request them. When you enter your job from cards, Batch requests compilation listings for you unless you specify otherwise. Cross-reference listings are generated for you only if you specifically ask Batch for them. When you enter your job from a terminal, you must request the listings in the COMPILE command. Also, if you request a cross-reference listing, you must run the CREF program (by means of the CREF command) to get your listing printed.

If you enter your job from cards and include a \$DATA card to request execution of a program, Batch requests a loader map for you. This map shows the locations in memory into which your program was placed. If you enter your job from a terminal, you must request a loader map in the EXECUTE command if you wish to have one. If you wish to know the locations into which your program was loaded, the loader map can be of use to you. Otherwise, you can ignore it. A loader map is shown in the sample output in Section 4.4, however, it is not interpreted in this manual.

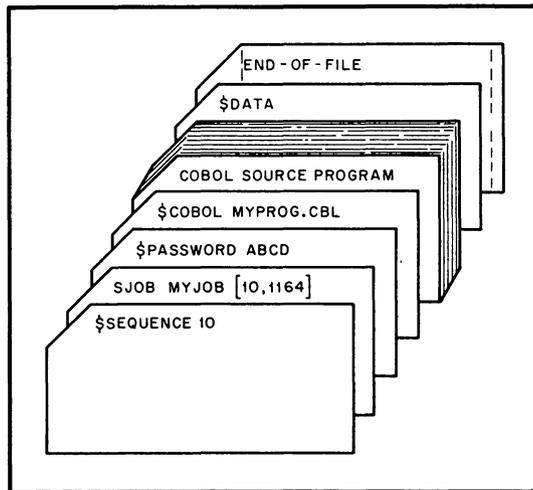
If a fatal error occurs in a program in your job and you have not included an error recovery command to Batch, Batch will not try to recover from the error for you. Instead, it will write the error message in the control file, request dump of your memory area, and terminate your job. The dump is then printed with your output. If you can read dumps, the dump that Batch requests for you may be helpful in finding your errors. Otherwise, you can ignore the dump and use the error messages to locate the errors in your program. A sample dump is shown in Section 4.4, but it is not interpreted. It is shown so that you can recognize it if you ever receive one.

### 4.4 SAMPLE BATCH OUTPUT

Two sample jobs and their output are shown in the following sections. The first shows a job entered from cards, the second shows a job entered from a terminal. The log file is somewhat different for the two types of jobs. Following the sample jobs is a sample dump.

#### 4.4.1 Sample Output from a Job on Cards

This example shows a job in which a small COBOL program is compiled and executed. The card deck is as follows.



The COBOL program is as follows.

10-0924

```

IDENTIFICATION DIVISION.
PROGRAM-ID. MYPROG.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
START.
DISPLAY "THIS IS TO SHOW SAMPLE OUTPUT FROM MPB.".
DISPLAY "THESE TWO LINES ARE OUTPUT FROM THE PROGRAM.".
STOP RUN.

```

When the job is run, the program is compiled and a compilation listing is produced. The listing is shown below. Note that the compiler put sequence numbers on the program even though they were not in the original program.

```

PROGRAM MYPROG,                COBOL 3(43)      21-MAR-72  10:42

0001  IDENTIFICATION DIVISION.
0002  PROGRAM-ID. MYPROG.
0003  ENVIRONMENT DIVISION.
0004  DATA DIVISION.
0005  PROCEDURE DIVISION.
0006  START.
0007  DISPLAY "THIS IS TO SHOW SAMPLE OUTPUT FROM MPB.".
0008  DISPLAY "THESE TWO LINES ARE OUTPUT FROM THE PROGRAM.".
0009  STOP RUN.

NO ERRORS DETECTED

```

After the program is compiled, it is loaded and executed. Since Batch requests a loader map when it puts the EXECUTE command in the control file, the loader map is the next thing printed from your job. It is shown below. Note that each of these print-outs are preceded by headers, which are not shown in these examples.

001246 IS THE LOW SEGMENT BREAK

MAP STORAGE MAP 10142 21-MAR-72

STARTING ADDRESS 001200 PROG COBOL FILE MYPROG

.COMM, 000140 001040

MYPROG, 001200 001103

START, 001200  
 ALTER, 000143  
 MONEY, 000147  
 TRAC3, 000154

FILES, 000140  
 OVRFN, 000144  
 MEMRY, 000150  
 XNM, 000155

USES, 000141  
 POINT, 000145  
 TRAC1, 000152  
 XDT, 000156

SEGWD, 000142  
 COMMA, 000146  
 TRAC2, 000153  
 XPR, 000157

TRACED 001243 000003

BTRAC, 001244  
 TRPOP, 001244

PTFLG, 001245

TRACE, 001243

TRPD, 001244

COBOL 1K CORE, 345 WORDS FREE  
 LOADER USED 2+4K CORE

Following loading, the program is executed. The program in this example does not have output to the line printer, instead its output is written to a terminal. Because this is a Batch job, the terminal output is written in the log file. The log file is printed next because the end of the job is reached. The log file contains all the dialog between your job and the monitor and system programs, and some commands that Batch sent to the monitor for you. An annotated log file is shown on the following pages. Note that each line in the log file is preceded by the time of day when the line was written. Following the time is a word that describes what kind of information is on each line. You do not need to know what each of these words means because much of the information is system information.

```

10141143 DATE      21-MAR-72      554A1E DUAL CPU   CDRSTK VER 12(17) DSK
10141143 CARD      $JOB MYJOB [10,1164]
10141144 STSUM     END OF FILE AFTER 15 CARDS, 04 FILES, 03 BLKS

10141150 BVERS     BATCON 7(35) INP: SUBJOB 01 OF 06
10141150 BDATE     21-MAR-72
10141150 BASUM     MYJOB[10,1164] FOR **[10,1164] LOG FILE IN [10,1164]
                    REQUEST CREATED AT 10:41:00 21-MAR-72
                    UNIQUE: 2 RESTART: 1

10141150 MONTR
1014115  MONTR     .LOGIN 10/1164
10142101 USER      JOB 24      554A1E DUAL CPU TTY102
10142101 USER      OTHER JOBS SAME PPN
10142101 USER      1041      21-MAR-72      TUE
10142101 MONTR
10142101 MONTR     .SET TIME 300.
10142108 MONTR
10142108 MONTR     .SET SPOOL ALL
10142108 MONTR
10142108 MONTR
10142108 MONTR     .
                    $SEQUENCE 10
                    $JOB MYJOB [10,1164]
                    $COBOL MYPROC.CBL
10142108 MONTR     .COMP /COMPILE MYPROC.CBL/LIST ;CREATED BY CDRSTK
10142110 USER      COBOL: MYPROC
10142132 MONTR     EXIT
10142132 MONTR
10142132 MONTR

```

This is system information that Batch enters. It need not concern you.

Batch logs your job into the system. The information that follows it is the system response.

These are commands that Batch entered for you.

These are the cards that you entered.

This is the command entered by Batch for you.

The answer to the COMPILE command from the monitor.

```

$DATA
10142:32 MONTR .SET CDR QAA,CDR ;CREATED BY CDRSTK
10142:32 MONTR
10142:32 MONTR .,EXEC /MAP:MAP.LPT /REL MYPROG.REL ;CREATED BY CDRSTK
10142:39 USER LOADING
10143:00 USER 001246 IS THE LOW SEGMENT BREAK
10143:00 USER
10143:00 USER COBOL 1K CORE
10143:00 USER EXECUTION
10143:00 USER THIS IS TO SHOW SAMPLE OUTPUT FROM MPB.
10143:00 USER THESE TWO LINES ARE OUTPUT FROM THE PROGRAM,
10143:00 MONTR
10143:00 MONTR EXIT
10143:00 MONTR
10143:00 MONTR .
10143:00 MONTR *FINI
10143:21 USER .DEL MYPROG.REL,QAA,CDR,MYPROG.CBL
10143:21 USER FILES DELETED:
10143:21 USER MYPROG.REL
10143:21 USER QAA,CDR
10143:21 USER MYPROG.CBL
10143:21 USER 03 BLOCKS FREED
10143:21 MONTR
10143:21 MONTR .
10143:22 MONTR .KJOB DSKB1:MYJOB.LOG[10,1164]=/W/Z:4/B/VS:10/VL:1200/VD:10
10143:25 K-QUE TOTAL OF 7 BLOCKS IN LPT REQUEST
10143:30 KJOB OTHER JOBS SAME PPN
10143:43 LGOUT JOB 24, USER [10,1164] LOGGED OFF TTY102 1043 21-MAR-72
10143:45 LGOUT SAVED ALL 4 FILES (25, DISK BLOCKS)
10143:45 LGOUT ANOTHER JOB STILL LOGGED IN UNDER [10,1164]
10143:45 LGOUT RUNTIME 0 MIN. 03,97 SEC

10143:54 LPMMSG LPTSPL VERSION 4(125) RUNNING ON LPT3
10144:20 LPMMSG JOB MYJOB FILE DSKB1:MYPROG.LST[10,1164] FOR [10,1164] STARTED
10145:20 LPMMSG DSKB1:MYPROG.LST[10,1164] DONE
10145:21 LPMMSG JOB MYJOB FILE DSKB1:MAP.LPT[10,1164] FOR [10,1164] STARTED
10146:25 LPMMSG DSKB1:MAP.LPT[10,1164] DONE

```

} Your \$DATA card.

} Commands entered by Batch for you.

} Monitor response to the EXECUTE command.

} This is the output from your program.

} Monitor indicates that execution of your program has ended.

} Command entered by Batch.

} Response to the DELETE Command.

} This is the LOGOUT dialog, giving system information.

} This is more system information.

## 4.4.2 Sample Output from a Job from a Terminal

This example shows the same job described above as it would be entered from a terminal. You would first create the program as a file on disk.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MYPROG.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
START.
DISPLAY "THIS IS TO SHOW SAMPLE OUTPUT FROM MPB.".
DISPLAY "THESE TWO LINES ARE OUTPUT FROM THE PROGRAM.".
STOP RUN.
```

Then you would make up a control file to compile and execute the COBOL program.

```
.COMPILE MYPROG.CBL
.EXECUTE MYPROG
```

You must then submit the job to Batch using the SUBMIT command.

```
SUBMIT MYJOB
```

When the job is run, the program is compiled and a listing is produced, even though you did not request it. This is because the COBOL compiler always produces a listing. Note that the compiler adds sequence numbers to the listing, even though you did not include these numbers on the program.

```
P R O G R A M   M Y P R O G .                COBOL 3(43)      22-MAR-72 15:10
0001  IDENTIFICATION DIVISION.
0002  PROGRAM-ID. MYPROG.
0003  ENVIRONMENT DIVISION.
0004  DATA DIVISION.
0005  PROCEDURE DIVISION.
0006  START.
0007  DISPLAY "THIS IS TO SHOW SAMPLE OUTPUT FROM MPB.".
0008  DISPLAY "THESE TWO LINES ARE OUTPUT FROM THE PROGRAM.".
0009  STOP RUN.
NO ERRORS DETECTED
```

Because you did not request it specifically in the EXECUTE command, you will not get a loader map of your program. The log file is printed next as the last of your output. The output from the program is written in the log file because it is output to the terminal and the log file simulates a terminal dialog. The log file also contains some commands that Batch sent to the monitor for you and some additional system information. An annotated log file is shown on the following page. Note that each line in the log file is preceded by the time of day when the line was written. Following the time is a word that describes what kind of information is on each line. You do not have to know what each of these words means because much of the information is system information.

```

15109:06 BVERS  BATCON 7(36) INP: SUBJOB 01 OF 06
15109:06 BDATE  22-MAR-72
15109:06 BASUM  MPB[10,1164] FOR *SMITH *[10,1164] LOG FILE IN [10,1164]
                REQUEST CREATED AT 15:07:32 22-MAR-72
                UNIQUE: 2 RESTART: 0

15109:06 MONTR  .LOGIN 10,1164
15109:06 MONTR  .JOB 35      554A1F DUAL CPU TTY102
15109:27 USER   OTHER JOBS SAME PPN
15109:27 USER   1509      22-MAR-72          WED
15109:51 USER

15110:06 MONTR  .SET TIME 300
15110:06 MONTR  .SET SPOOL ALL
15110:07 MONTR
15110:07 MONTR

15110:07 MONTR  ..COMPILE MYPROG.CBL
15110:31 USER   COBOL: MYPROG.
15111:18 MONTR  EXIT
15111:18 MONTR  MONTR
15111:18 MONTR  ..EXECUTE MYPROG.CBL
15111:30 USER   LOADING
15111:30 USER
15111:30 USER   COBOL 1K CORE
15111:30 USER   EXECUTION
15111:30 USER   THIS IS TO SHOW SAMPLE OUTPUT FROM MPB.
15111:30 USER   THESE TWO LINES ARE OUTPUT FROM THE PROGRAM.
15111:30 MONTR  EXIT
15111:30 MONTR
15111:30 MONTR
15111:30 MONTR  .
15111:30 MONTR  .KJOB USKB0:MPB.LOG[10,1164]=W/Z:4/B/VR:10/VS:425/V/L:120Z/VD:1P
1511:58 K=QUE  TOTAL OF 4 BLOCKS IN LPT REQUEST
1515:35 LGOUT  JOB 35, USER [10,1164] LOGGED OFF TTY102 1515 22-MAR-72
1515:36 LGOUT  SAVED ALL 10 FILES (125, DISK BLOCKS)
1515:36 LGOUT  RUNTIME 0 MIN, 04.05 SEC
1515:44 LPMMSG LPTSPL VERSION 4(125) RUNNING ON LPT3
1516:06 LPMMSG JOB MPB FILE DSKB1:MYPROG.LST[10,1164] FOR [10,1164] STARTED
1517:05 LPMMSG DSKB1:MYPROG.LST[10,1164] DONE

```

} This is sytem information that Batch enters. It need not concern you.

} Batch logs your job into the system. The information that follows is the system response.

} These are commands that Batch enters for you.

} This is the command from your control file and the response.

} This is another command from your control file and its response.

} This is the output from your program.

} This indicates that execution has ended.

} This is the LOGOUT dialog, which gives system information.

} This is more system information.

4-9

- 161 -

BEGINNER'S BATCH

#### 4.4.3 Sample Dump

Shown on the following pages is the log file containing an error message and the dump that Batch requested as a result of the message. The error resulted from use of a logical name in a program without assigning the logical name to a physical device at run time.

The dump lists the assembly language equivalent of your program, and the location in memory in octal, decimal, ASCII code, and SIXBIT code. (SIXBIT code is a compressed form of ASCII used in COBOL and some system programs.) Only the first three pages of the dump are shown.

```
14125142 BVERS  BATCON 7(36) INP: SUBJOB 01 OF 06
14125142 BDATE  22-MAR-72
14125142 BASUM  INJOB[10,1164] FOR *GORFINKLE *[10,1164] LOG FILE IN
                REQUEST CREATED AT 14:24:35 22-MAR-72
                UNIQUE! 2 RESTART! 0

14125142 MONTR
14125142 MONTR  .LOGIN 10,1164
14125145 USFR   JOB 30      554A1F DUAL CPU TTY102
14125145 USER   OTHER JOBS SAME PPN
14125145 USER   1425      22-MAR-72      WED
14125146 MONTR
14125146 MONTR  .SET TIME 300
14125146 MONTR
14125146 MONTR  .SET SPOOL ALL
14125146 MONTR
14125146 MONTR  .,COMPILE EXAMPLE,CBL
14125147 MONTR
14125147 MONTR  EXIT
14125147 MONTR
```

```

14:25:47 MONTR  ..EXECUTE EXAMPLE.CBL
14:25:50 USER   LOADING
14:25:55 USER
14:25:55 USER   COBOL 1K CORE
14:25:55 USER   EXECUTION
14:25:55 USER
14:25:55 USER   INIT TOOK THE ERROR RETURN
14:25:55 USER   DEVICE MAG1 IS NOT A DEVICE OR IS NOT AVAILABLE TO THIS JOB
14:25:55 USER   INIT TOOK THE ERROR RETURN
14:25:55 USER   DEVICE MAG2 IS NOT A DEVICE OR IS NOT AVAILABE TO THIS JOB
14:25:55 USER   ?LAST COBOL UO0 CALLED FROM USER LOCATION 4001155
14:25:55 MONTR
14:25:55 MONTR  EXIT
14:25:55 MONTR
14:25:55 MONTR  .

14:25:56 MONTR  .CLOSE
14:25:56 MONTR
14:25:56 MONTR
14:25:56 MONTR  .DUMP
14:26:04 USER
14:26:04 USER   0 SYMBOLS EXTRACTED
14:26:15 MONTR
14:26:15 MONTR  EXIT
14:26:15 MONTR
14:26:15 MONTR  .KJOB DSKB1:INJOB.LOG[10,1164]=/W/Z:4/B/VR:10/VS:422/VL:200/VD:P
14:26:17 K=QUE  TOTAL OF 30 BLOCKS IN LPT REQUEST
14:26:21 KJOB   OTHER JOBS SAME PPN

14:26:31 LGOUT  JOB 30, USER [10,1164] LOGGED OFF TTY102 1426 22-MAR-72
14:26:32 LGOUT  SAVED ALL 6 FILES (100. DISK BLOCKS)
14:26:32 LGOUT  ANOTHER JOB STILL LOGGED IN UNDER [10,1164]
14:26:32 LGOUT  RUNTIME 0 MIN, 10.79 SEC
14:26:40 LPMSG  LPTSPL VERSION 4(125) RUNNING ON LPT3
14:27:04 LPMSG  JOB INJOB FILE DSKB0:0300AE[10,1164] FOR [10,1164] STARTED
14:29:24 LPMSG  DSKB0:0300AE[10,1164] DONE

```

} This is the error message that caused Batch to request the dump.

QUICK DUMP VERSION X3(24) [FILE SYS:QUIKDM,CCL]

MONITOR INFORMATION

MONITOR NAME 554A1F DUAL CPU BUILT ON 03-21-72

SYSTEM SERIAL NUMBER IS 160

MONITOR VERSION IS 000000,050400

JOB INFORMATION

DUMP TAKEN 3-22-72 AT 14:25

DAEMON VERSION 6(21)-0

JOB NUMBER 30

TTY102 PPN [10,1164] CHARGE NUMBER 0

RUN TIME 00 MIN. 50 SECONDS

TOTAL KCS 6

TOTAL OF 128 DISK READS, 10 DISK WRITES

PRIV. BITS 0

THERE ARE 0 REAL TIME DEVICES IN USE

CURRENT HPQ IS 0 LAST HPQ COMMAND WAS 0

HISEG NAME DSKB:LIBOL .SHR

HISEG DIRECTORY [1,4]

USER NAME IS GORFINKLE

USER CORE LIMIT IS 261632 WORDS

USER TIME LIMIT IS 299 SECONDS

PROGRAM NAME IS COBOL

CORE INFORMATION

PC = 700000,057777 OPC = 000000,000000  
 LAST UO AT 440004,000006

SYMBOLIC LOCATIONS

PC = BLKI 57777  
 OPC = 7  
 LAST UO AT ANDCB 6(4)

ACS IN OCTAL:

|     |               |               |               |
|-----|---------------|---------------|---------------|
| 0/  | 010500,000002 | 000000,000000 | 522202,715530 |
| 3/  | 554147,220000 | 000000,000002 | 000000,000000 |
| 6/  | 000000,000000 | 200022,001361 | 000000,001777 |
| 11/ | 000000,000204 | 777777,000000 | 000000,001425 |
| 14/ | 310000,001412 | 004001,001424 | 000000,000000 |
| 17/ | 777601,001463 |               |               |

ACS IN DECIMAL:

|     |             |      |              |              |     |             |   |
|-----|-------------|------|--------------|--------------|-----|-------------|---|
| 0/  | 1157627906  | 0    | -23319569576 | -19837149184 | 2   | 0           | 0 |
| 7/  | 17184588529 | 1023 | 132          | -262144      | 789 | 26843546378 |   |
| 15/ | 537133844   | 0    | -33291469    |              |     |             |   |

SELECTED CORE AREAS DUMPED AS INSTRUCTION,OCTAL,DECIMAL,SIXBIT,ASCII

AROUND C(AC17) [HOPEFULLY A PUSH DOWN LIST]

|       |        |           |               |             |        |     |   |
|-------|--------|-----------|---------------|-------------|--------|-----|---|
| 1443/ | DPB    | 5,814     | 137240,001456 | 12792628014 | +Z# ,N | J   |   |
| 1444/ | UU0002 | 3,728     | 002140,001330 | 293602008   | 1# +8  | F   | L |
| 1445/ | UU0012 |           | 012000,000000 | 1342177280  | !0     | #   |   |
| 1446/ | JRST   | 795       | 254000,001433 | 23085450011 | 5# ,/  | +   |   |
| 1447/ | PUSHJ  | 15,815    | 260740,001457 | 23748150063 | 6!# ,0 | ,   |   |
| 1450/ | Z      | 17        | 000000,000021 |             | 17     |     |   |
| 1451/ | UU0001 | 1,728     | 001040,001330 | 142607064   | (# +8  | "   | L |
| 1452/ | UU0001 | 1,661     | 001040,001225 | 142606997   | (# *5  | "   | J |
| 1453/ | PUSHJ  | 15,0113   | 260760,000161 | 23752343665 | 6!P !Q | ,   | 8 |
| 1454/ | UU0001 | 5,753(14) | 001256,001361 | 179831537   | *N +Q  | *P  | X |
| 1455/ | CATL   | 8,699     | 301400,001273 | 25971131067 | 8, *{  | 00  | J |
| 1456/ | CATL   | 8,766     | 301400,001376 | 25971131134 | 8, +   | 00  |   |
| 1457/ | AOS    | (15)      | 350017,000000 | 31142445056 | = /    | ! X |   |
| 1460/ | POPJ   | 15,       | 263740,000000 | 24150802432 | 6?#    | ,   |   |
| 1461/ | Z      |           | 000000,000000 |             | 0      |     |   |
| 1462/ | Z      |           | 000000,000000 |             | 0      |     |   |
| 1463/ | UU0010 | 131191    | 010000,400167 | 1073073015  | ! #!W  |     | ! |
| 1464/ | CAM    | 131497    | 310000,400651 | 26843677097 | 9 #&I  | 2   | T |
| 1465/ | CAM    | 131555    | 310000,400743 | 26843677155 | 9 #!C  | 2   | Q |
| 1466/ | CAM    | 132709    | 310000,403145 | 26843678309 | 9 #9E  | 2   | 2 |
| 1467/ | Z      |           | 000000,000000 |             | 0      |     |   |
| 1470/ | Z      |           | 000000,000000 |             | 0      |     |   |
| 1471/ | Z      |           | 000000,000000 |             | 0      |     |   |
| 1472/ | Z      |           | 000000,000000 |             | 0      |     |   |
| 1473/ | Z      |           | 000000,000000 |             | 0      |     |   |
| 1474/ | Z      |           | 000000,000000 |             | 0      |     |   |
| 1475/ | Z      |           | 000000,000000 |             | 0      |     |   |
| 1476/ | Z      |           | 000000,000000 |             | 0      |     |   |
| 1477/ | Z      |           | 000000,000000 |             | 0      |     |   |
| 1500/ | Z      |           | 000000,000000 |             | 0      |     |   |
| 1501/ | Z      |           | 000000,000000 |             | 0      |     |   |
| 1502/ | Z      |           | 000000,000000 |             | 0      |     |   |
| 1503/ | Z      |           | 000000,000000 |             | 0      |     |   |

BEGINNER'S BATCH

- 168 -

## CHAPTER 5 PERFORMING COMMON TASKS WITH BATCH

This chapter shows some sample jobs that are run from a terminal and from cards. Section 5.1 illustrates entering jobs from a terminal. Section 5.2 shows entering jobs from cards. The examples are the same in both cases, the difference is only in the way that they are entered.

### 5.1 USING THE TERMINAL TO ENTER JOBS

#### ALGOL Example

The first job is a simple ALGOL program that writes output to the terminal. Since the job is being entered through Batch, the output is written in the log file instead of on the terminal.

```
BEGIN
  REAL X; INTEGER I;
  X := 1;
  FOR I := 1 UNTIL 1000 DO X := X+I;
  PRINT(X);
END
```

The control file for the program is as follows.

```
.COMPILE MYPROG.ALG/LIST
.EXECUTE MYPROG.ALG

SUBMIT MYFILE
```

When Batch starts the job, the statements in the control file call the ALGOL compiler to compile the program. Batch then calls the loader to load the program for execution. A listing of the program will be printed with the log file, as shown below.

```
DECSYSTEM 10 ALGOL-60, V. 2A(145);
13-APR-72 15:25:57

000003 B1 1 BEGIN
START OF BLOCK 1
000006 2 REAL X; INTEGER I;
000006 3 X :=1;
000016 4 FOR I :=1 UNTIL 1000 DO X :=X+I;
000023 5 PRINT(X);
000026 E1 6 END

END BLOCK 1, CONT 0
0 ERRORS
```

```

15:25:50 BVERS BATCON 7(52) INP: SUBJOB 01 OF 06
15:25:50 BDATE 13-APR-72
15:25:50 BASUM MYFILE[10,1461] FOR *SMITH *[10,1461] LOG FILE IN [10,1461]
REQUEST CREATED AT 15:24:39 13-APR-72
UNIQUE: 2 RESTART: 0

15:25:50 MONTR
15:25:50 MONTR .LOGIN 10/1461
15:25:51 USER JOB 20 55425E DUAL CPU TTY110
15:25:51 USER OTHER JOBS SAME PPN
15:25:51 USER 1525 13-APR-72 THUR
15:25:52 MONTR
15:25:52 MONTR .SET TIME 300
15:25:52 MONTR
15:25:52 MONTR .SET SPOOL ALL
15:25:53 MONTR
15:25:53 MONTR ..COMPILE MYPROG.ALG/LIST
15:25:56 USER ALGOL: MYPROG
15:25:57 MONTR
15:25:57 MONTR EXIT
15:25:58 MONTR
15:25:58 MONTR ..EXECUTE MYPROG.ALG
15:25:58 USER LOADING
15:26:06 USER
15:26:06 USER MYPROG 1K CORE
15:26:06 USER EXECUTION
15:26:07 USER 5.0050100& 5
15:26:07 USER
15:26:07 USER END OF EXECUTION - 1K CORE
15:26:07 USER
15:26:07 USER EXECUTION TIME: 0.08 SECS.
15:26:07 USER
15:26:07 USER ELAPSED TIME: 0.15 SECS.
15:26:07 MONTR
15:26:07 MONTR .
15:26:07 MONTR .KJOB DSKB0:MYFILE.LUG[10,1461]=/W/Z:4/B/VR:10/V:384/VL:200/VD:P
15:26:08 K-QUE TOTAL OF 3 BLOCKS IN LPT REQUEST
15:26:12 KJOB OTHER JOBS SAME PPN
15:26:15 LGOUT JOB 20, USER [10,1461] LOGGED OFF TTY110 1525 13-APR-72
15:26:15 LGOUT SAVED ALL 40 FILES (650. DISK BLOCKS)
15:26:15 LGOUT ANOTHER JOB STILL LOGGED IN UNDER [10,1461]
15:26:15 LGOUT RUNTIME 0 MIN, 03.25 SEC
15:26:21 LPMSG LPTSPL VERSION 4A(141) RUNNING ON LPT2
15:26:42 LPMSG JOB MYFILE FILE DSKB0:MYPROG.LST[10,1461] FOR [10,1461] STARTED
15:27:35 LPMSG DSKB0:MYPROG.LST[10,1461] DONE

```

BEGINNER'S BATCH

- 170 -

BASIC Example

The next sample shows how to enter a BASIC program to Batch. You must make up the file and save it on disk. Then make up a control file that simulates the dialog with the BASIC system. The program is shown below.

```
5 INPUT D
10 IF D = 2 THEN 110
20 PRINT "X VALUE", "SINE", "RESOLUTION"
30 FOR X=0 TO 3 STEP D
40 IF SIN(X) <= M THEN 80
50 LET X0=X
60 LET M=SIN(X)
80 NEXT X
90 PRINT X0, M, D
100 GO TO 5
110 END
```

The program requests data from the user when it is running. You include the data in the control file. The final data item must be 2 to conclude the program. The control file follows.

```
.R BASIC
*OLD
*DSK:MYBAS.BAS
*RUN
.1
.01
.001
2
*BYE
```

The output from the program will be printed in the control file because it would normally be printed on the terminal. The command to submit the job to Batch is as follows.

```
SUBMIT = BAS.CTL
```

```
15:41:37 BVERS BATCON 7(52) INP: SUBJOB 02 OF 06
15:41:37 BDATE 13-APR-72
15:41:37 BASUM BAS[10,1461] FOR *SMITH *[10,1461] LOG FILE IN [10,1461]
REQUEST CREATED AT 15:40:23 13-APR-72
UNIQUE: 2 RESTART: 0

15:41:37 MONTR
15:41:37 MONTR .LOGIN 10/1461
15:41:39 USER JOB 15 55425E DUAL CPU TTY115
15:41:40 USER OTHER JOBS SAME PPN
15:41:40 USER 1541 13-APR-72 THUR
15:41:41 MONTR
15:41:41 MONTR .SET TIME 300
15:41:41 MONTR
15:41:41 MONTR .SET SPOOL ALL
15:41:41 MONTR
15:41:41 MONTR ..R BASIC
15:41:41 USER
15:41:42 USER
15:41:42 USER NEW OR OLD--*OLD
```

```

15:41:42 USER OLD FILE NAME--*DSK:MYBAS
15:41:43 USER
15:41:43 USER READY
15:41:43 USER *RUN
15:41:47 USER
15:41:47 USER MYBAS          15:41          13-APR-72
15:41:47 USER
15:41:47 USER
15:41:47 USER ? .1
15:41:47 USER X VALUE          SINE          RESOLUTION
15:41:47 USER 3.          0.14112          0.1
15:41:47 USER ? .01
15:41:47 USER X VALUE          SINE          RESOLUTION
15:41:48 USER 3.          0.141121          0.01
15:41:48 USER ? .001
15:41:48 USER X VALUE          SINE          RESOLUTION
15:41:48 USER 2.99999          0.14113          0.001
15:41:49 USER ?2
15:41:49 USER
15:41:49 USER TIME: 1.50 SECS.
15:41:49 USER
15:41:50 USER READY
15:41:50 USER *BYE
15:41:50 USER JOB 15, USER [10,1461] LOGGED OFF TTY115 1541 13-APR-72
15:41:52 USER SAVED ALL 33 FILES (600, DISK BLOCKS)
15:41:52 USER ANOTHER JOB STILL LOGGED IN UNDER [10,1461]
15:41:53 MONTR RUNTIME 0 MIN, 03.05 SEC

```

## FORTRAN Example

The third example shows a FORTRAN program that prints output on the line printer. In the control file, you want to tell Batch to delete your relocatable binary file if an error occurs when your program is executed. Otherwise, you want Batch to save your relocatable binary file as it normally would. The program is shown below.

```

C      THIS PROGRAM CALCULATES PRIME NUMBERS FROM 11 TO 50.
      DO 10 I =11,50,2
      J=1
4      J=J+2
      A=J
      A=I/A
      L=I/J
      B=A-L
      IF (B) 5,10,5
5      IF (J.LT.SQRT(FLOAT(I))) GO TO 4
      PRINT 105,I
10     CONTINUE
105    FORMAT (I4, ' IS PRIME. ')
      END

```

The control file to compile and execute this program, deleting the relocatable binary file if there is an execution error, is as follows.

```
.COMPILE MYPROG.F4
.EXECUTE MYPROG.F4
.IF (NOERROR) .GOTO END
.DELETE MYPROG.REL
END:; ;END OF JOB
```

The command to submit this job is as follows.

```
SUBMIT MYFOR.CTL,MYFOR.LOG/DISPOSE:DELETE
```

The log file will be deleted after the output has been printed.

```
09:50:07 BVERS  BATCON 7(52) INP: SUBJOB 02 OF 06
09:50:07 BDATE  14-APR-72
09:50:07 BASUM  MYFOR[10,1461] FOR *SMITH :[10,1461] LOG FILE IN [10,1461]
                REQUEST CREATED AT 09:49:19 14-APR-72
                UNIQUE: 2 RESTART: 0

09:50:07 MONTR  .LOGIN 10/1461
09:50:07 MONTR  .JOB 23      55425I DUAL CPU TTY115
09:50:09 USER   OTHER JOBS SAME PPN
09:50:13 USER   0950      14-APR-72      FRI
09:50:13 MONTR  .SET TIME 300
09:50:14 MONTR  .SET SPOOL ALL
09:50:14 MONTR  ..COMPILE MYPROG.F4
09:50:16 USER   FORTRAN: MYPROG.F4
09:50:17 MONTR  EXIT
09:50:17 MONTR  ..EXECUTE MYPROG.F4
09:50:17 USER   LOADING
09:50:23 USER   MYPROG 2K CORE
09:50:23 USER   EXECUTION
09:50:23 USER   11 IS PRIME.
09:50:23 USER   13 IS PRIME.
09:50:23 USER   17 IS PRIME.
09:50:23 USER   19 IS PRIME.
09:50:23 USER   23 IS PRIME.
09:50:23 USER   29 IS PRIME.
09:50:23 USER   31 IS PRIME.
09:50:23 USER   37 IS PRIME.
09:50:23 USER   41 IS PRIME.
09:50:23 USER   43 IS PRIME
09:50:23 USER   47 IS PRIME.
09:50:23 USER   CPU TIME:  37  ELAPSED TIME: 0.60
09:50:23 USER   NO EXECUTION ERRORS DETECTED
09:50:25 MONTR  EXIT
09:50:25 MONTR
09:50:27 MONTR
09:50:27 MONTR  .
                END:;
                ;END OF JOB
```

```

09:50:27 MONTR .KJOB DSKB1:MYFOR.LOG[10,1461]=/W/Z:4/B/VR:10/VS:420/VL:200/VD:P
09:50:28 K-QUE TOTAL OF 3 BLOCKS IN LPT REQUEST
09:50:32 KJOB OTHER JOBS SAME PPN
09:50:34 LGOUT JOB 23, USER [10,1461] LOGGED OFF TTY115 0950 14-APR-72
09:50:34 LGOUT SAVED ALL 33 FILES (610. DISK BLOCKS)
09:50:34 LGOUT ANOTHER JOB STILL LOGGED IN UNDER [10,1461]
09:50:34 LGOUT RUNTIME 0 MIN, 05.39 SEC

```

## COBOL Example

The fourth program shows a COBOL program that reads a magnetic tape and writes output on another magnetic tape. To have your magnetic tapes mounted on drives and assigned to you, you must request that the operator mount them. Since you do not know which drives will be assigned to your job, you must assign them in your job with logical device names. The MOUNT command assigns the drive to your job and associates the logical name that you specify in it with the physical drive assigned. You should include a PLEASE command to the operator to tell him that you want two magnetic tape drives. If he can't let you have the drives because they are in use, you can ask him to enter your job again. Your magnetic tapes, one with the input data, the other blank so that you can write on it, should be given to the operator or kept at the central site, so that the operator can find your tapes. The program is as follows.

```

IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT INFIL, ASSIGN MA
    SELECT OUTFIL, ASSIGN MAG2.
DATA DIVISION.
FILE SECTION.
FD  INFIL, LABEL RECORDS ARE STANDARD,
    VALUE OF IDENTIFICATION IS "INFIL DAT",
    BLOCK CONTAINS 20 RECORDS.
01  INREC, PIC X(80).
FD  OUTFIL, LABEL RECORDS ARE STANDARD,
    VALUE OF IDENTIFICATION IS "OUTFIL DAT",
    BLOCK CONTAINS 12 RECORDS.
01  OUTREC, PIC X(80).
PROCEDURE DIVISION.
START.
    OPEN INPUT INFIL, OUTPUT OUTFIL.
LOOP.
    READ INFIL; AT END GO TO FIN.
    WRITE OUTREC FROM INREC.
    GO TO LOOP.
FIN.
    CLOSE OUTFIL, INFIL.
    STOP RUN.

```

The control file and the SUBMIT command to enter this program to Batch is as follows.

```

.PLEASE NEED TWO MAGTAPES, IF I CAN'T HAVE THEM, REQUEUE.
.MOUNT MTA:MAG1/VID:INFIL /RONLY
.MOUNT MTA:MAG2/VID:OUTFIL/WENABLE
.COMPILE MYPROG.CBL
.EXECUTE MYPROG.CBL
.DISMOUNT MAG1:
.DISMOUNT MAG2:
.DELETE MYPROG.*

.SUBMIT MYJOB=MYJOB.CTL

```

The log file is shown below.

```

11:53:26 BVERS  BATCON 7(53) INP: SUBJOB 01 OF 06
11:53:26 BDATE  20-APR-72
11:53:26 BASUM  MYJOB[10,1461] FOR *SMITH *[[10,1461] LOG FILE IN [10,1461]
                REQUEST CREATED AT 11:52:31 20-APR-72
                UNIQUE: 2 RESTART: 0

11:53:26 MONTR
11:53:26 MONTR  .LOGIN 10/1461
11:53:30 USER  JOB 17      554250 DUAL CPU TTY103
11:53:30 USER  OTHER JOBS SAME PPN
11:53:30 USER  1153      20-APR-72      THURS
11:53:30 MONTR
11:53:30 MONTR  .SET TIME 300
11:53:30 MONTR
11:53:30 MONTR  .SET SPOOL ALL
11:53:30 MONTR
11:53:30 MONTR  ..PLEASE NEED TWO MAG TAPES, IF I CAN'T HAVE THEM, REQUEUE.
11:53:50 MONTR  .MOUNT MTA:MAG1/VID:INFIL/RONLY
11:53:50 USER  OPERATOR NOTIFIED
11:53:52 USER  WAITING...
11:54:19 USER  MAG1 (MTA1) MOUNTED
11:54:19 USER
11:54:21 MONTR  ..MOUNT MTA:MAG2/VID:OUTFIL/WENABL
11:54:22 USER  OPERATOR NOTIFIED
11:54:25 USER  WAITING...
11:57:23 USER  MAG2 (MTA2) MOUNTED
11:57:23 USER
11:57:23 MONTR  ..COMPILE MYPROG.CBL
11:57:25 MONTR
11:57:25 MONTR  EXIT
11:57:25 MONTR
11:57:25 MONTR  ..EXECUTE MYPROG.CBL
11:57:48 USER  LOADING
11:58:05 USER
11:58:05 USER  COBOL 1K CORE
11:58:05 USER  EXECUTION
11:58:05 MONTR
11:58:05 MONTR  EXIT
11:58:05 MONTR  ..DISMOUNT MAG1:
11:58:12 USER  OPERATOR NOTIFIED
11:58:12 USER  WAITING...
11:58:45 USER  MAG1 DISMOUNTED
11:58:46 MONTR
11:58:46 MONTR  ..DISMOUNT MAG2:
11:58:58 USER  OPERATOR NOTIFIED
11:58:58 USER  WAITING...
12:00:07 USER  MAG2 DISMOUNTED

```

```

12:00:07 MONTR
12:00:07 MONTR
12:00:07 MONTR .KJOB DSK00:MYJOB.LOG(10,1461)=/W/Z:4/B/VR:10/VS:607/VL:200/VP:10/VD
12:00:10 K-GUE TOTAL OF 4 BLOCKS IN LPT REQUEST
12:00:14 KJOB OTHER JOBS SAME PPN
12:00:17 LGOUT JOB 17, USER (10,1461) LOGGED OFF TTY103 1200 20-APR-72
12:00:18 LGOUT SAVED ALL 38 FILES (645. DISK BLOCKS)
12:00:18 LGOUT ANOTHER JOB STILL LOGGED IN UNDER (10,1461)
12:00:18 LGOUT RUNTIME 0 MIN, 06.39 SEC
12:22:10 LPMMSG LPTSPL VERSION 4A(141) RUNNING ON LPT0
    
```

5.2 USING CARDS TO ENTER JOBS

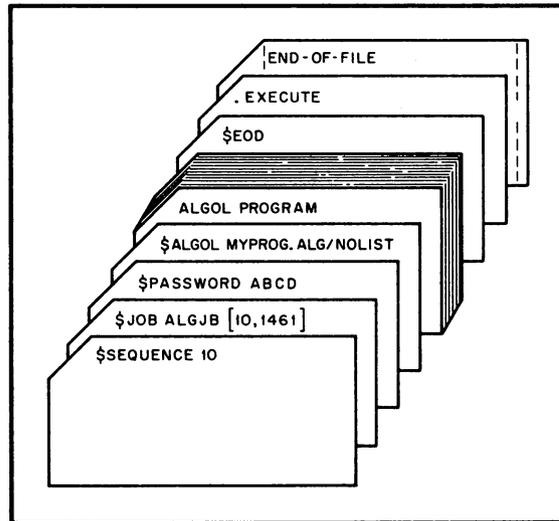
ALGOL Example

The first job is a simple ALGOL program that writes its output into the log file because it has statements that would cause it normally to write to the terminal. The program is as follows.

```

BEGIN
  REAL X; INTEGER I;
  X := 1;
  FOR I := 1 UNTIL 1000 DO X := X+I;
  PRINT(X);
END
    
```

The cards to enter this program are as follows.



10-0925

The control file that MPB makes up for you contains the following commands.

```

.COMPILE MYPROG.ALG /COMPILE /LIST
.EXECUTE
    
```

The output, including the log file is shown below.

```
09:01:43 DATE 13-APR-72 55425E DUAL CPU CDRSTK VER 12(26) DSK
09:01:43 CARD $JOB ALGJB[10/1461]
$ALGOL MYPROG,ALG/NOLIST

09:01:45 STSUM END OF FILE AFTER 12 CARDS, 03 FILES, 03 BLKS

09:01:53 0VERS BATCON 7(52) INP: SUBJOB 02 OF 06
09:01:53 BDATE 13-APR-72
09:01:53 BASUM ALGJB[10,1461] FOR **[10,1461] LOG FILE IN [10,1461]
REQUEST CREATED AT 09:01:08 13-APR-72
UNIQUE: 2 RESTART: 1

09:01:53 MONTR
09:01:53 MONTR .LOGIN 10,1461
09:01:58 USER JOB 13 55425E DUAL CPU TTY115
09:01:58 USER OTHER JOBS SAME PPN
09:01:58 USER 0901 13-APR-72 THUR
09:01:58 MONTR
09:01:58 MONTR .SET TIME 300
09:01:58 MONTR
09:01:58 MONTR .SET SPOOL ALL
09:01:59 MONTR
09:01:59 MONTR
.
$JOB ALGJB[10/1461]
09:02:01 MONTR
.
$ALGOL MYPROG,ALG/NOLIST
09:02:01 MONTR .COMP /COMPILE MYPROG,ALG /N ;CREATED BY CDRSTK
09:02:08 USER ALGOL: MYPROG
09:02:08 MONTR
09:02:08 MONTR EXIT
09:02:08 MONTR
09:02:08 MONTR
.
$EOD
09:02:08 MONTR .EXECUTE
09:02:10 USER ALGOL: MYPROG
09:02:27 USER LOADING
09:02:39 USER
09:02:39 USER MYPROG 1K CORE
09:02:39 USER EXECUTION
09:02:41 USER 5.0050100& 5
09:02:41 USER
09:02:41 USER END OF EXECUTION - 1K CORE
09:02:41 USER
09:02:41 USER EXECUTION TIME: 0.08 SECS.
09:02:41 USER
09:02:41 USER ELAPSED TIME: 0.12 SECS.
09:02:41 MONTR
.
%FIN:
09:02:41 MONTR .DEL MYPROG.REL,MYPROG.ALG
09:02:42 USER FILES DELTED:
09:02:42 USER MYPROG.REL
09:02:43 USER MYPROG.ALG
09:02:43 USER 02 BLOCKS FREED
09:02:43 MONTR
09:02:43 MONTR
.
09:02:44 MONTR .KJOB DSKB:ALGJB.LOG[10,1461]=/W/Z:4/B/V:320/VL:10/VD:0
09:02:45 K-QUE TOTAL OF 4 BLOCKS IN LPT REQUEST
```

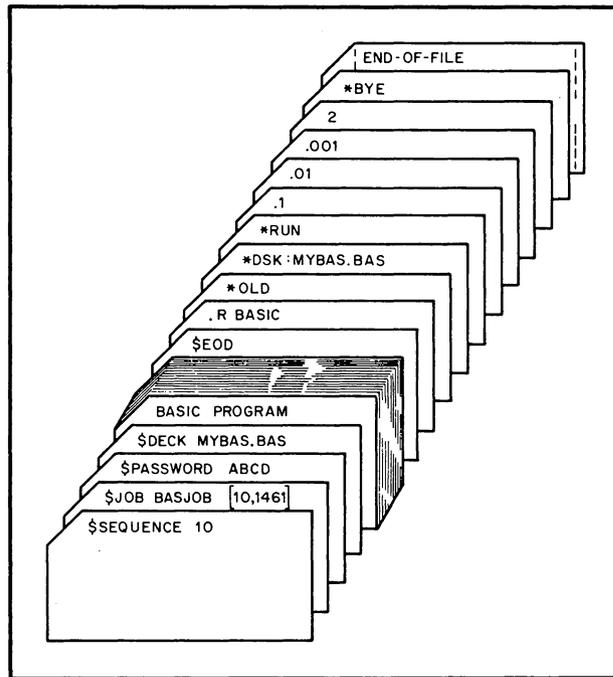
BASIC Example

The next example shows entering a BASIC program. You must include the program after a \$DECK card so that it will be copied into a file on disk. No \$DATA card can be used because BASIC does not use the EXECUTE command and because the data is entered in the control file. The program requests data when it is running; it finds the data in the control file. The final data item must be 2 so that the program can be concluded. The program is shown below.

```

5   INPUT D
10  IF D = 2 THEN 110
20  PRINT "X VALUE", "SINE", "RESOLUTION"
30  FOR X = 0 TO 3 STEP D
40  IF SIN(X) = M THEN 80
50  LET X0 = X
60  LET M = SIN(X)
80  NEXT X
90  PRINT X0, M, D
100 GO TO 5
110 END
    
```

The cards to enter the program and run it are as follows.



10-0926

The output from the program will be printed in the log file because it would normally be printed on the terminal. The log file is shown below.

```

11:10:45 DATE      13-APR-72  55425E DUAL CPU CDRSTK VER 12 (26) DSK
11:10:45 CARD      $JOB BASJOB[10/1461]
11:10:46 STSUM     END OF FILE AFTER 24 CARDS, 03 FILES, 04 BLKS

11:10:49 BVERS     BATCON 7(52) INP: SUBJOB 01 OF 14
11:10:49 BDATE     13-APR-72
11:10:49 BASUM     BASJOB[10,1461] FOR **[10,1461] LOG FILE IN [10,1461]
                    REQUEST CREATED AT 11:09:57 13-APR-72
                    UNIQUE: 2 RESTART: 1

11:10:49 MONTR
11:10:49 MONTR     .LOGIN 10,1461
11:10:51 USER     JOB 19      55425E DUAL OPU TTY114
11:10:51 USER     OTHER JOBS SAME PPN
11:10:52 USER     1110      13-APR-72      THUR
11:10:53 USER
11:10:53 MONTR
11:10:53 MONTR     .SET TIME 300
11:10:53 MONTR
11:10:53 MONTR     .SET SPOOL ALL
11:10:53 MONTR
11:10:53 MONTR     .
                    $JOB BASJOB[10/1461]
                    $DECK MYBAS.BAS
                    $EOD
                    .R BASIC

11:10:53 MONTR
11:10:53 USER
11:10:54 USER
11:10:54 USER     NEW OR OLD--*OLD
11:10:55 USER     OLD FILE NAME--*DSK:MYBAS
11:10:55 USER
11:10:55 USER     READY
11:10:56 USER     *RUN
11:10:56 USER
11:10:56 USER     MYBAS              11110              13-APR-72
11:10:56 USER
11:10:56 USER
11:10:56 USER     ? .1
11:10:57 USER     X VALUE          SINE              RESOLUTION
11:10:57 USER     3.              0.14112          0.1
11:10:57 USER     ? .01
11:10:59 USER     X VALUE          SINE              RESOLUTION
11:10:59 USER     3.              0.141121         0.01
11:10:59 USER     ? .001
11:10:59 USER     X VALUE          SINE              RESOLUTION
11:11:00 USER     2.99999         0.14113          0.001
11:11:00 USER
11:11:00 USER     ?2
11:11:00 USER
11:11:00 USER     TIME: 1.50 SECS.
11:11:00 USER
11:11:01 USER     READY
11:11:01 USER     *BYE
11:11:03 USER     JOB 19, USER [10,1461] LOGGED OFF TTY114 1111 13-APR-72
11:11:03 USER     SAVED ALL 33 FILED (600, DISK BLOCKS)
11:11:03 USER     ANOTHER JOB STILL LOGGED IN UNDER [10,1461]
11:11:03 MONTR     RUNTIME 0 MIN. 03.05 SEC

```

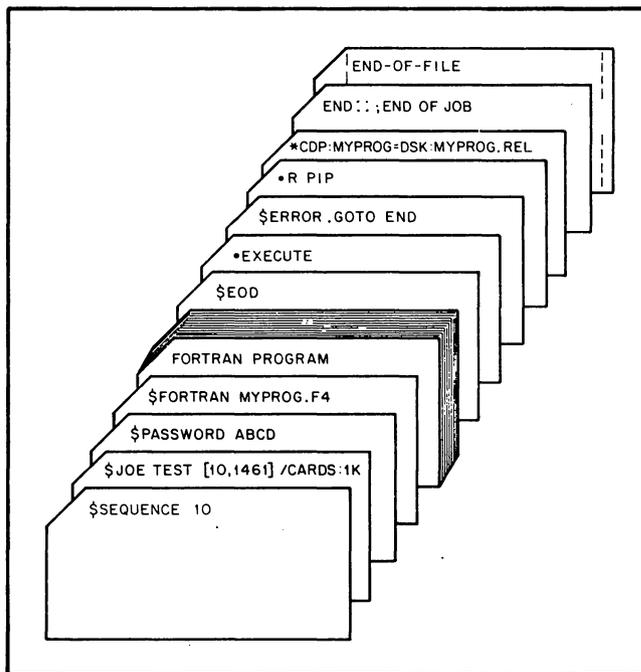
FORTRAN Example

The third example shows a FORTRAN program that prints output on the line printer. In the control file, you want to tell Batch to punch your relocatable binary program if it executes correctly. Otherwise, you want to end your job so that you can find your error from the message in the log file. The program is shown below.

```

C   THIS PROGRAM CALCULATES PRIME NUMBERS FROM 11 TO 50.
    DO 10 I=11,50,2
      J=1
4     J=J+2
      A=J
      A=1/A
      L=1/J
      B=A-L
      IF (B) 5,10,5
5     IF (J.LT.SQRT(FLOAT(I))) GO TO 4
      PRINT 105,I
10    CONTINUE
105  FORMAT (14, ' IS PRIME. ')
      END
    
```

The cards used to enter this program are as follows.



10-0927

Batch puts the following commands into the control file as a result of the cards you entered.

```

.COMPILE MYPROG.F4 /COMPILE /LIST
.EXECUTE MYPROG.REL /MAP:MAP.LPT
.IF (ERROR) .GOTO END
.R PIP
*CDP:MYPROG = DSK:MYPROG.REL
END: : ;END OF JOB

```

The printed output from the job, including the log file is shown below.

```

\MYPROG,F4      F40      V25      12=APR-72      13:43      PAGE 1

                DO 10 I=11,50,2
                J=1
                4  J=J+2
                A=J
                A=I/A
                L=I/J
                B=A-L
                5  IF (B) 5,10,5
                IF (J._T,SQRT (FLOAT (I))) GO TO 4
                TYPE 105,I
                10  CONTINJE
                105 FORMAT (I4, '#IS PRIME.')
                END

SUBPROGRAMS
FORSE, JOEFF  FLOAT  SQRT  INTO,  INTI.  EXIT

SCALARS
I          61          J          62          A          63          L          64          B          65

```

```

13:43:01 DATE      12-APR-72      554A48 DUAL CPU      CDRSTK VER 12(26) DSK
13:43:01 CARD      $JOB TEST[10,1461]/CARD:1K
13:43:03 STSUM     END OF FILE AFTER 19 CARDS, 03 FILES, 04 BLKS

```

```

13:43:21 BVERS     BATCCN 7(52) INP: SUBJOB 01 OF 14
13:43:21 BDATE     12-APR-72
13:43:21 BASUM     TEST[10,1461] FOR **[10,1461] LOG FILE IN [10,1461]
                    REQUEST CREATED AT 13:42:04 12-APR-72
                    UNIQUE: 2 RESTART: 1

```

```

13:43:21 MONTR     .LOGIN 10,1461
13:43:21 MONTR     .JOB 11      554A48 DUAL CPU TTY102
13:43:24 USER      OTHER JOBS SAME PPN
13:43:26 USER      1343      12=APR=72      WED
13:43:28 MONTR     .SET TIME 20
13:43:28 MONTR     .SET SPOOL ALL
13:43:28 MONTR     .
                    $JOB TEST[10,1461]/CARD:1K
                    $FORTRAN MYPROG.F4

```

```

13143:28 MONTR .COMP /COMPILE MYPROG.F4/LIST ;CREATED BY CDRSTK
13143:30 USER FORTRAN: MYPROG.F4
13143:33 MONTR
13143:33 MONTR EXIT
13143:33 MONTR
13143:33 MONTR .
13143:33 MONTR $EOD
13143:33 MONTR .EXECUTE
13143:34 USER FORTRAN: MYPROG.F4
13143:37 USER LOADING
13143:41 USER
13143:41 USER MYPROG 2K CORE
13143:42 USER EXECUTION
13143:42 USER
13143:42 USER 11 IS PRIME.
13143:42 USER 13 IS PRIME.
13143:42 USER 17 IS PRIME.
13143:42 USER 19 IS PRIME.
13143:42 USER 23 IS PRIME.
13143:42 USER 29 IS PRIME.
13143:42 USER 31 IS PRIME.
13143:43 USER 37 IS PRIME.
13143:43 USER 41 IS PRIME.
13143:43 USER 43 IS PRIME.
13143:43 USER 47 IS PRIME.
13143:43 USER
13143:43 USER CPU TIME: 0.27 ELAPSED TIME: 1.82

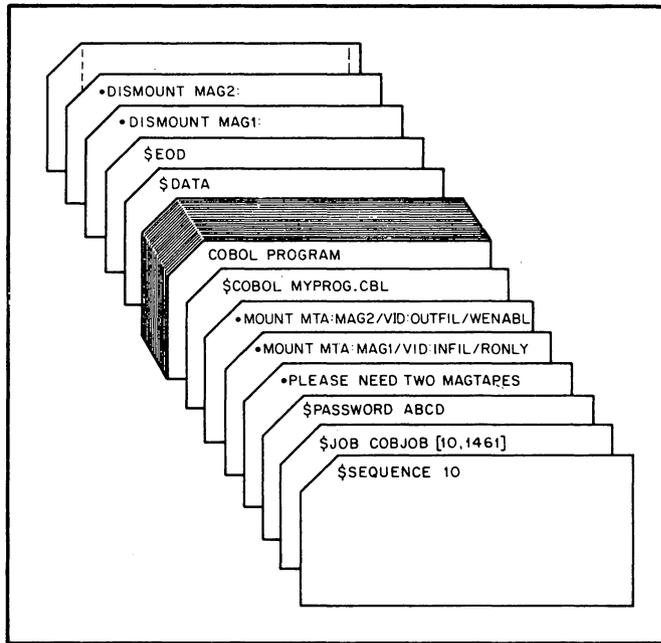
13143:43 USER NO EXECUTION ERRORS DETECTED
13143:43 MONTR
13143:43 MONTR EXIT
13143:43 MONTR .R PIP
13143:43 USER *COPY MYPROG=DSK:MYPROG.REL
13143:43 MONTR .
13143:43 MONTR END;
13143:43 MONTR ;END OF JOB
13143:44 MONTR .
13143:44 MONTR %FIN;
13143:44 USER .DEL MYPROG.REL,MYPROG.F4
13143:44 USER FILES DELETED:
13143:45 USER MYPROG.REL
13143:46 USER MYPROG.F4
13143:46 USER 03 BLOCKS FREED
13143:46 MONTR
13143:46 MONTR .
13143:48 MONTR .KJOB DSKB:TEST.LOG[10,1461]=/W/Z:4/P/VS:277/VL:200/VD:0
13143:48 K=QUE TOTAL OF 6 BLOCKS IN LPT REQUEST
13143:52 KJOB OTHER JOBS SAME PPN
13143:54 LGOUT JOB 11: USER [10,1461] LOGGED OFF TTY102 1343 12-APR-72
13143:54 LGOUT SAVED ALL 30 FILES (585, DISK BLOCKS)
13143:54 LGOUT ANOTHER JOB STILL LOGGED IN UNDER [10,1461]
13143:54 LGOUT RUNTIME 0 MIN. 05.64 DEC
13143:57 LPMSG LPTSPL VERSION 4A(141) RUNNING ON LPT1
13144:02 LPMSG JOB TEST FILE DSKB1:MYPROG.LST[10,1461] FOR [10,1461] STARTED
13144:09 LPMSG DSKB1:MYPROG.LST[10,1461] DONE
    
```

### COBOL Example

The fourth program shows a COBOL program that reads data from a magnetic tape and writes output on another magnetic tape. To have your magnetic tapes mounted on drives and assigned to you, you must request that the operator mount them. Since you do not know which drives will be assigned to your job, you must assign them in your job with logical device names. The MOUNT command assigns the drive to your job and associates the logical name that you specify in it with the physical drive assigned. You should include a PLEASE command to the operator to tell him that you want two magnetic tape drives. If he can't let you have the drives because they are in use, you can ask him to enter your job again. Your magnetic tapes, one with the input data, the other blank so that you can write on it, should be given to the operator with your card deck or kept at the central site, so that the operator can find your tapes. The program is as follows.

```
IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT INFIL, ASSIGN MAG1.
    SELECT OUTFIL, ASSIGN MAG2.
DATA DIVISION.
FILE SECTION.
FD  INFIL, LABEL RECORDS ARE STANDARD,
    VALUE OF IDENTIFICATION IS "INFIL DAT",
    BLOCK CONTAINS 20 RECORDS.
01  INREC, PIC X(80).
FD  OUTFIL, LABEL RECORDS ARE STANDARD,
    VALUE OF IDENTIFICATION IS "OUTFILDAT",
    BLOCK CONTAINS 12 RECORDS.
01  OUTREC, PIC X(80).
PROCEDURE DIVISION.
START.
    OPEN INPUT INFIL, OUTPUT OUTFIL.
LOOP.
    READ INFIL; AT END GO TO FIN.
    WRITE OUTREC FROM INREC.
    GO TO LOOP.
FIN.
    CLOSE OUTFIL, INFIL.
    STOP RUN.
```

The cards to enter this job are shown below.



10-0928

Batch puts the following commands into the control file for you.

```

.PLEASE NEED TWO MAG TAPES, IF I CAN'T HAVE THEM, REQUEUE.
.MOUNT MTA:MAG1/VID:INFIL /RONLY
.MOUNT MTA:MAG2/VID:OUTFIL /WENABL
.COMPILE /COMPILE MYPROG.CBL /LIST
.EXECUTE MYPROG.REL /MAP:MAP.LPT
.DISMOUNT MAG1:
.DISMOUNT MAG2:
    
```

The printed output from your job is shown below.

```

001462 IS THE LOW SEGMENT BREAK
MAP          STORAGE MAP      15142 20-APR-72
STARTING ADDRESS 001406  PROG COBOL  FILE MYPROG
.COMM.         000140   001040
COBOL          001200   001317  FILES. 000140  USES.  000141  SEGWD. 000142
              START,  001406  OVRFN. 000144  POINT, 000145  COMMA. 000146
              ALTER,  000143  MEMRY. 000152  TRAC1, 000152  TRAC2. 000153
              MONEY,  000147  %NM.   000155  %DT.   000156  %PR.   000157
              TRAC3,  000154
TRACED        001457/  000003  PFLG. 001461  TRACE. 001457  TRPD,  001460
              BTRAC,  001460
              TRPOP,  001460
COBOL 1K CORE, 205 WORDS FREE
LOADER USED 2+4K CORE
    
```

PROGRAM COBOL . COBOL 3(43) 20-APR-72

15:41 PAGE 1

```

0001 IDENTIFICATION DIVISION.
0002 ENVIRONMENT DIVISION.
0003 INPUT-OUTPUT SECTION.
0004 FILE-CONTROL.
0005     SELECT INFIL, ASSIGN MAG1.
0006     SELECT OUTFIL, ASSIGN MAG2.
0007 DATA DIVISION.
0008 FILE SECTION.
0009 FD INFIL, LABEL RECORDS ARE STANDARD,
0010     VALUE OF IDENTIFICATION IS "INFIL DAT",
0011     BLOCK CONTAINS 20 RECORDS.
0012 01 INREC, PIC X(80).
0013 FD OUTFIL, LABEL RECORDS ARE STANDARD,
0014     VALUE OF IDENTIFICATION IS "OUTFILDAT".
0015     BLOCK CONTAINS 12 RECORDS.
0016 01 OUTREC, PIC X(80).
0017 PROCEDURE DIVISION.
0018 START.
0019     OPEN INPUT INFIL, OUTPUT OUTFIL.
0020
0021 LOOP
0022     READ INFIL AT END GO TO FIN.
0023     WRITE OUTREC FROM INREC.
0024     GO TO LOOP.
0025 FIN.
0026     CLOSE OUTFIL, INFIL.
0027     STOP RUN.

```

NO ERRORS DETECTED

```

15:37:37 DATE 20-APR-72 554250 DUAL CPU DSRSTK VER 12(26) DSK
15:37:37 CARD $JOB COBJOB[10/1461]
15:37:38 STSUM END OF FILE AFTER 37 CARDS, 04 FILES, 06 BLKS

```

```

15:37:46 BVERS BATCON 7(53) INP: SUBJOB 01 OF 06
15:37:46 BDATE 20-APR-72
15:37:46 BASUM COBJOB[10,1461] FOR **[10,1461] LOG FILE IN [10,1461]
REQUEST CREATED AT 15:36:34 20-APR-72
UNIQUE! 2 RESTART! 1

```

```

15:37:46 MONTR .LOGIN 10/1461
15:37:48 USER JOB 24 554250 DUAL CPU TTY103
15:37:50 USER OTHER JOBS SAME PPN
15:37:53 USER 1537 20-APR-72 THUR
15:37:53 MONTR

```

```

15:37:53 MONTR .SET TIME 300
15:37:54 MONTR
15:37:54 MONTR .SET SPOOL ALL
15:37:54 MONTR
15:37:54 MONTR

```

```

15:37:54 MONTR $JOB COBJOB[10/1461]
15:38:17 MONTR .PLEASE NEED TWO MAG TAPES, IF CAN'T HAVE THEM, REQUEUE.
15:38:18 USER .MOUNT MTAIMAG1/VID:INFIL/RONOLY
15:38:18 USER OPERATOR NOTIFIED
15:38:18 USER WAITING...
15:39:59 USER MAG1 (MTA0) MOUNTED
15:39:59 USER

```

```

15139:59 MONTR  ..MOUNT MTA:MAG2/VID:OUTFIL/WENABL
15140:01 USER  OPERATOR NOTIFIED
15140:01 USER  WAITING...
15141:31 USER  MAG2 (MTA1) MOUNTED
15141:31 USER
15141:31 MONTR  .
                $COBOL MYPROG.CBL
15141:31 MONTR  .COMP /COMPILE MYPROG.CBL/LIST ;CREATED BY CDRSTK
15141:35 USER  COBOL:
15141:59 MONTR  EXIT
15141:59 MONTR
15141:59 MONTR  .
                $DATA
15141:59 MONTR  .SET CUR QAA.CDR ;CREATED BY CDRSTK
15141:59 MONTR
15141:59 MONTR  ..EXEC /MAP:MAP.LPT /REL MYPROG.REL ;CREATED BY CDRSTK
15142:04 USER  LOADING
15142:04 USER  021462 IS THE LOW SEGMENT BREAK
15142:05 USER
15142:06 USER  COBOL 1K CORE
15142:07 USER  EXECUTION
15142:09 MONTR
15142:09 MONTR  EXIT
15142:09 MONTR
15142:09 MONTR  .
                $EOD
15142:09 MONTR  .DISMOUNT MAG1:
15142:09 USER  OPERATOR NOTIFIED
15142:10 USER  WAITING...
15142:29 USER  MAG1 DISMOUNTED
15142:29 MONTR
15142:29 MONTR  ..DISMOUNT MAG2:
15142:30 USER  OPERATOR NOTIFIED
15142:31 USER  WAITING...
15142:47 USER  MAG2 DISMOUNTED
15142:47 MONTR
15142:47 MONTR  .
                %FIN:
15142:47 MONTR  .DEL MYPROG.REL,QAA.CDR,MYPROG.CBL
15142:51 USER  FILES DELETED:
15142:53 USER  MYPROG.REL
15142:57 USER  QAA.CDR
15142:59 USER  MYPROG.CBL
15143:00 USER  07 BLOCKS FREE
15143:01 MONTR
15143:01 MONTR  .
15143:01 MONTR  .KJOB DSKB1:COBJOB,LOG[10,1461]=/1/Z:4/B/VS:628/VL:200/VD:1D
15143:03 K-QUE  TOTAL OF 9 BLOCKS IN LPT REQUEST
15143:09 KJOB  OTHER JOBS SAME PPN
15143:12 LGOUT  JOB 24, USER [10,1461] LOGGED OFF TTY103 1543 20-APR-72
15143:13 LGOUT  SAVED ALL 43 FILES (855, DISK BLOCKS)
15143:13 LGOUT  ANOTHER JOB STILL LOGGED IN UNDER [10,1461]
15143:13 LGOUT  RUNTIME 0 MIN, 07,14 SEC
15144:07 LPMSG  LPTSPL VERSION 4A(141) RUNNING ON LPT2
15144:15 LPMSG  JOB COBJOB FILE DSKB1:MYPROG.LST[10,1461] FOR [10,1461] STARTED
15144:25 LPMSG  DSKB1:MYPROG.LST[10,1461] DONE
15144:25 LPMSG  JOB COBJOB FILE DSKB1:MAP.LPT[10,1461] FOR [10,1461] STARTED
15144:35 LPMSG  DSKB1:MAP.LPT[10,1461] DONE

```

**decsystem10**  
INTRODUCTION TO TECO  
(TEXT EDITOR AND CORRECTOR)

This document represents the software as of version 23 of TECO.

Copyright © 1972 by Digital Equipment Corporation

The material in this manual is for informational purposes and is subject to change without notice.

The following are trademarks of Digital Equipment Corporation, Maynard, Massachusetts:

|           |              |
|-----------|--------------|
| DEC       | PDP          |
| FLIP CHIP | FOCAL        |
| DIGITAL   | COMPUTER LAB |

CONTENTS

|           |                                       |     |
|-----------|---------------------------------------|-----|
| CHAPTER 1 | INTRODUCTION TO TECO                  | 191 |
| 1.1       | General Operating Procedure           | 191 |
| 1.2       | Initialization                        | 192 |
| 1.3       | Special Symbols Used in this Document | 193 |
| 1.4       | General Command String Syntax         | 194 |
| 1.5       | Erasing Commands                      | 195 |
| 1.6       | Command Arguments                     | 196 |
| CHAPTER 2 | TECO COMMANDS                         |     |
| 2.1       | Input Commands                        | 197 |
| 2.2       | Buffer Pointer Positioning            | 198 |
| 2.3       | Text Typeout                          | 200 |
| 2.4       | Deletion Commands                     | 201 |
| 2.5       | Insertion Command                     | 202 |
| 2.6       | Output Commands                       | 204 |
| 2.7       | Special Exit Commands                 | 205 |
| 2.8       | Search Commands                       | 206 |
| CHAPTER 3 | ERROR MESSAGES                        |     |

TABLES

|     |                     |     |
|-----|---------------------|-----|
| 1-1 | Special Symbols     | 193 |
| 3-1 | TECO Error Messages | 210 |



CHAPTER 1  
INTRODUCTION TO TECO

TECO, a very powerful text editor, enables the advanced DECsystem-10 user to edit any ASCII text with a minimum of effort. All editing can be accomplished by using only a few simple commands; or the user may select any of a large set of sophisticated commands such as character string searching, command repetition, conditional commands, programmed editing, and text block movement. In this description of TECO only the basic commands are described. If the user requires information about the more advanced uses of TECO, he can refer to the TECO manual in the DECsystem-10 Users Handbook.

TECO is a character-oriented editor. One or more characters in a line can be modified without retyping the rest of the line. Any sort of document can be edited: programs written in FORTRAN, COBOL, MACRO-10, or any other language; memoranda; specifications; and other types of arbitrarily formatted text. TECO does not require that line numbers or any other extraneous information be associated with the text.

### 1.1 GENERAL OPERATING PROCEDURE

TECO operates on ASCII data files. A file is an ordered set of data on some peripheral device. In the case of TECO, a data file is some type of document. An input file may be a named file on disk or DECTape, a file on magnetic tape, a deck of punched cards, or a punched paper tape. An output file can be written onto any of these same devices. The input file for a given editing operation is the file to which the user wishes to make changes. If the user is using TECO to create a new file, there is no input file. The output file is either the newly created file or the edited version of the input file. An output file is not required if the user wishes merely to examine a file without making any changes.

In general, the process of editing proceeds as follows. The user first specified the file he wishes to edit and then reads in a "page" of text. A page is normally an amount of text that is intended for a single sheet of paper. Form feeds are used to separate a document into pages. On input, TECO interprets form feeds as end-of-page indicators. It is not required, however, that a document be so divided into pages. If a form feed is not encountered, TECO simply reads as much text as will reasonably fit into its editing buffer. For the purposes of this document, the word page is used to mean the segment of text in TECO's editing buffer.

When a page has been read into the buffer, the user can modify this text by using the various editing commands. When he has finished editing the page, he outputs it and reads in the next page. This process continues until, after the last page has been output, the user closes the output file. If there are several pages where no editing is required, there are commands which may be used to skim over them.

## 1.2 INITIALIZATION

The two main uses of TECO are (1) to create a new disk file, and (2) to edit an existing disk file. These are the only uses of TECO described in this document. In particular, the use of TECO with devices other than disk is not described. The beginner can get around this limitation by using PIP to transfer files to and from disk. (Refer to the PIP manual in the DECsystem-10 Users Handbook for information about PIP.)

The two main uses of TECO are so common that there are direct monitor commands to initialize TECO for executing them. The command

```
._ MAKE filename.ext )
```

is used to initialize TECO for creating a new disk file. Filename.ext is the name that the user gives to the new file. The filename can be from one to six alphanumeric characters. This is followed (optionally) by a period (.) and a filename extension of from one to three alphanumeric characters. The most commonly used filename extensions are:

|      |                              |
|------|------------------------------|
| .F4  | for FORTRAN source programs  |
| .CBL | for COBOL source programs    |
| .MAC | for MACRO-10 source programs |

The MAKE command opens a new disk file to receive output from TECO and gives it the name specified by the user. Once the file has been opened it is then actually created by using the insert and output commands, which are explained in sections 2.5 and 2.6 of this document.

The command

```
._ TECO filename.ext )
```

is used to initialize TECO for editing an existing disk file, named filename.ext. The filename and filename extension must be exactly the same as those of the file that is to be edited. The TECO command opens the specified file for input by TECO and opens a new file, with a temporary name, for output of the edited version. When output of the new version is completed, the original version of the file is automatically renamed filename.BAK, and the newly edited version is given the name of the original file. The filename extension .BAK is used for backup files.

After TECO has been initialized for a particular job, it responds by typing an asterisk (\*). The asterisk indicates that TECO is ready to accept commands; it is typed at the beginning of TECO's operation and at the completion of execution of every command string.

Examples:

```

. MAKE EARNNG.F4 ↵      This command initializes TECO for creation
*                       of a new disk file called EARNNG.F4.
-                       The extension .F4 is used because the
                        file is to be a FORTRAN source file.

. TECO LIB40.MAC ↵     This command initializes TECO for editing
*                       the existing disk file LIB40.MAC. At the
-                       completion of editing, TECO automatically
                        changes the name of the original version of
                        LIB40.MAC to LIB40.BAK and gives the
                        name LIB40.MAC to the new version.
    
```

NOTE

The TECO command cannot be used to edit a file which has the filename extension .BAK. To edit a backup file the user must first rename the backup file. For example, to edit the file LIB40.BAK the user should proceed as follows:

```

. RENAME LIB40.OLD=LIB40.BAK ↵
. TECO LIB40.OLD ↵
*
-
    
```

1.3 SPECIAL SYMBOLS USED IN THIS DOCUMENT

Table 1-1  
Special Symbols

| Symbol | Character Represented | Comment  |
|--------|-----------------------|--|
| ↵      | Carriage Return       | Whenever the RETURN key is typed, TECO automatically appends a line feed to the carriage return.   |
| Ⓢ      | Altmode               | On most terminals, the altmode key is labeled "ALTMODE", but on some it is labeled "ESC" or "PREFIX". Since the altmode is a non-printing character, TECO indicates that it has received an altmode type-in by echoing a dollar sign (\$). |
| ␣      | Control C             | This character is typed by typing the letter C while holding down the CTRL key. Other control characters are represented in similar fashion.   |

Table 1-1 (Cont)  
Special Symbols

| Symbol | Character Represented | Comment   |
|--------|-----------------------|---|
| FORM   | Form Feed             | Form feed is typed by typing (tF) (control F).  |
| ↓      | Line Feed             | This symbol is used only when a line feed is explicitly typed. It is not used for the line feed which is automatically assumed when a carriage return is typed. |
| →      | Tab                   | Tab is typed by typing (tI) (control I).  |
| Δ      | Space                 | This symbol is used occasionally for emphasis.  |
| ⊙      | Rubout                | This key is used to nullify a character erroneously typed in a command string. Its use is explained fully in Section 1.5.                                       |

## 1.4 GENERAL COMMAND STRING SYNTAX

TECO commands are usually given by typing the one- or two- letter name of the command. However, many of the commands take arguments. Some typical examples are shown below, to give the reader an idea how TECO commands look. These commands are fully explained later in the manual.

```
L
PW
ISAMPLE ($)
3K
```

TECO commands may be given one at a time. However, it is usually more convenient to type, in a single command string, several commands that form a logical group. An example of a command string is shown below.

```
*IHEADING ($)NTAG: ($)2LT ($) ($)
```

A command string may be typed after TECO indicates its readiness by printing an asterisk. Command strings are formed by merely writing one command after another. Command strings are terminated by typing two consecutive altmodes.

Execution of the command string begins only after the double altmode has been typed. At that point each command in the string is executed in turn, starting at the left. When all commands in the string have been executed, TECO prints another asterisk, indicating its readiness to accept another command.

If some command in the string cannot be executed because of a command error, execution of the command string stops at that point, and an error message is printed. Commands preceding the bad command are executed. The bad command and those following it are not executed.

### 1.5 ERASING COMMANDS

Typographical errors, if discovered while typing a command string, may be "erased" by use of the rubout key. This process is best explained by an example.

\*3LKILEIF ERICXON

After typing this much of the command string, the user discovers that he has misspelled the name "Ericson." To nullify his error, he types three successive rubouts. As he does this, TECO responds by retyping the characters which are being rubbed out.

\*3LKILEIF ERICXON (RO) N (RO) O (RO) X

Of course, rubout is a non-printing character so the actual line looks like this:

\* 3LKILEIF ERICXONNOX

Once he has rubbed out the bad character, the user continues the command string from the last correct character.

\* 3LKILEIF ERICXONNOXSON (\$) 0LT (\$) (\$)

The actual function of the rubout character is to delete the last typed character in t. Consequently, if the bad character is not the last in the string, all characters back to that point must be deleted. Rubout characters do not enter the command string.

An entire command string may be erased, if it has not yet been terminated, by typing two successive tG (control G) characters.

Example:

\* 3LKILIEF ERICXON tG tG

tG tG causes the entire command string to be rejected. TECO types a new asterisk and awaits a new command.

### 1.6 COMMAND ARGUMENTS

There are two types of arguments for TECO commands. Some commands require numeric arguments and some require alphanumeric (text) arguments.

Numeric arguments, and also all numeric type-outs by TECO, are decimal integers. Numeric arguments always precede the command to which they apply. A typical example of a command taking a numeric argument is the command to delete three characters: "3D".

Alphanumeric arguments are textual arguments meant to be interpreted as ASCII code by TECO. Alphanumeric arguments always follow the command to which they apply, and they must always be terminated by an altmode. Examples of alphanumeric arguments are (1) text to be inserted, and (2) character strings to be searched for.

Example:

\* ISOMETHING  $\text{\$}$   $\text{\$}$

The argument is "SOMETHING".

As shown in the above example, the altmode used to terminate an alphanumeric argument may also serve as one of the two altmodes necessary to terminate a command string. Any ASCII character except null, altmode, and rubout may be included in an alphanumeric argument.

CHAPTER 2  
TECO COMMANDS

2.1 INPUT COMMANDS

The Y (yank) command first clears the editing buffer and then reads the next page of the input file into the buffer.

A single Y command is automatically performed by the command

```
._ TECO filename.ext )
```

so that when editing with this command the first page of the input file is automatically read in before TECO prints the first asterisk.

The Y command may be used to delete entire pages of a file, since the editing buffer is completely cleared before the input is performed.

The A (append) command reads in the next page of the input file without clearing the current contents of the editing buffer. This command is used to combine several pages of a document. When the A command is used, the form feed separating the page already in the buffer and the page to be read in is removed. Thus after the A command the two pages are combined into one.

If the editing buffer does not have enough room to accommodate an A command which has been given, TECO automatically expands its buffer and then executes the A command. The user is notified of this action by a message of the following form

```
[3K CORE]
```

If sufficient core is not available to allow buffer expansion, the user is notified by an error message.

NOTE

On either an A or a Y command the form feed terminating the page to be read in is not actually read into the buffer. It is removed on input and a single form feed is appended to the end of the buffer when the buffer is output.

Examples:

```

_ TECO REPORT.CBL)
*
-

* Y ($ $)
*
-

* AA ($ $)
*
-

* A ($ $)
[4K CORE]
*
-
    
```

This command, as part of the process of initializing TECO for editing the disk file REPORT.CBL, automatically clears the buffer and then reads in the first page of the file.

This command deletes the entire contents of the buffer and then reads in the next page of the input file.

Read the next two pages of the input file into the buffer, combining them with the page already in the buffer.

The buffer is expanded as required by the A command. In most cases this message need be of no concern to the user. It is important only if the system is low on core and does not have swapping capability.

## 2.2 BUFFER POINTER POSITIONING

Since TECO is a character-oriented editor, it is very important that the user understand the concept of the buffer pointer. The position of the buffer pointer determines the effect of many of the editing commands. For example, insertion and deletion always take place at the current position of the buffer pointer.

The buffer pointer is simply a movable position indicator. It is always positioned between two characters in the editing buffer, or before the first character in the buffer, or after the last character in the buffer. It is never positioned "on" a particular character, but rather before or after the character. The pointer may be moved forward or backward over any number of characters.

The J command moves the buffer pointer to the beginning of the buffer, i.e., to the position immediately before the first character in the buffer.

The ZJ command moves the pointer to the end of the buffer, i.e., to the position following the last character in the buffer.

The C command advances the pointer over one character in the buffer. The C command may be preceded by a (decimal) numeric argument. The command nC moves the pointer forward over n characters. (The pointer cannot be advanced beyond the end of the buffer.)

The R command moves the pointer backward over one character in the buffer. This command may also be preceded by a numeric argument. The command nR moves the pointer backward over n characters. (The pointer cannot be moved backward beyond the beginning of the buffer.)

The L command is used to advance the buffer pointer or move it backward, on a line-by-line basis. The L command takes a numeric argument, which may be positive, negative, or zero, and is understood to be one (1) if omitted.

The action of the L command with various arguments is best explained in a more concrete way. Suppose the buffer pointer is positioned at the beginning of line b, or at some position within line b.

The command L, or 1L, advances the pointer to the beginning of line b+1, i.e., to the position following the line feed which terminates line b.

The command nL, where n > 0, advances the pointer to the beginning of line b+n.

The command 0L moves the pointer to the beginning of line b. If the pointer is already at the beginning of line b, nothing happens.

The command -L moves the pointer back to the beginning of line b-1.

The command -nL moves the pointer back to the beginning of line b-n.

NOTE

After execution of a Y command, the buffer pointer is always positioned before the first character in the buffer. (The Y command automatically executes an implicit J command.) The A command does not change the position of the buffer pointer.

In examples, the position of the buffer pointer is often represented in this manual by the symbol ↑ just below the line of text.

Examples:

\* J3L (\$) (\$)  
\_ \*  
\_

The J command moves the pointer to the beginning of the first line in the buffer. The 3L command then moves it to the beginning of the fourth line.

\* ZJ-2L (\$) (\$)  
\_ \*  
\_

This moves the pointer to the beginning of the next to last line in the buffer.

\* L4C (\$) (\$)  
\_ \*  
\_

Advances the pointer to the position following the fourth character in the next line.

\* 0L2R (\$) (\$)  
\_ \*  
\_

0L moves the pointer back to the beginning of the line it is currently on. Then 2R moves it back over the carriage return-line feed pair which terminates the preceding line.

ABCDEF  
↑

In this example of text stored in the buffer, the position of the buffer pointer is shown to be between B and C.

### 2.3 TEXT TYPE-OUT

Various parts of the text in the buffer can be typed out for examination. This is done by use of the T command. Just what is typed out depends on the position of the buffer pointer and the argument given. The T command never moves the buffer pointer.

The command T types out everything from the buffer pointer through the next line feed. Thus, if the pointer is at the beginning of a line, the command T causes that line to be typed out. If the pointer is in the middle of a line, T causes the portion of the line following the pointer to be typed.

The command nT (n>0) is used to type out n lines, i.e., everything from the buffer pointer through the nth line feed following it.

The command OT types out everything from the beginning of the current line up to the buffer pointer. This is useful for determining the position of the pointer.

The command HT types out the entire contents of the buffer.

The user, especially one new to TECO, should use the T command often, to make sure the buffer pointer is where he thinks it is.

During execution of any T command, the user may stop the terminal output by typing the 1O (control O) character. This command causes TECO to finish execution of the command string, omitting all further type-out. The 1O command does not carry over to the next command string.

Examples:

```
*_OLT ($) ($)
ENTIRE LINE TYPED
*
_
```

This command string is used to move the pointer back to the beginning of a line and then type out the entire line. It is frequently used after insertion and search commands.

```
*_OTT ($) ($)
ENTIRE LINE TYPED
*
_
```

This command string causes the entire line to be typed without moving the pointer. It is useful after insertion and search commands when it is not convenient to move the pointer back to the beginning of the line.

```
*_2T ($) ($)
EF
GHIJKL
*
_
```

If the buffer contains the text below with the pointer between D and E,  
 ABCD,EF ) ↓  
 GHIJKL ) ↓  
 MNOPQR ) ↓

this command causes the typeout shown.

"ABCD" is not typed because these characters precede the pointer. MNOPQR is not typed because these characters follow the second line feed.

### 2.4 DELETION COMMANDS

Characters are deleted individually by using the D command. The command D deletes the character immediately following the buffer pointer. The command nD, where n > 0, deletes the n characters immediately following the pointer. The commands -D and -nD delete the character or the n characters, respectively, which immediately precede the buffer pointer.

Lines are deleted using the K command. The K command may be preceded by a numeric argument, which is understood to be 1, if omitted. The command nK (n > 0) deletes everything from the current position of the pointer through the nth line-feed character following the pointer. The command HK deletes the entire contents of the buffer.

At the conclusion of a D or K command the buffer pointer is positioned between the characters which precede and follow the deletion.

Examples:

The editing buffer contains the following three lines of text, and the pointer is positioned between the G and H.

```
ABCDEF^G^HIJKLM )↓
NOPQRSTU^VWXY^Z )↓
1234567890 )↓
```

```
* 4D ($) ($) Delete HIJK.
_
*
_
```

```
* -D ($) ($) Delete G.
_
*
_
```

```
* -3D ($) ($) Delete EFG.
_
*
_
```

```
* 7D ($) ($) Delete HIJKLM )↓ but do not delete the line
_ feed at the end of the first line.
*
_
```

```
* K ($) ($) Delete HIJKLM )↓.
_ Since the carriage return and line feed at the end
* of the first line are deleted, the text in the buffer
_ after this command would be:
ABCDEF^GNOPQRSTU^VWXY^Z )↓
1234567890 )↓
```

```
* 2K10D ($) ($) This would leave the buffer containing only
_ ABCDEF^G )↓
*
_
```

```

* 0LK ($) ($)
_
*
_
* L2K ($) ($)
_
*
_
* HK ($) ($)
_
*
_

```

This is the command string that is required to kill (delete) the entire first line.

This kills the last two lines.

Kill the entire buffer.

### 2.5 INSERTION COMMAND

The only insertion command is the I command. The ASCII text that is to be inserted into the buffer is typed immediately after the letter I. The text to be inserted is terminated by an altmode.

Any ASCII character except null, altmode, and rubout may be included in the text to be inserted. Specifically, spaces, tabs, carriage returns, form feeds, line feeds, and control characters are all allowed. If a carriage return is typed in an insertion, it is automatically followed by a line feed.

The text to be inserted is placed in the buffer at the position of the buffer pointer, i.e., between the characters. At the conclusion of the insertion command the buffer pointer is positioned at the end of the insertion.

Any number of lines may be inserted with a single I command. For the user's protection, however, no more than 10 to 20 lines should be inserted with each I command.

Examples:

If the buffer contains ABCD,EF) ↓ with the pointer between D and E, the command

```

* IXYZ ($) ($)
_
*
_
* I)
_
_ ($) ($)
_
*
_
* I↓
_
_ ($) ($)
_
*
_
* 3RI Δ ($) 4CI Δ ($) ($)
_
*
_
* I (FORM)
_ ($) ($)
_
*
_

```

produces ABCDXYZ,EF) ↓

produces ABCD) ↓  
           ↓,EF) ↓

produces ABCD ↓  
                   ↓,EF) ↓

produces A Δ BCDE Δ F) ↓

This command is used to separate the page in the buffer into two pages. Both pages, however, remain in the buffer. They are not actually separated until output.

```

* JILINE ONE )
_ LINE TWO )
LINE THREE )
( $ )
*
-
* KI )
( $ )
*
-

```

This example shows insertion of several lines of text at the beginning of the buffer.

This is the command string used to delete the tail of a line without removing the carriage return-line feed at the end of the line. If the buffer contains

```

ABCD ) ↓
EFGH ) ↓

```

This command will produce

```

AB ) ↓
,EFGH ) ↓

```

### 2.6 OUTPUT COMMANDS

The command P causes (1) the entire contents of the editing buffer to be output to the output file and (2) an implicit Y command to be performed which reads in the next page of the input file. This command is used after editing of a given page is complete and the user is ready to move on to the next page.

The P command may be used with a positive numeric argument to skim over several pages. Specifically, the nP command causes the n consecutive pages of the input file, starting with the page in the editing buffer, to be output, and then the n+1st page to be yanked in.

The PW command merely outputs the page currently in the editing buffer. It does not clear the buffer, it does not read in any more text, and it does not move the buffer pointer. This command is used when creating a new file. It is also used to output the last page of a file.

If the buffer is empty, the PW and P commands have no effect.

The EF command must be used to close the output file after all output to it is complete. EF is normally used after the PW command which outputs the last page of the file.

Examples:

```

* PWEF ( $ ) ( $ )
*
-
* PT ( $ ) ( $ )
FIRST LINE
*
-

```

This is the command string usually used to close out a file when the last page of the file is in the buffer.

This command string outputs the current page, reads in the next page, and then types the first line of the new page.

```
* 8P ($) ($)
_
*
```

If, for example, page 6 of a document is in the editing buffer, this command causes pages 6 through 13 of the document to be output, one after the other, and then reads in page 14.

## 2.7 SPECIAL EXIT COMMANDS

The EX command is used to conclude an editing job with a minimum of effort. Its use is best shown by an example.

Suppose the user is editing a 30-page file and suppose that the last actual change to the file is made on page 10. At this point the user gives the command

```
* EX ($) ($)
_
_
```

In this case the action performed by TECO is equivalent to the command string 20PPWEF, with an automatic return to the monitor at the end. Thus, the action of TECO is (1) to rapidly move all the rest of the input file on to the output file, (2) close the output file, and (3) to return control to the monitor.

The EG command is even more efficient. This command performs exactly the same functions as the EX command, but after that it causes re-execution of the last COMPILE, LOAD, EXECUTE, or DEBUG command attempted before TECO was called.

For example, suppose the user gives the command

```
_ COMPILE PLOT.F4 )
```

To request compilation of a FORTRAN source program, but the compiler discovers errors in the code. The user would then call TECO to correct these errors:

```
_ TECO PLOT.F4 )
*
_
```

When all the errors are edited, the user exits from TECO with the command

```
* EG ($) ($)
_
```

This causes the COMPILE command to be executed again on the file PLOT.F4, after TECO has finished output of the file.

Any TECO job may be aborted by using the standard return-to-monitor command: ↑C ↑C (control C typed twice). However, if this command is typed before the output file is closed, the output file is lost.

If no input or output operations are in progress a single ↑C is sufficient to exit from TECO to the monitor. In such a case, the user may reenter TECO without destroying the job he was previously executing. This is illustrated in the following example.

|  |   |
|--|---|
| <pre> _ TECO SOURCE.MAC ) * ICOMMENTS (\$) (\$) * ↑C _ DEASSIGN LPT ) _ DAYTIME ) 24-MAY-72 10:34 _ REE ) * _ </pre> | <p>A TECO job is started.</p> <p>The user exits to perform a few simple monitor commands.</p> <p>The user reenters TECO. The previous buffer is still intact.</p> |
|--|---|

### 2.8 SEARCH COMMANDS

In many cases the simplest way to position the buffer pointer is by using a character string search. A search command causes TECO to scan through the text until a specified string of characters is found, and then to position the pointer at the end of this string. There are two main search commands.

The S command is used to search for a character string within the editing buffer. The string to be searched for is specified as an alphanumerical argument following the S command. This argument must be terminated by an altmode. The character string to be searched for may contain any ASCII character except null, altmode, or rubout.

The S command may be preceded by a numerical argument n > 1. This argument is used to search for the nth occurrence of a character string. Thus a 2S command searches for the second occurrence of the particular character string, skipping the first occurrence. If n is omitted, n = 1 is assumed.

Execution of the S command begins at the position of the buffer pointer and continues to the end of the buffer. If the specified character string is not found in this range, an error message is printed and the buffer pointer is set to the beginning of the buffer.

Examples:

|                                     |   |
|-------------------------------------|---|
| <pre> * SA → B (\$) (\$) * _ </pre> | <p>This causes the pointer to be positioned after the B in the first occurrence of the string.</p> <p>A - tab - B past the current position of the pointer.</p> |
|-------------------------------------|---|

```
* J2$NAME ($) ($)
_
*
```

This causes the pointer to be positioned after the second occurrence of the string "NAME" in the buffer.

```
* S20 )
TAG: ($) OLT ($) ($)
TAG: REST OF LINE
_
*
```

This moves the pointer to the position just following the colon in the string "20 ) TAG:", then repositions the pointer to the beginning of the line (just before the "TAG:") and types out the entire line starting with "TAG:".

Warning: When attempting a search it is very easy to overlook an occurrence of the search string preceding the one which the user desires. For example, he may want to move the pointer after the word "AND" but erroneously position it after a preceding occurrence of a word like "THOUSAND". For this reason the user, especially the novice, is strongly urged to execute a T command to ascertain the position of the pointer after each search command.

Example:

```
* SWORD ($) OTT ($) ($)
FORMAT(1X, 'WORD')
* I Δ WORD2 ($) ($)
_
*
```

Here the user wishes to insert "Δ WORD2" after "WORD". He wisely types out the line to make sure he is at the right place, before inserting "WORD2".

The other principle search command is the N command. The difference is that an S search ends at the end of the current buffer, whereas an N search does not. An N search begins like an S search, but if the character string is not found in the current buffer, an automatic P command is executed. The current page is outputted, the next page read in, and the search continued on the new page. This process continues until either the string is found or the input file is exhausted.

If the N search does find the specified character string, the pointer is positioned at its end.

If the string is not found, an error message is generated. In this case the user caused himself a fair amount of delay. If an N search fails, the user must close the file with an EX command, then reopen it and try the N search again with a character string that can be found. The user is strongly urged to be careful when typing search character strings. Remember also that a search string must be terminated with an altmode.

Example:

```
|
* NSTRING - 3D ($) ($)
?SRH CANNOT FIND "STRING-3D"
* EX ($) ($)
. TECO filename.ext )
* NSTRING ($) - 3D ($) ($)
*
-
```

Here the user meant to search for the character string "STRING", and to delete the last three characters of the string. However, he forgot to terminate the search string with an altmode and this caused the unsatisfied search request error message (?SRH).



CHAPTER 3  
ERROR MESSAGES

When TECO encounters an illegal command or a command that for any other reason cannot be executed, an error message is printed on the user's terminal. Such messages contain a three-character code of the form ?aaa and a one-line description of the error.

To get more information about the error, the user can type a slash (/) immediately after he receives the error message. TECO will type an additional message that describes the error in more detail. All three parts of the error messages from TECO are given in Table 3-1.

When an error message is generated, the command to which it refers is not executed, the remainder of the command string is ignored, and TECO retruns to the idle state by typing an asterisk and awaiting a new command string.

The novice user is especially warned that there are a great many TECO commands that have not been described in this introductory material. Almost every letter of the alphabet and many of the special characters have meanings as TECO commands. Hence, the user should be careful when typing command strings. The beginner should probably stick to relatively short command strings.

In the following table, all TECO error messages are listed, even though some of them refer to the more advanced commands not described in this manual. Error messages referring to the advanced commands will probably be encountered by the user of this introductory material only if he has typed an unintended command letter.

The complete set of TECO commands is fully described in the TECO manual in the DEC-system-10 Users Handbook. Since most editing can be done using only the basic commands covered in this introductory material, most users should be able to get along without the more advanced description for some time. The novice should gain complete mastery of the basic commands before attempting to use any of the advanced commands.

Table 3-1  
TECO Error Messages

|      |   |
|------|---|
| ?ARG | <p>Improper Arguments<br/>The following argument combinations are illegal:</p> <ol style="list-style-type: none"> <li>1) , (no argument before comma)</li> <li>2) m,n, (where m and n are numeric terms)</li> <li>3) H, (because H=B,Z is already two arguments)</li> <li>4) ,H (H following other arguments)</li> </ol>  |
| ?BAK | <p>Cannot Delete Old Backup File<br/>Failure in rename process at close of editing job initiated by an EB command or a TECO command. There exists an old backup file filnam.BAK with a protection &lt;nnn&gt; such that it cannot be deleted. Hence the input file filnam.ext cannot be renamed to "filnam.BAK". The output file is closed with the filename "nnnTEC.TEMP", where nnn is the user's job number. The RENAME UO error code is nn.</p>   |
| ?COR | <p>Storage Capacity Exceeded<br/>The current operation requires more memory storage than TECO now has and TECO is unable to obtain more core from the monitor. This message can occur as a result of any one of the following things:</p> <ol style="list-style-type: none"> <li>1) command buffer overflow while a long command string is being typed,</li> <li>2) Q-register buffer overflow caused by an X or [ command,</li> <li>3) editing buffer overflow caused by an insert command or a read command.</li> </ol> |
| ?COS | <p>Contradictory Output Switches<br/>The GENLSN and SUPLSN switches may not both be used with the same output file.</p>   |
| ?EBD | <p>EB with Device dev Is Illegal<br/>The EB command and the TECO command may be specified only with file structured devices, i.e., disk and DECTape.</p>  |
| ?EBF | <p>EB with Illegal File filnam.ext<br/>The EB command and the TECO command may not be used with a file having the filename extension ".BAK" or with a file having the name "nnnTEC.TMP". Where nnn is the user's job number, the user must either use an ER-EW sequence, or rename the file.</p>  |
| ?EBO | <p>EB, EW, or EZ Before Current EB Job Closed<br/>After an output file has been opened by a TECO command or an EB command, no further EB, EW, or EZ commands may be given until the current output file is closed.</p>  |
| ?EBP | <p>EB Illegal Because of File filnam.ext Protection<br/>The file filnam.ext cannot be edited with an EB command or a TECO command because it has a protection &lt;nnn&gt; such that it cannot be renamed at close time.</p>   |

Table 3-1 (Cont)  
TECO Error Messages

|         |   |
|---------|---|
| ?EEE    | <p>Unable to Read Error Message File</p> <p>An error, whose code was typed previous to this error message, has occurred, and while TECO was trying to find the proper error message in the error message file, one of the following errors occurred: 1) the error message file, TECO.ERR, could not be found on device SYS:, 2) an input error occurred while TECO was reading the file TECO.ERR, 3) The error message corresponding to that error code is missing from TECO.ERR, 4) the user's TECO job does not currently have enough room for a buffer to read the error message into, and no more core can be obtained from the monitor, 5) for some strange reason device SYS: could not be initialized for input.</p> |
| ?EMA    | <p>EM with Illegal Argument nn</p> <p>The argument n in an nEM command must be greater than zero.</p>   |
| ?EMD    | <p>EM with No Input Device Open</p> <p>EM commands apply only to the input device, and so should be preceded by an ER (or equivalent) command. To position a tape for output, that unit should be temporarily opened for input while doing the EM commands.</p>   |
| ?ENT-00 | <p>Illegal Output Filename "filnam.ext"</p> <p>ENTER UJO failure 0. The filename "filnam.ext" specified for the output file cannot be used. The format is invalid.</p>  |
| -01     | <p>Output UFD dev:[pj,pg] Not Found</p> <p>ENTER UJO failure 1. The file filnam.ext[pj,pg] specified for output by an EW, EZ, or MAKE command cannot be created because there is no user file directory with project-programmer number [pj,pg] on device dev.</p>   |
| -02     | <p>Output Protection Failure</p> <p>ENTER UJO failure 2. The file filnam.ext[pj,pg] specified for output by an EW, EZ, EB, MAKE, or TECO command cannot be created either because it already exists and is write-protected &lt;nnn&gt; against the user, or because the UFD it is to be entered into is write-protected against the user.</p>   |
| -03     | <p>Output File Being Modified</p> <p>ENTER UJO failure 3. The file filnam.ext specified for output by an EW, EZ, EB, MAKE, or TECO command cannot be created because it is currently being created or modified by another job.</p>  |
| -06     | <p>Output UFD or RIB Error</p> <p>ENTER UJO failure 6. The output file filnam.ext cannot be created because a bad directory block was encountered by the monitor while the ENTER was in progress. The user may try repeating the EW, EB, or TECO command, but if the error persists, it is impossible to proceed. Notify your system manager.</p>   |
| -14     | <p>No Room or Quota Exceeded on dev:</p> <p>ENTER UJO failure 14. The output file filnam.ext cannot be created because there is no more free space on device dev:, or because the user's quota is already exceeded there.</p>   |

Table 3-1 (Cont)  
TECO Error Messages

|         |  |
|---------|--|
| -15     | Write Lock on dev:<br>ENTER UOO failure 15. The output file filnam.ext cannot be created because the output file structure is write-locked.  |
| -16     | Monitor Table Space Exhausted<br>ENTER UOO failure 16. The output file filnam.ext cannot be created because there is not enough table space left in the monitor to allow the ENTER. The user may try repeating the EW, EB, or TECO command, but if the error persists he will have to wait until conditions improve. |
| -23     | Output SFD not Found<br>ENTER UOO failure 23. The output file filnam.ext cannot be created because the sub-file-directory on which it should be ENTERed cannot be found.   |
| -24     | Search List Empty<br>ENTER UOO failure 24. The output file filnam.ext cannot be created because the user's file structure search list is empty.  |
| -25     | Output SFD Nested too Deeply<br>ENTER UOO failure 25. The output file filnam.ext cannot be created because the specified SFD path for the ENTER is nested too deeply.  |
| -26     | No Create for Specified SFD Path<br>ENTER UOO failure 26. The output file filnam.ext cannot be created because the specified SFD path for the ENTER is set for no creation.  |
| -nn     | ENTER Failure nn on Output File filnam.ext<br>The attempted ENTER of the output file filnam.ext has failed and the monitor has returned an error code of nn. This error is not expected to occur on an ENTER. Please send the TTY printout showing what you were doing to DEC with an SPR form.                      |
| ?EOA    | nEO Argument Too Large<br>The argument n given with an EO command is larger than the standard (maximum) setting of EO=n for this version of TECO. This must be an older version of TECO than the user thinks he is using; the features corresponding to EO=n do not exist.   |
| ?FNF-00 | Input File filnam.ext Not Found<br>LOOKUP UOO failure 0. The file filnam.ext specified for input by an ER, EB, or TECO command was not found on the input device dev.  |
| -01     | Input UFD dev:[pj,pg] Not Found<br>LOOKUP UOO failure 1. The file filnam.ext[pj,pg] specified for input by an ER, EB, or TECO command cannot be found because there is no User File Directory with project-programmer number [pj,pg] on device dev.  |
| -02     | Input Protection Failure<br>LOOKUP UOO failure 2. The file filnam.ext[pj,pg] specified for input by an ER, EB, or TECO command cannot be read because it is read-protected <nnn> against the user.   |

Table 3-1 (Cont)  
TECO Error Messages

|      |  |
|------|--|
| -06  | Input UFD or RIB Error<br>LOOKUP UUO failure 6. The input file filnam.ext cannot be read because a bad directory block was encountered by the monitor while the LOOKUP was in progress. The user may try repeating the ER, EB, or TECO command, but if the error persists all is lost. Notify your system manager.               |
| -16  | Monitor Table Space Exhausted<br>LOOKUP UUO failure 16. The input file filnam.ext cannot be read because there is not enough table space left in the monitor to allow the LOOKUP. The user may try repeating the ER, EB, or TECO command, but if the error persists he will have to wait until system conditions improve.        |
| -23  | Input SFD not Found<br>LOOKUP UUO failure 23. The input file filnam.ext cannot be found because the sub-file-directory on which it should be looked up cannot be found.  |
| -24  | Search List Empty<br>LOOKUP UUO failure 24. The input file filnam.ext cannot be found because the user's file structure search list is empty.  |
| -25  | Input SFD Nested too Deeply<br>LOOKUP UUO failure 25. The input file filnam.ext cannot be found because the specified SFD path for the LOOKUP is nested too deeply.  |
| -nn  | LOOKUP Failure nn on Input File filnam.ext<br>The attempted LOOKUP on the Input file filnam.ext has failed and the monitor has returned an error code of nn. This error is not expected to occur on a LOOKUP. Please send the TTY printout showing what you were doing to DEC with an SPR form.                                  |
| ?FUL | Device dev: Directory Full<br>ENTER UUO failure n. The file filnam.ext specified for output by an EW or MAKE command cannot be created on DECTape dev because the tape directory is full.  |
| ?IAB | Incomplete <...> or (...) in Macro<br>A macro contained in a Q-register and being executed by an M command contains an iteration that is not closed within the Q-register by a >, or a parenthetical expression that is not closed within the Q-register by a).  |
| ?ICE | Illegal Control-E Command in Search Argument<br>A search argument contains a (tE) command that is either not defined or incomplete.  |
| ?ICT | Illegal Control Command †<char> in text Argument<br>In order to be entered as text in an Insert command or search command, all control characters (†@ - †H and †N - †←) must be preceded by †R or †T. Otherwise they are interpreted as commands. The control character "†<char>" is an undefined text argument control command. |

Table 3-1 (Cont)  
TECO Error Messages

|             |  |
|-------------|--|
| ?IDV        | Input Device dev Not Available<br>Initialization failure. Unable to initialize the device dev for input. Either the device is being used by someone else right now, or else it does not exist in the system.   |
| ?IEC        | Illegal Character "<char>" After E<br>The only commands starting with the letter E are EB, EF, EG, EH, EM, EO, ER, ET, EU, EW, and EZ. When used as a command (i.e., not in a text argument) E may not be followed by any character except one of these.   |
| ?IEM        | Re-Init Failure on Device dev After EM<br>Unable to re-initialize the device dev after executing an EM command on it. If this error persists after retrying to initialize the device with an ER command (or EW command if output to the device is desired), consult your system manager.   |
| ?IFC        | Illegal Character "<char>" After F<br>The only commands starting with the letter F are FS and FN. When used as a command (other than EF or in a text argument) F may not be followed by any character other than one of these.   |
| ?IFN        | Illegal Character "<char>" in Filename<br>File specifications must be of the form dev:filnam.ext[m,n] (\$) where dev, filnam, and ext are alphanumeric, and m and n are numeric. No characters other than the ones specified may appear between the EB, ER, EW, or EZ command and the altmode terminator (\$).   |
| ?ILL        | Illegal Command <char><br>The character "<char>" is not defined as a valid TECO command.   |
| ?ILR        | Cannot Lookup Input File filnam.ext to Rename It<br>Failure in rename process at close of editing job initiated by an EB command or a TECO command. Unable to do a LOOKUP on the original input file dev:filnam.ext in order to rename it "filnam.BAK". The output file is closed with the name "nnnTEC.TMP", where nnn is the user's job number. The LOOKUP UO error code is nn.  |
| ?INP-nn0000 | Input Error nn0000 on File filnam.ext.<br>A read error has occurred during input. The input file filnam.ext has been released. The user may try again to read the file, but if the error persists, the user will have to return to his backup file. The input device status word error flags are nn0000.<br>(Note: This number represents the I/O status word (rh) with bits 22-35 masked out.)<br>(040000 -- block too large).<br>(100000 -- parity or checksum error).<br>(140000 -- block too large and parity error).<br>(200000 -- device error, data missed).<br>(240000 -- block too large and device error).<br>(300000 -- parity error and device error).<br>(340000 -- block too large, parity error, and device error). |

Table 3-1 (Cont)  
TECO Error Messages

|      |  |
|------|--|
|      | <p>(400000 -- improper mode).<br/>                 (440000 -- block too large and improper mode).<br/>                 (500000 -- parity error and improper mode).<br/>                 (540000 -- block too large, parity error, and improper mode).<br/>                 (600000 -- device error and improper mode).<br/>                 (640000 -- block too large, device error, and improper mode).<br/>                 (700000 -- parity error, device error, and improper mode).<br/>                 (740000 -- block too large, parity error, device error, and improper mode).</p> |
| ?IOS | <p>Illegal Character "&lt;char&gt;" in I/O Switch<br/>                 The only valid characters in switches used with file selection commands are the alphabetic characters.</p>  |
| ?IQC | <p>Illegal command " &lt;char&gt;<br/>                 The only valid "commands are "G, "L, "N, "E, "C, "A, "D, "V, "W, "T, "F, "S, and "U.</p>  |
| ?IQN | <p>Illegal Q-register Name "&lt;char&gt;"<br/>                 The Q-register name specified by a Q, U, X, G, %, M, [, ], or * command must be a letter (A through Z) or a digit (0 through 9).</p>  |
| ?IRB | <p>Cannot Rename Input File filnam.ext to filnam.BAK<br/>                 Failure in rename process at close of editing job initiated by an EB command or a TECO command. The attempt to rename the original input file filnam.ext to the backup filename "filnam.BAK" has failed. The output file is closed with the name "nnnTEC.TMP", where nnn is the user's job number. The RENAME UO error code is nn.</p>   |
| ?IRN | <p>Cannot RE-Init Device dev for Rename Process<br/>                 Failure in rename process at close of editing job initiated by an EB command or a TECO command.<br/>                 Cannot reinitialize the original input device dev in order to rename the input file filnam.ext to "filnam.BAK". The output file is closed with the name "nnnTEC.TMP", where nnn is the user's job number.</p>  |
| ?ISA | <p>n Argument with Search Command<br/>                 The argument preceding a search command indicates the number of times a match must be found before the search is considered successful. This argument must be greater than 0.</p>   |
| ?MAP | <p>Missing '<br/>                 In attempting to execute a conditional skip command (a " command whose argument does not satisfy the required condition) no ' command closing the conditional execution string can be found. Note: n'...' strings must be complete within a single macro level.</p>  |

Table 3-1 (Cont)  
TECO Error Messages

|      |   |
|------|---|
| ?MEE | <p>Macro Ending with E<br/>A command macro being executed from a Q-register ends with the character "E". This is an incomplete command. E is the initial character of an entire set of commands. The other character of the command begun by E must be in the same macro with the E.</p>  |
| ?MEF | <p>Macro Ending with F<br/>A command macro being executed from a Q-register ends with the character "F" (not an, EF). This is an incomplete command. F is the initial character of an entire set of commands. The other character of the command begun by F must be in the same macro with the F.</p>   |
| ?MEO | <p>Macro Ending with Unterminated O Command<br/>The last command in a command macro being executed from a Q-register is an O command with no altmode to mark the end of the tag-name argument. The argument for the O command must be complete within the Q-register.</p>   |
| ?MEQ | <p>Macro Ending with "<br/>A command macro being executed from a Q-register ends with the " character. This is an incomplete command. The " command must be followed by one of the characters G, L, N, E, C, A, D, V, W, T, F, S, or U to indicate the condition under which the following commands are to be executed. This character must be in the Q-register with the "<br/> </p> |
| ?MEU | <p>Macro Ending with †<br/>A command macro being executed from a Q-register ends with the † character. This is an incomplete command. The † command takes a single character text argument that must be in the Q-register with the †.</p>   |
| ?MIQ | <p>Macro Ending with &lt;char &gt;<br/>A command macro being executed from a Q-register ends with the character "&lt;char&gt;". This is an incomplete command. The &lt;char&gt; command takes a single character text argument to name the Q-register to which it applies. This argument must be in the same macro as the &lt;char&gt; command itself.</p>                            |
| ?MLA | <p>Missing &lt;<br/>There is a right angle bracket not matched by a left angle bracket somewhere to its left. (Note: an iteration in a macro stored in a Q-register must be complete within the Q-register.)</p>  |
| ?MLP | <p>Missing (<br/>Command string contains a right parenthesis that is not matched by a corresponding left parenthesis.</p>   |
| ?MRA | <p>Missing &gt;<br/>In attempting to exit from an iteration field with a ; command (or to skip over an iteration field with a 0 argument) no &gt; command closing the iteration can be found. Note: iteration fields must be complete within a single macro level.</p>  |

Table 3-1 (Cont)  
TECO Error Messages

|      |  |
|------|--|
| ?MRP | Missing )<br>The command string contains, within an iteration field, a parenthetical expression that is not closed by a right parenthesis.   |
| ?MUU | Macro Ending with † †<br>A command macro being executed from a Q-register ends with control-† or † †. This is an incomplete command. The † † command takes a single character text argument that must be in the Q-register with the † †. |
| ?NAE | No Argument Before =<br>The command n= or n== causes the value n to be typed. The = command must be preceded by either a specific numeric argument or a command that returns a numeric value.  |
| ?NAI | No Altmode after nI<br>Unless the EO value has been set to 1, the numeric insert command nI must be immediately followed by altmode.   |
| ?NAQ | No Argument Before ''<br>The '' command must be preceded by a single numeric argument on which the decision to execute the following commands or skip to the matching ' is based.  |
| ?NAU | No Argument Before U<br>The command nUi stores the value n in Q-register i. The U command must be preceded by either a specific numeric argument or a command that returns a numeric value.  |
| ?NCS | No Command String Seen Prior to *i<br>The *i command saves the preceding command string in Q-register i. In this case no command string has previously been given.   |
| ?NFI | No File for Input<br>Before issuing an input command (Y or A) it is necessary to open an input file by use of an ER, EB, or TECO command.  |
| ?NFO | No File for Output<br>Before giving an output command (PW, P, N, EX, or EG) it is necessary to open an output file by use of an EB, EW, EZ, MAKE, or TECO command.   |
| ?NTQ | No Text in Q-register x<br>Q-register x, specified by a G or M command, does not contain text.   |
| ?OCT | "8" or "9" in Octal Digit String<br>In a digit string preceded by †O, only the octal digits 0-7 may be used.   |
| ?ODV | Output Device dev Not Available<br>Initialization failure. Unable to initialize the device dev for output. Either the device is being used by someone else right now, or it is write locked, or else it does not exist in the system.    |

Table 3-1 (Cont)  
TECO Error Messages

|             |  |
|-------------|--|
| ?OLR        | <p>Cannot Lookup Output File dev:filnam.ext to Rename It<br/>Failure in rename process at close of editing job initiated by an EB command or a TECO command. The special LOOKUP on the output file filnam.ext required for DECTape in order to rename the file to 'filnam.ext' has failed. The original input file filnam.ext has been renamed 'filnam.BAK', but the output file is closed with the name 'nnnTEC.TMP', where nnn is the user's job number. The LOOKUP UO error code is nn.</p>   |
| ?OUT-nn0000 | <p>Output Error nn0000 - Output File nnnTEC.TMP Closed<br/>An error on the output device is fatal. The output file is closed at the end of the last data that was successfully output. It has the filename 'nnnTEC.TMP', where nnn is the user's job number. See Section 4.3 for a recovery technique. The output device status word error flags are nn0000. (Note: This number represents the I/O status word (rh) with bits 22-35 masked out.)<br/>(000000 -- end of tape).<br/>(040000 -- block number too large: device full or quota exceeded).<br/>(100000 -- parity or checksum error).<br/>(140000 -- block number too large and parity error).<br/>(200000 -- device error, data missed).<br/>(240000 -- block number too large and device error).<br/>(300000 -- parity error and device error).<br/>(340000 -- block number too large, parity error, and device error).<br/>(400000 -- improper mode or device write locked).<br/>(440000 -- block number too large and improper mode).<br/>(500000 -- parity error and improper mode).<br/>(540000 -- block number too large, parity error, and improper mode).<br/>(600000 -- device error and improper mode).<br/>(640000 -- block number too large, device error, and improper mode).<br/>(700000 -- parity error, device error, and improper mode).<br/>(740000 -- block number too large, parity error, device error, and improper mode).</p> |
| ?PAR        | <p>Confused Use of Parentheses<br/>A string of the form (...&lt;...) has been encountered. Parentheses should be used only to enclose combinations of numeric arguments. An iteration may not be opened and not closed between a left and right parenthesis.</p>   |
| ?POP        | <p>Attempt to Move Pointer Off Page with J, C, R, or D<br/>The argument specified with a J, C, R, or D command must point to a position within the current size of the buffer, i.e., between 0 and Z, inclusive.</p>   |
| ?PPN        | <p>Illegal Character '&lt;char&gt;' in Project-programmer Number<br/>Project-programmer numbers in file specifications must be given in the form [m,n] where m and n are octal digit strings separated by a comma. No characters other than the ones specified may appear between the enclosing brackets.</p>  |

Table 3-1 (Cont)  
TECO Error Messages

|      |  |
|------|--|
| ?RNO | <p>Cannot Rename Output File nnnTEC.TMP<br/>Failure in rename process at close of editing job initiated by an EB command or a TECO command. The attempt to rename the output file nnnTEC.TMP to the name "filnam.ext" originally specified in the EB or TECO command has failed. The original input file filnam.ext has been renamed "filnam.BAK", but the output file is closed with the name "nnnTEC.TMP", where nnn is the user's job number. The RENAME UO error code is nn.</p>   |
| ?SAL | <p>Second Argument Less Than First<br/>In a two-argument command, the first argument must be less than or equal to the second.</p>   |
| ?SNA | <p>Initial Search with No Argument<br/>A search command with null argument has been given, but there was no preceding search command from which the argument could be taken.</p>   |
| ?SNI | <p>; Not in an Iteration<br/>The semicolon command may be used only with a string of commands enclosed by angle brackets, i.e., in an iteration field.</p>   |
| ?SRH | <p>Cannot Find "&lt;text&gt;"<br/>A search command not preceded by a colon modifier and not within an iteration has failed to find the specified character string "&lt;text&gt;". After an S search fails the pointer is left positioned at the beginning of the buffer. After an N or ← search fails the last page of the input file has been input and, in the case of the N, output, and the buffer cleared. Note that when this message occurs, the text string printed includes all control-character commands included in the search argument.</p> |
| ?STC | <p>Search String Too Long<br/>The maximum length of a search string is 80 characters including all string control commands and their arguments.</p>  |
| ?STL | <p>Search String too Long<br/>The maximum length of a search string is 36 character positions, not counting extra characters required to specify a single position.</p>  |
| ?TAG | <p>Missing Tag !xxx!<br/>The tag !xxx! specified by an O command cannot be found. This tag must be in the same macro level as the O command referencing it.</p>  |
| ?TAL | <p>Two Arguments with L<br/>The L command takes at most one numeric argument, namely, the number of lines over which the buffer pointer is to be moved.</p>  |
| ?TTY | <p>Illegal TTY I-O Device<br/>A terminal may be specified as an input-output device in an ER, EW, EZ, or MAKE command only if it is not being used to control an attached job, the user's own terminal included.</p>   |

Table 3-1 (Cont)  
TECO Error Messages

|      |  |
|------|--|
| ?UCA | <p>Unterminated ↑A Command<br/>A ↑A message type-out command has been given, but there is no corresponding ↑A to mark the end of the message. ↑A commands must be complete within a single command level.</p>  |
| ?UFS | <p>Macro Ending with Unterminated File Selection Command<br/>The last command in a command macro being executed from a Q-register is a file selection command (ER, EW, EB, or EZ) with no altmode to mark the end of the file specifications. The file selection command must be complete within the Q-register.</p> |
| ?UIN | <p>Unterminated Insert Command<br/>An insert command (possibly an @ insert command) has been given without terminating the text argument at the same macro level.</p>  |
| ?UIS | <p>Undefined I/O Switch "/xxx"<br/>The switch "/xxx" is not defined with either input or output file selection commands. The only switches currently defined for input or output file selection commands are /GENLSN and /SUPLSN.</p>  |
| ?USR | <p>Unterminated Search Command<br/>A search command (possibly an @ search command) has been given without terminating the text argument at the same macro level.</p>   |
| ?UTG | <p>Unterminated Tag<br/>A command string tag has been indicated by a ! command, but there is no corresponding ! to mark the end of the tag. Tags must be complete within a single command level.</p>   |
| ?UUO | <p>Illegal UUO<br/>Internal error. The illegal instruction &lt;lh,rh&gt; has been encountered at address nnnnnn. This is caused by either a TECO bug or a monitor bug. Please give this printout to your system manager, or submit it to DEC with an SPR.</p>  |

reference



**decsystem10 TECO**  
**TEXT EDITOR AND CORRECTOR PROGRAM**  
**PROGRAMMER'S REFERENCE MANUAL**

This manual reflects the software as of Version 23 of TECO.

1st Printing January 1968  
2nd Printing October 1968  
3rd Printing August 1969  
4th Printing April 1970  
5th Printing (Rev) October 1970  
6th Printing (Rev) May 1972

Copyright © 1968, 1969, 1970, 1971, 1972 by Digital Equipment Corporation

The material in this manual is for information purposes and is subject to change without notice.

The following are trademarks of Digital Equipment Corporation, Maynard, Massachusetts:

|           |              |
|-----------|--------------|
| DEC       | PDP          |
| FLIP CHIP | FOCAL        |
| DIGITAL   | COMPUTER LAB |

#### NEW AND CHANGED INFORMATION

This manual reflects the software as of version 23. It has been revised to include all new and changed material since version 21A of the TECO software. Change bars in the left margin are used to indicate the new and revised information.

TECO

- 224 -

CONTENTS

|   | Page |
|---|------|
| CHAPTER 1 INTRODUCTION                                      | 231  |
| CHAPTER 2 CONCEPTS  | 233  |
| 2.1 Data Files  | 233  |
| 2.2 Character Set   | 234  |
| 2.2.1 Special Characters                                    | 235  |
| 2.2.2 Control Characters                                    | 235  |
| 2.2.3 Carriage Control Functions                            | 236  |
| 2.2.4 Symbols   | 236  |
| 2.3 Data Format   | 237  |
| 2.4 Editing Buffer  | 238  |
| 2.5 Buffer Pointer  | 239  |
| 2.6 General Command String Syntax                           | 239  |
| 2.7 Arguments   | 240  |
| 2.7.1 Alphanumeric Arguments                                | 240  |
| 2.7.2 Numeric Arguments                                     | 241  |
| 2.7.3 Commands That Return a Value                          | 243  |
| 2.8 Q-Registers   | 243  |
| 2.9 Core Expansion  | 244  |
| CHAPTER 3 COMMANDS  |      |
| 3.1 Initialization Commands                                 | 247  |
| 3.1.1 R TECO Command  | 247  |
| 3.1.2 MAKE Command  | 247  |
| 3.1.3 TECO Command  | 248  |
| 3.1.4 Examples of the Use of Initialization Commands        | 250  |
| 3.2 File Selection Commands                                 | 250  |
| 3.2.1 ER Command  | 251  |
| 3.2.2 EM Command  | 251  |
| 3.2.3 EW Command  | 251  |
| 3.2.4 EZ Command  | 253  |
| 3.2.5 EB Command  | 253  |
| 3.2.6 Editing Line-Sequence Numbered Files                  | 254  |
| 3.2.7 Examples of the Use of File Section Commands          | 254  |
| 3.3 Input Commands  | 255  |
| 3.3.1 Y Command   | 256  |
| 3.3.2 A Command   | 257  |
| 3.3.3 Examples of the Use of Input Commands                 | 257  |
| 3.4 Special Characters as Buffer Position Numeric Arguments | 257  |
| 3.5 Buffer Pointer Positioning Commands                     | 258  |
| 3.5.1 J Command   | 258  |

## CONTENTS (Cont)

|         | Page   |     |
|---------|--|-----|
| 3.5.2   | C Command  | 258 |
| 3.5.3   | R Command  | 258 |
| 3.5.4   | L Command  | 259 |
| 3.5.5   | Examples of the Use of Buffer Pointer Positioning Commands | 259 |
| 3.6     | Text Type-Out Commands                                     | 260 |
| 3.6.1   | T Command  | 260 |
| 3.6.2   | Ⓢ Command  | 260 |
| 3.6.3   | Ⓛ Command  | 261 |
| 3.6.4   | nET Command  | 261 |
| 3.6.5   | Case Flagging On Type-out                                  | 262 |
| 3.6.6   | Examples of the User Text Typeout Commands                 | 262 |
| 3.7     | Deletion Commands  | 264 |
| 3.7.1   | K Command  | 264 |
| 3.7.2   | D Command  | 264 |
| 3.7.3   | Examples of the Use of Deletion Commands                   | 265 |
| 3.8     | Insertion Commands   | 265 |
| 3.8.1   | I Command  | 266 |
| 3.8.2   | Tab Command  | 266 |
| 3.8.3   | @I Command   | 266 |
| 3.8.4   | nI Ⓢ Command   | 266 |
| 3.8.5   | n\ Command   | 267 |
| 3.8.6   | Examples of the Use of Insertion Commands                  | 267 |
| 3.8.7   | Case Control with Insert Commands                          | 268 |
| 3.8.7.1 | Alphabetic Case Control                                    | 268 |
| 3.8.7.2 | Special "Lower Case" Characters                            | 270 |
| 3.8.8   | Inserting Control Characters                               | 271 |
| 3.9     | Output Commands  | 272 |
| 3.9.1   | PW Command   | 272 |
| 3.9.2   | P Command  | 272 |
| 3.9.3   | EF Command   | 274 |
| 3.9.4   | Examples of the Use of Output Commands                     | 274 |
| 3.10    | Exit Commands  | 275 |
| 3.10.1  | EX Command   | 275 |
| 3.10.2  | EG Command   | 276 |
| 3.10.3  | Ⓢ and Ⓢ Commands   | 276 |
| 3.11    | Search Commands  | 278 |
| 3.11.1  | S Command  | 279 |
| 3.11.2  | FS Command   | 279 |
| 3.11.3  | N Command  | 279 |

CONTENTS (Cont)

|          | Page   |     |
|----------|--|-----|
| 3.11.4   | FN Command   | 280 |
| 3.11.5   | Backarrow Command                                    | 280 |
| 3.11.6   | Search Command Modifiers                             | 281 |
| 3.11.6.1 | @Modifier  | 281 |
| 3.11.6.2 | Colon Modifier                                       | 281 |
| 3.11.7   | Automatic Typeout After Searches                     | 282 |
| 3.11.8   | Case Control in Searches                             | 282 |
| 3.11.8.1 | Alphabetic Case Control in Search Arguments          | 282 |
| 3.11.8.2 | Special "Lower Case" Characters                      | 285 |
| 3.11.8.3 | Control Characters in Search Arguments               | 285 |
| 3.11.8.4 | Case Match Mode Control in Searches                  | 285 |
| 3.11.9   | Special Match Control Characters                     | 286 |
| 3.11.10  | Examples of the Use of Search Commands               | 287 |
| 3.12     | Iteration Commands                                   | 289 |
| 3.12.1   | Angle Bracket (<...>)                                | 289 |
| 3.12.2   | Semicolon Command                                    | 289 |
| 3.13     | Flow Control Commands                                | 291 |
| 3.13.1   | Command String Tags                                  | 291 |
| 3.13.2   | O Command  | 292 |
| 3.13.3   | Conditional Execution Commands                       | 292 |
| 3.13.4   | Examples of the Use of Flow Control Commands         | 293 |
| 3.14     | Q-Register Commands                                  | 295 |
| 3.14.1   | Commands for Storing Integers                        | 295 |
| 3.14.1.1 | U Command  | 295 |
| 3.14.1.2 | Q Command  | 295 |
| 3.14.1.3 | % Command  | 295 |
| 3.14.2   | Commands for Storing Character Strings               | 295 |
| 3.14.2.1 | X Command  | 295 |
| 3.14.2.2 | G Command  | 296 |
| 3.14.2.3 | M Command  | 296 |
| 3.14.3   | Saving the Previous Command String                   | 296 |
| 3.14.4   | Q-Register Pushdown List                             | 297 |
| 3.14.5   | Examples of the Use of Q-Register Commands           | 297 |
| 3.15     | Numeric Typeout Command                              | 300 |
| 3.16     | Special Numeric Values                               | 300 |
| 3.16.1   | Examples of the Use of the Special Numeric Arguments | 302 |
| 3.17     | TECO Programming Aids                                |     |
| 3.17.1   | Ⓜ Command  | 303 |
| 3.17.2   | Question Mark (?) Command                            | 304 |
| 3.17.3   | The EO Value   | 304 |

## CONTENTS (Cont)

|            | Page  |     |
|------------|---|-----|
| 3.18       | Command String Type-in Control Commands               | 306 |
| 3.18.1     | Carriage Return, Line Feed, and Spaces                | 306 |
| CHAPTER 4  | TECHNIQUES  |     |
| 4.1        | Creation, Execution, and Editing of a FORTRAN Program | 307 |
| 4.2        | Rearranging a File                                    | 309 |
| 4.3        | Splitting and Merging Files                           | 310 |
| 4.4        | Example of an Advanced TECO Macro                     | 313 |
| CHAPTER 5  | USER ERRORS   |     |
| 5.1        | Erasing Commands                                      | 319 |
| 5.1.1      | Rubout Command  | 319 |
| 5.1.2      | Double $\text{tG}$ Command                            | 320 |
| 5.1.3      | $\text{tU}$ Command                                   | 320 |
| 5.1.4      | Bell-Space Command                                    | 321 |
| 5.2        | Error Messages  | 321 |
| 5.2.1      | Question Mark Command                                 | 322 |
| 5.2.2      | Slash Command   | 323 |
| 5.2.3      | EH Command  |     |
| APPENDICES |   |     |
| APPENDIX A | TECO ERROR MESSAGES                                   | 325 |
| APPENDIX B | ASCII CHARACTERS                                      | 337 |
| APPENDIX C | SUMMARY OF COMMANDS                                   | 345 |

CONTENTS (Cont)

|     |                                   | Page |
|-----|-----------------------------------|------|
|     | TABLES                            |      |
| 2-1 | Special Characters                | 235  |
| 2-2 | Special Symbols                   | 236  |
| 2-3 | Numeric Operators                 | 241  |
| 3-1 | EM Commands                       | 252  |
| 3-2 | Special Buffer Position Arguments | 258  |
| 3-3 | L Commands                        | 259  |
| 3-4 | T Commands                        | 260  |
| 3-5 | K Commands                        | 264  |
| 3-6 | P Commands                        | 273  |
| 3-7 | Conditional Execution Commands    | 293  |
| A-1 | TECO Error Messages               | 325  |
| B-1 | ASCII Characters                  | 337  |
| C-1 | Command Description               | 345  |

TECO

- 230 -

## Chapter 1

### Introduction

This manual is a complete reference manual for the advanced TECO user. It is not designed to be used as a beginner's text, and people who are learning TECO should not use it as such. Beginners are referred to the tutorial "Introduction to TECO", which appears in Section I of the DECsystem-10 Users Handbook.

TECO is a powerful text editor for use with all DECsystem-10 systems. TECO enables the advanced user to easily edit any ASCII text. Most editing can be accomplished using a few simple commands; or the user can select any of a large set of sophisticated commands, such as character string searching, command repetition, conditional commands, programmed editing, and text block movement. Refer to Appendix C for a summary of the commands available.

TECO editing is normally done on-line, using the terminal. However, the user can also write his editing commands as a TECO command file and have his editing task run by an operator.

TECO is a character-oriented editor; one or more characters in a line can be modified without re-typing the rest of the line. Any source document can be edited: programs written in FORTRAN, COBOL, MACRO-10, or any other language, as well as memoranda, specifications, and other types of arbitrarily-formatted text. TECO does not require that line numbers or any other extraneous information be associated with the text. The full ASCII character set, printing and nonprinting characters alike, can be processed.

TECO requires a minimum of 5K of core memory, 3K of which is shared in a reentrant system. TECO takes advantage of any additional core available to expand its buffers, as required.

A single terminal is required for typing in commands. Data can be input or output on any standard I/O device.



## Chapter 2

### Concepts

#### 2.1 DATA FILES

DECsystem-10 TECO operates on ASCII data files. The input file is the file that the user wishes to change. The output file is the file that receives the newly created or edited data.

Inputting is defined as the process of reading in data that already exists in some computer-readable form (paper tape, disk file, etc.). Data can be input from any device except the user's terminal (or another user's terminal). Inserting is defined as the actual typing in of new data and is done only at the user's terminal.

In the case of such hard-copy devices as the card reader and the paper-tape reader, only the device need be specified to open a file for input or output. For disk and DECTape files, filenames, as well as the device, must be specified. If no device is specified, the device DSK: is assumed. Magnetic tape files are specified by naming the tape drive and by using special TECO commands to position the tape properly.

Any I/O device name acceptable to the monitor can be used. Some examples are:

|       |   |
|-------|---|
| DSK:  | Disk (including drums)  |
| DTAn: | DECTape (n is the number of the drive on which the tape is mounted)       |
| MTAn: | Magnetic tape (n is the number of the drive on which the tape is mounted) |
| CDR:  | Card reader   |
| CDP:  | Card punch  |
| PTR:  | Paper-tape reader   |
| PTP:  | Paper-tape punch  |
| LPT:  | Line printer  |
| TTYn: | Terminal number n, usually a terminal having a low-speed reader or punch  |

## NOTE

TTYn: used as an I/O device must be different from the user's terminal and must not be the terminal of any attached user.

Filenames for disk and DECtape files consist of two parts: the first part, the filename proper, consists of from one to six alphanumeric characters; the second part, which is optional, is called the "filename extension." If given, the filename extension consists of from one to three alphanumeric characters and is separated from the filename proper by a period. If the filename extension is not given, it is defined as null and as such is distinctive. In the case of a null filename extension, the period after the filename proper can be omitted.

## Examples of filenames:

|            |  |
|------------|--|
| TECO.21    | The source file for TECO version 21        |
| EARNING.F4 | A FORTRAN source program                   |
| 0015J.CBL  | A COBOL source program                     |
| GLOB.MAC   | A MACRO-10 source program                  |
| GLOB.BAK   | A backup file                              |
| FRMTR.TEC  | A file containing a TECO macro             |
| M20        | A filename with null extension             |
| M20.1      | A similar filename with non-null extension |

## 2.2 CHARACTER SET

The TECO character set is the full ASCII set. To obtain particular information about individual characters, the user should refer to the table of ASCII characters in Appendix B. This table contains the following:

- a. A list of all ASCII characters and the symbols used in this manual to represent them,
- b. octal and decimal values of the characters, and,
- c. comments concerning any special significance of each character.

In general, the user must be concerned with the character set on two levels: the data level and the command level.

Every ASCII character from control-A (decimal value 01) through rubout (decimal value 127) is legal in TECO data. They can all be input and output, and they can all be inserted. The only character that is not completely legal as data is the null character (decimal value 0). The null character can be inserted and output, but it is ignored on input. Form feed characters (decimal value 12) are completely legal in data but are treated specially on input (see Sections 2.3 and 3.3).

Most of the ASCII characters have some meaning when used as commands. Some are monitor commands. When used as commands, the lower-case characters have the same meaning as their upper-case

equivalents. The table in Appendix B tells where in this manual the uses of the various characters as commands are explained.

### 2.2.1 Special Characters

Because of their use as special immediate-action commands (monitor control commands or erasing commands), certain characters must not be typed in explicitly as alphanumeric arguments. All of them, however, are legal as data (except the null character) and can be inserted using special techniques. The characters to which this restriction applies are referred to in this manual as "special characters." These special characters are listed in Table 2-1.

Table 2-1  
Special Characters<sup>†</sup>

| Character                        | Remarks   |
|----------------------------------|---|
| ⓐ (control-C)                    | A monitor command   |
| ⓑ ⓐ (two successive control-G's) | An erasing command (A single control-G is acceptable.)                                  |
| ⓐ␣ (control-G, space)            | Immediate editing command (causes current line to be retyped).                          |
| ⓐ (control-O)                    | A monitor command   |
| ⓑ (control-U)                    | An erasing command  |
| ESCAPE or PREFIX                 | Equivalent to ALTmode   |
| ALTmode or ⓐ                     | Standard text argument terminator (Two successive ALTmodes terminate a command string.) |
| Rubout                           | An erasing command  |

<sup>†</sup> In monitors preceding the 5.02 monitor the characters ⓑ, ⓐ, and ⓐ are also monitor commands and must be included in the above list for these systems.

### 2.2.2 Control Characters

Control characters are characters that are typed by holding down the CTRL key while striking a character key. The control characters have decimal values 0 through 31. When TECO is printing text, a control character is printed as an up-arrow, followed by the character which is typed to produce the control character. For example, control-A prints as "↑A".

In many cases the control character commands can be typed into command strings by using an alternate procedure to the standard method of holding down the CTRL key while striking the desired character.

Instead, the user can first type up-arrow and then type the desired character without depressing the CTRL key. For example, when used as a command, the two-character sequence up-arrow, H (denoted by ↑H) is equivalent to the single character control-H (denoted by ⓂH). This method can be used only when the control character is typed as a command, not when it is typed as text or as an alpha-numeric argument. Control characters appearing as text arguments must be preceded by a Ⓜ. Exceptions are noted at appropriate places throughout the manual.

### 2.2.3 Carriage Control Functions

A few of the control characters are the special terminal functions: bell, tab, line feed, vertical tab, form feed, and carriage return. All of these characters echo by performing their particular function; they also perform this function when TECO is printing out text from the buffer.

When a carriage return is typed in, the monitor automatically generates a line feed following it. The echo to the carriage return type-in is a carriage return followed by a line feed. If the carriage return is typed as an insert, a line feed is automatically inserted immediately after the carriage return.

Altmode (or escape or prefix) echoes and prints out as a dollar sign.

### 2.2.4 Symbols

In the examples in this manual, some special symbols are used to clearly indicate what the user must type. These special symbols are listed in Table 2-2.

In all examples containing both characters typed by the monitor or TECO and characters typed by the user, the characters typed by the monitor or TECO are underlined. Carriage control characters (carriage return, form feed, etc.) typed by the user are indicated through use of the special symbols.

Table 2-2  
Special Symbols

| Symbol                                       | Character              |
|--|------------------------|
| →  | tab                    |
| ↓  | line feed              |
| ⓂVT  | vertical tab           |
| ⓂFORM  | form feed              |
| ↵  | carriage return        |
| ␣  | space                  |
| Ⓜ\$  | altmode                |
| ⓂRC  | rubout                 |
| ⓂA   | control-A              |
| ↑A   | up-arrow followed by A |
| (Other control characters similarly denoted) |                        |

### 2.3 DATA FORMAT

TECO is capable of editing text written in any format. There are, however, features in TECO that make use of the concept of a line and the concept of a page. Therefore, the user must know how these concepts are defined in TECO.

Lines can be of any length. The characters that define the end of a line are the line feed, vertical tab, and form feed. The end of the editing buffer also counts as an end-of-line character if there is no other end-of-line character at the end of the buffer. When TECO counts lines, it does so by counting these end-of-line characters. An end-of-line character is considered to belong to the line that it terminates.

Examples:

The following text comprises three lines of text as defined by TECO:

```
LINE ONE ↓
      LINE TWO ↵ ↓
LINE THREE ↵ ↓
```

The following text is considered to be two lines:

```
BEGINNING ↵ OVERPRINT (VT) CONTINUATION ↵ ↓
```

The first line is terminated by the (VT) character and the second by the ↓ character.

Text to be edited by TECO does not have to contain end-of-line characters; however, if it does not contain them, those features of TECO that count lines will not be useful.

#### NOTE

If the EO value has been set to 1, the only end-of-line character is the line feed (refer to Paragraph 3.17.3 for a description of the EO value).

Pages are defined in TECO by form feed characters, which act as page separators. They are not considered to belong to either of the two pages that they separate. Two consecutive form feed characters delimit a null page. A form feed character at the beginning of a file delimits a null page at the beginning of the file. A form feed character at the end of a file has no effect in TECO. It can be omitted.

## Examples:

The following file consists of two pages:

```

LINE ONE ↵↓
LINE TWO ↵↓
(FORM) LINE THREE ↵↓
LINE FOUR ↵↓

```

The following consists of four pages; the first and third pages are null:

```

(FORM) LINE ONE ↵↓
LINE TWO ↵↓
(FORM) (FORM) LINE THREE ↵↓
LINE FOUR ↵↓

```

TECO operates most efficiently with files that are divided into pages of approximately fifty or fewer lines. Files with longer pages or files containing no form feed characters can be edited with TECO; but, this process requires either additional core storage or more care when editing.

The processing of form feed characters by TECO must be thoroughly understood by the user. The page concept is further discussed in relation to the size of the editing buffer in Section 2.4, and the relation of form feed characters to input and output commands is discussed in Sections 3.3, 3.9, 3.10, and 3.11.

TECO may be used to edit files containing the special line-sequence numbers produced by BASIC, the PIP /S switch, LINED, and several other editors, but TECO does not need these numbers and makes no special use of them (nor does it destroy them). See Section 3.2.6 for an explanation of how these numbers may be processed.

## 2.4 EDITING BUFFER

Editing is accomplished by:

- a. Reading text into the editing buffer
- b. Making changes to the text in this buffer
- c. Writing the modified text out to a new file

The editing buffer is a block of core memory within TECO. Data is put in the editing buffer when it is read in or inserted; it is kept in the editing buffer while it is being modified.

Text is packed in the editing buffer with five 7-bit ASCII characters per 36-bit word. When TECO is running in the minimum 5K of core, the editing buffer holds approximately 3600 characters. Each additional 1K of core assigned to TECO increases the size of the editing buffer by 5120 characters.



line feed. At that point, each command in the string is executed in turn, starting at the left. When all commands in the string have been executed, TECO prints another asterisk indicating it is ready to accept another command string.

If a command in the string cannot be executed due to a command error, execution of the command string stops at that point, and an error message is printed. Commands preceding the command in error are executed. The erroneous command and the commands following it are not executed. Errors, error messages, and recovery techniques are fully discussed in Chapter 5.

There are exceptions to the general rule that commands are not executed until the end of the command string has been indicated by two consecutive altmodes. These exceptions are the commands listed in Table 2-1 in Section 2.2.

## 2.7 ARGUMENTS

### 2.7.1 Alphanumeric Arguments

Most alphanumeric arguments are text arguments that are interpreted as ASCII data by TECO. Some examples of text arguments are: data to be inserted in the buffer, search character strings, and command string tags. Other types of alphanumeric arguments are device and filenames and Q-register names.

An alphanumeric argument always follows the command to which it applies. As a rule, most commands that take text arguments require that the argument be terminated by an altmode; however, there are exceptions to this rule which are explained at appropriate places in the manual.

An altmode used to terminate an alphanumeric argument can also function as one of the two altmodes necessary to terminate a command string.

Example:

\* ITEXT (\$) STTEXT2 (\$) (\$)  
\*  
-

The alphanumeric argument, "TEXT", is terminated by an altmode. The second argument, "TEXT2", is also terminated by an altmode, but this altmode is also used as one of the altmodes terminating the command string.

Any printable ASCII character is legal in an alphanumeric argument with the exception of the special characters listed in Table 2-1, Section 2.2. In addition, non-printing characters are legal when they are preceded by a (R).

### 2.7.2 Numeric Arguments

Numeric arguments always precede the command to which they apply. In some cases, only a single numeric argument is required; in others, a pair of numeric arguments is required.

When two numeric arguments are used, they are separated by a comma. In most cases, numeric arguments must be positive; however, some commands allow a numeric argument to be negative or zero. The number and type of numeric arguments allowed by each command are stated in the section in which that command is explained.

Where a numeric argument is used to specify a buffer position, the number used is the number of characters in the buffer to the left of that position. Thus, n means the position to the right of the nth character in the buffer (between the nth and n+ 1st characters).

Numeric arguments used in pairs are always buffer position arguments. Such a pair specifies all the characters in the buffer that lie between the two buffer positions represented by the two arguments. This definition is precise because the term "buffer position" always indicates a position before or after a given character, not "on" or "at" the character.

Example:

12,20 This argument pair specifies the thirteenth (13th) through the twentieth (20th) characters in the buffer. These characters are specified because the 12 indicates the position between the 12th and 13th characters, and the 20 indicates the position between the 20th and 21st characters.

Numeric arguments can be used in arithmetic/logical combinations. The characters shown in Table 2-3 are used as operators.

Table 2-3  
Numeric Operators

| Operator   | Function   | Example                   |
|------------|--|---------------------------|
| +          | Ignored, if used before the first term in a string.      | +2=2                      |
| +<br>space | Addition, if used between two terms.<br>Equivalent to +. | 5+6=11<br>└ 2=2<br>5└6=11 |
| -          | Negation, if used before the first term in a string.     | -2=-2                     |

Table 2-3 (Cont)  
Numeric Operators

| Operator | Function   | Example        |
|----------|--|----------------|
| -        | Subtraction, if used between terms.  | 8-2=6          |
| *        | Multiply. (Used between two terms.)  | 8*2=16         |
| /        | Integer Divide (and drop the remainder).<br>(Used between two terms.)                      | 8/2=4<br>8/3=2 |
| &        | Bitwise logical AND of the binary representations of two terms, if used between the terms. | 12 & 10=8      |
| #        | Bitwise logical OR of the binary representations of two terms, if used between the terms.  | 12# 10=14      |

When more than one arithmetic/logical operator is used in a single numeric argument, the operations are performed from left to right. This sequence can be overridden through use of parentheses (). All operations within parentheses are performed before those outside parentheses. Parentheses can be nested:

In TECO, numbers are ordinarily assumed to be decimal integers. Preceding a number with  $\uparrow$ O (uparrow-O, not control-O) causes the number to be read in octal radix.

Example:

$\uparrow$ O177 is equivalent to 127.

Examples:

$3 * \uparrow$ O10=24  
 $2 + 3 * 4 = 20$   
 $2 + (3 * 4) = 14$   
 $2 + (3 * (16 / (3 - 1)) / 2 + (2 * 5)) = 24$   
 $2 \& (3 \# 5) \# 16 = 18$   
 $-(2 + (3 * 4) - 1 \& (6 + 8)) / 2 = -6$

The arithmetic/logical operators and parentheses can be used to form one or both of the numeric arguments in a pair.

Example:

$260 - (3 * 42), 250 + (77/3)$

### 2.7.3 Commands That Return a Value

Generally speaking, there are two main categories of TECO commands: 1) those that perform some operation, such as inserting text, and 2) those that "return" a value, such as the number of characters in the editing buffer. (There are also some commands that do both.)

A command is said to "return" a value if the command causes the current value of some quantity to be calculated, and then the command takes on this value, becoming itself a numeric argument that may be used by another command. Using such a command is equivalent to typing the particular number that the command returns as a value, except that the value is not usually known in advance. This value can then be used as an argument by the next command in the command string, provided that the command is one that can take a numeric argument. Otherwise, it is ignored.

An example of a command that returns a value is the Z command (see Section 3.4). The Z command returns a value equal to the number of characters in the buffer. It has no other function. Thus, in order to be useful, Z must be used as a numeric argument preceding another command.

Commands that return values may be used in arithmetic/logical combinations with each other and with explicit numbers. All the same rules apply. Each command that returns a value has all the properties of a number that has been explicitly typed in.

If commands that return values are concatenated with each other or with digits, the value returned is that of the last command or number in the string. An operator preceding such a string continues to apply.

Examples:

```
ZZ = Z
Z48 = 48
-2Z = -Z
3+ZZ = 3+Z
```

## 2.8 Q-REGISTERS

Q-registers are data storage registers that are available to the TECO user. Q-registers give a great amount of editing power to the user by enabling programmed editing and text block movement. Data stored in Q-registers is not disturbed by the flow of data into and out of the editing buffer. It can be preserved throughout an entire TECO job, and it is available for retrieval or change at any time.

There are 36 Q-registers; each Q-register has a single character name, which is either one of the digits 0 through 9, or one of the letters A through Z. Also, there is a Q-register pushdown stack that effectively makes available an additional 32 Q-registers for certain applications.<sup>1</sup>

<sup>1</sup>The number of entries in the pushdown stack can be increased by changing the parameter LPF in TECO. MAC and reassembling TECO.

Two types of data can be stored in Q-registers: decimal integers or alphanumeric character strings.

For numeric storage, a Q-register can be used to hold a single positive, negative, or zero decimal integer in the range  $-2^{35} + \leq n \leq 2^{35} - 1$ . Numbers stored in Q-registers can be incremented, tested, or recalled. Hence, Q-registers can be used as switches and counters, as well as for simple data-save functions.

For text storage, a Q-register can be used to hold a character string of any length. Two types of character strings can be stored: ordinary text and TECO command strings. Ordinary textual data stored in a Q-register is copied into the Q-register from the editing buffer without destroying the copy in the editing buffer. Storing text in a Q-register is useful for functions such as making many copies of a given segment of text throughout a file without retyping it each time, for moving a block of text from one position to another in a file, and for moving a block of text to another file.

Textual data in the form of TECO command strings can also be stored in Q-registers. Such a command string can be executed over and over throughout an editing job, much like calling a subroutine. This feature also enables an editing job to be typed up off-line and then executed by an operator at a later time. Such command strings can be edited just as any other text.

## 2.9 CORE EXPANSION

The minimum 5K of core memory is allocated within TECO in the following manner. The executable code is allocated 3K of core memory; this code is pure and is shared in a reentrant system. The other 2K of core memory is allocated to the data segment. Part of the data segment is used for program variables and fixed-length I/O buffers, while the rest is used for three variable-length storage areas:

- a. The editing buffer,
- b. the command string buffer, and
- c. the storage area for Q-registers containing text.

When TECO is initialized, the three variable-length storage areas are assigned a specific amount of space. After a command string is executed, the command string buffer is cleared. When text is deleted from the editing buffer, the formerly occupied space is reclaimed. However, during a TECO job, conditions can arise where the available space is not sufficient for the three variable-length storage areas. For example, a command string having a single insert command with many lines of text to be inserted may overflow the command string buffer. In such a case, TECO attempts to obtain the required space from one of the other variable-length storage areas. If, however, all three areas are filled to such an extent that the total amount of space allotted to all three is insufficient, TECO automatically requests another 1K of core memory from the monitor.

If the request for more core is successful, operation continues normally. TECO prints a message of the form "[nK CORE]" (where n is the new number of 1K segments of (low) core allocated to the

user) to inform the user that his core has been expanded to the specified amount. (This message is suppressed while the user is typing a command string.) If the request for more core is unsuccessful, TECO stops execution of the command string at this point and prints the error message ?COR Storage Capacity Exceeded.

TECO

## Chapter 3 Commands

### 3.1 INITIALIZATION COMMANDS

TECO is called by giving one of three different initialization commands to the monitor. An initialization command can be given whenever the monitor has typed a period to indicate that it is waiting for a new command.

#### 3.1.1 R TECO Command

The general TECO initialization command is the command:

```
.R TECO )  
*  
-
```

This command calls TECO into core and initializes the program for general use. It does not automatically initialize any particular devices or files for input or output.

When initialization is complete, an asterisk is typed to indicate that TECO is ready to receive a command. This state, in which TECO waits for command string type in, is called command mode or the idle state.

The R TECO command can be given with an argument:

```
.R TECO n )
```

where n is a decimal integer. The argument is used to request more than the minimum of 5K of core memory for the TECO job. If n is greater than 5, the monitor initializes the user's TECO job with nK of core, if possible. If n is not greater than 5, it has no effect.

#### 3.1.2 MAKE Command

- The two main uses of TECO are (1) to create a new file, and (2) to edit an existing file. These two uses are so common that there are special monitor commands to initialize TECO for executing them.

The command:

- ```
.MAKE dev:filnam.ext[proj,prog] )
```

is used to initialize TECO for creating a new file. Filnam.ext is the name that the user, using this command, gives to the new file. Dev: is the device on which the file is to be created; it can be any output device. If dev: is omitted, DSK: is assumed. If the output device is a disk device, [proj,prog] is used to specify the user area in which the file is to be created; if [proj,prog] is omitted and the device is DSK:, the file will be created in the user's own disk area. For a more precise explanation of file specifications (dev:filnam.ext[proj,prog]), see Section 3.2.1.

The MAKE command opens a new file to receive output from TECO and gives it the name specified. Once the file has been opened, it is then actually created by using the insert and output commands.

Care should be used in the choice of the filename used with a MAKE command. If there is already a file on the system device with the name specified, the MAKE command will cause the old file to be overwritten and TECO will output the warning message %SUPERSEDING EXISTING FILE:. If the user does not wish to supersede the file, he should type  $\textcircled{C}$  to return to the monitor. If no filename is used with a MAKE command, the name of the last ASCII file used in a MAKE command or any other edit-class command (MAKE, TECO, EDIT, or CREATE) is used. If no filename is given in a MAKE command and no edit-class command was previously given, the error message "COMMAND ERROR" is typed.

When initialization is completed, TECO types an asterisk to indicate its readiness to receive a command string. Usually the first command following a MAKE command is an insert command.

```
._MAKE dev:filnam.ext[proj,prog] )
```

is equivalent to

```
._R TECO )
*_EWdev:filnam.ext[proj,prog]  $\textcircled{C}$   $\textcircled{C}$ 
```

### 3.1.3 TECO Command

The command

```
._TECO dev:filnam.ext[proj,prog] )
```

is used to initialize TECO for editing an existing file on disk or DECTape. The file specifications dev:filnam.ext[proj,prog] are interpreted in the same way as for the MAKE command, except that the device must be a directory-structured device (disk or DECTape).

The filename and filename extension must be exactly the same as those of the file that is to be edited.

The TECO command opens the specified file for input and reads in the first page of that file. It also opens a new file, with a temporary name, for output of the edited version. The temporary name is of the form nnnTEC.TMP, where nnn is the user's job number, including leading zeros. When output of the new version is completed, the original (input) version of the file is automatically renamed filnam.BAK, and the new version is given the name of the original file. This operation is identical to that used for the EB command (see Section 3.2.5).

If no filename is specified in a TECO command, the name of the ASCII file last referenced in any edit-class command is assumed. If no filename is specified and no edit-class command has previously been given, the error message "COMMAND ERROR" is typed. The TECO command cannot be used with a file having the filename extension .BAK, nor with a file name nnnTEC.TMP, where nnn is the user's job number.

When initialization is completed, TECO types an asterisk to indicate its readiness to receive a command string.

The command

`._TECO dev:filnam.ext [proj,prog]`

is equivalent to

`._R TECO`  
`*EBdev:filnam.ext [proj,prog] (Y) (Y) (Y)`

If the project-programmer number specified in a TECO `filnam.ext [proj,prog]` command is different from the user's project-programmer number, the action of the TECO command is somewhat different from that of the standard TECO command explained above. In this case the named file is taken for input from the specified project-programmer area, but the output file is written in the user's own disk area with the same name as the input file. This operation is identical to that used for the EB command (see Section 3.2.5).

If `[proj,prog]` is not the user's project-programmer number, the command:

`._TECO filnam.ext [proj,prog]`

is equivalent to

`._R TECO`  
`*EBfilnam.ext [proj,prog] (Y) (Y) (Y)`

or to

`._R TECO`  
`*ERfilnam.ext [proj,prog] (EWfilnam.ext) (Y) (Y) (Y)`

and the input file is not renamed to `filnam.BAK`.

## NOTE

The R TECO command must be used for jobs involving editing a file on a device other than disk or DECTape, or for editing a file named nnnTEC.TMP, or a file with the filename extension .BAK. The R TECO command is also preferred with complex editing jobs, where user errors are likely, because of the greater control it gives over the input and output files. The R TECO command requires the use of file selection commands (see Section 3.2), whereas the MAKE and TECO commands do not.

## 3.1.4 Examples of the Use of Initialization Commands

|                                           |                                                                                                                                                                                                                                             |
|-------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>._MAKE EARNNG.F4 )</code><br>*<br>- | This command initializes TECO for creation of a FORTRAN file named EARNNG.F4.                                                                                                                                                               |
| <code>._TECO LIB40.MAC )</code>           | This command initializes TECO for editing the existing file LIB40.MAC. At the completion of editing, TECO automatically changes the name of the original version of LIB40.MAC to LIB40.BAK and gives the name LIB40.MAC to the new version. |
| <code>._TECO )</code>                     | This initializes TECO for editing the disk file last referenced in an edit-class command (MAKE, TECO, EDIT, or CREATE).                                                                                                                     |
| <code>._R TECO )</code><br>*<br>-         | This is the command to initialize TECO for general-purpose editing. FILE selection commands (see Section 3.2) should follow.                                                                                                                |

## 3.2 FILE SELECTION COMMANDS

File selection is the specification of the device from which input is to be taken and the device to which output is to go. In the case of magnetic tape, file selection also involves positioning the tape. In the case of directory-structured devices, disk and DECTape, a filename must be specified in addition to the device.

If the user wants only to create a file, or to edit an existing disk or DECTape file, file selection can be done by using either of the previously described initialization commands.

```
._MAKE dev:filnam.ext[proj,prog] )
      or
._TECO dev:filnam.ext[proj,prog] )
```

In all other cases, and in particular if the user initializes TECO with the R TECO command, one or more of the file selection commands described in this section must be used.

### 3.2.1 ER Command

The ER command is used to select a file for input. The general form is

\*ERdev:filnam.ext [proj,prog] Ⓢ

where

- a. dev: is the device name, which can be any name acceptable to the monitor. The device name must be followed by a colon. If dev: is omitted, the default value DSK: is assumed.
- b. [proj,prog] is ignored when used with a device other than disk. proj is the project number and prog is the programmer number of the disk area where the specified file resides or, in the case of output, is to be written. If [proj,prog] is omitted and the device is a disk, the user's project-programmer number is assumed.
- c. filnam.ext need be used only if the device is a directory device, i.e., disk or DECTape. filnam is the one-to-six character filename, and ext is the one-to-three character filename extension conforming to the rules stated in Section 2.1. If the device is a disk or DECTape, filnam must not be omitted; .ext must not be omitted unless the null extension is explicitly intended.
- d. The Ⓢ (altmode) functions as the argument terminator.

The ER command terminates input from any file that may have been previously opened for input, and then opens the specified file for input.

The user may open one file for input, read only part of that file, and then, with another ER command, release the first file and open a new file for input. It is not necessary to read to the end of a file before opening another. However, opening the second file does end input from the first. There is never more than one input file active. In Section 4.4, an example is given showing how to use multiple ER commands to merge parts of several files. Data cannot be input without first giving an ER, or equivalent, command.

### 3.2.2 EM Command

EM commands are used to position a magnetic tape for input or output. However, EM command apply only to the magnetic tape that is currently open for input (i.e., opened by the latest ERMTAn: Ⓢ command). To position a magnetic tape for output, it is necessary to first initialize the tape for input, then do the desired EM function, and then reopen the device for output.

The function of an EM command is determined by the value of a single numeric argument preceding the EM. The various EM commands are shown in Table 3-1.

### 3.2.3 EW Command

The EW command is used to select a file for output. The general form is

\*EWdev:filnam.ext [proj,prog] Ⓢ

The EW command opens the specified file for output. If any output file is already active, a new EW command closes that file before opening the new file. Only one output file can be active at any one time. If a previously active output file is closed by an EW command, that closed file contains all and only that data supplied to it by output commands preceding the new EW command.

If there is already an output file with the name specified, the EW command causes the old file to be overwritten and TECO outputs the warning message %SUPERSEDING EXISTING FILE.

Multiple EW commands may be used without changing the input file. In Section 4.3, an example is given showing how to use this technique in order to split a single input file into several parts.

The MAKE filnam.ext initialization command causes an automatic EWDSK:filnam.ext  $\text{\textcircled{S}}$  command to be executed. Output may not be done without first giving an EW, or equivalent, command.

Table 3-1  
EM Commands

| Command                                                                                                                               | Function                                                                                                                                                                                                                     |
|---------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EM or 1EM                                                                                                                             | Rewind the currently-selected input magnetic tape to load point.                                                                                                                                                             |
| 3EM                                                                                                                                   | Write an end-of-file record on the input tape.                                                                                                                                                                               |
| 6EM                                                                                                                                   | Skip ahead one record.                                                                                                                                                                                                       |
| 7EM                                                                                                                                   | Back up one record.                                                                                                                                                                                                          |
| 8EM                                                                                                                                   | Skip ahead to logical end-of-tape (defined by two successive end-of-file marks). The 8EM command leaves the tape positioned between the two end-of-file marks so that successive output correctly overwrites the second EOF. |
| 9EM                                                                                                                                   | Rewind and unload.                                                                                                                                                                                                           |
| 11EM                                                                                                                                  | Write 3 in. of blank tape.                                                                                                                                                                                                   |
| 14EM                                                                                                                                  | Advance tape one file. This leaves the tape positioned so that the next item read will be the first record of the next file (or the second end-of-file mark at the logical end-of-tape).                                     |
| 15EM                                                                                                                                  | Backspace tape one file. This leaves the tape positioned so that the next item read will be the end-of-file mark preceding the file backspaced over (unless the file is the first on the tape).                              |
| NOTE                                                                                                                                  |                                                                                                                                                                                                                              |
| The EM commands do not clear the internal input buffers. It is best to reinitialize with a new ER command before doing an EM command. |                                                                                                                                                                                                                              |

### 3.2.4 EZ Command

The EZ command is used only with disk, DECTape, or magnetic tape. Its function is equivalent to that of the EW command except that before opening the specified output file it zeros the output device directory if the device is a disk or DECTape, or it rewinds the tape if the device is a magnetic tape. For other devices, it is treated exactly like an EW. The form is

\*EZdev:filnam.ext [proj,prog] (\$)

### 3.2.5 EB Command

The EB command is used to open a file for editing in a manner similar to the initialization command `TECO dev:filnam.ext [proj,prog]`. It can be used only for files on a disk or DECTape. The general form of the command is

\*EBdev:filnam.ext [proj,prog] (\$)

The exact operation of the EB command is as follows:

First, the EB command executes an automatic `ERdev:filnam.ext` (\$) command, opening the specified file for input and releasing any previously opened input file. Then, it opens a temporary file to receive the output of the edited version of the input file. This temporary file is named `nnnTEC.TMP`, where `nnn` is the user's job number with leading zeros. This action is equivalent to executing the command `EWdev:nnnTEC.TMP` (\$). The output device is the same as the input device. Finally, the EB command sets an internal flag indicating that special action must be taken when the EB file is closed (by an EF, EX, or EG command - see Sections 3.9 and 3.10). It also prohibits any further EW, EZ, or EB commands until the file is closed.

When the EB file is closed, the following action takes place. First, if there already exists on the device a file with the name `filnam.BAK`, it is deleted. Then, the input file `filnam.ext` is renamed `filnam.BAK`. Finally, the output file, `nnnTEC.TMP`, is renamed `filnam.ext`.

The effect of using the EB command is analogous to editing a file in place, to itself, and converting the original version into a backup file. It updates the specified file and keeps the most recent previous version as a backup file.

If the project-programmer number specified in an `EBfilnam.ext [proj,prog]` (\$) command is different from the user's, then the input file is taken from the specified area, but the output file is written in the user's own area with the same name as the input file. In other words, if `[proj,prog]` is not the user's project-programmer number,

\*EBfilnam.ext [proj,prog] (\$)

is equivalent to

\*ERfilnam.ext [proj,prog] (\$) `EW`filnam.ext (\$)

The EB command cannot be used with a file having the filename extension `.BAK` nor with a file named `nnnTEC.TMP`. The `TECO dev:filnam.ext [proj,prog]` initialization command causes an automatic `EBdev:filnam.ext [proj,prog]` (\$) to be executed (followed by an automatic Y command).

### 3.2.6 Editing Line-Sequence Numbered Files

Some ASCII files, e.g., those created by BASIC, PIP with the /S and /O switches, and LINED, have a special type of line number at the beginning of each line. These "line-sequence numbers" conform to certain rules so that they may be ignored or treated specially by compilers and other programs. The standards for line-sequence numbers are given in the LINED Program Reference Manual.

TECO does not need line-sequence numbers for operation, but TECO can be used to edit files containing them. If such a file is edited with TECO the line-sequence numbers are, in the normal case, simply preserved as additional text at the beginning of each line. The line-sequence numbers may be deleted, edited, and inserted exactly like any other text. On output the line-sequence numbers are output according to the standard, except that the tab after the number is output only if it is already there. Leading zeros are added as necessary. If a line without a line-sequence number is encountered, a line-sequence number word of five spaces is placed at the beginning of the line.

The following switches are available for use with line-sequence-numbered files. These switches are merely added to the appropriate file selection command.

ERdev:filnam.ext[proj,prog]/SUPLSN (Ⓢ) (Ⓢ)

EBdev:filnam.ext[proj,prog]/SUPLSN (Ⓢ) (Ⓢ)

causes line-sequence numbers to be suppressed at input time. The numbers will not be read into the editing buffer. Also, the tabs following the line-sequence numbers, if they exist, will be suppressed.

EWdev:filnam.ext[proj,prog]/SUPLSN (Ⓢ) (Ⓢ)

causes the line-sequence numbers to be suppressed at output time. Tabs following the line-sequence numbers will also be suppressed if they exist.

EWdev:filnam.ext[proj,prog]/GENLSN (Ⓢ) (Ⓢ)

EBdev:filnam.ext[proj,prog]/GENLSN (Ⓢ) (Ⓢ)

causes line-sequence numbers to be generated for the output file if they did not already exist in the input file. Generated line-sequence numbers begin at 00010 and continue with increments of 10 for each line.

Note that these switches are needed only if a change is to be made in the format of the file being edited. If no switches are specified, a file is output in the same form as it was input.

### 3.2.7 Examples of the Use of File Selection Commands

\* ERDTA2:CREF.2 (Ⓢ) EWDSK:CREF.3 (Ⓢ) (Ⓢ)

\*  
-

This command string selects the DECtape file CREF .2 on DECtape drive 2 for input and opens a file called CREF.3 on the disk for output. If there is a file named CREF.3 already on the disk, it will be overwritten.

\* ERCDR: (\$) EWPTP: (\$) (\$)

\*

-

\*ERMTA1: (\$) EM14EM14EMEZDTA5:PROFIT.CBL (\$) (\$)

\*

-

\*ERPULSE.F4[11,14] (\$) (\$)

\*

-

\* EZMTA3: (\$) (\$)

\*

-

\*ERMTA1: (\$) 8EMEWMTA1: (\$) (\$)

\*

-

\* EB22.F4 (\$) (\$)

\*

-

\* n<14EM> (\$) (\$)

\*

-

\*EBCHESS.MAC[1,4] (\$) (\$)

\*

-

Select the card reader for input and the paper tape punch for output.

This command string selects the tape on magnetic tape drive 1 for input, then positions the tape at the beginning of the third file on that tape, and finally zeros the directory of the DECTape on drive 5 and opens an output file named PROFIT.CBL on it.

Select the file PULSE.F4 in project-programmer area [11,14] on the disk for input. If this file is read-protected against the current user, an error message results.

Rewind the magnetic tape on drive 3 and select it for output.

To position a magnetic tape for output (other than just a rewind), the user must first select the tape for input, then use EM commands to position the tape, and finally select the tape for output. In this example, the 8EM command positions the tape at the end of data that had previously been written on the tape. This enables new output to the tape without overwriting any of the previous data.

This command selects the disk file 22.F4 for editing. When the editing is completed, the file 22.F4 is the new version. The old version is changed to the backup file 22.BAK, and any previous backup file 22.BAK is deleted.

Advance magnetic tape n files.

This command opens the file CHESS.MAC on the [1, 4] disk area for input, and opens a file CHESS.MAC on the user's own disk area for output (assuming the user's project-programmer number is not [1,4]).

### 3.3 INPUT COMMANDS

Input commands are used to read data from the input file, which must previously have been opened, into the editing buffer. Input commands can be used only after an ER command (or the equivalent)

has been given. Input always begins at the beginning of the selected input file. Successive input commands then read successive segments of data from the input file.

The amount of data read on an input command depends on the buffer size, the particular input command used, and the data itself, as explained in the paragraphs below.

### 3.3.1 Y Command

The Y (yank) command first clears the editing buffer and then reads text into the buffer until one of the following conditions is met:

- a. The end of the input file is reached;
- b. A form feed character is read;
- c. the buffer is two-thirds full and a line feed is read (or filled to within 128 characters of capacity);
- d. the buffer is completely filled.

The usual effect of the Y command is to clear the editing buffer and then read the next page of the input file into it. Less than the entire next page is read in only if that page is too large to fit within two-thirds of the buffer's capacity. If the cleared buffer is not large enough to accommodate at least 3000 characters, TECO automatically expands its buffer by 1K, if possible, before beginning to input. The user is notified of the buffer expansion by a message of the form [nK CORE], where n is the new number of 1K segments of (low) core allocated to the user.

If the end of the input file has previously been read, the Y command only clears the buffer.

If a form feed is read (i.e., if input stops because of condition b), the form feed flag (ⓉE) is set to -1. The form feed itself is not packed in the buffer with the rest of the text. A succeeding input command begins input at the character following the form feed. If a form feed is not read, the form feed flag is set to 0, and the next input command begins input at the character following the last character previously read in. The form feed flag may be tested by the user (see Section 3.16), but ordinarily this is not necessary.

A single Y command is automatically executed by the TECO filnam.ext initialization command causing the first page of the input file to be read into the buffer before TECO prints the first asterisk.

The Y command sets the buffer pointer to the position preceding the first character in the buffer.

The Y command does not accept a numeric argument. If multiple Y commands are desired, n<Y> (where n is the number of pages to be ignored) can be typed.

### 3.3.2 A Command

The A (append) command reads in the next page of the input file without clearing the current contents of the editing buffer. The new input data is appended to that which is already in the buffer (at the end of that data). The position of the buffer pointer is not changed. If there was a form feed character in the input file separating the data already in the buffer and the data read in, it is removed. Thus, the A command can be used to combine several pages of a file.

If the editing buffer does not have sufficient space to accommodate 3000 more characters, TECO automatically expands its buffer by 1K, if possible, and then completes execution of the A command. The user is notified of the buffer expansion by a message of the form [nK CORE].

Input begun by an A command is terminated by any of the same four conditions that terminate a Y command. The A command processes form feeds and the form feed flags in the same manner as the Y command.

The A command does not accept a numeric argument. If multiple appends are desired, the user can type n<A> where n is the number of pages to be appended to the buffer. Note that nA is a different command (refer to Paragraph 3.16).

If the end of the input file was previously read, the A command has no effect.

### 3.3.3 Examples of the Use of Input Commands

\*ERREPORT.CBL (\$) Y (\$)

This command string opens the disk file REPORT.CBL for input and reads in the first page of that file.

\*YA (\$) (\$)

This deletes the page of text currently in the editing buffer, reads in the next two pages of the current input file, appending the second page to the first.

\*A (\$) (\$)  
[3K CORE]

This inputs the next page of the file, appending it to the data already in the buffer. The previous contents of the buffer are not altered and the pointer is not moved.

The buffer is expanded automatically, as required by the A command. In most cases, this message is of no concern to the user. It is important only if the system is nearly overloaded.

\*ERDTA6:DATA.DOC (\$) YYY (\$)

This command string reads in and discards the first two pages of the DECtape file DATA.DOC, and then reads in the third page of that file.

### 3.4 SPECIAL CHARACTERS AS BUFFER POSITION NUMERIC ARGUMENTS

In many cases, numeric arguments are used to specify buffer positions. Because such arguments tend to be large and not easily determined by counting, the buffer positions commonly used as arguments are represented by special characters. These special characters are shown in Table 3-2.

Table 3-2  
Special Buffer Position Arguments

| Character | Value                                                                                                                                                                                                                              |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| B         | Equivalent to 0. It represents the position at the beginning of the buffer, i.e., preceding the first character in the buffer.                                                                                                     |
| Z         | Equals the total number of characters in the buffer. Thus, Z represents the position at the end of the buffer, immediately after the last character in the buffer.                                                                 |
| .         | Equals the number of characters to the left of the current position of the buffer pointer, and hence represents the buffer pointer position itself.                                                                                |
| H         | Equivalent to the numeric argument pair B, Z. Thus, in those commands that take two numeric buffer position arguments, H represents the whole of the buffer. This letter is particularly useful with type-out and output commands. |

The characters B, Z and . can be used in arithmetic expressions.

### 3.5 BUFFER POINTER POSITIONING COMMANDS

This section describes the most elementary commands for moving the buffer pointer. In addition to these elementary commands, the search commands make up an entire set of powerful pointer-positioning commands. The search commands are described in Section 3.11.

#### 3.5.1 J Command

The  $nJ$  command moves the buffer pointer to the position immediately after the  $n$ th character in the buffer. The command  $0J$  moves the pointer to the beginning of the buffer, i.e., to the position immediately preceding the first character in the buffer. The command  $J$ , not preceded by an argument, is equivalent to  $0J$ .

#### 3.5.2 C Command

If  $n \geq 0$ ,  $nC$  moves the pointer forward over  $n$  characters in the buffer. If  $n < 0$ ,  $nC$  moves the pointer backward over  $n$  characters. The  $nC$  command is equivalent to  $.+nJ$ . The command  $C$  is equivalent to  $1C$ ;  $-C$  is equivalent to  $-1C$ .

#### 3.5.3 R Command

The  $R$  command is equivalent to  $-C$ . The  $nR$  command is equivalent to  $\neg nC$ . If  $n \geq 0$ ,  $nR$  moves the pointer backward over  $n$  characters in the buffer. If  $n < 0$ ,  $nR$  moves the pointer forward over  $n$

characters. The nR command is equivalent to .-nJ. The command R is equivalent to 1R; -R is equivalent to -1R.

### 3.5.4 L Command

The L command is used to move the buffer pointer over entire lines. The use of the L command with various arguments is shown in Table 3-3.

Table 3-3  
L Commands

| Command | Argument   | Function                                                                            |
|---------|------------|-------------------------------------------------------------------------------------|
| L       | 1 assumed  | Advances the pointer to the beginning of the line following the current line.       |
| nL      | n > 0      | Advances the pointer to the beginning of the nth line following the current line.   |
| 0L      | 0          | Moves the pointer back to the beginning of the current line.                        |
| -L      | -1 assumed | Moves the pointer back to the beginning of the line preceding the current line.     |
| nL      | n < 0      | Moves the pointer back to the beginning of the nth line preceding the current line. |

If the user attempts to move the buffer pointer backward beyond the position immediately prior to the first character in the buffer, or forward beyond the position immediately after the last character in the buffer with a C, R, or J command, an error message is printed, and the pointer is not moved from the position it had before the illegal command was given. With the L command no such error message results, but the pointer will be moved beyond the boundary of the buffer.

### 3.5.5 Examples of the Use of Buffer Pointer Positioning Commands

```
*J3L ($) ($)  
*  
-
```

The J command moves the pointer to the beginning of the first line in the buffer. The 3L command then moves it to the beginning of the fourth line.

```
*ZJ-2L ($) ($)  
*  
-
```

The ZJ command moves the pointer to the end of the last line in the buffer. Then the -2L command moves the pointer to the beginning of the next to last line in the buffer (assuming that the last line is terminated by a line feed).

```
*L4C ($) ($)  
*  
-
```

Advance the pointer to the position following the fourth character in the next line.

\*OL2R (Ⓢ) (Ⓢ)  
\*  
-

The OL command moves the pointer back to the beginning of the current line. Then the 2R command moves it back past the last two characters in the preceding line (the second of which must be a line terminator).

\*J-L (Ⓢ) (Ⓢ)

The J command moves the pointer to the beginning of the buffer, and the -L command then has no effect and therefore does not return an error message.

\*ZJC (Ⓢ) (Ⓢ)

The ZJ command moves the pointer to the end of the buffer, and the C command then causes the error message.

?POP

Attempt to move pointer off the page with the C command.

### 3.6 TEXT TYPE-OUT COMMANDS

#### 3.6.1 T Command

Any part of the text in the editing buffer can be typed out for examination. This is accomplished by using the T commands. The text typed out depends on the position of the buffer pointer and the argument(s) given. The T commands never move the buffer pointer.

When used with a single numeric argument, T is a line-oriented type-out command; when used with a pair of numeric arguments, T is a character-oriented type-out command. The various T commands are described in Table 3-4.

#### 3.6.2 (↑) Command

During the execution of any T command, the user can stop the terminal output by typing the special monitor control-character (↑). The (↑) command causes TECO to finish execution of the command string omitting all further type-outs. The effect of the (↑) command does not carry over to the next command string. (This command may only be typed as a control character. The combination ↑O (uparrow, O) does not have the same effect.) Occasionally the asterisk output by TECO when a command is finished is also suppressed by (↑). If this occurs, the user can type (↑U). TECO will respond with an asterisk if it is waiting for a command.

Table 3-4  
T Commands

| Command | Argument  | Function                                                                                                                                                                                                                                                                                   |
|---------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| T       | 1 assumed | Types out everything from the buffer pointer through the next line terminator. If the pointer is at the beginning of a line, T causes the entire line to be typed out. If the pointer is in the middle of a line, T causes that portion of the line following the pointer to be typed out. |

Table 3-4 (Cont)  
T Commands

| Command | Argument   | Function                                                                                                                                                                                                       |
|---------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| nT      | n > 0      | Types out everything from the buffer pointer through the nth line terminator following it. If the pointer is at the beginning of a line, this command types out the next n lines (including the current line). |
| OT      | 0          | Types out everything from the beginning of the current line up to the pointer. This command is especially useful for determining the position of the buffer pointer.                                           |
| -T      | -1 assumed | Types out everything in the line preceding the current line, plus everything in the current line up to the pointer.                                                                                            |
| nT      | n < 0      | Types out everything in the n lines preceding the current line, plus everything in the current line up to the pointer.                                                                                         |
| m,nT    | m < n      | Types the m+1st through the nth characters in the buffer.                                                                                                                                                      |
| .,.tnT  | n > 0      | Types the n characters immediately following the buffer pointer.                                                                                                                                               |
| .,.n.,T | n > 0      | Types the n characters immediately preceding the buffer pointer.                                                                                                                                               |
| HT      | H = B, Z   | Types out the entire contents of the buffer.                                                                                                                                                                   |

### 3.6.3 tL Command

If a form feed character,  $\textcircled{\text{tL}}$  or tL, is included in a command string as a command, it causes a form feed to be printed on the terminal when TECO reaches that point in execution of the command string. This feature is useful for obtaining a clean printout of the text in the buffer.

### 3.6.4 nET Command

In normal typeout mode, most control characters print in the up-arrow form and altmodes print as dollar signs. For the benefit of users with special terminal equipment, this feature can be suppressed. The command 1ET (any nonzero argument has the same effect as 1) changes the typeout commands so that every ASCII character is delivered to the typeout device literally, i.e., with its own octal mode. This is called literal type-out mode.

When TECO is in literal type-out mode, it can be restored to normal type-out mode, i.e., with substitutions for control characters and altmodes, by using the command 0ET.

The ET command (with no argument) returns the value (0 or 1) of the current setting of the type-out mode switch. See Section 3.16 for an explanation of this command.

### 3.6.5 Case Flagging On Type-out

TECO has three text type-out case-flagging modes: (1) lower case flagging, (2) upper case flagging, and (3) no case flagging. In lower case flagging mode, all characters in the range octal 140 to 177. are preceded by ' (apostrophe) when typed out. In upper case flagging mode characters in the range octal 100 to 137 are flagged with a preceding '. TECO is initially set for lower case flagging.

The case flagging mode may be set as follows:

|             |                                                                         |
|-------------|-------------------------------------------------------------------------|
| nEU (n > 0) | sets the typeout mode to flag upper case characters,                    |
| OEU         | sets the mode to lower case flagging (standard),                        |
| nEU (n < 0) | sets the mode to no flagging,                                           |
| EU          | (without argument) returns the value of the current case flagging mode. |

If TTY LC is on (i.e., the user's terminal handles lower case) or if the ET flag is on, no case flagging ever occurs regardless of the EU setting.

### 3.6.6 Examples of the User Text Typeout Commands

The following examples assume the buffer contains the text shown at the right, with the buffer pointer positioned between the M and the N

```
ABCDE) ↓
FGHIJ) ↓
KLM,NO) ↓
PQRST) ↓
UVWXY) ↓
Z) ↓
```

Examples:

```
*T ($) ($)
NO
*
-
*3T ($) ($)
NO
PQRST
UVWXY
*
-

*0T ($) ($)
KLM*
```

Note that no carriage return-line feed exists between the beginning of the line the pointer is on and the pointer itself, therefore, none are typed. The second asterisk indicates that TECO is ready for the next command.

\*OTT (\$) (\$)  
RLMNO

This pair of commands causes the entire current line to be typed out without moving the pointer.

\*  
-  
\*-2T (\$) (\$)

ABCDE  
FGHIJ  
KLM\*

\*.,.+6T (\$) (\$)

The six characters typed are NO) !PQ.

NO  
PQ\*  
\*.-2, .T (\$) (\$)  
LM\*

\*0LT (\$) (\$)

This pair of commands types out the entire current line and leaves the pointer at the beginning of this line.

KLMNO

\*  
-  
\*HT (\$) (\$)

The user requests type-out of the whole buffer, but stops it with a (10) immediately after the G is typed.

ABCDE

FG (10)

\*  
-  
\*↑LHT↑L (\$) (\$)

This command string causes the entire contents of the buffer to be typed out, with a form feed printed before and after the text is printed.

ABCDE  
FGHIJ  
KLMNO  
PQRST  
UVWXY  
Z

\*  
-  
\*0ETH11ETH (\$) (\$)  
X↑AY\$Z  
XYZ

If the buffer contains the text X (↑A) Y (\$) Z)!, this command string causes it to be typed out in both normal and literal modes, as shown. In the first line typed out, the control-A and altmode are typed in normal mode as up-arrow, A and dollar sign. In the second line, typed in literal mode, ↑A and \$ do not appear because they are delivered to the console device in their true values, which are nonprinting characters on most terminals.

\*T (\$) (\$)  
TECO M'A'N'U'A'L

The appearance of apostrophes in the typed text indicates that "anual" is lower case.

\*1EUT (\$) (\$)  
'T'E'C'O 'MANUAL

1EU changes TECO so that upper case characters are flagged.

\*-1EUT (\$) (\$)  
TECO MANUAL

-1EU stops case flagging.

\*  
-

### 3.7 DELETION COMMANDS

The K and D commands are used to delete characters from the editing buffer. The K command used with a single numeric argument is a line-oriented deletion command. The D command and the K command used with a pair of numeric arguments are character-oriented deletion commands.

#### 3.7.1 K Command

The various K commands are described in Table 3-5.

Table 3-5  
K Commands

| Command | Argument   | Function                                                                                                                                                                                                                                                                                                                            |
|---------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| K       | 1 assumed  | Deletes everything from the buffer pointer through the next line terminator. If the pointer is at the beginning of a line, the K command causes the entire line to be deleted. If the pointer is in the middle of a line, the K command deletes only the portion of the line following the pointer (including the line terminator). |
| nK      | $n > 0$    | Deletes everything from the buffer pointer through the nth line terminator following it.                                                                                                                                                                                                                                            |
| OK      | 0          | Deletes everything from the pointer back to the beginning of the current line.                                                                                                                                                                                                                                                      |
| -K      | -1 assumed | Deletes everything from the pointer back to the beginning of the line preceding the current line.                                                                                                                                                                                                                                   |
| nK      | $n < 0$    | Deletes everything from the pointer back to the beginning of the nth line preceding the current line.                                                                                                                                                                                                                               |
| m,nK    | $m < n$    | Deletes the m+1st through the nth characters in the buffer and positions the pointer at the point of deletion (that is, the pointer is set equal to m).                                                                                                                                                                             |

#### 3.7.2 D Command

Using the D command, characters can be deleted individually and in short strings. The nD command, where  $n \geq 0$ , deletes the n characters immediately following the buffer pointer. If the argument n is omitted,  $n = 1$  is assumed. The command nD, where  $n < 0$ , deletes the n characters immediately preceding the pointer; -D is equivalent to -1D.

At the conclusion of any K or D command, the buffer pointer is positioned between the characters that preceded and followed the deletion.

### 3.7.3 Examples of the Use of Deletion Commands

The following examples assume that the buffer contains the text shown at the right; the buffer pointer is positioned between the M and the N.

```

ABCDE ) ↓
FGHIJ ) ↓
KLM,NO ) ↓
PQRST ) ↓
UVWXY ) ↓
Z ) ↓

```

#### Examples:

```

_ *6D ($) ($)
_ *
_

```

Deletes NO ) ↓ PQ, changing the third and fourth lines to KLMRST ) ↓.

```

_ *-D ($) ($)
_ *
_

```

Deletes M.

```

_ *-5D ($) ($)
_ *
_

```

Deletes ) ↓ KLM, changing the second and third lines to FGHIJNO ) ↓.

```

_ *-2D2D ($) ($)
_ *
_

```

Deletes LMNO, changing the third line to K ) ↓.

```

_ *HK ($) ($)
_ *
_

```

Deletes everything in the buffer, but does not delete the form feed marking the end of the page (if there is one).

```

_ *0,.K ($) ($)
_ *
_

```

Deletes everything from A through M.

```

_ *,ZK ($) ($)
_ *
_

```

Deletes everything from N through Z ) ↓.

```

_ *K ($) ($)
_ *
_

```

Deletes NO ) ↓ changing the third and fourth lines to KLMPQRST ) ↓.

```

_ *0LK ($) ($)
_ *
_

```

Deletes the entire third line.

```

_ *L3K ($) ($)
_ *
_

```

Deletes the last three lines (everything from P through Z ) ↓).

```

_ *KD ($) ($)
_ *
_

```

Deletes NO ) ↓ P, changing the third and fourth lines to KLMQRST ) ↓.

```

_ *OK ($) ($)
_ *
_

```

Deletes KLM.

```

_ *-K ($) ($)
_ *
_

```

Deletes FGHIJ ) ↓ KLM.

### 3.8 INSERTION COMMANDS

The insertion commands are used to insert characters into the editing buffer from the user's terminal.

### 3.8.1 I Command

The basic text insertion command is the I command used with the desired text as its argument. The text argument is terminated by an altmode. The general form is

`*|text` Ⓢ

This command inserts the ASCII text string, "text", into the editing buffer just ahead of the buffer pointer. After the insertion, the buffer pointer is positioned immediately after the last inserted character. The altmode terminating the text argument is not inserted. The text to be inserted may contain any character except the special characters (see Table 2-1), but control characters must be treated specially (see Section 3.8.8).

### 3.8.2 Tab Command

The tab command is equivalent to the I command, except that the tab command causes the tab itself as well as all the following text up to the altmode to be inserted. In other words, if the first character of a text string to be inserted by an I command is a tab, the I may be omitted. The general form of the tab command is

`*-|text` Ⓢ

### 3.8.3 @I Command

The @I command is slightly more powerful than the I command. This command enables the user to insert single (but not double) altmode characters in addition to the characters that can be inserted with the I command. (To insert a double altmode, the second altmode must be preceded by a Ⓢ.) The @I command is useful for inserting TECO command strings into the editing buffer. The general form is

`*@I/text/`

In this form, "text" is the text string to be inserted. The text argument must be immediately delimited, both before and after by any single character which is not itself a part of the text to be inserted. In this example, the delimiting character is the slash character. Altmode is not required to terminate the text string; the second occurrence of the delimiting character terminates the text string. The text is inserted immediately preceding the buffer pointer, as it is with the I command. The delimiting character is not inserted.

### 3.8.4 nl Ⓢ Command

Any ASCII character can be inserted into the buffer using the nl Ⓢ command. This includes all characters that the I and @I commands cannot insert. However, the nl command inserts only one character at a time. The command nl Ⓢ inserts the character with the ASCII value n (decimal) into the buffer immediately preceding the pointer.

Unless the EO value has been set to 1, the nl command must be followed by an altmode (refer to Paragraph 3.17 for a description of the EO value).

### 3.8.5 n\ Command

The n\ command is used to insert the ASCII representation of a decimal number n into the buffer. For example, 349\ inserts the ASCII characters 3, 4 and 9 into the buffer immediately preceding the pointer. Note that n does not have to be a number typed in by the user. It can be a value returned by some other TECO command. Note that the n\ command always inserts the decimal representation of n.

### 3.8.6 Examples of the Use of Insertion Commands

The following examples assume that the buffer contains ABCD\EF) ↓ with the pointer positioned between D and E.

|                               |                                                                                                                                    |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| *IXYZ (\$ (\$)                | Produces ABCDXYZ\EF) ↓                                                                                                             |
| *I)                           | Produces ABCD) ↓                                                                                                                   |
| (\$ (\$)                      | ^EF) ↓                                                                                                                             |
| *                             |                                                                                                                                    |
| *I ↓                          | Produces ABCD ↓                                                                                                                    |
| (\$ (\$)                      | ^EF) ↓                                                                                                                             |
| *                             |                                                                                                                                    |
| *3RI _(\$ 4CI _(\$ (\$)       | Produces A _BCDE _F) ↓                                                                                                             |
| *                             |                                                                                                                                    |
| *→XYZ (\$ (\$)                | Produces ABCD →XYZ\EF) ↓                                                                                                           |
| *@#IA (\$ SA (\$ PW# (\$ (\$) | Produces ABCDIA (\$ SA (\$ PW\EF) ↓                                                                                                |
| *↑O33I (\$ (\$)               | Produces ABCD(\$EF) ↓                                                                                                              |
| *10I (\$ 10I (\$ (\$)         | Produces ABCD ↓                                                                                                                    |
| *                             |                                                                                                                                    |
| *Z \ (\$ (\$)                 | Produces ABCD8\EF) ↓ because Z has the value 8.                                                                                    |
| *Z \Z \Z \ (\$ (\$)           | Produces ABCD8910\EF) ↓ because Z successively returns the values 8, 9, and 10.                                                    |
| *I (FORM)                     | This command is used to separate the page in the editing buffer into two pages. Both pages, however, remain in the editing buffer. |
| (\$ (\$)                      |                                                                                                                                    |
| *12I (\$ (\$)                 | This is equivalent to the command in preceding example. It is convenient because it avoids the form feed echo.                     |
| *                             |                                                                                                                                    |
| -                             |                                                                                                                                    |

```

*JILINE ONE)
LINE TWO)
LINE THREE)
($) ($)
*
*KI)
($) ($)
*

```

This example shows insertion of several lines of text at the beginning of the buffer. Note that line feeds are inserted automatically as the user types the carriage returns.

This command string is used to delete the tail of a line without removing the carriage return-line feed at the end of the line. If the buffer contains

```
AB^CD) ↓
```

```
EFGH) ↓
```

this command produces

```
AB) ↓
```

```
^EFGH) ↓
```

This is used to insert a carriage return without a line feed following it. The single rubout deletes the line feed but not the carriage return. (See Section 5.1 for an explanation of rubout.)

```

*|)
RO)
($) ($)
*

```

```

*|@I%TEXT ($) x (RO) ($) % ($) ($)
*

```

This is a convenient method for inserting multiple altmodes when using the @I command.

The sequence x (RO), where 'x' is any character except altmode, is typed between the successive altmodes.

```

*|O777 \ ($) ($)

```

This is used to insert the ASCII characters 511 at the current pointer position.

### 3.8.7 Case Control with Insert Commands

With the I, @I, and tab insert commands TECO ordinarily inserts text in the same case in which it appears in the command string. The user may, however, alter the case of text being inserted by use of the special case control commands described in this section.

3.8.7.1 Alphabetic Case Control - The features described in this section provide the method by which alphabetic characters in the upper case range can be converted to the equivalent characters in the lower case range, and vice-versa. Alphabetic case conversion is done by use of two control-character commands,

(tV) is used for translation to lower case,

(tW) is used for translation to upper case.

These two commands may be used within insert text arguments to cause case conversion on a temporary basis for that text argument, or as independent commands to cause case conversion in all insert and search text arguments.

Note that (tV) and (tW) affect only alphabetic characters. They have no effect on non-alphabetic characters.

- (1) (tV) (tV) and (tW) (tW) used within text arguments.

When used inside an insert text argument, two successive (tV) or (tW) commands cause translation, to the specified case, of all following alphabetic characters in that text argument.

Example:

\*IF (tV) (tV) OR USERS OF (tW) (tW) TECO. (\$)\$

The above command inserts "For users of TECO." with the initial "F" and "TECO" capitalized, and all the other letters in lower case.

- (2) Single (tV) and (tW) used within text arguments.

When used inside an insert text argument, a single (tV) or (tW) command causes translation of the next single character (if it is alphabetic) to the specified case. The single (tV) or (tW) in a text argument takes precedence over the case conversion mode defined by double (tV) or (tW) commands.

Example:

\*I (tV) (tV) USER (tW) PROGRAM (\$)\$

The above command causes the string "user Program" with the "P" in upper case, and all the other letters in lower case to be inserted.

- (3) Independent (tV) and (tW) Commands.

As explained above, when (tV) and (tW) commands are used inside a text argument, they affect only that particular text string. When used as independent commands, however, (tV) and (tW) set TECO to a prevailing case conversion mode that affects all insert and search text arguments (except as specified by (tV) and (tW) commands within the text arguments). The independent command (tV) or tV (or n (tV), where n does not equal 0) sets the prevailing case conversion mode so that all upper case alphabetic characters in insert and search text arguments are translated to lower case, except where (tW) commands within individual text arguments override the independent (tV).

Example:

\*tV\$\$  
 \*I (tW) FOR USERS OF (tW) (tW) TECO. (\$)\$  
 \*IEXAMPLE (\$)\$

The above commands cause "For users of TECO." and "example" to be inserted with all letters lower case except the "F" and "TECO". Likewise, the independent command (tW) or tW (or n (tW), where n does not equal 0) sets the prevailing case conversion mode so that all lower case alphabetic characters

in insert and search text arguments are translated to upper case, except where  $\textcircled{tV}$  commands within individual text arguments override the independent  $\textcircled{tW}$ .

The independent  $\textcircled{tW}$  command has the use explained above, obviously, only when the user TTY has lower case capability and TTY LC is on. Otherwise the  $\textcircled{tW}$  command serves merely to turn off the  $\textcircled{tV}$  command.

(4)  $0 \textcircled{tV}$  and  $0 \textcircled{tW}$

The independent  $0 \textcircled{tV}$  and  $0 \textcircled{tW}$  commands both have the same effect, namely, to restore TECO to the default condition where neither case of alphabetic characters are translated to the opposite case, except by  $\textcircled{tV}$  and  $\textcircled{tW}$  commands within text arguments.

TECO is initially set for no prevailing case conversion.

Note that the prevailing case conversion mode can have one, and only one, setting at any one time. The possible settings are:

|                                              |                                  |
|----------------------------------------------|----------------------------------|
| $\textcircled{tV}$                           | convert upper case to lower case |
| $\textcircled{tW}$                           | convert lower case to upper case |
| $0 \textcircled{tV}$ or $0 \textcircled{tW}$ | no prevailing conversion         |

When any of these prevailing modes is put into effect, it cancels any of the others that were in effect.

The order of precedence of the case conversion commands is as follows:

|          |                                                              |
|----------|--------------------------------------------------------------|
| Highest: | single $\textcircled{tV}$ and $\textcircled{tW}$ inside text |
| Next:    | double $\textcircled{tV}$ and $\textcircled{tW}$ inside text |
| Lowest:  | independent $\textcircled{tV}$ and $\textcircled{tW}$        |

NOTE

If the EO value has been set to 1,  $\textcircled{tW}$  and  $\textcircled{tV}$  have no special effect when used inside text arguments (refer to Paragraph 3.17 for a description of the EO value).

3.8.7.2 Special "Lower Case" Characters - When used inside an insert text argument, the control command  $\textcircled{t\uparrow}$  causes the immediately following character (if it is one of the special characters @, [, \, ], ↑, or ←) to be converted to the equivalent character in the lower case ASCII range (i.e., octal 140 or octal 173-177). That is,

|                                     |                              |
|-------------------------------------|------------------------------|
| $\textcircled{t\uparrow}$ @ becomes | ASCII 140                    |
| $\textcircled{t\uparrow}$ [ becomes | ASCII 173                    |
| $\textcircled{t\uparrow}$ \ becomes | ASCII 174                    |
| $\textcircled{t\uparrow}$ ] becomes | ASCII 175                    |
| $\textcircled{t\uparrow}$ ↑ becomes | ASCII 176                    |
| $\textcircled{t\uparrow}$ ← becomes | $\textcircled{RO}$ ASCII 177 |

$\textcircled{t\uparrow}$  has no special effect within text arguments if the EO value has been set to 1.

Examples:

|                                   |                           |
|-----------------------------------|---------------------------|
| *iVI (tW) EXAMPLES FOR THE        | Inserts "Examples for the |
| (tW) (tW) TECO M (tW) (tW) ANUAL. | TECO Manual.              |
| (t\$) (t\$)                       | EXAMPLE 1.                |
| *0iVIEXAMPLE 1.                   | nI Command."              |
| (tV) NI C (tV) (tV) OMMAND.       |                           |
| (t\$) (t\$)                       |                           |
| *                                 |                           |
| -                                 |                           |
| *i (t↑) [(t\$) (t\$)]             | Inserts a right brace (}) |

3:8.8 Inserting Control Characters

As of version 22 of TECO all of the control characters (tA) - (tG) , (tN) - (tZ) , and (t\ ) , (tj) , (t↑) , and (t←) have been reserved as inside-text-commands (some as yet undefined). In order to insert these characters, the user must employ either the (tR) or (tT) command.

(tR) when used inside an insert text argument causes the next single character to be interpreted as text rather than as a command, and accordingly to be inserted in the buffer. This applies to all control characters including (tR) itself. It also applies to Altmode. (It does not, however, apply to (tC) , (tO) , (tU) , or RUBOUT.)

(tT) when used inside an insert text argument causes all succeeding instances of the above mentioned control characters except (tR) and (tT) itself to be interpreted as text rather than as commands. (tT) does not affect altmodes. A second instance of (tT) in the same text argument nullifies the effect of the first.

If the EO value has been set to 1, (tR) and (tT) have no special effect when used inside text arguments, and all control characters can be inserted with no special treatment (refer to Paragraph 3.17 for a description of the EO value).

NOTE

The clever way to create a TECO macro is simply to type the macro as a long command string just as if it were to be executed immediately, but instead of typing (t\$) (t\$) at the end, type (tG) (tG). Then type \*i to place the command string in Q-register i. (This stores the macro, ready for execution, in Q-register i. (Refer to Paragraph 3.14.3 for the description of the \*i command.)

Examples:

|                                         |                                     |
|-----------------------------------------|-------------------------------------|
| *i (tR) (tA) TEXT (tR) (tA) (t\$) (t\$) | Inserts the text " (tA) TEXT (tA) " |
| *                                       |                                     |
| -                                       |                                     |

|                                                                                                                                                              |                                                                                                                  |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|
| <pre> *INSTRING (1R) (\$) (\$) (\$) * _ *  (1T) (1A) SEARCH (1A) NSTRING (1R) (\$) I (1V) (1V) TEXT (1R) (\$) (1T) !E (1V) (1V) XAMPLE! (\$) (\$) * _ </pre> | <pre> Inserts "NSTRING (\$) ".  Inserts " (1A) SEARCH (1A) NSTRING (\$) I (1V) (1V) TEXT (\$) !Example!". </pre> |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|

### 3.9 OUTPUT COMMANDS

Output commands are used to transfer data from the editing buffer to the output file.

#### 3.9.1 PW Command

The PW command is the basic output command. It does nothing but output. Depending on the argument used with it, the PW command outputs all or any part of the data in the editing buffer. It does not, however, delete any data from the buffer, and it never moves the buffer pointer.

The PW command outputs the entire contents of the buffer and always appends a form feed to it.

The nPW command ( $n > 0$ ) outputs  $n$  copies of the text in the buffer, appending a form feed to each copy.

#### 3.9.2 P Command

The P command is a combination command; when used with a single numeric argument (or no argument), the P command does both output and input. The various functions of the P command are described in Table 3-6.

Note that the P command (with a single argument) always clears the editing buffer before it inputs the next page, and it leaves the pointer at the beginning of the new page. If a P command is executed after the end of the input file has already been reached or when there is no input file, the buffer is simply cleared. No data is read in.

Unlike the PW command, the P command does not always cause a form feed to be output at the end of the data output from the editing buffer. The P command outputs a form feed at the end of the data only if a form feed was encountered to terminate the last input command.

Table 3-6  
P Commands

| Command                                                                                                                               | Argument  | Function                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------------------------------------------------------------------------------------------------------------------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| P                                                                                                                                     | 1 assumed | Similar to PWY. Outputs the entire contents of the buffer, then clears the buffer and reads in the next page of input. The buffer pointer is left at the beginning of the page that is read in. If there is no input file, or no more data in the input file, the buffer is left cleared. A form feed character is appended to the end of the data that is output only if the last input command was terminated by a form feed. |
| nP                                                                                                                                    | n > 0     | Executes the P command n times. This command can be used to skip over several pages of text when no editing is required. The nP command causes the n pages of the input file, starting with the page currently in the editing buffer, to be output, and then the nth page after the current page to be yanked in.                                                                                                               |
| m,nP                                                                                                                                  | m < n     | When used with a pair of numeric arguments, the P command does output only; it does not clear any data from the buffer, it does not input any more data, and it does not move the buffer pointer. Also, the m,nP command never causes a form feed to be appended to output <sup>1</sup> . The only action of m,nP is to output the m+1st through the nth characters in the buffer. (m,nP and m,nPW are equivalent.)             |
| HP                                                                                                                                    | H = B, Z  | Outputs the entire contents of the buffer without appending a form feed to it; the buffer is not cleared, and no new data is read in. (HP and HPW are equivalent.)                                                                                                                                                                                                                                                              |
| <sup>1</sup> However, if a form feed character has been inserted in the buffer between the mth and nth characters, it will be output. |           |                                                                                                                                                                                                                                                                                                                                                                                                                                 |

The PW command does not clear the buffer and does not move the buffer pointer. The same is true of a P command used with two arguments.

Note also that when a PW command is used, a form feed character is always automatically sent to the output file immediately following the data from the buffer. (Recall that when the page was read into the buffer, the form feed character that terminated it, if any, was discarded and not read into the buffer.) The form feed character is appended to the outgoing data regardless of whether or not a form feed character was encountered when the data was read in, i.e., regardless of the setting of the form feed flag. This is not true of the P command.

NOTE

If the EO value has been set to 1, the P command behaves like the PW command with regard to form feeds.

When a P or PW command is used with a double numeric argument (including an H argument), a form feed character is never appended to the output data. This is true regardless of whether or not a form feed character was encountered when the data was read in.

#### NOTE

The discussion in this section does not apply to the form feed characters that the user has inserted into the editing buffer using 12I (\$) or I (FORM) (\$) commands. Form feed characters in the buffer are output exactly as other characters in the buffer.

If the editing buffer is empty when a P or PW command is executed, no output of any kind takes place. No form feed character is output. If the user wants to create a blank page, an example of the procedure is shown below.

As shown in the discussion above, the nP command can be used to skip over several pages to get to the next page where editing is required. The nP command can also be used with a very large argument, e.g., 10000, in order to skip to the end of the input file without doing any more editing. The N and EX commands are other commands which can be used for this purpose.

#### 3.9.3 EF Command

The EF command is the output file closing command. The EF command, or an equivalent command, must be used to close the output file after all output to it is complete. The EF command is normally used after the P command which outputs the last page of a file. The special exit commands EX and EG (see Section 3.10) automatically cause an EF to be executed. Also, a new EW command causes an EF to be executed on the previous output file, if any, before opening the new output file. Note that if an EF command is executed in the middle of the file, all succeeding pages of that file are lost.

#### 3.9.4 Examples of the Use of Output Commands

\*PT (\$) (\$) FIRST LINE OF NEXT PAGE

Output the current page, clear the buffer, read in the next page, then type out the first line of the new page.

\*PEF (\$) (\$) \*

Output the current page to the output file, and then close the output file. This command string is used to close a file (after writing the last page) when it is not desirable to exit from TECO.

\*PWEF (\$) (\$) \*

Equivalent to the preceding example, except that the buffer is not altered.

\*.,ZP0,.P (\$)(  
\*  
-

This command string outputs the entire contents of the buffer, but it rearranges the data as it is output. The part of the page that follows the buffer pointer is output first by the .,ZP command. Then that part of the data which precedes the pointer is output by the 0,.P command. No form feed character is appended to either section of the output.

| \*.,ZP12I (\$ 0,.P (\$)(  
\*  
-

This performs the same function as the preceding command string except that it does append a form feed character to that part of the page that is output last.

| \*\_HK12I (\$ HP (\$)(  
\*  
-

This command string produces a single blank page.

| \*\_HK12I (\$ PW (\$)(  
\*  
-  
\*\_8P (\$)(  
\*  
-

This produces two successive blank pages.

If page 6 of a file is in the editing buffer, this command causes pages 6 through 13 of the file to be output one after the other, and then reads in page 14.

\*300PW(\$)(  
\*  
-

This outputs 300 copies of the current page.

\*PWJKIJ.DOE (\$ PW (\$)(

This outputs the current buffer, the modifies the first line and outputs the buffer again.

MAKE FILE  
\*1page of text (\$)(  
\*PI2nd page of text (\$)(  
\*PIlast page of text (\$ EX (\$)(

This is the usual method for creating a text file.

### 3.10 EXIT COMMANDS

Exit commands are used to terminate a TECO job and return to the monitor. There are four exit commands: EX, EG, (IZ), and (IC).

#### 3.10.1 EX Command

The EX command is used to bring an editing job to a satisfactory conclusion with a minimum of effort. Its use is shown in the example below.

The user is editing a 30-page file and that the last actual change to the file is made on page 10. At this point the user gives the command:

\*EX (\$)(  
EXIT  
IC  
-

In this case, the action performed by TECO is equivalent to the command string 21PEF, with an automatic exit to the monitor at the end. Thus, the action of TECO is (1) to rapidly move all the rest of the input file, including the page currently in the buffer, on to the output file; (2) to close the output file; and (3) to return control to the monitor.

The EX command is the easiest method of finishing an editing job, with the latter part of the input file being properly output and the output file closed.

The EX command performs both input and output functions.

The EX command causes a form feed character to be output after the output of the buffer, only if a form feed was encountered when that buffer of text was read in. In this way, the EX command maintains existing page sizes.

### 3.10.2 EG Command

The EG command first performs exactly the same functions as the EX command, and then causes the last compile-class command (COMPILE, EXECUTE, LOAD, or DEBUG) attempted before TECO was called, to be re-executed (with the same arguments). Generally, the EG command is used only to exit from an editing job that was initialized by an EB command or a TECO filnam.ext command.

As an example, suppose the user gives the command

```
._COMPILE PLOT.F4)
```

to request compilation of a FORTRAN source program, but the compiler encounters errors in the code. The user then calls TECO to correct these errors with the command:

```
._TECO PLOT.F4)
*
-
```

When all the errors are edited, the user exits from TECO with the command

```
*EG ($) ($)
```

This command causes (1) the rest of the file PLOT.F4 to be output and closed, and (2) the command COMPILE PLOT.F4 to be re-executed automatically.

### 3.10.3 (tZ) and (tC) Commands

The (tZ) and (tC) commands do not perform any input or output. They are used strictly for exiting to the monitor.

The command (tZ) (or tZ) is the simple exit command that can be entered into command strings. It allows any I/O commands that have already been given to be completed, then closes the output file, and then returns the user to the monitor.

Example:

```

*PWEF (tZ) ($) ($)
EXIT
tC
:

```

The (tZ) is executed as a regular command in the command string when its turn comes.

NOTE

If the EO value has been set to 1 (refer to Paragraph 3.17.3), a single (tG) is equivalent to (tZ).

The (tC) command is a monitor command that is used to immediately exit to the monitor. The (tC) command can be typed at any time, while typing a command string or while a command string is being executed, and it will override everything else. It cannot be entered in the up-arrow, C form. If there are any input/output functions in progress when (tC) is typed, a single (tC) will allow them to be completed before exiting to the monitor. Double (tC) ( (tC) (tC) ) interrupts everything, even I/O in progress, and exits to the monitor immediately. The (tC) command does not cause the output file to be closed.

- Both (tZ) and (tC) are abortive exit commands. However, when they are used, it is possible to return to the TECO job provided no other program has been called into core over the TECO job. Simple monitor commands such as ASSIGN, or PJOB, can be executed without damaging the TECO job.
- After an exit to monitor level, even if the exit was caused not by a user (tC), or (tZ), but instead by some problem detected by the monitor itself, the user can return to his TECO job by using either the CONTINUE or the REENTER command.

The command CONT causes TECO to begin operations exactly where it left off. Even I/O can be interrupted and then continued.

Example:

```

*ERPTR: ($) EWLPT: ($) Y3P ($) ($)
DEVICE LPT OK?
.CONT )
*
-

```

Here the monitor causes an exit to monitor level because of a device problem. After the user corrects the problem, he continues the job and the current command string executes to completion.

REENTER causes the TECO job to be reentered with the contents of the editing buffer (when the exit occurred) intact. After reentry by a REENTER, TECO reinitializes itself for a new command string. Any previous commands still unexecuted at the time of the exit are lost. If a command string was being executed when the exit occurred, the part of the string that was not executed before the exit will

not be executed after the REENTER command. The user must determine how much of the command string was executed. If I/O is interrupted, some portion of the input or output files is frequently either lost or duplicated.

Examples:

```
*ICOMME (tC)
.DEASSIGN LPT )
.DAYTIME )
T4-APR-70 10:34
.REE )
*ICOMMENTS ($) ($)
*
* <SFOO ($) OL > ($) ($)
(tC) (tC)
.REE )
*
*50P ($) ($)
(tC)
(tC)
.REE )
*
```

Before finishing a command string the user exits to perform a monitor command.

He then reenters TECO. The command string must be retyped, but the buffer is still intact.

This is an infinite loop (if FOO is in the buffer). (tC) (tC) stops execution and returns the user to the monitor. REE restarts TECO with the editing buffer intact and the command buffer empty.

This is an example of what should not be done. Interrupting execution of an I/O command does not permit reentry. In this case, some of the output file will almost certainly be duplicated.

The contents of any Q-registers (refer to Paragraph 2.8) remain intact after a (tC) , CONT or (tC) , REENTER command sequence.

### 3.11 SEARCH COMMANDS

In many cases the simplest way to reposition the buffer pointer is by using a character string search. A search command causes TECO to scan through the text until a specified string of characters is found, and then to position the pointer at the end of this string.

The string of characters to be searched for is supplied as a text argument with the search command.

The search string can be from 1 to 36 character positions in length or up to 80 characters including all control commands.

If an exact match for the search string is found in the text, the buffer pointer is positioned immediately after the last character in this match. If the string is not found, TECO positions the pointer at the beginning of the buffer and notifies the user of the failure. The failure notice may take one of two forms, depending on the type of search command used. For further explanation see the paragraph below.

All searches begin at the current position of the buffer pointer.

If no text argument is provided with a search command, e.g., S (\$) or @N//, the search is executed using the last previous search command argument.

### 3.11.1 S Command

The S Command is used to search for a character string within the current editing buffer. If the string is not found between the current buffer pointer position and the end of the buffer, the search fails. After an unsuccessful S search, the buffer pointer is reset to the beginning of the buffer, and, unless the : modifier (explained below) was used or the search is within an iteration (see Section 3.12), an error message is printed.

The general form of the S command is

Sstring (\$)

For the standard S command, the search string is provided as a normal alphanumeric argument following the S and terminated by an altmode. "string" can contain any character except the special characters listed in Table 2-1.

The S command may be used with a single numeric argument. The command nS causes a search for the nth occurrence of the specified search string. When n is omitted, n=1 is assumed. n must be greater than 0.

### 3.11.2 FS Command

The FS command is used to search for a character string within the current editing buffer (function of the S command) and replace it with another string. If the string to be replaced is not found after the current buffer pointer position and before the end of the buffer, the search fails and no replacement is made.

The general form of the FS command is

FSstring1 (\$) string2 (\$)

where string 1 is the string to be deleted and string2 is the string to be inserted in its place. If string 2 is omitted, string 1 is deleted without any string replacing it. However, even when string2 is omitted, its terminating altmode must be present as shown in the form:

FSstring1 (\$)

### 3.11.3 N Command

The N command combines the S command with input/output functions. The N command is used to search for a character string in a page of the input file which may not yet have been read into the buffer. The N command has the same form as the S command.

The N command functions exactly like the S command except that an N search does not terminate at the end of the page currently in the buffer. If no match for the search string is found between the current buffer pointer position and the end of the buffer, the current page is output, the buffer is cleared, the next page is read in, and the search starts over at the beginning of the new page. This process continues until a match is found or the input file is exhausted.

If an N search fails, the entire input file has been passed through the buffer and delivered to the output file, and the buffer cleared. The output file is not closed. Unless the : modifier was used or the search is within an iteration, an error message is typed to notify the user that the search has failed.

An N search will not detect a match when the matching characters are split across two buffer loads.

The output function of the N command is exactly like the P command and the EX command. If a form feed character was encountered when a given page was read in, a form feed character is appended to that page when it is output; otherwise, no form feed character is output.

The N command can be used with a single numeric argument. The command nN causes a search for the nth occurrence of the specified search string. When n is omitted, n=1 is assumed. n must be greater than 0.

#### 3.11.4 FN Command

The FN command is used to search for a character string in a page of the input file which may not yet have been read into the buffer (function of the N command) and to replace it with another string. The FN command operates like the N command when searching for the string. If the search fails, no replacement occurs.

The general form of the FN command is

`*FNstring1 ($) string2 ($)`

where string1 is the string to be deleted and string2 is the string to be inserted in its place. If string2 is omitted, string1 is deleted without any string replacing it. However, even when string2 is omitted, its terminating altmode must be present as shown in the form

`*FNstring1 ($) ($)`

#### 3.11.5 Backarrow Command

The backarrow command is identical to the N command except that a backarrow search generates no output. Generally, where the N command executes a P, the backarrow command executes a Y. The backarrow search is used for examination functions and for discarding parts of a file. The general form of the backarrow command is

`*←string ($) ($)`

The backarrow command can also be used with a single numeric argument. The command `n←` causes a search for the nth occurrence of the specified search string. When n is omitted, n=1 is assumed. n must be greater than 0.

### 3.11.6 Search Command Modifiers

3.11.6.1 @ Modifier - There are two search command modifiers. The @ modifier is used to alter the method which TECO reads the search command's text argument from the command string. The general form of a @ search command is the same for S, FS, N, FN, and backarrow. It is

`*@nS/string/`

The @ modifier is placed before the S, FS, N, FN, or backarrow, and before the numeric argument, if any. When the @ modifier is used, the search string argument is delimited, not by the search command and an altmode, but by the first character typed after the search command and the next recurrence of this character. In the example above, the delimiting character is a slash. The delimiting character may be any character except a character that appears in the search string itself. With the @ modifier, single (but not double) altmodes can be used in the search string. The @ modifier can be used in an FS or FN command to separate the strings with a delimiting character other than altmode. This is useful in cases where a double altmode cannot terminate the command. A double altmode terminates an FS or FN command when the replacement string is omitted to allow deletion of the string for which the search is made. Use of the @ search commands is similar to the use of the @I insert command (refer to Paragraph 3.8.3).

3.11.6.2 Colon Modifier - The colon modifier is used to alter the execution of a search command in the event the search fails. Without the colon modifier, a search that fails causes an error message to be printed; if the colon modifier is used, no error message is printed. Instead, every colon search command executed returns a numeric value that can be printed out, stored in a Q-register, or tested by a conditional branch. A colon search command returns the value -1 if the search is successful, and the value 0 if the search fails.

The general form of a colon search command is the same for S, FS, N, FN, and backarrow searches:

`*:nString ($)`

The colon precedes the search command letter and its numeric argument, if any. Both the colon and @ modifiers may be used on a search command, in either order.

The concept of a command returning a value is explained in Section 2.7.3. Just as the Z command takes on a value that may be used as a numeric argument, so also the command `:Sstring ($)` takes on a value of 0 or -1 after it is executed. If this is the last command in a command string, or if the command following it does not take a numeric argument, the value returned by the colon search is discarded. Hence, a colon search should be followed by a command that takes a numeric argument.

The colon search commands reposition the buffer pointer in the same manner as other search commands, regardless of whether or not the returned value is used.

The colon searches are used primarily in programmed editing and are usually followed by a conditional command. Examples of the uses of colon searches are given in Sections 3.13 and 3.14.

### 3.11.7 Automatic Timeout After Searches

The ES command allows the user to specify automatic timeout of the line where a successful search has terminated. The search cannot be in an iteration, nor can the search command be preceded by a colon. When the FS or FN command is used, the timeout occurs after the insertion has taken place. The user can also specify in the ES command that either a line feed or a character be inserted into the timeout to indicate the position of the pointer. Unless the ES value is set, the default is that no automatic timeout after searches will be performed.

The user can set the ES value in the following manner:

|            |                                                                                                                                                                                                                                                                                       |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OES        | Restore TECO to the default of no automatic timeout.                                                                                                                                                                                                                                  |
| -IES       | Set the ES value to cause automatic timeout of a line on which a successful search has terminated.                                                                                                                                                                                    |
| nES(n > 0) | Set the ES value to n. If n is in the range 1 through 31, a single line feed character is included in the timeout at the position of the pointer. If n is 32 or greater, the character with the ASCII value specified by n is included in the timeout at the position of the pointer. |
| ES         | Examine the setting of the ES flag.                                                                                                                                                                                                                                                   |

### 3.11.8 Case Control in Searches

When searching for alphabetic characters TECO will normally accept either upper or lower case characters as a match. This is called "either-case mode". TECO may, however, be forced to execute any or all searches in "exact mode". In exact mode TECO will accept an alphabetic character or a search match only if it has the same case as the corresponding character given by the user in the text argument.

Before the techniques for match mode control can be explained, we must first explain the various techniques for case control. Match mode control is explained in Section 3.11.8.4.

3.11.8.1 Alphabetic Case Control in Search Arguments - The case of alphabetic characters in search text argument is controlled by the same set of commands used to control case in insert text arguments.

The features described in this section provide the method by which alphabetic characters in the upper case range can be converted to the equivalent characters in the lower case range, and vice-versa.

Alphabetic case conversion is done by use of two control-character commands.

- ⓧ is used for translation to lower case.
- ⓨ is used for translation to upper case.

These two commands may be used within search text arguments to cause case conversion on a temporary basis for that text argument, or as independent commands to cause case conversion in all insert and search text arguments.

Note that ⓧ and ⓨ affect only alphabetic characters. They have no effect on non-alphabetic characters.

- (1) ⓧ ⓧ and ⓨ ⓨ used within text arguments.

When used inside a search text argument, two successive ⓧ or ⓨ commands cause translation, to the specified case, of all following alphabetic characters in that text argument.

Example:

\*SF ⓧ ⓧ OR USERS OF ⓨ ⓨ TECO. \$ \$

The above command searches for "For users of TECO." with the initial "F" and "TECO" capitalized, and all the other letters in lower case.

- (2) Single ⓧ and ⓨ used within text arguments.

When used inside a search text argument, a single ⓧ or ⓨ command causes translation of the next single character (if it is alphabetic) to the specified case. The single ⓧ or ⓨ in a text argument take precedence over the case conversion mode defined by double ⓧ or ⓨ commands.

Example:

\*S ⓧ ⓧ USER ⓨ PROGRAM \$ \$

The above command causes a search for the string "user Program" with the "P" in upper case, and all the other letters in lower case.

- (3) Independent ⓧ and ⓨ commands.

As explained above, when ⓧ and ⓨ commands are used inside a text argument, they affect only that particular text string. When used as independent commands, however, ⓧ and ⓨ set TECO to a prevailing case conversion mode that affects all insert and search text arguments (except as specified by ⓧ and ⓨ commands within the text arguments).

The independent command  $\text{\textcircled{tV}}$  or  $\text{tV}$  (or  $n \text{\textcircled{tV}}$  where  $n$  does not equal 0) sets the prevailing case conversion mode so that all upper case alphabetic characters in insert and search text arguments are translated to lower case, except where  $\text{\textcircled{tW}}$  commands within individual text arguments override the independent  $\text{\textcircled{tV}}$ .

Likewise, the independent command  $\text{\textcircled{tW}}$  or  $\text{tW}$  (or  $n \text{\textcircled{tW}}$ , where  $n$  does not equal 0) sets the prevailing case conversion mode so that all lower case alphabetic characters in insert and search text arguments are translated to upper case, except where  $\text{\textcircled{tV}}$  commands within individual text arguments override the independent  $\text{\textcircled{tW}}$ .

The independent  $\text{\textcircled{tW}}$  command has the use explained above, obviously, only when the user TTY has lower case capability and TTY LC is on. Otherwise the  $\text{\textcircled{tW}}$  command serves merely to turn off the  $\text{\textcircled{tV}}$  command.

(4)  $0 \text{\textcircled{tV}}$  and  $0 \text{\textcircled{tW}}$

The independent  $0 \text{\textcircled{tV}}$  and  $0 \text{\textcircled{tW}}$  commands both have the same effect, namely, to restore TECO to the default condition where neither case of alphabetic characters are translated to the opposite case, except by  $\text{\textcircled{tV}}$  and  $\text{\textcircled{tW}}$  commands within text arguments.

TECO is initially set for no prevailing case conversion.

Note that the prevailing case conversion mode can have one, and only one, setting at any one time. The possible settings are:

|                                |                                  |
|--------------------------------|----------------------------------|
| $\text{tV}$                    | convert upper case to lower case |
| $\text{tW}$                    | convert lower case to upper case |
| $0 \text{tV}$ or $0 \text{tW}$ | no prevailing conversion         |

When any of these prevailing modes is put into effect, it cancels any of the others that were in effect.

The order of precedence of the case conversion commands is as follows:

|          |                                                                            |
|----------|----------------------------------------------------------------------------|
| Highest: | single $\text{\textcircled{tV}}$ and $\text{\textcircled{tW}}$ inside text |
| Next:    | double $\text{\textcircled{tV}}$ and $\text{\textcircled{tW}}$ inside text |
| Lowest:  | independent $\text{\textcircled{tV}}$ and $\text{\textcircled{tW}}$        |

#### NOTE

If the EO Value has been set to 1 (refer to Paragraph 3.17.3),  $\text{\textcircled{tW}}$  and  $\text{\textcircled{tV}}$  have no special effect when encountered inside text arguments.

3.11.8.2 Special "Lower Case" Characters - When used inside a search text argument, the control command  $\text{\textcircled{t}}\text{\textcircled{t}}$  causes the immediately following character (if it is one of the special characters @, [, \, ], t, or ←) to be converted to the equivalent character in the lower case ASCII range (i.e., octal 140 or octal 173 to 177).  $\text{\textcircled{t}}\text{\textcircled{t}}$  has no special effect within text arguments if the EO value has been set to 1. Refer to Paragraph 3.8.7.2 for examples.

3.11.8.3 Control Characters in Search Arguments - As of version 22 of TECO all of the control characters  $\text{\textcircled{t}}\text{\textcircled{A}}$  -  $\text{\textcircled{t}}\text{\textcircled{G}}$ ,  $\text{\textcircled{t}}\text{\textcircled{N}}$  -  $\text{\textcircled{t}}\text{\textcircled{Z}}$ , and  $\text{\textcircled{t}}\text{\textcircled{\backslash}}$ ,  $\text{\textcircled{t}}\text{\textcircled{J}}$ ,  $\text{\textcircled{t}}\text{\textcircled{t}}$ , and  $\text{\textcircled{t}}\text{\textcircled{\leftarrow}}$  have been reserved as inside-text-commands (some as yet undefined). In order to search for these characters, the user must employ either the  $\text{\textcircled{t}}\text{\textcircled{R}}$  or  $\text{\textcircled{t}}\text{\textcircled{T}}$  command.

$\text{\textcircled{t}}\text{\textcircled{R}}$  when used inside a search text argument causes the next single character to be interpreted as text rather than as a command. This applies to all control characters including  $\text{\textcircled{t}}\text{\textcircled{R}}$  itself. It also applies to altmode. (It does not, however, apply to  $\text{\textcircled{t}}\text{\textcircled{C}}$ ,  $\text{\textcircled{t}}\text{\textcircled{O}}$ ,  $\text{\textcircled{t}}\text{\textcircled{U}}$ , or RUBOUT.)

$\text{\textcircled{t}}\text{\textcircled{T}}$  when used inside a search text argument causes all succeeding instances of the above mentioned control characters except  $\text{\textcircled{t}}\text{\textcircled{R}}$  and  $\text{\textcircled{t}}\text{\textcircled{T}}$  itself to be interpreted as text rather than as commands.

$\text{\textcircled{t}}\text{\textcircled{T}}$  does not affect altmodes. A second instance of  $\text{\textcircled{t}}\text{\textcircled{T}}$  in the same text argument nullifies the effect of the first.

If the EO value has been set to 1,  $\text{\textcircled{t}}\text{\textcircled{R}}$  and  $\text{\textcircled{t}}\text{\textcircled{T}}$  have no special effect when used inside text arguments, and all control characters (except the special characters) can be searched for with no special treatment.

3.11.8.4 Case Match Mode Control in Searches - Unless special action is taken all searches are executed in "either-case mode". This means that regardless of the setting of the prevailing case mode by an independent  $\text{\textcircled{t}}\text{\textcircled{V}}$  or  $\text{\textcircled{t}}\text{\textcircled{W}}$  command, a search for an alphabetic character will accept either the corresponding upper or lower case character as a match.

However, if  $\text{\textcircled{t}}\text{\textcircled{V}}$  or  $\text{\textcircled{t}}\text{\textcircled{W}}$  case control commands are used within a search text argument, it is assumed that the user desires an exact mode search, and a match will be accepted only for the corresponding characters in the exact case specified by the user.

If the user desires a search to be executed partly with exact mode and partly with either-case mode, he should bracket the characters to be taken in either case with  $\text{\textcircled{t}}\text{\textcircled{\backslash}}$  characters. (The  $\text{\textcircled{t}}\text{\textcircled{\backslash}}$  character is entered by simultaneously depressing the CTRL, SHIFT, and L keys.)

For example, S  $\text{\textcircled{t}}\text{\textcircled{V}}$   $\text{\textcircled{t}}\text{\textcircled{V}}$  ABC  $\text{\textcircled{t}}\text{\textcircled{\backslash}}$  DEF  $\text{\textcircled{t}}\text{\textcircled{\backslash}}$   $\text{\textcircled{\$}}$  will be successful only with strings containing lower case abc, but it will accept either upper or lower case def as a match for the last 3 characters.

## NOTE

If EO=1, all searches are executed in exact mode and

Ⓢ has no special effect in text arguments.

The search mode can be forced to exact mode for all searches by use of the independent command  $n \text{Ⓢ}$ , where  $n$  does not equal 0.  $0 \text{Ⓢ}$  resets the search mode to 'either' mode.  $\text{Ⓢ}$  without an argument returns the value of the search mode flag.

## 3.11.9 Special Match Control Characters

There are five special control characters that can be used in search character string arguments. These characters alter the usual character-matching process that goes on when a search is in progress. They actually reside in the search string and are interpreted by the search routine itself.

The presence of a  $\text{Ⓢ}$  command in a search string is a signal that this particular character position in the string is unimportant and that any character is to be accepted as a match for it. The  $\text{Ⓢ}$  command is a free variable in the search string. To find a match, some character must be present in the position occupied by the  $\text{Ⓢ}$  command; however, it does not matter what this character is.

The  $\text{Ⓢ}$  command in a search string is a restricted variable. Its presence indicates that any separator character is to be accepted as a match in its position. A separator character in any character except a letter, a digit, a period, a dollar sign, or a percent sign; i.e., any character except a character that is commonly used in symbols.  $\text{Ⓢ}$  also accepts the beginning of the editing buffer as a match.

The  $\text{Ⓢ}$  command is another restricted variable. It must be followed by a single character argument:  $\text{Ⓢ} x$ . The  $\text{Ⓢ}$  command signals that, in the position occupied by the  $\text{Ⓢ}$  and its argument, any character is to be accepted as a match except the argument.

The  $\text{Ⓢ}$  command is used in a search string to indicate that the character following the  $\text{Ⓢ}$  is to be interpreted literally rather than as a command, even if this character is one of the special match control characters. The  $\text{Ⓢ}$  command has the same function as  $\text{Ⓢ}$ , but it is better to use  $\text{Ⓢ}$  because  $\text{Ⓢ}$  will not allow insertion of  $\text{Ⓢ}$  as a text character while  $\text{Ⓢ}$  will.

The  $\text{Ⓢ}$  command when used with an argument in a search string indicates particular groups of characters to be accepted as a match. Depending on the argument, this command matches on the first occurrence of one of the following groups.

|              |                                                                                                    |
|--------------|----------------------------------------------------------------------------------------------------|
| $\text{Ⓢ} A$ | any alphabetic character.                                                                          |
| $\text{Ⓢ} D$ | any digit.                                                                                         |
| $\text{Ⓢ} L$ | any end of line character (or end of buffer character in the absence of an end of line character). |

|               |                                                                                                      |
|---------------|------------------------------------------------------------------------------------------------------|
| ⓔ S           | any string of spaces and/or tabs..                                                                   |
| ⓔ V           | any lower case alphabetic character.                                                                 |
| ⓔ W           | any upper case alphabetic character.                                                                 |
| ⓔ <nnn >      | the ASCII character whose octal value is nnn.                                                        |
| ⓔ [a,b,c,...] | any one of the characters a,b,c,... (a,b,c,... can be any symbols that represent single characters). |

Since the five commands ⓧ, Ⓞ, Ⓝ, Ⓡ, and ⓔ are used in the middle of ASCII search strings, they cannot be entered in the up-arrow, character form allowable for some control character commands. They must be typed as a single control character.

### 3.11.10 Examples of the Use of Search Commands

Examples:

```
*SA →|B Ⓞ Ⓞ
*
-
```

This causes the pointer to be positioned immediately after the B, in the first occurrence of the string A →|B after the current position of the pointer.

```
*SNIX Ⓞ Ⓞ
?SRH CANNOT FIND "NIX"
*
-
```

The string NIX is not found between the current pointer position and the end of the buffer. The error message is typed and the pointer moved to the beginning of the buffer. The user may have typed an incorrect search string, the pointer may have been positioned somewhere in the buffer after the N, or the string NIX may not have been read into the current buffer.

```
*NDIGITAL Ⓞ Ⓞ
*
-
```

If page 5 of the text is currently in the buffer and the string DIGITAL does not occur until page 15, this command causes pages 5 through 14 to be output and page 15 to be read in. The pointer will be set immediately after the L.

```
*NLAST LIN PG1
TST LIN PG2
Ⓞ Ⓞ
?SRH CANNOT FIND "LAST LIN PG1
TST LIN PG 2"
||
*
-
```

If this string actually exists in the file but the two lines are not read into the same buffer load, the N search will fail.

```
*12FSOF Ⓞ FOR Ⓞ Ⓞ
```

This command causes TECO to search the current buffer for the 12th occurrence of the string "OF" and replace it with the string "FOR".

\*5←VERSION88 (\$ \$)

\*-IESSWORD (\$ \$)  
60 FORMAT ('WORD')

\*

-

\*5FSINTEREST (\$ \$)

\*NMASSACHUSETS (\$ \$)  
?SRH CANNOT FIND "MASSACHUSETS

\*EF (\$ \$)

\*EBOUTPUT.FIL (\$ Y (\$ \$)

\*NMASSACHUSETTS (\$ \$)

\*

\*@3S+ (\$ +IEF (\$ \$)

\*

-

\*@FN/WRITE#/PRINT#/ (\$ \$)

\*NA (tX) B (tS) C (tN) .D (tR) (tX) (\$ \$)

\*

-

This command can be used to determine if the string VERSION88 occurs in the input file five times. If it does, the pointer is positioned immediately after the fifth occurrence, and everything in the input file, preceding the page on which the fifth occurrence is located, is discarded.

The ES value is set to -1 to cause the line where the search ended to be typed. This makes certain that the search has actually found the right occurrence of the string. It is easy to overlook an occurrence of a string preceding the one which the user desires.

This command causes TECO to search the current page for the fifth occurrence of the string "INTEREST" and delete it. Two (\$)'s must be present following the string to be deleted; the first delimits the string to be searched for and the second tells TECO that there is no replacement string.

An N search should not be used where an S search would suffice, because user errors with the N command, such as the spelling error shown here, can cause considerable delay. In this example, the user's error caused him to have to pass over the entire file twice instead of just once.

The command @3S+ (\$ + searches for the third occurrence of the altmode character following the buffer pointer. When this altmode is found, the characters EF are inserted immediately after it. The plus characters serve as the delimiters for the one-character search string (\$). The plus characters are not part of the search string.

This command causes TECO to search for the string "WRITE#" and replace it with the string "PRINT#." Each page of the text is searched until the string is found.

Any of the following three strings of characters would serve as a match for this N search:

A6B-C?D (tX)

A\_B →C\_D (tX)

AAB,C (\$ D (tX)

None of the following four strings would serve as a match:

AJB C-D3

A.B.C.D. (tX)

AABBCCD (tX)

AXB\_CAX (tX)

```

*1ESSFOUR ($) ($)
FOUR
    SCORE AND SEVEN YEARS AGO
    
```

Because the ES value was set to 1, automatic typeout of the line occurs after the string "FOUR" was found. A line feed was inserted at the pointer position in the line to allow the user to easily locate the pointer.

```

*1ESFSI/O ($) I-O ($) ($)
I-O
    CONTROL
    
```

This command string causes TECO to search for the string "I/O" on the current page and replace it with the string "I-O". The line is then typed with a line feed at the position of the pointer.

### 3.12 ITERATION COMMANDS

#### 3.12.1 Angle Bracket (<...>)

The user can cause a group of command to be iterated (repeatedly executed) any number of times by placing these commands within angle brackets. The left angle bracket marks the beginning of a command string loop and the right angle bracket marks the end of the loop. These command string loops can be nested in the same manner as arithmetic expressions are nested within parentheses. Loops should be nested to no more than approximately 20 levels; otherwise, pushdown list overflow may occur.

A numeric argument can be used to specify the number of times a given loop is executed. The argument is placed before the left angle bracket in the form  $n < \dots >$ . This causes the group of commands within the brackets to be iterated  $n$  times. In a command of the form  $n < \dots >$ , if the argument  $n$  is less than or equal to zero, the commands contained within the angle brackets are skipped. If no argument is given, the number of iterations is assumed to be infinite ( $2^{35}$ ).

Example:

```

*_J8< -> ($) L> ($) ($)
*
-
    
```

This command string inserts a tab at the beginning of the first eight lines in the buffer and leaves the pointer positioned at the beginning of the ninth line. The J command starts the pointer off at the beginning of the first line. The first command in the loop,  $-> ($)$  inserts a tab. Then the next command, L, moves the pointer to the next line to prepare for the next iteration of the loop.

#### 3.12.2 Semicolon Command

Iteration of a command string loop can be terminated before the iteration count is satisfied by using the conditional iteration exit command, semicolon. The semicolon command can be used only within angle brackets. It can be used with or without a numeric argument.

When used without a numeric argument, the semicolon command evaluates the outcome of the last search (of any kind) that was executed before the semicolon command was encountered. If this search was successful, command execution continues within the loop, as if no semicolon were present. If, however, the most recent search failed, the semicolon command causes all those commands that follow

the semicolon in the loop to be skipped over, and command execution to pass on to the first command following the right angle bracket which closes the innermost loop that the semicolon is in.

## NOTE

Within a command loop, all searches are colon searches. They do not generate error messages when a failure occurs, instead they return a value of -1 if successful and 0 if unsuccessful.

The semicolon command can also be used with a numeric argument. The command  $n;$  is ignored if  $n < 0$ . However, if  $n \geq 0$ , the command  $n;$  causes command execution to exit from the loop just as the semicolon command exits from the loop when a search fails.

Examples:

```
*J<0LIJAN ($) FS1969 ($) 70 ($) ;>HT ($)
```

```
JAN REPORT  
DEPT:
```

```
JAN 1970 SALES  
WHOLESALE:  
RETAIL:  
JAN 1970 EXPENSES:  
OVERHEAD:  
ADVERTISING:  
COMMISSIONS:  
JAN 1970 RETURNS:  
JAN 1970 INVENTORY:
```

```
*  
-
```

```
*<S1969 ($) ;0LIDEC ($) > ($)
```

```
[6K CORE]  
[7K CORE]  
[8K CORE]
```

```
(C) (C)
```

```
.REE ↵
```

```
*  
-
```

This command string inserts JAN at the beginning of the first line in the buffer and at the beginning of each line that contains the string 1970. It also changes the 69 in every occurrence of 1969 to 70. The action is as follows: The J command starts the operation at the beginning of the buffer. The first execution of the OL does nothing.

IJAN (\$) then inserts JAN at the beginning of the first line. Now, a search is made for 1969. When 1969 is found, FS1969 (\$) 70 (\$) changes the 69 to 70. This completes the first iteration; execution loops back to the <. OL moves the pointer to the beginning of the line where the 1969 was found. Here JAN is inserted and then a search is begun for the next 1969. This continues until the search command fails to find another 1969. When the search fails, the pointer is moved to the beginning of the buffer. HT is the next command which is executed. (It is assumed that no line contains more than one "1969.")

This command puts TECO into an infinite loop because the OL causes the search command to keep finding the same 1969 over and over again. If left to run long enough the IDEC (\$) command will eventually exhaust available core and stop execution. In this example, the user has stopped the loop with (C) (C) , and then REEntered.

```

*Y<NEXAMPLES: ($) ;<S)
($); →($ L >> ($ ($
*
-

```

This is an example of nested loops. The main loop searches for pages in a file that contain the heading EXAMPLES:. When this is found, execution enters the secondary loop, which inserts a tab at the beginning of all the succeeding lines on that page (i.e., after every ↵ on that page). When the second semicolon causes an exit from the inner loop, execution loops back to the N search. Finally, when the N search fails, execution is completed.

```

*EBfilnam.ext ($) 50000<YHP>EX ($) ($)

```

This example shows how to remove all form feeds from a file.

```

* <FSREAD ($) WRITE ($) ;>

```

This command causes a search of the current page for all occurrences of the string "READ" and replacement of them with the string "WRITE".

```

* <@FN/ERROR//;>
/

```

This command causes TECO to search all the following pages for the string "ERROR" and delete every occurrence of it. The @ construction must be used in this case because it allows the user to specify a delimiting character other than (\$). The delimiting character (in this case /) must be specified twice after the string; the first to end the string and the second to indicate that a replacement string is not present. If (\$) were used as the delimiter, a double (\$) would be present which would cause an erroneous result.

Only the methods described in this section should be used to exit from a loop. Specifically, the flow control commands described in Section 3.13 should not be used. Some violations of this rule may be successful, but generally they will not succeed.

Matching pairs of angle brackets defining loops within the loop may, however, occur following the semicolon.

### 3.13 FLOW CONTROL COMMANDS

TECO contains commands that enable the user to write editing programs capable of solving most complex editing problems. The iteration commands discussed in Section 3.12 are a specialized example. In addition to these, TECO has an unconditional branch command and a set of conditional execution commands that can be used to create any kind of conditional branch or conditional skip.

#### 3.13.1 Command String Tags

To have branching in a command string, there must be a method of naming locations in the command string. Location tags in the general form

!tag!

may be placed anywhere in a command string (except in text arguments). A tag is delimited before and after by an exclamation point and may contain any number of any ASCII characters except the special characters listed in Table 2-1 and exclamation points.

Command string tags are also the recommended method for putting comments in TECO macros; they need not be referenced.

### 3.13.2 O Command

The unconditional branch command is the O command. The general form is

\*Otag (\$)

The text argument following the O command and delimited by an altmode is the tag naming the destination of the branch. The tag location itself may be either before or after the O command in the command string. The O command causes the command string execution pointer to be moved to the first character following the exclamation point that terminates the tag, and command execution continues from that point.

Tags are ignored except when an O command forces TECO to scan the command string for them.

### 3.13.3 Conditional Execution Commands

All conditional execution commands have the following general form:

\*n'x...'

In this form, n is the numeric argument on which the decision to execute or not to execute is based. The quotation mark (") is the first character of all conditional execution commands. The letter x represents the second character of the conditional execution command. The letter x may be any one of several letters depending on which conditional execution command is intended. The two command characters, 'x, may be followed by any string of commands terminated by an apostrophe('). If the condition specified by x is satisfied by the argument n, all the commands between 'x and ' are executed in the usual manner. If there is no branch command within the range 'x...'; then after the last command in the range is executed, command execution falls through the apostrophe and executes the next command following it. If n does not satisfy the condition specified by x, then all the commands between 'x and the matching ' are skipped, and command execution continues with the first command following the apostrophe.

The commands 'x and ' must be used in matching pairs and they may be nested in the same manner that parentheses surrounding arithmetic expressions may be nested.

The individual conditional execution commands are shown in Table 3-7.

Table 3-7  
Conditional Execution Commands

| Command | Function                                                                                                                                                                                     |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| n"G     | Execute the commands that follow if n > 0; otherwise, skip to the matching apostrophe on the right.                                                                                          |
| n"L     | Execute the commands that follow if n < 0; otherwise, skip to the matching apostrophe on the right.                                                                                          |
| n"E     | Execute the commands that follow if n = 0; otherwise, skip to the matching apostrophe on the right.                                                                                          |
| n"N     | Execute the commands that follow if n ≠ 0; otherwise, skip to the matching apostrophe on the right.                                                                                          |
| n"C     | Execute the commands that follow if n is the decimal value of an ASCII symbol constituent character (a letter, digit, \$, ., or %); otherwise, skip to the matching apostrophe on the right. |
| n-1"L   | Execute the commands that follow if n < 0; otherwise, skip to the matching apostrophe on the right.                                                                                          |
| n+1"G   | Execute the commands that follow if n > 0; otherwise, skip to the matching apostrophe on the right.                                                                                          |
| n"D     | Execute the commands that follow if n is in the digit range (octal 60 to 71).                                                                                                                |
| n"A     | Execute the commands that follow if n is in the alphabetic range (octal 101 to 132 or 141 to 172).                                                                                           |
| n"V     | Execute the commands that follow if n is in the lower case alphabetic range (octal 141-172).                                                                                                 |
| n"W     | Execute the commands that follow if n is in the upper case alphabetic range (octal 101 to 132).                                                                                              |
| n"T     | Execute the commands that follow if n is 'true' (flag is on) (i.e., if n < 0).                                                                                                               |
| n"F     | Execute the commands that follow if n is 'false' (flag is off) (i.e., if n = 0).                                                                                                             |
| n"S     | Execute the commands that follow if n is 'successful' (i.e., if n < 0).                                                                                                                      |
| n"U     | Execute the commands that follow if n is 'unsuccessful' (i.e., if n = 0).                                                                                                                    |

3.13.4 Examples of the Use of Flow Control Commands

\*!START! J → → PDP-10 TECO )

Ⓢ

<S 5K Ⓢ ;R-DI6 Ⓢ>

<SWAR Ⓢ ;-3DILOVE Ⓢ >

PZ"NOSTART Ⓢ '

EF Ⓢ Ⓢ

!INSERT PAGE HEADING!

!CHANGE 5K TO 6K!

!CHANGE WAR TO LOVE!

!GET NEXT PAGE AND!

!RESTART IF NOT NULL!

This small editing program contains an example of the O command, i.e., the OSTART (\$) command which causes a jump back to !START!. It also contains examples of command string tags used purely for documentation, e.g., !INSERT PAGE HEADING!. Normally, comments would be used only for lengthy and complex macros that the user expects to maintain.

This example also shows how a conditional execution command may be combined with an O command to produce a conditional branch. When all three of the editing functions have been performed on the page, the P command is executed to output this page and read in the next. The program then tests Z (the number of characters in the buffer) to determine if any data was read in. If  $Z \neq 0$ , data was read in, therefore a branch is taken to restart the program. When finally  $Z=0$ , the command OSTART (\$) is skipped, and execution branches to the concluding EF command. This technique fails when a file contains null pages (consecutive form feed characters). Therefore, the (N) end-of-file test is preferred.

```
*YZ'N!##!Z-4000+1'G4000JOL12I ($) 0,.P0,.KO## ($) 'ZJA.-Z'NO## ($) ''PEF ($) ($)
```

This slightly more complex command string shows how conditional execution commands may be nested. If the first Y command produces no data, the 'N command sends execution to the matching apostrophe on the right. This is the last apostrophe, immediately prior to the PEF. Otherwise, the commands following the 'N are executed.

The function of this command string is to convert a file with pages of arbitrary lengths to one with pages of approximately 4000 characters each.

The command string operates as follows:  $Z-4000 + 1'G$  means if  $Z \geq 4000$ , i.e., there are at least 4000 characters on the current page, execute the following commands; otherwise, skip to the matching apostrophe (between (\$) and Z). If  $Z \geq 4000$ , 4000JOL moves the pointer to the end of last complete line before the 4000th character in the buffer. Then, 12I (\$) 0,.P outputs this much of the buffer with a form feed character after it, and 0,.K deletes that which has been output. Now, go back to !##! and test Z again. Stay in this loop until  $Z < 4000$ . Execution then skips to the apostrophe. ZJ moves the pointer to the end of the current buffer. A appends another page, but leaves the pointer (.) at the end of the previous page. .-Z'N checks to determine if any data was actually read in. If so, the loop is reentered at !##!; otherwise the end of the file has been reached. When .-Z=0, execution skips to the matching apostrophe and then falls through the next apostrophe to the PEF that closes the output file.

```
*<NSIN ($) ;:SCOS ($) ''S-3DITAN ($) 'ZJ> ($) ($)
```

This example shows how the value returned by a colon search can be used as the argument for a conditional execution command. The N command searches through the file for the first occurrence of SIN on any page. When SIN is found, the command :SCOS (\$) checks for an occurrence of COS

following SIN on the same page. The colon search command returns the value -1 if the search is successful, and 0 if there is no COS following SIN on the page. This value is then used as the numeric argument for the "S command. If :SCOS (\$) has a value of -1, the occurrence of COS that was found is replaced by TAN. If :SCOS (\$) has a value of 0, the commands -3DITAN (\$) are skipped. We then jump to the end of this page, ignoring all further occurrences of SIN and COS on it, and continue the iteration process.

### 3.14 Q-REGISTER COMMANDS

Q-registers are a powerful feature of TECO with many different uses. The general concept of Q-registers is explained in Section 2.8. Section 3.14 explains the TECO commands that enable the use of Q-registers.

The 36 Q-registers have the single character names A, B, C, ..., Z, and 0, 1, 2, ..., 9. In this section, the letter i is used to represent the name of an arbitrary Q-register.

#### 3.14.1 Commands for Storing Integers

The following commands enable the use of Q-registers for storing single 36-bit integers.

3.14.1.1 U Command - The command nUi stores the decimal integer n in Q-register i. n may be any integer in the range  $-2^{35} + \leq n \leq 2^{35} - 1$ . If anything was previously in Q-register i, it is destroyed.

3.14.1.2 Q Command - The command Qi is used to read the numeric value in Q-register i. Qi has no function other than returning the value in the specified Q-register as a numeric argument. It does not alter the value in the Q-register. In order to be useful, Qi must be used as a numeric argument for another command. Qi is often used in conjunction with conditional commands.

3.14.1.3 % Command - The command %i adds 1 to the integer in Q-register i and then returns the new value in the same manner as a Qi command. If the user wants to increment the value in Q-register i, but does not want the returned value to be used as an argument for the next command, he should type an altmode after the %i command.

#### 3.14.2 Commands for Storing Character Strings

The following commands enable the user to store character strings of any length consistent with the amount of core available.

3.14.2.1 X Command - The X command copies characters from the editing buffer into a Q-register. These characters are not removed from the editing buffer. Any data previously in the Q-register is destroyed.

The various uses of the X command are as follows:

- a.  $m, nXi$  ( $m < n$ ) copies the  $m + 1$ st through the  $n$ th characters in the buffer into Q-register  $i$ .
- b. If  $n > 0$ ,  $nXi$  copies everything from the current buffer pointer position through the  $n$ th following line terminator character into Q-register  $i$ .  $Xi$  is equivalent to  $1Xi$ .
- c.  $0Xi$  copies everything from the beginning of the current line up to the buffer pointer into Q-register  $i$ .
- d. If  $n < 0$ ,  $nXi$  copies everything from the beginning of the  $n$ th line preceding the current line up to the buffer pointer into Q-register  $i$ .  $-Xi$  is equivalent to  $-1Xi$ .

An X command may require more core space for storage than is available. If so, TECO automatically tries to expand its core. If successful, TECO prints a message in the form [nK CORE] to show the new amount of core being used. If unsuccessful, TECO prints an error message and does not execute the X command.

3.14.2.2 G Command - The command  $Gi$  fetches a copy of the entire character string stored in Q-register  $i$  and inserts it into the editing buffer at the current position of the buffer pointer. The contents of Q-register  $i$  are not changed. The buffer pointer is positioned at the right end of the character string that was inserted by the G command.

3.14.2.3 M Command - TECO command strings are basically ASCII character strings and, as such, can be inserted or read into the editing buffer just like any other text. When a command string is in the editing buffer, it can be edited but it cannot be executed, because at that point it appears to be data to TECO. However, if the user copies a command string from the editing buffer into a Q-register (using an X command), then this command string can be executed. The command that accomplishes this is the  $Mi$  command.

The command  $Mi$  executes the text in Q-register  $i$  just as if this text had been typed in the command string instead of  $Mi$ . Using an  $Mi$  command is analogous to calling a subroutine. Any TECO commands may be included in the command string or "macro" which is stored in and executed from the Q-register. Even double altmodes can be included if there are conditions under which the user wants execution to stop. The only restriction is that the commands must all be complete within the macro in the Q-register. For example, a command and its argument must not be split apart, one in the main command string with the  $Mi$  command and the other in the Q-register. Iterations and conditional execution strings, if included, must be complete within the Q-register. If an O command is used in the Q-register macro, the tag to which it branches must be in the Q-register also. M commands may be nested up to approximately 10 levels, depending on the contents of the internal pushdown list.

### 3.14.3 Saving the Previous Command String

After a command string has completed execution or if it has been aborted by means of the   command, it may be stored in a Q-register. This is done by using an  $*i$  command as the first command in the next command string.

\*i causes the entire previous command string, less one of the two concluding altmodes, to be stored in Q-register i. If the command string was aborted by (tG) (tG), neither (tG) is stored with the command string. The previous contents of Q-register i are lost. The asterisk has this function only when used as the first command in a command string. At any other position in a command string, asterisk has its usual meaning of multiplication (see Section 2.7.2).

If the user intended to use \*i as the first command but typed some other command first instead, he may recover the ability to use \*i as the first command by typing enough rubouts to cause TECO to respond with a carriage return/line feed and a new asterisk. This technique will not work perfectly if some of the characters typed before the \*i command were break characters (altmode, carriage return, etc.). In this case some of the leading characters of the preceding command string will be overwritten.

The \*i command is especially useful when an error occurs in a long command string. See the example in Section 3.14.5.

#### 3.14.4 Q-Register Pushdown List

An additional Q-register feature is the Q-register pushdown list, which may be used for temporary storage during the execution of a command string.

The command [i pushes the contents of Q-register i onto the stack. It does not change the contents of i.

The command ]i pops the last pushed entry from the top of the pushdown list into Q-register i. The previous contents of Q-register i are lost; the entry which was popped off the pushdown list is erased from the top of the list.

#### NOTE

The Q-register pushdown list is cleared after the execution of each complete command string (i.e., every time TECO types an \* to indicate readiness to accept a new command string).

The maximum depth of the Q-register pushdown list is 32 entries. (This number can be changed by redefining LPF in TECO.MAC and reassembling TECO.)

#### 3.14.5 Examples of the Use of Q-Register Commands

\*QR-3UR (\$) (\$)

This command subtracts 3 from the value in Q-register R

YISTIOUCIST + 11:S↓

                  Ⓢ ''S%C-50''LOST + 1 Ⓢ'12I Ⓢ 0, .P0, .KOST Ⓢ'↓  
 ZUEAQE-Z''NQEJOST + 1 Ⓢ'PWEF Ⓢ Ⓢ

This command string arranges a file into pages of 50 lines each. The Y command starts operation at the beginning of the file. At IST! the command OUC sets the value 0 in Q-register C. At IST+1! search begins for a line feed. The command :S↓ Ⓢ

returns a value of -1 if a line feed is found, in which case ''S causes the following commands to be executed. The %C command increments Q-register C by 1 and returns the new value in C. If %C<50, jump back to IST+1! and search for another line feed. However, if %C=50, proceed as follows: (1) insert a form feed character because the output command used does not output one automatically, (2) output everything from the beginning of the buffer through the form feed character, then (3) delete everything that was output and (4) go back to IST! where the counter is reinitialized and start over.

If the search command fails to find another line, with the value in Q-register C less than 50, it returns the value 0, therefore the ''S command causes a skip to the apostrophe at the end of the second line. The carriage return is ignored (see Section 3.18). The ZUE command stores the number of characters currently in the buffer in Q-register E. The A command reads in more data without moving the buffer pointer, while QE-Z''N checks the old value of Z with the new value to see if any data was actually read. If data was read, QEJ sets the pointer at the end of the old data and before the new data, then continue the line count at IST+1!. If not, output the last page and close the file.

0, .X10, .KZJG1 Ⓢ Ⓢ

This command string moves everything to the left of the pointer from its position at the beginning of the page to the end of the page. The 0, .X1 command puts everything from the top of the page to the pointer in Q-register I. The 0, .K command deletes this data from its present position. The ZJ command moves the pointer to the end of the page. At this point the command G1 copies the contents of Q-register I into the buffer at the position of the pointer.

ZJ-5XAJ8LGA Ⓢ Ⓢ

This command string puts a copy of the last five lines of the page into Q-register A and then puts a copy of these five lines immediately after the eighth line in the page. It does not delete the five lines from their position at the end of the page.

\_HK@I#J<SREAD (\$) ; -4DIACCEPT (\$) >#HXS (\$) (\$)

\_Y4PMS6PMS2PMSEX (\$) (\$)

EXIT

IC

-

In this example, the @I command inserts a short macro into the buffer. The # character is used to delimit the insertion. The HXS command stores this macro in Q-register S. In the second command string, the MS command executes the stored macro on pages 5, 11, and 13 of the input file. Note that the initial Y command clears the macro from the buffer before the first page is read in. The EX command copies all remaining pages, closes the output file, and returns to the monitor.

\_J16<[DSDIMENSION (\$) 0L1XDK >J4L16<GD]> (\$) (\$)

\*

-

The 16 <[DSDIMENSION (\$) 0L1XDK > command locates the first 16 lines on the current page that have the word DIMENSION in them, stores them on the Q-register pushdown list, and then deletes them from their present positions. Then the J4L16<GD]> command brings these 16 lines back onto the page immediately after the fourth line from the top.

\_A LOT OF TEXT (\$) (\$)

?NFI NO FILE FOR INPUT

\*\* Z (\$) (\$)

\_GZ (\$) (\$)

\*-D (\$) (\$)

\*

-

Assume the user meant to insert "A LOT OF TEXT" but forgot the "" at the beginning. The following technique illustrates the simple way to recover from this common error.

Move the entire command string (with just one altmode at the end) into Q-register Z.

Move the command string from Q-register Z into the editing buffer at the current pointer position.

Delete the altmode at the end of the command string. The rest of the command string is the text that was to be inserted, and it is now inserted.

\*5DITITLE (\$) NLONG STRING (\$)

-8DIA LOT OF TEXT (\$) (\$)

?NFO No File for Output

An error is encountered early in a long command string. (The N-search failed because it could not output the page in the editing buffer. The commands preceding the N-search have been executed.)

Save that entire command string in Q-register Z.

Save the current pointer position. Move the pointer to the beginning of the buffer (a convenient place to edit the command string), and get the string back from Q-register Z.

\*\*Z (\$) (\$)

\*.UP (\$) (\$)

\_JGZ (\$) (\$)

Delete the commands "5DITITLE (\$) " that have already been executed.

\*J9D (\$) (\$)

-

```

EWOUT.FIL ($) ($)
STEXT ($) D' ($) ($)

0,.XZ ($) ($)
0,.K ($) ($)
QPJ ($) ($)

MZ ($) ($)
*
W<SDIVIS ($) ;S= ($)RINOT _ ($)
LIX1KLG1> ($) ($)
?ILL Illegal Command W

**ZHKGZ ($) ($)

JDHXZ ($) ($)

```

Correct the error.

Get back to the end of the command string. The D command deletes the (\$) at the end of the command string.

Put the corrected string back into Q-register Z.

Delete the command string from the editing buffer.

Move the pointer back to its previous position. (In this particular case this step is not actually necessary.)

Execute the corrected command string.

This example shows a simple technique for creating a TECO macro. The user purposely begins the command string with an illegal command. The rest of the command string is the TECO macro the user wishes to create.

When the expected error occurs, move the command string to Q-register Z, then move it into the editing buffer.

Delete the W from the beginning of the macro, then save the correct macro in Q-register Z.

### 3.15 NUMERIC TYPEOUT COMMAND

The numeric typeout command is n=, where n is the numeric value to be typed in decimal radix. If a double = sign is used, the numeric value is typed in octal radix.

Example:

```

YZ = ($) ($)
2529
*
IA== ($) ($)
40

```

This reads in a page and then types out the (decimal) number of characters in the page.

This types the octal representation of the next character in the buffer.

### 3.16 SPECIAL NUMERIC VALUES

Several TECO commands, which have no other purpose than to return some particular numeric value, have already been discussed in this manual. These commands are B, Z, ., and Qi. Some commands that execute a function while returning a numeric value have also been discussed. These commands are %i; colon searches, and all searches within iterations. The concept of a command returning a numeric value is explained in Section 3.11.

All of these commands can be used as numeric arguments for commands that take a numeric argument, e.g., nI, n=, n;, nD, nUi, etc. To perform this function place the command, which returns a numeric value, in the position of n immediately before the command that takes a numeric argument.

There are several other commands that return numeric values; these commands are listed below.

The nA command (where n can be any numeric value, and serves only to differentiate this command from the A (append) command) is equivalent to the ASCII value of the character immediately to the right of the buffer pointer. The nA command equals 0, if the pointer is at the end of the buffer. The nA command is used primarily with conditional commands where one is checking for a particular character or range of characters.

The (tE) (or tE) command returns the value of the form feed flag. If, on the last input command (Y or A), input was terminated because a form feed character was encountered, E equals -1; otherwise, E equals 0. For further discussion of the form feed flag, see Sections 2.4, 3.3, 3.9, 3.10, 3.11 and 4.2.

The (tN) (or tN) command returns the value of the end-of-file flag. If the end of the input file was seen on the last input command (Y or A), tN = -1; otherwise, tN=0. When tN is set to -1, it will remain -1 until cleared by an ER or EB command. When tN is first set to -1, new data may or may not have been read into the editing buffer. Consequently, the tN flag should usually be tested after processing the input data.

The tF (or (tF)<sup>1</sup>) command is equivalent to the value of the console data switches.

The (tH) (or tH) command is equivalent to the time of day in 60th's of a second (50th's where 50 Hz power is used).

The ET command (without a numeric argument) returns the value of the ET flag. The ET command equals -1 if the flag is on and equals 0 if the flag is off. The significance of this flag is discussed in Section 3.6. When the ET flag is on, the T command delivers all characters, including altmodes and control characters, to the terminal in their exact form rather than substituting other characters.

The EU command returns the value of the case flag. The EU value is 1 if upper case characters are flagged on typeout; 0, if lower case characters are flagged on typeout (default); and -1, if no case flagging is being performed. Refer to Section 3.6.

The EH command returns the value of the error message flag. The EH value is 1 if only the error is typed; 2, if the error code plus one line is typed (default); and 3 if the full error message is typed. Refer to Section 5.2.

The EO command returns the value of the version number flag. The EO value is 1 for version 21A of TECO and 2 for versions 22 and 23 of TECO. Refer to Section 3.17.

The ES command returns the value of the automatic typeout flag. The ES value is -1 for automatic typeout after successful searches, 1 through 31 for automatic typeout with a line feed to indicate the pointer position, a decimal number greater than 31 for automatic typeout with the character equal to the ASCII value of the decimal number indicating the pointer position, and 0 for no automatic typeout (default). Refer to Section 3.11.

---

<sup>1</sup>When using TECO with monitors prior to the 5.02 monitor, the tF TECO command must be entered in the up-arrow, F form because control-F is interpreted as a special monitor command (see Section 3.18).

The  $\text{\textcircled{tt}}$  (or  $\text{\textcircled{tt}}$ ) command, followed by an arbitrary character  $x$ , is equivalent to the ASCII value of the character that immediately follows the  $\text{\textcircled{tt}}$  in the command string. For example, in the command  $\text{\textcircled{tt}} A$ , the character  $A$  is an argument for  $\text{\textcircled{tt}}$  and is not interpreted as a command. ( $\text{\textcircled{tt}} A$  equals 65.)

The backslash(\) command (without a numeric argument) is equivalent to the decimal value of the digit string (optionally preceded by a + or - sign) immediately following the current position of the buffer pointer. The value is terminated by the first nondigit character encountered. If there is no digit string immediately following the buffer pointer, backslash equals 0. The backslash command moves the buffer pointer to the right end of the digit string and assumes the value of the digit string.

The  $\text{\textcircled{tT}}$  (or  $\text{\textcircled{tT}}$ ) command is used to enable type-in of characters while the command string is being executed. When the  $\text{\textcircled{tT}}$  command is encountered in a command string, execution of the command string stops and waits for the user to type any single character. When this character is typed, the  $\text{\textcircled{tT}}$  command assumes the value of this character. Hence, the  $\text{\textcircled{tT}}$  command is useful only as a numeric argument for another command, e.g., the command  $\text{\textcircled{tT}}UC$  puts the ASCII value of the typed character into Q-register C.

The  $\text{\textcircled{tT}}$  command is most often used with a  $\text{\textcircled{tA}}$  message string preceding it (see Section 3.17). The message string is used to inform the user that TECO is waiting for a character to be typed in.

### 3.16.1 Examples of the Use of the Special Numeric Arguments

$\text{\_}J3C1A== \text{\textcircled{\$}} \text{\textcircled{\$}}$

$\text{\_}71$

$\text{\_}^*$

$\text{\_}$

$\text{\_}^*J!A!1A-97"G1A-123"L1A-32UCDQCI \text{\textcircled{\$}} \text{\textcircled{OB}} \text{\textcircled{\$}} ' '$

$\text{\_}CIB!2A"NOA \text{\textcircled{\$}} ' \text{\textcircled{\$}} \text{\textcircled{\$}}$

$\text{\_}^*$

$\text{\_}$

If the fourth character in the buffer is 9, the command string returns the indicated result.

This command string converts all lower case alphabetic characters in the buffer to upper case. Starting at the beginning of the buffer (J), if the next character has a decimal ASCII value between 97 and 122 inclusive ( $1A-97"G1A-123"L$ ), store the upper case value of this character in Q-register C ( $1A-32UC$ ), delete the character (D) and replace it with the value in Q-register C ( $QCI \text{\textcircled{\$}}$ ). Then TECO skips to  $IB!$  ( $OB \text{\textcircled{\$}}$ ); otherwise, it advances to the next character (C). In either case, at  $IB!$  TECO checks to determine if there is another character in the buffer ( $2A"N$ ) and if so, returns to  $!A!$  ( $OA \text{\textcircled{\$}}$ ). When  $2A$  equals 0, execution stops.

$\text{\_}^*P<-1- \text{\textcircled{tE}} ;A> \text{\textcircled{\$}} \text{\textcircled{\$}}$

$\text{\_}[3K CORE]$

$\text{\_}[4K CORE]$

$\text{\_}^*$

$\text{\_}$

This command string outputs the current page, and then continues input until a form feed character is detected. This command string could be used on a file that is not divided into pages of a reasonable size. The A command is repeatedly executed until  $\text{\textcircled{tE}}$  equals -1. When  $\text{\textcircled{tE}}$  equals -1, the semicolon command causes an exit from the loop.

```
*tF= tH=ET= ($) ($)
23094886497
1823373
-1
*
```

This command string causes the (decimal) value of the console data switches, the time of day in 60th's of a second, and the value of the ET flag to be typed out. At this execution, the console switches were set to octal 254064000141, the time was 08:26:29:33, and the ET flag was on.

```
*t↑ MU0 ($) ($)
*
*YCHAPTER \= ($) ($)
T6
*
```

This command string stores the ASCII value of the letter M (77) in Q-register 0.

This command string searches for the next chapter heading and then types out the number of the chapter. The buffer pointer is positioned immediately following the 6, after the command in this example has been executed.

```
*<SFUNCTION ($) ; (tA)
FUNCTION LETTER (tA) (tT) | ($) > ($) ($)
FUNCTION LETTER M
FUNCTION LETTER K
FUNCTION LETTER C
```

Here, the (tT) command is used as the argument for an ni (\$) insert command. This command string inserts the letter which is typed in following each occurrence of the string FUNCTION that is found by the search command.

```
*
*YITITLE
 ($) PWtN; > ($) ($)
*
```

This command string inserts "TITLE" at the top of each page of a file.

### 3.17 TECO PROGRAMMING AIDS

Bugs can occur in editing macros written in TECO language as in any other program; therefore, TECO provides the following debugging aids for the TECO user.

#### 3.17.1 (tA) Command

The user can cause a statement to be typed out at any point in the execution of a command string. The (tA) command is used to perform this function. The general form of this command is

(tA) text (tA)

The first (tA) is the actual command. It can be entered either as (tA) or tA. The string "text" is the character string that TECO types out when the (tA) command is encountered. The second (tA) command marks the end of the text to be typed and must be entered as (tA). The text string can contain any characters except (tA) and the special characters listed in Table 2-1.

Example:

```
*Y!ST! (A) NEW PAGE
(A) OUC!ST+1!:S↓
      ($) "N%C-50"LOST+1 ($) ' 12 I ($) 0,. P0,. KOST ($) ' )
ZUEAQE-Z"NOST=1 ($) ' (A) END )
(A) PWEF ($) ($)
NEW PAGE
NEW PAGE
NEW PAGE
NEW PAGE
NEW PAGE
END
*
-
```

This command string is identical to an example used in Section 3.14; however two (A) commands have been added.

### 3.17.2 Question Mark (?) Command

The question mark command has two uses in TECO. When question mark is the first character typed by the user after TECO has typed out an error message, it has the special function described in Section 5.2. However, at any other time the question mark can be entered in a command string exactly like any other command. This use of the question mark command causes TECO to enter trace mode. In trace mode, TECO types out each command as it is executed. A second question mark command takes TECO out of trace mode.

Example:

```
*JHT?!L!1A-9" N!M!1A-58" NCOM ($) 'CD → ($) 'LOL ($) ($)
AB: LINE1
      LINE 2
C:  LINE 3
      LINE 4
!L!1A-9" N!M!1A-58" NCOM$1A-58" NCO!M!1A-58" NCD $ 'LOL$1A-9" NLO!L!1A-9" N!
M!1A-58" NCO!M!1A-58" NCD $ 'LO!L!1A-9" NLO!L!1A-9" N!M!1A-58" NC?POP
*J?HT ($) ($)
J?
AB: LINE1
      LINE2
C:  LINE3
      LINE4
*
-
```

After the first question mark command, TECO begins typing out each command as it is executed. This enables the user to see exactly what the command string is doing. The ?POP error message is caused by the attempt to move the pointer beyond the end of the fourth (and last) line (the end of the buffer) with the C command.

The second question mark command turns off the trace feature so that the "HT" following it is not printed.

### 3.17.3 The EO Value

The EO (Edit Old) feature enables TECO users to protect existing TECO macros from future changes to the TECO specifications. In most cases when features are added to TECO, the changes merely

involve additional commands whose existence in no way affects old TECO macros. The EO feature does not apply to changes such as these. Occasionally, however, a new feature would cause old macros not to run properly. The EO feature is designed to protect old macros from such changes.

Every version of TECO has an EO value. For all versions of TECO up through version 21A, the EO value is 1. For TECO versions 22 and 23, and all succeeding versions until the next specification change that would affect old macros, the EO value is 2.

The EO value is always initially set to the maximum value for the version of TECO being run. This enables all new features.

By using the EO command the EO value can be set to a lower value so as to disable features of TECO that were implemented since the macro was written and which would cause the macro not to function properly. The EO command does not disable all new features, but only those that affect old macros.

- 0EO or                      resets the EO value to the maximum (standard)
- nEO (n<0)                    for the version of TECO in use.
- nEO (0<n<=max)            sets the EO value to n.
- EO (no argument)        returns the current value of the EO flag.

All TECO macros written before version 22 should be edited by putting "1EO" at the beginning and "0EO" at the end. All macros written with version 22 should have "2EO" at the beginning and "0EO" at the end, etc.

Table 3-8  
Features Enabled by EO Values Greater Than 1

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EO=1 | Base value.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| EO=2 | <ul style="list-style-type: none"> <li>(1) Standard altmode changed from ASCII 175 to 033.</li> <li>(2) All control characters within text arguments reserved as commands, instead of only <math>\text{\textcircled{tN}}</math>, <math>\text{\textcircled{tQ}}</math>, <math>\text{\textcircled{tS}}</math>, <math>\text{\textcircled{tX}}</math> in search strings.</li> <li>(3) Standard searches accept either upper or lower case alphabetic characters as a match.</li> <li>(4) Vertical tab and form feed recognized as end-of-line characters in addition to line feed.</li> <li>(5) The P command does not create form feeds.</li> <li>(6) Command string jumps will not accept instances of the target characters occurring within text arguments.</li> <li>(7) Because of (6) comments should be enclosed only by !...!.</li> <li>(8) The nl command must be followed by altmode.</li> <li>(9) The <math>\text{\textcircled{tG}}</math> exit command is changed to <math>\text{\textcircled{tZ}}</math>.</li> </ul> |

Examples:

|                                                             |                            |
|-------------------------------------------------------------|----------------------------|
| <u>*EO=</u> (Ⓢ) (Ⓢ)<br><u>2</u>                             | Initial setting is EO=2.   |
| <u>*1EOEO==</u> (Ⓢ) (Ⓢ)<br><u>1</u>                         | Set EO value to 1.         |
| <u>*0EOEO==</u> (Ⓢ) (Ⓢ)<br><u>2</u><br><u>*</u><br><u>-</u> | Revert back to EO=maximum. |

### 3.18 COMMAND STRING TYPE-IN CONTROL COMMANDS

The use of two successive altmodes as the command string terminator has already been discussed in Section 2.6. The use of rubout, (Ⓣ), and double (Ⓒ) as command string erasing commands is discussed in Section 5.1. There are other characters, however, that are useful in the creation of command strings.

#### 3.18.1 Carriage Return, Line Feed, and Spaces

Except as text arguments, the characters carriage return and line feed are ignored in command strings. Spaces are also ignored except (1) when used in text arguments, and (2) when used between two numeric arguments as a + (see Section 2.7.2). Hence, these characters can be employed by the user when formatting command strings. The carriage return (and the monitor-supplied line feed following it) is used to enable the user to conveniently type command strings much longer than a single line. Spaces are used to lend clarity to more complicated macros.

# Chapter 4 Techniques

## 4.1 CREATION, EXECUTION, AND EDITING OF A FORTRAN PROGRAM

This section demonstrates the use of TECO's multi-purpose commands to simplify the creation and editing of programs.

The following example shows the creation and immediate execution of a FORTRAN program.

```

_ MAKE ATEST.F4 )           Give the command to create the
                               disk file ATEST.F4 using TECO.
* - TYPE 1 )                Begin insertion with the TAB command.
1  FORMAT ('COMPILER
   ARITHMETIC TEST') )
   J=3 )
   K=7 )
   X.5 (RO) 5 (RO) .=.5      Rub out erroneous .5.
($)-T ($)$                 Stop insertion and use the -T command
                               to verify last line inserted.
   X=.5
* - II=K/J*(X*1.E+2-K*K/    Continue insertion.
   (3.*J)) )
   R=10.6 )
   S=3.5 )
   I=5 )
   J=2 )
   N=7 )
   Z=R+S*I/J*N/3 )
   TYPE 2,II,Z )
1  FORMAT (18,F20.12) )
   END )
($) EX ($)$                 End insertion, and then use the EX
                               command to output and close the file.

EXIT
tC

```

```

.EXECUTE ATEST)
FORTRAN: ATEST.F4
UNDEFINED LBLs
2
MULTIPLY DEFINED LBLs
1
MAIN. ERRORS DETECTED: 2
?TOTAL ERRORS DETECTED: 2
LOADING
LOADER 4K CORE
?EXECUTION DELETED
EXIT
IC

```

Give the command to compile and execute ATEST.F4.

The FORTRAN compiler discovers errors in the program.

```

.TECO)
*SF20. ( ( OLDI2 ( ( OTT ( ( (
2 FORMAT (18,F20.12)

```

Call TECO to edit ATEST.F4.

Change the second label 1 to 2, and then verify the change.

```

*EG ( (
FORTRAN: ATEST.F4
LOADING
LOADER 4K CORE
EXECUTION
COMPILER ARITHMETIC TEST
103 21.683333118562
EXIT
IC

```

Output the new version and automatically cause a repeat of the execution by using the EG command.

Success.

⋮

#### NOTES

- a. The command MAKE ATEST.F4) is equivalent to the following sequence of commands:

```

.R TECO)
*EWATEST.F4 ( (
⌘

```

- b. The -T command does not move the buffer pointer, therefore, the user can continue insertion from the point he left off.
- c. In this example, the EX command is equivalent to PWEF (C .
- d. No filename is given with the command TECO, therefore the name of the file used in the most recent edit-class command (i.e., MAKE, or TECO command) is assumed. In the example, the command TECO) is equivalent to

```

.R TECO)
*EBATEST.F4 ( Y ( (
⌘

```

NOTES (Cont)

- e. The command SF20. (\$) moves the pointer to the line the user wishes to correct. The OL command positions the pointer immediately prior to the bad character 1. The D command deletes the 1; the I2 (\$) command inserts 2 in its place. The OTT command types out this entire line.
- f. The command EG (\$) (\$) is equivalent (in this example) to

```
*PWEF IG ($) ($)
EXIT
TC
EXECUTE ATEST.F4)
```

4.2 REARRANGING A FILE

In Section 3.14, an example shows the use of a Q-register in moving a segment of text from one place on a page to another place on the same page. This section describes how to move blocks of text, or whole pages, to entirely different places in a file.

Example:

The user has a file named PGM.MAC on the disk and this file contains data in the following form:

AB (FORM) CD (FORM) EF (FORM) GH (FORM) IJ (FORM) KL (FORM) MN (FORM) OP

where each of the letters A, B, C... represents 20 lines of text.

The user intends to rearrange the file, as shown in the following example:

AOB (FORM) D (FORM) MN (FORM) EF (FORM) ICJ (FORM) KL (FORM) P (FORM) GH

The following commands achieve this rearrangement.

```
.R TECO 6)
_EBPGM.MAC ($) Y ($) ($)
_NC ($) ($)
*_J20X1 ($) ($)
*_20K ($) ($)
*_NG ($) ($)

*_HX2 ($) ($)
*_Y ($) ($)
*_20L ($) ($)

```

Call TECO with extra core.  
 Specify the file and get the first page.  
 Output AB (FORM) and input CD.  
 Save all of C in Q-register 1.  
 Delete C from its position in the editing buffer.  
 Output D (FORM) and input EF. Output EF (FORM) and input GH.  
 Save all of GH in Q-register 2.  
 Delete GH and input IJ.  
 Move the pointer to the beginning of J.

```

*G1 ($) ($)
*NM ($) ($)

*HX1 ($) ($)

*Y ($) ($)
*_J20X3 ($) ($)
*_20K ($) ($)
*_P ($) ($)

*_G2 ($) ($)

*_HPEF ($) ($)

*_EBPGM.MAC ($) Y ($) ($)

*_20L ($) ($)
*_G3 ($) ($)
*_ND ($) ($)
*_PWHK ($) ($)

*_G1 ($) ($)
*_EX ($) ($)

EXIT
1C
_

```

Bring in all of C from Q-register 1.  
Output ICJ (FORM) input KL, output KL (FORM), and input MN.  
Save all of MN in Q-register 1 (thereby discarding the previous contents).  
Delete MN and input OP.  
Save all of O in Q-register 3.  
Delete O from the editing buffer.  
Output P (FORM) and clear the editing buffer.  
Bring GH into the buffer from Q-register 2.  
Output GH, close the output file (now called nnnTEC.TMP), rename the input file PGM.BAK, and then rename the output file PGM.MAC.  
Now edit the partially revised file just output. Loop around to the beginning of the file.  
Move the pointer to the beginning of B.  
Bring in all of O from Q-register 3.  
Output AOB (FORM) and input D.  
Output D (FORM), and then clear the buffer.  
Bring in all of MN from Q-register 1.  
Output MN (FORM) and continue the input/output sequence until GH has been output. Then close the output file (called nnnTEC.TMP), delete the previous PGM.BAK, rename the input file PGM.BAK, and then rename the new output file PGM.MAC. Finally, exit to the monitor.

### 4.3 SPLITTING AND MERGING FILES

This section demonstrates the procedure to split a file into several smaller files and the procedure to merge parts of several files.

#### Example 1: Splitting a File

Assume the user has a file named FILE.CBL on the disk; this file contains data in the following form:

AB (FORM) CD (FORM) EF (FORM) GH (FORM) IJ (FORM) KL (FORM) MN (FORM) OP

where each of the letters A, B, C, ... represents 20 lines of text. The user wants to separate FILE.CBL into two files:

- a. FILE.1 containing AB (FORM) CD and
- b. FILE.2 containing KL (FORM) M

And to discard the rest of the data. To accomplish this proceed as follows.

|                                                                                                                                                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> _R TECO) *ERFILE.CBL (\$) EWFILE.1 (\$ \$) *_Y (\$ \$) *_P (\$ \$) *_HPEF (\$ \$) *←K (\$ \$)  *EWFILE.2 (\$ \$) *_P (\$ \$) *_20L (\$ \$) *_0,.PEF (\$ \$) * (tC) _ </pre> | <p>Call TECO.</p> <p>Open the input file and the first output file.</p> <p>Input AB.</p> <p>Output AB (FORM) and input CD.</p> <p>Output CD and then close the output file FILE.1.</p> <p>Clear the buffer (deleting CD from it) and continue inputting pages of the file and searching for K. If K is not found on a given page, clear the buffer, and read in the next page. The ← command does not perform output. Thus EF, GH, and IJ are all read in and then deleted. When KL is read in, the search stops.</p> <p>Open the second output file.</p> <p>Output KL (FORM) and input MN.</p> <p>Position pointer at the end of M.</p> <p>Output M and then close the output file FILE.2.</p> <p>Exit to the monitor with the job completed.</p> |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Example 2: Merging Files

Assumed the user has two files:

- a. MATH.BAK containing  
 AB (FORM) CD (FORM) EF (FORM) GH (FORM) IJ (FORM) KL
- b. MATH.F4 containing  
 A'B' (FORM) C'D' (FORM) E'F'

Where A, B, C, ... each represents 20 lines of text, and A', B', ... represent updated versions of A, B, ...

The user wants to merge MATH.F4 with the latter half of MATH.BAK to produce:

MATH.NEW containing

A'B (FORM) C'D' (FORM) E'F' (FORM) GH (FORM) IJ (FORM) KL

He proceeds as follows.

```

_R TECO
_ERMATH.F4 ($)
EWMATH.NEW ($ $)
*_Y ($ $)
*_NF' ($ $)
*_PW ($ $)
_ERMATH.BAK ($ $)
*_Y ($ $)
*_←G ($ $)
*_NL ($ $)
*_HPEF (G) ($ $)
EXIT
TC
.
```

Call TECO.

Open the first input file and the output file.

Input A'B'.

Output A'B' (FORM), input C'D',  
output C'D' (FORM), and input E'F'.  
Output E'F' (FORM).

Close input from MATH.F4, and open  
MATH.BAK for input.

Delete E'F' from the buffer and input  
AB.

Delete AB, input and delete CD and  
EF, then input GH.

Output GH (FORM), input and then  
output IJ (FORM), then input KL.

Output KL, close the output file  
MATH.NEW, and then exit to the  
monitor with the job completed.

The technique shown in Example 2 illustrates the best method for recovering from the error indicated by the error message:

```
?OUT-200000 Output Error 200000 - Output File 018TEC.TMP Closed
```

If this error occurs during an editing job initialized by the TECO filnam.ext command or an EB command, the incomplete output file has a temporary name of the form nnnTEC.TMP (see Section 3.2); otherwise, the incomplete output file will have the name specified by the user. (Refer to Appendix A for a list of error messages and their meanings.)

Example 3 is more explicit illustration of recovery from the foregoing error.

Example 3: Recovery from an Output Error

```

._TECO FIL.DOC)
*_edit a few pages ($) ($)
*_P ($) ($)
?OUT-200000 Output Error 200000 - Output File 018TEC.TMP Closed

*_ER018TEC.TMP ($) EWFIL.NEW ($) Y ($) ($)
*_Nlast page edited and successfully output ($) ($)
*_PW ($) ($)
*_ERFIL.DOC ($) Y ($) ($)
*_←last page edited and successfully output ($) ($)
*_Y and edit next page ($) ($)
*_Nnext place to edit ($) ($)
*_finish editing normally ($) ($)
*_EX ($) ($)
EXIT
!C
._RENAME FIL.DOC=FIL.NEW)

```

4.4 EXAMPLE OF AN ADVANCED TECO MACRO

This section demonstrates a TECO macro for formatting DECsystem-10 Macro assembly language programs.

The procedure for executing this macro is as follows:

|                                                                                                                                           |                                                                                                                                                                                                                                                                                                                                                               |
|-------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> ._R TECO 6) *_ERDTA7:PGMFMF.TEC (\$) (\$) *_YHX1 (\$) (\$) *_EBPROGRM.MAC (\$) (\$) *_Y (\$) (\$) *_M1 (\$) (\$) *_ (tC) _ _ </pre> | <p>Call TECO with enough core to cover the maximum page size.</p> <p>Open the file containing the macro itself for input.</p> <p>Input the macro and save it in Q-register 1.</p> <p>Open for editing the file that is to be formatted by the macro.</p> <p>Read in the first page of the file.</p> <p>Execute the macro.</p> <p>Exit with job completed.</p> |
|-------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Formatting Macro (PGMFMT.TEC)

```

1EO !START!OUL<S!
  ($) ;%L>ZJR1A-10''N%L'                                !COUNT LINES ON PAGE!
!LOOP!JQL<OUC   !EXECUTE LOOP ONCE FOR EACH LINE!
!FSTCH1!1A''COTAG ($) '
!FSTCH2!1A-9''ECOOP ($) '1A-32''NOZ ($) '
!FSTCH3!%C-8''GOZ ($) 'C1A-32''EOFSTCH3 ($) '1A-9''EQC-7''GOZ ($) 'COFSTCH4 ($) '
QC-8''GOZ ($) '
!FSTCH4!OUS   !CHANGE LEADING SPACES TO A TAB!
!FSTCH5!-D%S-QC''LOFSTCH5 ($) ' → ($) OOP ($)
!TAG!%C-6''GOZ ($) 'C1A''COTAG ($) '1A-58''NOZ ($) '
!COLON!OUSC1A-9''ECOOP ($) '1A-32''NOZ ($) '           !LOOK FOR A COLON!
!COLON2!%S ($) C1A-32''EOCOLON2 ($) 'QC+QS-7''GOZ ($) 'QC+QS-7''EOCOLON3
 ($) '1A-9''NOZ ($) 'D
!COLON3!R1A-32''EDOCOLON3 ($)
'C → ($)   !CHG SPACES AFTER COLON TO TAB!
!OP!1A-90''GOZ ($) '1A-65''LOZ ($) 'OUC
!OP2!%C ($) C1A-90''GOZ ($) '1A-64''GOOP2 ($) '1A-57''GOZ ($) '1A-47''GOOP2 ($) '
  1A-9''EC1A-32''EOZ ($) '1A-9''EOZ ($)           !GIVE UP IF NO OPERANDS!
  1A-32''NOZ ($) 'QC-7''GOZ ($) 'C1A-32''EOZ ($) '1A-9''EOZ ($) '
  -D → ($)   !IF A SINGLE SPACE FOLLOWS OP, CHANGE IT TO A TAB!
!OP3!OUC
!EOL!%C ($)   !LOOK FOR END OF LINE OR SEMI-COLON!
!EOL2!1A-9''EOEOL ($) '1A-13''G1A-59''NOEOL ($) 'OUS
!SEMI!R1A-32''N1A-9''NOSEMI2 ($) ' '%S ($)           !LINE UP COMMENTS!
!SEMI2!QS''NC → ($) QC-QS-8''L → ($) ' ' '
!Z!L>
PZ''NOSTART ($) 'OUC                                   !LOOK FOR NEXT PAGE!
!GET!YZ''NOSTART ($) '%C-10''NOGET ($) 'EF0EO !QUIT WHEN 10 YANKS YIELD NO DATA!

```

An explanation of the macro follows.

1EO

The 1EO command enables only those features found in versions prior to 21A for which this macro was written.

!START!

It is assumed that the pointer is at the beginning of the first page of the file.

OUL

Initialize line counter.

<S! (\$) ;%L>

Count the line feed characters on the page.

ZJR1A-10''N%L'

If the last character on the page is not a line feed, count those characters following the last line feed character as one more line.

ICOUNT LINES ON PAGE!

This is the standard technique for including comments in TECO macros.

JQL<

Execute everything which follows, down to the > character on the second to the last line, once for each line on the page.

0UC

Initialize first character counter for the line.

1A-90"GOZ (\$)'

If the first character in the line is greater than Z (decimal 90) in the ASCII set, skip this line by jumping to !Z!.

1A"COTAG (\$)'

If the first character is alphabetic or period, or %, or a dollar sign (i.e., legal as the first character of a Macro language symbol), go to !TAG!. Otherwise, go to !FSTCH2!.

!FSTCH2!1A-9"ECOOP (\$)'

If the first character is a tab, move the pointer past the tab, then go to !OP!.

1A-32"NOZ (\$)'

If the first character is a space, continue on to !FSTCH3!; otherwise, skip this line.

!FSTCH3!%C-8"GOZ (\$) 'C

Increment the character counter (counting leading spaces), and if the new total is more than eight spaces, skip to the next line; otherwise, move the pointer to the next character.

1A-32"EOFSTCH3 (\$)'

If the next character is another space, go back to !FSTCH3!.

1A-9"EQC-7"GOZ (\$)'

If the character is neither a tab nor a space, and if more than eight spaces preceded this character, skip to the next line. If the character is a tab, but more than seven spaces preceded this tab, skip to the next line. Otherwise, go to !FSTCH4!.

COFSTCH4 (\$) 'QC-8"GOZ (\$)'

Initialize space deleted counter.

!FSTCH4!0US

Delete last space seen.

!FSTCH5!-D

Increment space deleted counter. Then, if the new value of this counter is still less than the number of characters (spaces) counted on the line, go back to !FSTCH5!.

%S-QC"LOFSTCH5 (\$)'

When the count of spaces deleted reaches the number of spaces there were, insert a tab and then go to !OP!.

-> (\$) OOP (\$)'

!TAG!%C-6"GOZ (\$) 'C

Increment the character counter (counting characters in the tag), and if the new total is more than six spaces, skip to the next line. Otherwise, move the pointer to the next character.

1A"COTAG (\$)'

If the next character is a symbol constituent, go back to !TAG!.

1A-58"NOZ (\$) 'OCOLON (\$)'

If the character is a colon, go on to !COLON!; otherwise, skip to the next line.

!COLON!0USC

Initialize counter of spaces following the colon, and move the pointer to the next character.

1A-9''ECOOP (\$)'

If the character after the colon is a tab, move the pointer to the next character and go to !OP!.

1A-32''NOZ (\$)'

If the character is not a space either, skip to the next line. Otherwise, continue on to !COLON2!.

!COLON2!%S (\$) C

Increment the space-following-colon counter, and then move the pointer to the next character. The altmode following %S prevents the value returned by the %S command from being used as an argument for the following C command.

1A-32''EOCOLON2 (\$)'

If the next character is another space, go back to !COLON2!.

QC+QS-7''GOZ (\$)'

If the total count of the symbol characters before the colon and the spaces after the colon is more than seven, skip to the next line.

QC+QS-7''EOCOLON3 (\$)'

If the count mentioned above exactly equals seven, go to !COLON3!.

1A-9''NOZ (\$) 'D

With the count mentioned above less than seven, if the next character is not a tab, skip to the next line. If this character is a tab, delete it and continue to !COLON3!.

!COLON3!R

Move pointer back one character (i.e., back past the next space or the colon).

1A-32''EDOCOLON3 (\$)'

If the character passed over is a space delete it and go back to !COLON3!.

C → (\$) OOP (\$)'

Otherwise, the pointer is now in front of the colon. Move it forward over the colon and then insert a tab to replace the deleted spaces. Then go to !OP!.

!OP!1A-90''GOZ (\$)'

If the first character in the operator field is not alphabetic, skip to the next line. Otherwise, initialize the op field character counter.

1A-65''LOZ (\$) '0UC

!OP2!%C (\$) C

Increment operator field character counter and then move pointer to the next character.

1A-90''GOZ (\$)'

If the next character is above Z in the ASCII set, skip to the next line. If it is alphabetic, go back to !OP2!.

1A-64''GOOP2 (\$)'

1A-57''GOZ (\$)'

If the character is greater than the digit nine in the ASCII set, skip to the next line. If it is a digit, go back to !OP2!.

1A-47''GOOP2 (\$)'

1A-9''E

If the character is not a tab, skip to the ' following the comment ''GIVE UP IF NO OPERANDS''. The leading spaces are for appearance only and are ignored. (A tab could not be used for this purpose.)

C1A-32''EOZ (\$)'

If it is a tab, move the pointer to the next character. If this character is a tab or a space, skip to the next line. If the character is anything else, go to !OP3!.

1A-9''EOZ (\$) 'OOP3 (\$)'



0UC

!GET!YZ''NOSTART (\$)'

%C-10''NOGET (\$)'

EF

0EO

Otherwise, initialize the yank counter.

Try another yank. If this produces any new data, go back to !START!.

Increment the yank counter, and if it is still less than 10, try again.

When a total of 10 straight yanks after the P command fails to produce any new data, close the output file.

The 0EO command re-enables TECO commands to the current version.

## Chapter 5 User Errors

This chapter describes two types of errors: (1) typing errors discovered by the user before a command string is completed, and (2) command errors detected by TECO. The user should realize, however, that there is a third class of error. Because TECO interprets almost every character as a command, there can be cases where, if the user fails to notice a command string typing error, TECO executes a command that the user did not intend. For example, if the user meant to type the command

\*INAME  $\text{\textcircled{\$}}$   $\text{\textcircled{\$}}$

but forgot to type the "I", then TECO is forced to interpret the command as an N-search for "AME" and act accordingly. There is no way to protect the user from errors of this type.

### 5.1 ERASING COMMANDS

If the user makes an error while typing a command string and discovers the error before terminating the command string (with a double altmode), the error can be corrected using one of three erasing commands described below. All of these must be typed before the double altmode that terminates the command string.

#### 5.1.1 Rubout Command

Rubout is used to erase typed-in characters one at a time starting with the last character typed in.

Example

After typing the portion of the command string shown below, the user discovers that he has misspelled the name "Ericson".

\*3LKILEIF ERICXON

To nullify the error, he types three successive rubouts. As he does this, TECO responds by retyping the characters which are being rubbed out.

\*LKILEIF ERICXON  $\text{\textcircled{R}}$  N  $\text{\textcircled{R}}$  O  $\text{\textcircled{R}}$  X

The actual function of the rubout character is to delete the last typed character in the command string. Consequently, if the incorrect character is not the last in the string, all characters back to that point must also be rubbed out.

Rubout is a nonprinting character; consequently, the actual line appears as follows:

```
*LKILEIF ERICXONNOX
```

When the user has rubbed out the incorrect character, he continues the command string from the last correct character.

```
*3LKLEIF ERICXONNOXSON ($) OTT ($) ($)
```

Two successive rubouts are required to erase a carriage return and the monitor-generated line feed following it.

### 5.1.2 Double $\text{\textcircled{tG}}$ Command

The command  $\text{\textcircled{tG}}$   $\text{\textcircled{tG}}$  (two successive control-Gs) is used to erase an entire command string.

In the following example the user has decided, after typing the "N", to quit and start over. He does this by typing two successive control-Gs. (Control-G echoes visibly as "tG" and audibly as a bell ring.)

```
*3LKILEEF ERIXON  $\text{\textcircled{tG}}$   $\text{\textcircled{tG}}$ 
_
*
_
```

$\text{\textcircled{tG}}$   $\text{\textcircled{tG}}$  cannot be typed in the alternate up-arrow, character form described in Section 2.2.

### 5.1.3 $\text{\textcircled{tU}}$ Command

The  $\text{\textcircled{tU}}$  command is another erasing command available to the TECO user. The  $\text{\textcircled{tU}}$  command erases everything in a command string back to the last carriage-return/end-of-line character pair. It does not erase the carriage-return nor end-of-line character. The end-of-line characters are line feed, vertical tab and form feed.

In monitors previous to 5.02B, control-U is intercepted by the monitor and erases only back to the most recent break character (carriage-return, linefeed, formfeed, altmode).

Example 1:

```
*ILINE ONE)
LINE TWO)
LINE THREE)
KINE FOUR  $\text{\textcircled{tU}}$ 
LINE FOUR)
($) ($)
*
_
```

The user makes an error typing the fourth line and uses the  $\text{\textcircled{tU}}$  command to erase the entire line. The  $\text{\textcircled{tU}}$  command causes a carriage return-line feed to be echoed but the carriage return and line feed are not inserted.

Example 2:

```

*LINE ONE )
LINE TWO )
KINE THREE )
LINE FOUR (tU)
      (RO)
      (tU)
LINE THREE )
LINE FOUR )
($) ($)
*
-

```

The user makes an error on the third line but does not notice it until he is on the fourth line. In order to erase back to his error without erasing the entire command string, he types control-U, rubout, control-U. The first (tU) erases "LINE FOUR". The rubout erases the line feed that marks the end of the third line, and the second (tU) erases "LINE THREE" and the carriage-return at its end.

5.1.4 Bell-Space Command

The bell-space command is not actually an erasing command, but it is usually used in conjunction with the erasing commands. Its function is to cause the current line of the command string to be re-typed. It is used when the user has typed so many rubouts on a line that he cannot tell exactly what has been typed.

Specifically, if the user types (tG) and space in succession, everything in the command string back to, but not including, the last carriage return line feed pair is immediately retyped on the next line. The user may then continue the command string just as if bell-space had not been typed. The bell-space is not stored in the command string. Neither does it remove anything from the command string.

Example:

```

*ISTAET: (RO) : (RO) I (RO) ERT: ->TRZE ->SW, (tG)
START: TRZE SW, CCLFLG ->; CLEAR FLAG
($) ($)
*
-

```

5.2 ERROR MESSAGES

When TECO encounters an illegal command or a command that cannot be executed, an error message is printed on the user's terminal. An error message consists of three parts, some of which are printed automatically and some of which can be printed at the user's option. The first part of the message is a question mark followed by a 3-letter mnemonic code for the error message. The second is a brief, one-line, statement of the error condition. The last part is a more complete explanation of the error.

In the standard version of TECO the first two parts of the error message are automatically printed; the third part is printed only if the user requests it. In Section 5.2.2 there is an explanation of how to

obtain the optional parts of the error message, and in Section 5.2.3 there is an explanation of how to change TECO so that more or less of the error message is printed automatically.

When an error message is generated, the command to which it refers is not executed, the remainder of the command string is ignored, and TECO returns to command mode. Also any commands that the user might have typed ahead are erased.

Example:

```
*SWORD ($) -4DUINEW ($) ($)
?NAU No Argument Before U
*
-
```

The error message points out the presence of a U command not preceded by a numeric argument. The commands SWORD (\$) -4D have been executed, but the commands UINEW (\$) have not.

After an error message has been printed, the user has the option to use either or both of two special commands, ? and /, that are designed to help the user after a command error has been encountered. These commands are described below. Note, however, that these two commands have the special properties described below only immediately after an error has occurred. If any other command is typed after an error has occurred, TECO assumes that a new command string is being typed and the ability to use the ? and / commands for this error is lost.

Also note that the \*i command described in Section 3.8.8 is frequently useful after an error is encountered.

### 5.2.1 Question Mark Command

In some cases, the user may not be able to determine immediately which command in the string caused the error. This could occur, for example, if there were several commands of the same type in the command string. In such a case, the user can use the question mark command to obtain more information. The question mark command, when used immediately after an error message typeout, causes the offending command and several of the preceding characters in the command string to be typed out. A maximum of 10 characters of the command string are typed; usually this number is sufficient to identify the command that caused the error. Note that when the question mark command is used in this manner, it is not necessary to type altmode or any other character after the question mark.

A second question mark is always typed after the last character of the group. The character at which the error was detected is the last character before the second question mark typed.

Another use of the question mark command is explained in Section 3.17.

Example:

```

_H X 2PG2ZJ-1U2PG2ZJ ($) ($)
?NTQ No Text in Q-register 2
*? 2ZJ-1U2PG2?
*
_

```

According to the error message, one of the G2 commands specifies a Q-register that does not contain text. The question mark command is used and the second G2 command is identified as the offending command.

### 5.2.2 Slash Command

When a command error occurs, one or more of the three parts of the corresponding error message is automatically printed. If all three parts of the error message have not yet been printed and the user needs a more detailed explanation of the error, he may type the slash command to obtain more information.

The slash command, when used immediately after an error message, causes the next unprinted part of the error message to be printed. It may be used enough times to cause all three parts of the error message to be printed, but no more. Note that when the slash command is used in this manner, it is not necessary to type altmode or any other character after the slash.

#### NOTE

The verbal parts 2 and 3 of the error messages printed by TECO are obtained from a system file (TECO.ERR) external to TECO itself. If for any reason this file cannot be read, only the code portion of the error message is printed, and this is followed by the special message "?EEE Unable to Read Error Message File". In this case the / command cannot be used.

Another use of / is described in Section 2.7.2.

Example:

```

*EBTEST.CBL ($) EX ($) ($)
?BAK Cannot Delete Old Backup File
*/ Failure in rename process at close of editing job
initiated by an EB command or a TECO command. There
exists an old backup file TEST.BAK with a protection
<???> such that it cannot be deleted. Hence the in-
put file TEST.CBL cannot be renamed to "TEST.BAK".
The output file is closed with the filename "009TEC.TMP".
The RENAME UO error code is 2.

```

### 5.2.3 EH Command

As was stated above, TECO error messages consist of three parts. The first, or code, part is always automatically typed. With the standard version of TECO, the second, brief message, part is also automatically typed. The third, more lengthy part is obtained by the / command at the option of the user.

By use of the EH command, the user may change TECO so that more or less of the error message is automatically typed. This is done as follows:

- 1EH sets TECO so that only the code part of the error message is automatically printed.
- 2EH sets TECO so that both the code and the 1-line message parts of the message are automatically printed.
- 3EH sets TECO so that all three parts of the error message are always automatically typed.
- 0EH resets TECO to the system standard mode of error message typeout. (Normally equivalent to 2EH.)
- EH (with no argument) returns the value of the current EH setting.

## Appendix A TECO Error Messages

The following table lists the error messages from TECO. The three-letter message preceded by a question mark is always typed; the second part of the error message, which is a short explanation of the error, is always typed in standard versions of TECO. The detailed message is typed if the user types a slash command (/) immediately following the short error message.

Table A-1  
TECO Error Messages

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ?ARG | Improper Arguments<br>The following argument combinations are illegal:<br>1) , (no argument before comma)<br>2) m,n, (where m and n are numeric terms)<br>3) H, (because H=B,Z is already two arguments)<br>4) ,H (H following other arguments)                                                                                                                                                                                                        |
| ?BAK | Cannot Delete Old Backup File<br>Failure in rename process at close of editing job initiated by an EB command or a TECO command. There exists an old backup file filnam.BAK with a protection<nnn> such that it cannot be deleted. Hence the input file filnam.ext cannot be renamed to "filnam.BAK". The output file is closed with the filename "nnnTEC.TEMP", where nnn is the user's job number. The RENAME UOO error code is nn.                  |
| ?COR | Storage Capacity Exceeded<br>The current operation requires more memory storage than TECO now has and TECO is unable to obtain more core from the monitor. This message can occur as a result of any one of the following things:<br>1) command buffer overflow while a long command string is being typed,<br>2) Q-register buffer overflow caused by an X or [ command,<br>3) editing buffer overflow caused by an insert command or a read command. |
| ?COS | Contradictory Output Switches<br>The GENLSN and SUPLSN switches may not both be used with the same output file.                                                                                                                                                                                                                                                                                                                                        |
| ?EBD | EB with Device dev Is Illegal<br>The EB command and the TECO command may be specified only with file structured devices, i.e., disk and DECTape.                                                                                                                                                                                                                                                                                                       |

Table A-1 (Cont)  
TECO Error Messages

|         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ?EBF    | <p>EB with Illegal File filnam.ext<br/>The EB command and the TECO command may not be used with a file having the filename extension ".BAK" or with a file having the name "nnnTEC.TMP". Where nnn is the user's job number, the user must either use an ER-EW sequence, or rename the file.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| ?EBO    | <p>EB, EW, or EZ Before Current EB Job Closed<br/>After an output file has been opened by a TECO command or an EB command, no further EB, EW, or EZ commands may be given until the current output file is closed.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| ?EBP    | <p>EB Illegal Because of File filnam.ext Protection<br/>The file filnam.ext cannot be edited with an EB command or a TECO command because it has a protection &lt;nnn&gt; such that it cannot be renamed at close time.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| ?EEE    | <p>Unable to Read Error Message File<br/>An error, whose code was typed previous to this error message, has occurred, and while TECO was trying to find the proper error message in the error message file, one of the following errors occurred:</p> <ol style="list-style-type: none"> <li>1) the error message file, TECO.ERR, could not be found on device SYS;</li> <li>2) an input error occurred while TECO was reading the file TECO.ERR,</li> <li>3) the error message corresponding to that error code is missing from TECO.ERR,</li> <li>4) the user's TECO job does not currently have enough room for a buffer to read the error message file into, and no more core can be obtained from the monitor,</li> <li>5) for some strange reason device SYS: could not be initialized for input.</li> </ol> |
| ?EMA    | <p>EM with Illegal Argument nn<br/>The argument n in an nEM command must be greater than zero.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| ?EMD    | <p>EM with No Input Device Open<br/>EM commands apply only to the input device, and so should be preceded by an ER (or equivalent) command. To position a tape for output, that unit should be temporarily opened for input while doing the EM commands.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| ?ENT-00 | <p>Illegal Output Filename "filnam.ext"<br/>ENTER UUO failure 0. The filename "filnam.ext" specified for the output file cannot be used. The format is invalid.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| -01     | <p>Output UFD dev:[pj,pg] Not Found<br/>ENTER UUO failure 1. The file filnam.ext[pj,pg] specified for output by an EW, EZ, or MAKE command cannot be created because there is no user file directory with project-programmer number [pj,pg] on device dev.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| -02     | <p>Output Protection Failure<br/>ENTER UUO failure 2. The file filnam.ext[pj,pg] specified for output by an EW, EZ, EB, MAKE, or TECO command cannot be created either because it already exists and is write-protected &lt;nnn&gt; against the user, or because the UFD it is to be entered into is write-protected against the user.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

Table A-1 (Cont)  
TECO Error Messages

|     |                                                                                                                                                                                                                                                                                                                                        |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -03 | Output File Being Modified<br>ENTER UUO failure 3. The file filnam.ext specified for output by an EW, EZ, EB, MAKE, or TECO command cannot be created because it is current being created or modified by another job.                                                                                                                  |
| -06 | Output UFD or RIB Error<br>ENTER UUO failure 6. The output file filnam.ext cannot be created because a bad directory block was encountered by the monitor while the ENTER was in progress. The user may try repeating the EW, EB, or TECO command, but if the error persists, it is impossible to proceed. Notify your system manager. |
| -14 | No Room or Quota Exceeded on dev:<br>ENTER UUO failure 14. The output file filnam.ext cannot be created because there is no more free space on device dev:, or because the user's quota is already exceeded there.                                                                                                                     |
| -15 | Write Lock on dev:<br>ENTER UUO failure 15. The output file filnam.ext cannot be created because the output file structure is write-locked.                                                                                                                                                                                            |
| -16 | Monitor Table Space Exhausted<br>ENTER UUO failure 16. The output file filnam.ext cannot be created because there is not enough table space left in the monitor to allow the ENTER. The user may try repeating the EW, EB, or TECO command, but if the error persists he will have to wait until conditions improve.                   |
| -23 | Output SFD Not Found<br>ENTER UUO failure 23. The output file filnam.ext cannot be created because the sub-file-directory on which it should be ENTERed cannot be found.                                                                                                                                                               |
| -24 | Search List Empty<br>ENTER UUO failure 24. The output file filnam.ext cannot be created because the user's file structure search list is empty.                                                                                                                                                                                        |
| -25 | Output SFD Nested too Deeply<br>ENTER UUO failure 25. The output file filnam.ext cannot be created because the specified SFD path for the ENTER is nested too deeply.                                                                                                                                                                  |
| -26 | No Create for Specified SFD Path<br>ENTER UUO failure 26. The output file filnam.ext cannot be created because the specified SFD path for the ENTER is set for no creation.                                                                                                                                                            |
| -nn | ENTER Failure nn on Output File filnam.ext<br>The attempted ENTER of the output file filnam.ext has failed and the monitor has returned an error code of nn. This error is not expected to occur on an ENTER. Please send the TTY printout showing what you are doing to DEC with an SPR form.                                         |

Table A-1 (Cont)  
TECO Error Messages

|         |                                                                                                                                                                                                                                                                                                                                   |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ?EOA    | <p>nEO Argument Too Large<br/>The argument n given with an EO command is larger than the standard (maximum) setting in EO=n for this version of TECO. This must be an older version of TECO than the user thinks he is using; the features corresponding to EO=n do not exist.</p>                                                |
| ?FNF-00 | <p>Input File filnam.ext Not Found<br/>LOOKUP UUU failure 0. The file filnam.ext specified for input by an ER, EB, or TECO command was not found on the input device dev.</p>                                                                                                                                                     |
| -01     | <p>Input UFD dev:[pj,pg] Not Found<br/>LOOKUP UUU failure 1. The file filnam.ext [pj,pg] specified for input by an ER, EB, or TECO command cannot be found because there is no User File Directory with project-programmer number [pj,pg] on device dev.</p>                                                                      |
| -02     | <p>Input Protection Failure<br/>LOOKUP UUU failure 2. The file filnam.ext [pj,pg] specified for input by an ER, EB, or TECO command cannot be read because it is read-protected &lt;nnn&gt; against the user.</p>                                                                                                                 |
| -06     | <p>Input UFD or RIB Error<br/>LOOKUP UUU failure 6. The input file filnam.ext cannot be read because a bad directory block was encountered by the monitor while the LOOKUP was in progress. The user may try repeating the ER, EB, or TECO command, but if the error persists all is lost. Notify your system manager.</p>        |
| -16     | <p>Monitor Table Space Exhausted<br/>LOOKUP UUU failure 16. The input file filnam.ext cannot be read because there is not enough table space left in the monitor to allow the LOOKUP. The user may try repeating the ER, EB, or TECO command, but if the error persists he will have to wait until system conditions improve.</p> |
| -23     | <p>Input SFD not Found<br/>LOOKUP UUU failure 23. The input file filnam.ext cannot be found because the sub-file-directory on which it should be looked up cannot be found.</p>                                                                                                                                                   |
| -24     | <p>Search List Empty<br/>LOOKUP UUU failure 24. The input file filnam.ext cannot be found because the user's file structure search list is empty.</p>                                                                                                                                                                             |
| -25     | <p>Input SFD Nested too Deeply<br/>LOOKUP UUU failure 25. The input file filnam.ext cannot be found because the specified SFD path for the LOOKUP is nested too deeply.</p>                                                                                                                                                       |
| -nn     | <p>LOOKUP Failure nn on Input File filnam.ext<br/>The attempted LOOKUP on the Input file filnam.ext has failed and the monitor has returned an error code of nn. This error is not expected to occur on a LOOKUP. Please send the TTY printout showing what you were doing to DEC with an SPR form.</p>                           |

Table A-1 (Cont)  
TECO Error Messages

|      |                                                                                                                                                                                                                                                                                                                                                                                 |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ?FUL | Device dev: Directory Full<br>ENTER UJO failure n. The file filnam.ext specified for output by an EW or MAKE command cannot be created on DECTape dev because the tape directory is full.                                                                                                                                                                                       |
| ?IAB | Incomplete <...> or (...) in Macro<br>A macro contained in a Q-register and being executed by an M command contains an iteration that is not closed within the Q-register by a >, or a parenthetical expression that is not closed within the Q-register by a ).                                                                                                                |
| ?ICE | Illegal Control-E Command in Search Argument<br>A search argument contains a (tE) command that is either not defined or incomplete.                                                                                                                                                                                                                                             |
| ?ICT | Illegal Control Command †<char> in text Argument<br>In order to be entered as text in an Insert command or search command, all control characters (†@ - †H and †N - †←) must be preceded by †R or †T. Otherwise they are interpreted as commands. The control character "†<char>" is an undefined text argument control command.                                                |
| ?IDV | Input Device dev Not Available<br>Initialization failure. Unable to initialize the device dev for input. Either the device is being used by someone else right now, or else it does not exist in the system.                                                                                                                                                                    |
| ?IEC | Illegal Character "<char>" After E<br>The only commands starting with the letter E are EB, EF, EG, EH, EM, EO, ER, ET, EU, EW, and EZ. When used as a command (i.e., not in a text argument) E may not be followed by any character except one of these.                                                                                                                        |
| ?IEM | Re-Init Failure on Device dev After EM<br>Unable to re-initialize the device dev after executing an EM command on it. If this error persists after retrying to initialize the device with an ER command (or EW command if output to the device is desired), consult your system manager.                                                                                        |
| ?IFC | Illegal Character "<char>" after F<br>The only commands starting with the letter F are FS and FN. When used as a command (other than EF or in a text argument) F may not be followed by any character other than one of these.                                                                                                                                                  |
| ?IFN | Illegal Character "<char>" in Filename<br>File specifications must be of the form dev:filnam.ext[m,n] (\$) where dev, filnam, and ext are alphanumeric, and m and n are numeric. No characters other than the ones specified may appear between the EB, ER, EW, or EZ command and the altmode terminator(\$).                                                                   |
| ?ILL | Illegal Command <char><br>The character "<char>" is not defined as a valid TECO command.                                                                                                                                                                                                                                                                                        |
| ?ILR | Cannot Lookup Input File filnam.ext. to Rename It<br>Failure in rename process at close of editing job initiated by an EB command or a TECO command. Unable to do a LOOKUP on the original input file dev:filnam.ext in order to rename it "filnam.BAK". The output file is closed with the name "nnnTEC.TMP", where nnn is the user's job number. The LOOKUP error code is nn. |

Table A-1 (Cont)  
TECO Error Messages

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ?INP-nn0000 | <p>Input Error nn0000 on File filnam.ext.<br/>A read error has occurred during input. The input file filnam.ext has been released. The user may try again to read the file, but if the error persists, the user will have to return to his backup file. The input device status word error flags are nn0000. (Note: This number represents the I/O status word (rh) with bits 22-35 masked out.)</p> <p>(040000 -- block too large).<br/>(100000 -- parity or checksum error).<br/>(140000 -- block too large and parity error).<br/>(200000 -- device error, data missed).<br/>(240000 -- block too large and device error).<br/>(300000 -- parity error and device error).<br/>(340000 -- block too large, parity error, and device error).<br/>(400000 -- improper mode).<br/>(440000 -- block too large and improper mode).<br/>(500000 -- parity error and improper mode).<br/>(540000 -- block too large, parity error, and improper mode).<br/>(600000 -- device error and improper mode).<br/>(640000 -- block too large, device error, and improper mode).<br/>(700000 -- parity error, device error, and improper mode).<br/>(740000 -- block too large, parity error, device error, and improper mode).</p> |
| ?IOS        | <p>Illegal Character "&lt;char&gt;" in I/O Switch<br/>The only valid characters in switches used with file selection commands are the alphabetic characters.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| ?IQC        | <p>Illegal command "&lt;char&gt;"<br/>The only valid " commands are "G, "L, "N, "E, "C, "A, "D, "V, "W, "T, "F, "S, and "U.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| ?IQN        | <p>Illegal Q-register Name "&lt;char&gt;"<br/>The Q-register name specified by a Q, U, X, G, %, M, [, ], or * command must be a letter (A thru Z) or a digit (0 thru 9).</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| ?IRB        | <p>Cannot Rename Input File filnam.ext to filnam.BAK<br/>Failure in rename process at close of editing job initiated by an EB command or a TECO command. The attempt to rename the original input file filnam.ext to the backup filename "filnam.BAK" has failed. The output file is closed with the name "nnnTEC.TMP", where nnn is the user's job number. The RENAME UO error code is nn.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| ?IRN        | <p>Cannot RE-Init Device dev for Rename Process<br/>Failure in rename process at close of editing job initiated by an EB command or a TECO command. Cannot reinitialize the original input device dev in order to rename the input file filnam.ext to "filnam.BAK". The output file is closed with the name "nnnTEC.TMP", where nnn is the user's job number.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| ?ISA        | <p>n Argument with Search Command<br/>The argument preceding a search command indicates the number of times a match must be found before the search is considered successful. This argument must be greater than 0.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

Table A-1 (Cont)  
TECO Error Messages

|      |                                                                                                                                                                                                                                                                                                                                                                                       |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ?MAP | <p>Missing '<br/>In attempting to execute a conditional skip command (a '' command whose argument does not satisfy the required condition) no ' command closing the conditional execution string can be found. Note: n''...' strings must be complete within a single macro level.</p>                                                                                                |
| ?MEE | <p>Macro Ending with E<br/>A command macro being executed from a Q-register ends with the character ''E''. This is an incomplete command. E is the initial character of an entire set of commands. The other character of the command begun by E must be in the same macro with the E.</p>                                                                                            |
| ?MEF | <p>Macro Ending with F<br/>A command macro being executed from a Q-register ends with the character ''F'' (not an EF). This is an incomplete command. F is the initial character of an entire set of commands. The other character of the command begun by F must be in the same macro with the F.</p>                                                                                |
| ?MEO | <p>Macro Ending with Unterminated O Command<br/>The last command in a command macro being executed from a Q-register is an O command with no altmode to mark the end of the tag-name argument. The argument for the O command must be complete within the Q-register.</p>                                                                                                             |
| ?MEQ | <p>Macro Ending with ''<br/>A command macro being executed from a Q-register ends with the '' character. This is an incomplete command. The '' command must be followed by one of the characters G, L, N, E, C, A, D, V, W, T, F, S, or U to indicate the condition under which the following commands are to be executed. This character must be in the Q-register with the '' .</p> |
| ?MEU | <p>Macro Ending with †<br/>A command macro being executed from a Q-register ends with the † character. This is an incomplete command. The † command takes a single character text argument that must be in the Q-register with the † .</p>                                                                                                                                            |
| ?MIQ | <p>Macro Ending with &lt;char&gt;<br/>A command macro being executed from a Q-register ends with the character ''&lt;char&gt;''. This is an incomplete command. The &lt;char&gt; command takes a single character text argument to name the Q-register to which it applies. This argument must be in the same macro as the &lt;char&gt; command itself.</p>                           |
| ?MLA | <p>Missing &lt;<br/>There is a right angle bracket not matched by a left angle bracket somewhere to its left. (Note: an iteration in a macro stored in a Q-register must be complete within the Q-register.)</p>                                                                                                                                                                      |
| ?MLP | <p>Missing (<br/>Command string contains a right parenthesis that is not matched by a corresponding left parenthesis.</p>                                                                                                                                                                                                                                                             |

Table A-1 (Cont)  
TECO Error Messages

|      |                                                                                                                                                                                                                                                          |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ?MRA | Missing ><br>In attempting to exit from an iteration field with a ; command (or to skip over an iteration field with a 0 argument) no > command closing the iteration can be found. Note: iteration fields must be complete within a single macro level. |
| ?MRP | Missing )<br>The command string contains, within an iteration field, a parenthetical expression that is not closed by a right parenthesis.                                                                                                               |
| ?MUU | Macro Ending with ††<br>A command macro being executed from a Q-register ends with control-† or ††. This is an incomplete command. The †† command takes a single character text argument that must be in the Q-register with the ††.                     |
| ?NAE | No Argument Before =<br>The command n= or n== causes the value n to be typed. The = command must be preceded by either a specific numeric argument or a command that returns a numeric value.                                                            |
| ?NAI | No Altmode after nI<br>Unless the EO value has been set to 1, the numeric insert command nI must be immediately followed by altmode.                                                                                                                     |
| ?NAQ | No Argument Before ''<br>The '' command must be preceded by a single numeric argument on which the decision to execute the following commands or skip to the matching ' is based.                                                                        |
| ?NAU | No Argument Before U<br>The command nUi stores the value n in Q-register i. The U command must be preceded by either a specific numeric argument or a command that returns a numeric value.                                                              |
| ?NCS | No Command String Seen Prior to *i<br>The *i command saves the preceding command string in Q-register i. In this case no command string has previously been given.                                                                                       |
| ?NFI | No File for Input<br>Before issuing an input command (Y or A) it is necessary to open an input file by use of an ER, EB, or TECO command.                                                                                                                |
| ?NFO | No File for Output<br>Before giving an output command (PW, P, N, EX, or EG) it is necessary to open an output file by use of an EB, EW, EZ, MAKE, or TECO command.                                                                                       |
| ?NTQ | No Text in Q-register x<br>Q-register x, specified by a G or M command, does not contain text.                                                                                                                                                           |
| ?OCT | "8" or "9" in Octal Digit String<br>In a digit string preceded by 10, only the octal digits 0-7 may be used.                                                                                                                                             |

Table A-1 (Cont)  
TECO Error Messages

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ?ODV        | Output Device dev Not Available<br>Initialization failure. Unable to initialize the device dev for output. Either the device is being used by someone else right now, or it is write locked, or else it does not exist in the system.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| ?OLR        | Cannot Lookup Output File dev:filnam.ext to Rename It<br>Failure in rename process at close of editing job initiated by an EB command or a TECO command. The special LOOKUP on the output file filnam.ext required for DECtape in order to rename the file to "filnam.ext" has failed. The original input file filnam.ext has been renamed "filnam.BAK", but the output file is closed with the name "nnnTEC.TMP", where nnn is the user's job number. The LOOKUP UUU error code is nn.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| ?OUT-nn0000 | Output Error nn0000 - Output File nnnTEC. TMP Closed<br>An error on the output device is fatal. The output file is closed at the end of the last data that was successfully output. It has the filename "nnnTEC.TMP", where nnn is the user's job number. See Section 4.3 for a recovery technique. The output device status word error flags are nn0000. (Note: This number represents the I/O status word (rh) with bits 22-35 masked out.)<br>(000000 -- end of tape).<br>(040000 -- block number too large: device full or quota exceeded).<br>(100000 -- parity or checksum error).<br>(140000 -- block number too large and parity error).<br>(200000 -- device error, data missed).<br>(240000 -- block number too large and device error).<br>(300000 -- parity error and device error).<br>(340000 -- block number too large, parity error, and device error).<br>(400000 -- improper mode or device write locked).<br>(440000 -- block number too large and improper mode).<br>(500000 -- parity error and improper mode).<br>(540000 -- block number too large, parity error, and improper mode).<br>(600000 -- device error and improper mode).<br>(640000 -- block number too large, device error, and improper mode).<br>(700000 -- parity error, device error, and improper mode).<br>(740000 -- block number too large, parity error, device error, and improper mode). |
| ?PAR        | Confused Use of Parentheses<br>A string of the form (...<...) has been encountered. Parentheses should be used only to enclose combinations of numeric arguments. An iteration may not be opened and not closed between a left and right parenthesis.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| ?POP        | Attempt to Move Pointer Off Page with J, C, R, or D<br>The argument specified with a J, C, R, or D command must point to a position within the current size of the buffer, i.e., between 0 and Z, inclusive.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

Table A-1 (Cont)  
TECO Error Messages

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ?PPN | <p>Illegal Character "&lt;char&gt;" in Project-programmer Number</p> <p>Project-programmer numbers in file specifications must be given in the form [m,n] where m and n are octal digit strings separated by a comma. No characters other than the ones specified may appear between the enclosing brackets.</p>                                                                                                                                                                                                                                        |
| ?RNO | <p>Cannot Rename Output File nnnTEC.TMP</p> <p>Failure in rename process at close of editing job initiated by an EB command or a TECO command. The attempt to rename the output file nnnTEC.TMP to the name "filnam.ext" originally specified in the EB or TECO command has failed. The original input file filnam.ext has been renamed "filnam.BAK", but the output file is closed with the name "nnnTEC.TMP", where nnn is the user's job number. The RENAME UJO error code is nn.</p>                                                                |
| ?SAL | <p>Second Argument Less Than First</p> <p>In a two-argument command, the first argument must be less than or equal to the second.</p>                                                                                                                                                                                                                                                                                                                                                                                                                   |
| ?SNA | <p>Initial Search with No Argument</p> <p>A search command with null argument has been given, but there was no preceding search command from which the argument could be taken.</p>                                                                                                                                                                                                                                                                                                                                                                     |
| ?SNI | <p>; Not in an Iteration</p> <p>The semicolon command may be used only with a string of commands enclosed by angle brackets, i.e., in an iteration field.</p>                                                                                                                                                                                                                                                                                                                                                                                           |
| ?SRH | <p>Cannot Find "&lt;text&gt;"</p> <p>A search command not preceded by a colon modifier and not within an iteration has failed to find the specified character string "&lt;text&gt;". After an S search fails the pointer is left positioned at the beginning of the buffer. After an N or ← search fails the last page of the input file has been input and, in the case of N, output, and the buffer cleared. Note that when this message occurs, the text string printed includes all control-character commands included in the search argument.</p> |
| ?STC | <p>Search String Too Long</p> <p>The maximum length of a search string is 80 characters including all string control commands and their arguments.</p>                                                                                                                                                                                                                                                                                                                                                                                                  |
| ?STL | <p>Search String too Long</p> <p>The maximum length of a search string is 36 character positions, not counting extra characters required to specify a single position.</p>                                                                                                                                                                                                                                                                                                                                                                              |
| ?TAG | <p>Missing Tag !xxx!</p> <p>The tag !xxx! specified by an O command cannot be found. This tag must be in the same macro level as the O command referencing it.</p>                                                                                                                                                                                                                                                                                                                                                                                      |
| ?TAL | <p>Two Arguments with L</p> <p>The L command takes at most one numeric argument, namely, the number of lines over which the buffer pointer is to be moved.</p>                                                                                                                                                                                                                                                                                                                                                                                          |

Table A-1 (Cont)  
TECO Error Messages

|      |                                                                                                                                                                                                                                                                                                                      |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ?TTY | <p>Illegal TTY I-O Device<br/>A terminal may be specified as an input-output device in an ER, EW, EZ, or MAKE command only if it is not being used to control an attached job, the user's own terminal included.</p>                                                                                                 |
| ?UCA | <p>Unterminated tA Command<br/>A tA message type-out command has been given, but there is no corresponding tA to mark the end of the message. tA commands must be complete within a single command level.</p>                                                                                                        |
| ?UFS | <p>Macro Ending with Unterminated File Selection Command<br/>The last command in a command macro being executed from a Q-register is a file selection command (ER, EW, EB, or EZ) with no altmode to mark the end of the file specifications. The file selection command must be complete within the Q-register.</p> |
| ?UIN | <p>Unterminated Insert Command<br/>An insert command (possibly an @ insert command) has been given without terminating the text argument at the same macro level.</p>                                                                                                                                                |
| ?UIS | <p>Undefined I/O Switch "/xxx"<br/>The switch "/xxx" is not defined with either input or output file selection commands. The only switches currently defined for input or output file selection commands are /GENLSN and /SUPLSN.</p>                                                                                |
| ?USR | <p>Unterminated Search Command<br/>A search command (possibly an @ search command) has been given without terminating the text argument at the same macro level.</p>                                                                                                                                                 |
| ?UTG | <p>Unterminated Tag<br/>A command string tag has been indicated by a ! command, but there is no corresponding ! to mark the end of the tag. Tags must be complete within a single command level.</p>                                                                                                                 |
| ?UOO | <p>Illegal UOO<br/>Internal error. The illegal instruction &lt;lh,rh&gt; has been encountered at address nnnnnn. This is caused by either a TECO bug or a monitor bug. Please give printout to your system manager, or submit it to DEC with an SPR.</p>                                                             |

TECO

- 336 -

## Appendix B ASCII Characters

Table B-1  
ASCII Characters

| Character               | Manual Symbol | Octal | Decimal | Comment and Section Reference                                                                                                                                                                |
|-------------------------|---------------|-------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Null or Control-Shift-P | ⓐ             | 000   | 0       | Ignored on input. Ignored on type-in. nlⓐinsert only.                                                                                                                                        |
| Control-A               | ⓐ             | 001   | 1       | TECO command (Section 3.17).                                                                                                                                                                 |
| Control-B               | ⓑ             | 002   | 2       | Monitor command (Section 3.18). A special character (Section 2.2).                                                                                                                           |
| Control-C               | ⓒ             | 003   | 3       | Monitor command (Section 3.10). A special character (Section 2.2). nlⓒinsert only. Echoes as ⓐ-carriage return-line feed.                                                                    |
| Control-D               | ⓓ             | 004   | 4       | TECO command (Section 3.17).                                                                                                                                                                 |
| Control-E               | ⓔ             | 005   | 5       | TECO command (Sections 3.11 and 3.16).                                                                                                                                                       |
| Control-F               | ⓕ             | 006   | 6       | TECO command (Section 3.16). Monitor command (Section 3.18). A special character (Section 2.2).                                                                                              |
| Bell                    | ⓖ             | 007   | 7       | Echoes and prints as a single bell ring and ⓖ. Double ⓖ and ⓖ␣are TECO commands (Section 5.1) and special characters (Section 2.2).                                                          |
| Backspace               | ⓗ             | 010   | 8       | TECO command (Section 3.16). Prints as ⓗ.                                                                                                                                                    |
| Tab                     | ⓓ             | 011   | 9       | TECO command (Section 3.8).                                                                                                                                                                  |
| Line Feed               | ⓓ             | 012   | 10      | Ignored in command strings except as a text argument (Section 3.18). The symbol ⓓ is used only to represent an explicitly-typed line feed. It is not used for the line feed that the monitor |

Table B-1 (Cont)  
ASCII Characters

| Character        | Manual Symbol | Octal | Decimal | Comment and Section Reference                                                                                                                                    |
|------------------|---------------|-------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Line Feed (Cont) |               |       |         | generates when a carriage return is typed. In data, line feed defines the end of a line (Section 2.3).                                                           |
| Vertical Tab     | Ⓥ             | 013   | 11      | In data, vertical tab defines the end of a line (Section 2.3).                                                                                                   |
| Form Feed        | Ⓣ             | 014   | 12      | TECO command (Section 3.6). In data, form feed defines the end of a page (Section 2.3).                                                                          |
| Carriage Return  | ↵             | 015   | 13      | Ignored in command strings except as a text argument. (Section 3.18). When this character is typed the monitor automatically generates a line feed following it. |
| Control-N        | Ⓝ             | 016   | 14      | TECO command (Section 3.11).                                                                                                                                     |
| Control-O        | Ⓞ             | 017   | 15      | Monitor command (Section 3.6). A special character (Section 2.2). nlⓄ insert only. Echoes as Ⓞ-carriage return-line feed.                                        |
| Control-P        | Ⓟ             | 020   | 16      | Monitor command (Section 3.18). A special character (Section 2.2).                                                                                               |
| Control-Q        | Ⓠ             | 021   | 17      | TECO command (Section 3.11).                                                                                                                                     |
| Control-R        | Ⓡ             | 022   | 18      | TECO command (Sections 3.8 and 3.11).                                                                                                                            |
| Control-S        | Ⓢ             | 023   | 19      | TECO command (Section 3.11).                                                                                                                                     |
| Control-T        | Ⓣ             | 024   | 20      | Two different uses as TECO commands (Sections 3.8, 3.11, 3.16).                                                                                                  |
| Control-U        | Ⓤ             | 025   | 21      | TECO command (Section 5.1). A special character (Section 2.2). nlⓊ insert only. Echoes as Ⓤ carriage return-line feed.                                           |
| Control-V        | Ⓥ             | 026   | 22      | TECO command (Sections 3.8 and 3.11).                                                                                                                            |
| Control-W        | Ⓦ             | 027   | 23      | TECO command (Sections 3.8 and 3.11).                                                                                                                            |
| Control-X        | Ⓧ             | 030   | 24      | Two different uses as TECO commands (Section 3.11).                                                                                                              |
| Control-Y        | Ⓨ             | 031   | 25      |                                                                                                                                                                  |
| Control-Z        | Ⓩ             | 032   | 26      | TECO command (Section 3.10). Echoes as Ⓩ-carriage return-line feed. Used as end-of-file signal when doing data input from a TTY.                                 |

Table B-1 (Cont)  
ASCII Characters

| Character                       | Manual Symbol | Octal | Decimal | Comment and Section Reference                                                                                                                                                            |
|---------------------------------|---------------|-------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Altmode or<br>(Control-Shift-K) | Ⓢ             | 033   | 27      | Alphanumeric argument terminator (Section 2.7). A special character (Section 2.2). Echoes and prints as Ⓢ. Two successive altmodes are used to terminate a command string (Section 2.6). |
| Control-Shift-L                 | Ⓛ             | 034   | 28      | TECO command (Section 3.11).                                                                                                                                                             |
| Control-Shift-M                 | Ⓜ             | 035   | 29      |                                                                                                                                                                                          |
| Control-Shift-N                 | Ⓝ             | 036   | 30      | Two different uses as TECO commands (Sections 3.8, 3.11, 3.16).                                                                                                                          |
| Control-Shift-O                 | Ⓞ             | 037   | 31      |                                                                                                                                                                                          |
| Space                           | ␣             | 040   | 32      | TECO command (Section 2.7). Ignored in command strings except as a text argument or when used instead of + with numeric arguments (Section 3.18).                                        |
| !                               |               | 041   | 33      | TECO command (Section 3.13).                                                                                                                                                             |
| "                               |               | 042   | 34      | Used as a prefix for a whole class of TECO commands (Section 3.13).                                                                                                                      |
| #                               |               | 043   | 35      | TECO command (Section 2.7).                                                                                                                                                              |
| \$                              |               | 044   | 36      |                                                                                                                                                                                          |
| %                               |               | 045   | 37      | TECO command (Section 3.14).                                                                                                                                                             |
| &                               |               | 046   | 38      | TECO command (Section 2.7).                                                                                                                                                              |
| '                               |               | 047   | 39      | TECO command (Section 3.13).                                                                                                                                                             |
| (                               |               | 050   | 40      | TECO command (Section 2.7).                                                                                                                                                              |
| )                               |               | 051   | 41      | TECO command (Section 2.7).                                                                                                                                                              |
| *                               |               | 052   | 42      | Two different uses as TECO commands (Sections 2.7 and 2.14).                                                                                                                             |
| +                               |               | 053   | 43      | TECO command (Section 2.7).                                                                                                                                                              |
| ,                               |               | 054   | 44      | TECO command (Section 2.7).                                                                                                                                                              |
| -                               |               | 055   | 45      | TECO command (Section 2.7).                                                                                                                                                              |
| .                               |               | 056   | 46      | TECO command (Sections 3.2 and 3.4).                                                                                                                                                     |
| /                               |               | 057   | 47      | Two different uses as TECO commands (Sections 2.7 and 5.2).                                                                                                                              |

Table B-1 (Cont)  
ASCII Characters

| Character | Manual Symbol | Octal | Decimal | Comment and Section Reference                                                                                                                                           |
|-----------|---------------|-------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0         |               | 060   | 48      |                                                                                                                                                                         |
| 1         |               | 061   | 49      |                                                                                                                                                                         |
| 2         |               | 062   | 50      |                                                                                                                                                                         |
| 3         |               | 063   | 51      |                                                                                                                                                                         |
| 4         |               | 064   | 52      |                                                                                                                                                                         |
| 5         |               | 065   | 53      |                                                                                                                                                                         |
| 6         |               | 066   | 54      |                                                                                                                                                                         |
| 7         |               | 067   | 55      |                                                                                                                                                                         |
| 8         |               | 070   | 56      |                                                                                                                                                                         |
| 9         |               | 071   | 57      |                                                                                                                                                                         |
| :         |               | 072   | 58      | TECO command (Section 3.11). Device name delimiter (Section 3.2).                                                                                                       |
| ;         |               | 073   | 59      | TECO command (Section 3.12).                                                                                                                                            |
| <         |               | 074   | 60      | TECO command (Section 3.12).                                                                                                                                            |
| =         |               | 075   | 61      | TECO command (Section 3.15).                                                                                                                                            |
| >         |               | 076   | 62      | TECO command (Section 3.12).                                                                                                                                            |
| ?         |               | 077   | 63      | Two different uses as TECO commands (Sections 3.17 and 5.2).                                                                                                            |
| @         |               | 100   | 64      | TECO command (Sections 3.8 and 3.11).                                                                                                                                   |
| A         |               | 101   | 65      | Two different uses as TECO commands (Sections 3.3 and 3.15).                                                                                                            |
| B         |               | 102   | 66      | TECO command (Section 3.4). Also used in the EB command (Section 3.2).                                                                                                  |
| C         |               | 103   | 67      | TECO command (Section 3.5). Also used in the "C command (Section 3.13).                                                                                                 |
| D         |               | 104   | 68      | TECO command (Section 3.7).                                                                                                                                             |
| E         |               | 105   | 69      | Used as a prefix for many TECO commands: EB, EF, EG, EH, EM, EO, ER, ES, ET, EU, EW, EX, EZ (Sections 3.2, 3.6, 3.9, 3.10). Also used in the "E command (Section 3.13). |

Table B-1 (Cont)  
ASCII Characters

| Character | Manual Symbol | Octal | Decimal | Comment and Section Reference                                                                                    |
|-----------|---------------|-------|---------|------------------------------------------------------------------------------------------------------------------|
| F         |               | 106   | 70      | Used in the EF commands (Section 3.9). Also in FS and FN commands (Section 3.11).                                |
| G         |               | 107   | 71      | TECO command (Section 3.14). Also used in the EG command (Section 3.10) and "G command (Section 3.13).           |
| H         |               | 110   | 72      | TECO command (Section 3.4).                                                                                      |
| I         |               | 111   | 73      | TECO command (Section 3.8).                                                                                      |
| J         |               | 112   | 74      | TECO command (Section 3.5).                                                                                      |
| K         |               | 113   | 75      | TECO command (Section 3.7).                                                                                      |
| L         |               | 114   | 76      | TECO command (Section 3.5). Also used in the "L command (Section 3.13).                                          |
| M         |               | 115   | 77      | TECO command (Section 3.14). Also used in the EM command (Section 3.2).                                          |
| N         |               | 116   | 78      | TECO command (Section 3.11). Also used in the "N command (Section 3.13). Also used in FN command (Section 3.11). |
| O         |               | 117   | 79      | TECO command (Section 3.13).                                                                                     |
| P         |               | 120   | 80      | TECO command (Section 3.9).                                                                                      |
| Q         |               | 121   | 81      | TECO command (Section 3.14).                                                                                     |
| R         |               | 122   | 82      | TECO command (Section 3.5). Also used in the ER command (Section 3.2).                                           |
| S         |               | 123   | 83      | TECO command (Section 3.11). Also used in ES and FS commands (Section 3.11).                                     |
| T         |               | 124   | 84      | TECO command (Section 3.6). Also used in the ET command (Sections 3.6 and 3.16).                                 |
| U         |               | 125   | 85      | TECO command (Section 3.14).                                                                                     |
| V         |               | 126   | 86      |                                                                                                                  |
| W         |               | 127   | 87      | Used in the EW command (Section 3.2) and the PW command (Section 3.4). Otherwise ignored in command strings.     |
| X         |               | 130   | 88      | TECO command (Section 3.14). Also used in the EX command (Section 3.10).                                         |

Table B-1 (Cont)  
ASCII Characters

| Character | Manual Symbol | Octal | Decimal | Command and Section Reference                                                                          |
|-----------|---------------|-------|---------|--------------------------------------------------------------------------------------------------------|
| Y         |               | 131   | 89      | TECO command (Section 3.3).                                                                            |
| Z         |               | 132   | 90      | TECO command (Section 3.4). Also used in the EZ command (Section 3.2).                                 |
| [         |               | 133   | 91      | TECO command (Sections 3.2 and 3.14).                                                                  |
| \         |               | 134   | 92      | Two different uses as TECO commands (Sections 3.8 and 3.14).                                           |
| ]         |               | 135   | 93      | TECO command (Sections 3.2 and 3.14).                                                                  |
| for ^     | †             | 136   | 94      | When used as a command, indicates that the next character is to be interpreted as a control character. |
| ← or _    | ←             | 137   | 95      | TECO command (Section 3.11).                                                                           |
| /         |               | 140   | 96      |                                                                                                        |
| a         |               | 141   | 97      | Equivalent to A in command strings.                                                                    |
| b         |               | 142   | 98      | Equivalent to B in command strings.                                                                    |
| c         |               | 143   | 99      | Equivalent to C in command strings.                                                                    |
| d         |               | 144   | 100     | Equivalent to D in command strings.                                                                    |
| e         |               | 145   | 101     | Equivalent to E in command strings.                                                                    |
| f         |               | 146   | 102     | Equivalent to F in command strings.                                                                    |
| g         |               | 147   | 103     | Equivalent to G in command strings.                                                                    |
| h         |               | 150   | 104     | Equivalent to H in command strings.                                                                    |
| i         |               | 151   | 105     | Equivalent to I in command strings.                                                                    |
| j         |               | 152   | 106     | Equivalent to J in command strings.                                                                    |
| k         |               | 153   | 107     | Equivalent to K in command strings.                                                                    |
| l         |               | 154   | 108     | Equivalent to L in command strings.                                                                    |
| m         |               | 155   | 109     | Equivalent to M in command strings.                                                                    |
| n         |               | 156   | 110     | Equivalent to N in command strings.                                                                    |
| o         |               | 157   | 111     | Equivalent to O in command strings.                                                                    |
| p         |               | 160   | 112     | Equivalent to P in command strings.                                                                    |
| q         |               | 161   | 113     | Equivalent to Q in command strings.                                                                    |

Table B-1 (Cont)  
ASCII Characters

| Character        | Manual Symbol | Octal | Decimal | Comment and Section Reference                                                                                                                                                                      |
|------------------|---------------|-------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| r                |               | 162   | 114     | Equivalent to R in command strings.                                                                                                                                                                |
| s                |               | 163   | 115     | Equivalent to S in command strings.                                                                                                                                                                |
| t                |               | 164   | 116     | Equivalent to T in command strings.                                                                                                                                                                |
| u                |               | 165   | 117     | Equivalent to U in command strings.                                                                                                                                                                |
| v                |               | 166   | 118     | Equivalent to V in command strings.                                                                                                                                                                |
| w                |               | 167   | 119     | Equivalent to W in command strings.                                                                                                                                                                |
| x                |               | 170   | 120     | Equivalent to X in command strings.                                                                                                                                                                |
| y                |               | 171   | 121     | Equivalent to Y in command strings.                                                                                                                                                                |
| z                |               | 172   | 122     | Equivalent to Z in command strings.                                                                                                                                                                |
| {                |               | 173   | 123     |                                                                                                                                                                                                    |
|                  |               | 174   | 124     |                                                                                                                                                                                                    |
| }                |               | 175   | 125     | Converted to altmode (033) when read from TTY unless user has specified TTY LC mode. Equivalent to altmode (033) when executing commands or being typed as text if the EO value has been set to 1. |
|                  |               | 176   | 126     | Converted to altmode (033) when read from TTY unless user has specified TTY LC mode. Equivalent to altmode (033) when executing commands or being typed as text if the EO value has been set to 1. |
| Rubout or Delete | Ⓜ             | 177   | 127     | TECO command (Section 5.1). A special character (Section 2.2).<br>nlⓂ insert only. Does not print.<br>Echoes as the character being erased.                                                        |

TECO

- 344 -

## Appendix C Summary of Commands

### C.1 INITIALIZATION AND FILE SELECTION

Table C-1  
Command Description

| Command                     | Function                                                                   | Reference     |
|-----------------------------|----------------------------------------------------------------------------|---------------|
|                             | INITIALIZATION AND FILE SELECTION                                          |               |
| dev:filnam.ext [proj, prog] | File specifications                                                        | (Section 3.2) |
| ERfilespecification (S)     | Select file for input.                                                     | (Section 3.2) |
| nEM                         | Position magnetic tape                                                     | (Section 3.2) |
| EWfilespecification (S)     | Select file for output.                                                    | (Section 3.2) |
| EZfilespecification (S)     | Zero directory and select file for output.                                 | (Section 3.2) |
| EBfilespecification (S)     | Select file for input and output, with back-up file protection.            | (Section 3.2) |
| MAKEfilespec)               | Equivalent to EWfilnam.ext (S).                                            | (Section 3.1) |
| TECOfilespec)               | Equivalent to EBfilnam.ext (S) Y.                                          | (Section 3.1) |
| /GENLSN                     | Used with EW or EB to cause line sequence numbers to be generated.         | (Section 3.2) |
| /SUPLSN                     | Used with ER, EB, or EW to suppress line sequence numbers.                 | (Section 3.2) |
|                             | INPUT                                                                      |               |
| Y                           | Clear Buffer and input one page.                                           | (Section 3.3) |
| A                           | Input one page and append to current buffer contents.                      | (Section 3.3) |
|                             | BUFFER POSITIONS                                                           |               |
| B                           | Before first character; 0.                                                 | (Section 3.4) |
| .                           | Current pointer position; number of characters to the left of the pointer. | (Section 3.4) |

Table C-1 (Cont)  
Command Description

| Command             | Function                                                                                                                  | Reference      |
|---------------------|---------------------------------------------------------------------------------------------------------------------------|----------------|
| Z                   | End of the buffer; number of characters in the buffer.                                                                    | (Section 3.4)  |
| m,n                 | m+1st through nth characters in the buffer.                                                                               | (Section 2.7)  |
| H                   | Entire buffer; B, Z.                                                                                                      | (Section 3.4)  |
| ARGUMENT OPERATORS  |                                                                                                                           |                |
| m+n                 | Add.                                                                                                                      | (Section 2.7)  |
| m <sub>□</sub> n    | Add.                                                                                                                      | (Section 2.7)  |
| m-n                 | Subtract.                                                                                                                 | (Section 2.7)  |
| m*m                 | Multiply.                                                                                                                 | (Section 2.7)  |
| m/n                 | Divide and truncate.                                                                                                      | (Section 2.7)  |
| m&n                 | Logical AND.                                                                                                              | (Section 2.7)  |
| m#n                 | Logical OR.                                                                                                               | (Section 2.7)  |
| ( )                 | Perform enclosed operations first.                                                                                        | (Section 2.7)  |
| tO                  | Accept number in octal radix.                                                                                             | (Section 2.7)  |
| POINTER POSITIONING |                                                                                                                           |                |
| nJ                  | Move pointer to position between nth and n+1st characters.                                                                | (Section 3.5)  |
| nC                  | Advance pointer n positions.                                                                                              | (Section 3.5)  |
| nR                  | Move pointer back n positions. Equivalent to -nC.                                                                         | (Section 3.5)  |
| nL                  | Move pointer to beginning of nth line from current pointer position.                                                      | (Section 3.5)  |
| TYPE-OUT            |                                                                                                                           |                |
| nT                  | Type all text in the buffer from the current pointer position to the beginning of the nth line from the pointer position. | (Section 3.6)  |
| m,nT                | Type the m+1st through the nth characters.                                                                                | (Section 3.6)  |
| n=                  | Type the decimal integer n.                                                                                               | (Section 3.15) |
| n==                 | Type the octal integer n.                                                                                                 | (Section 3.15) |
| 1ET                 | Change typeout mode so that no substitutions are made for nonprinting characters.                                         | (Section 3.6)  |
| 0ET                 | Restore typeout mode to normal.                                                                                           | (Section 3.6)  |

Table C-1 (Cont)  
Command Description

| Command         | Function                                                                                                                      | Reference      |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------|----------------|
| OEU             | Flag lower case characters on timeout (standard).                                                                             | (Section 3.6)  |
| 1EU             | Flag upper case characters on timeout.                                                                                        | (Section 3.6)  |
| -1EU            | No case flagging on timeout.                                                                                                  | (Section 3.6)  |
| -1ES            | Set automatic timeout after searches.                                                                                         | (Section 3.11) |
| nES(n>0)        | Set automatic timeout after searches and include a character to indicate the position of the pointer.                         | (Section 3.11) |
| OES             | Set to no automatic timeout after searches.                                                                                   | (Section 3.11) |
| Ⓜmessage Ⓜ      | Type the message enclosed.                                                                                                    | (Section 3.17) |
| ↑L or form feed | Type a form feed.                                                                                                             | (Section 3.6)  |
| Ⓢ               | Inhibit timeout.                                                                                                              | (Section 3.6)  |
| DELETION        |                                                                                                                               |                |
| nD              | Delete the n characters following the pointer position.                                                                       | (Section 3.7)  |
| -nD             | Delete the n characters preceding the pointer position.                                                                       | (Section 3.7)  |
| nK              | Delete all characters in the buffer from current pointer position to the beginning of the nth line from the pointer position. | (Section 3.7)  |
| m,nK            | Delete the m+1st through the nth characters.                                                                                  | (Section 3.7)  |
| INSERTION       |                                                                                                                               |                |
| ltext Ⓢ         | Insert the text delimited by l and altmode.                                                                                   | (Section 3.8)  |
| nI Ⓢ            | Insert the character with ASCII value n (decimal).                                                                            | (Section 3.8)  |
| @I/TEXT/        | Insert the text delimited by the arbitrary character following I.                                                             | (Section 3.8)  |
| n \             | Insert the ASCII representation of the decimal integer n.                                                                     | (Section 3.8)  |
| Ⓧ               | Translate to lower case.                                                                                                      | (Section 3.8)  |
| Ⓦ               | Translate to upper case.                                                                                                      | (Section 3.8)  |
| ⓍⓍ              | When used inside text arguments, this means translate special characters @, [, \, ], ↑, ← to "lower case" range.              | (Section 3.8)  |

Table C-1 (Cont)  
Command Description

| Command          | Function                                                                                                                                                                 | Reference      |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
|                  | <b>INSERTION (Cont)</b>                                                                                                                                                  |                |
| Ⓡ                | Accept next character as text.                                                                                                                                           | (Section 3.8)  |
| Ⓣ                | Used inside text arguments to cause all control characters except Ⓡ, Ⓣ, and altmode to be taken as text. Nullified by a second Ⓣ.                                        | (Section 3.8)  |
|                  | <b>OUTPUT AND EXIT</b>                                                                                                                                                   |                |
| PW               | Output the current page and append a form feed character to it.                                                                                                          | (Section 3.9)  |
| nP               | Output the current page, clear the buffer, and read in the next page. Continue this process until the nth page from the current page has been input.                     | (Section 3.9)  |
| m,nP             | Output the m+1st through the nth characters. Do not append a form feed character, and do not change the buffer.                                                          | (Section 3.9)  |
| EF               | Close the output file.                                                                                                                                                   | (Section 3.9)  |
| ⓉZ or tZ         | Close the output file and exit to the monitor.                                                                                                                           | (Section 3.10) |
| Ⓢ                | Exit to the monitor.                                                                                                                                                     | (Section 3.10) |
| EX               | Output the remainder of the file, close the output file, and then exit to the monitor.                                                                                   | (Section 3.10) |
| EG               | Output the remainder of the file, close and then re-execute the last compile-class command that was typed.                                                               | (Section 3.10) |
|                  | <b>SEARCH</b>                                                                                                                                                            |                |
| nStext Ⓢ         | Search for the nth occurrence (following the pointer) of the text delimited by S and altmode, but do not go beyond the end of the current page.                          | (Section 3.11) |
| nFStext Ⓢ text Ⓢ | Search for the nth occurrence (following the pointer) of the first text string and replace it with the second text string. Do not go beyond the end of the current page. | (Section 3.11) |
| nNtext Ⓢ         | Equivalent to nStext Ⓢ except that if the text is not found on the current page, pages are input and output until it is found.                                           | (Section 3.11) |

Table C-1 (Cont)  
Command Description

| Command                | Function                                                                                                                                                                                  | Reference      |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| SEARCH (Cont)          |                                                                                                                                                                                           |                |
| nFNtext (\$) text (\$) | Equivalent to nFStext (\$) text (\$) except that if the text is not found on the current page, pages are input and output until it is found.                                              | (Section 3.11) |
| n ← text (\$)          | Equivalent to nNtext (\$) except that it does input only, no output.                                                                                                                      | (Section 3.11) |
| :nStext (\$)           | Equivalent to nStext (\$) except that it returns a value of -1 if the search succeeds or 0 if it fails instead of an error message. The : command can also be used with FS, N, FN, and ←. | (Section 3.11) |
| @nS/text/              | Equivalent to nStext (\$) except that the text is delimited by the arbitrary character following the S. The @ command may also be used with FS, N, FN, and ←.                             | (Section 3.11) |
| 0 (tX) or 0 tX         | Reset search mode to accept either case.                                                                                                                                                  | (Section 3.11) |
| n (tX) or n tX (n≠0)   | Set search mode to "exact" mode.                                                                                                                                                          | (Section 3.11) |
| (tV)                   | Translate to lower case.                                                                                                                                                                  | (Section 3.11) |
| (tW)                   | Translate to upper case.                                                                                                                                                                  | (Section 3.11) |
| (tt)                   | When used inside text arguments, this means translate special characters @, [, \, ], ↑, ← to "lower case" range.                                                                          | (Section 3.11) |
| (tR)                   | Accept next character as text.                                                                                                                                                            | (Section 3.11) |
| (tT)                   | Used inside text arguments to cause all control characters except (tR), (tT), and altmode to be taken as text. Nullified by a second (tT).                                                | (Section 3.11) |
| (t\)                   | Used inside search arguments to indicate accept either case for following characters. Nullified by a second (t\).                                                                         | (Section 3.11) |
| (tX)                   | When used inside a text argument, accept any character at this position in the search string.                                                                                             | (Section 3.11) |
| (tS)                   | Accept any separator character at this position.                                                                                                                                          | (Section 3.11) |
| (tN) a                 | Accept any character except the arbitrary character a following (tN).                                                                                                                     | (Section 3.11) |

Table C-1 (Cont)  
Command Description

| Command        | Function                                                                                            | Reference      |
|----------------|-----------------------------------------------------------------------------------------------------|----------------|
|                | SEARCH (Cont)                                                                                       |                |
| Ⓚ              | Take the next character in the search string literally, even if it is a control character.          | (Section 3.11) |
| Ⓛ A            | Accept any alphabetic character as a match.                                                         | (Section 3.11) |
| Ⓛ V            | Accept any lower case alphabetic character as a match.                                              | (Section 3.11) |
| Ⓛ W            | Accept any upper case alphabetic character as a match.                                              |                |
| Ⓛ D            | Accept any digit as a match.                                                                        | (Section 3.11) |
| Ⓛ L            | Accept any end-of-line character as a match.                                                        | (Section 3.11) |
| Ⓛ S            | Accept any string of spaces and/or tabs as a match.                                                 | (Section 3.11) |
| Ⓛ <nnn>        | Accept the ASCII character whose octal value is nnn as a match.                                     | (Section 3.11) |
| Ⓛ [a,b,c...]   | Accept any one of the characters in the brackets as a match.                                        | (Section 3.11) |
|                | ITERATION AND FLOW CONTROL                                                                          |                |
| n<>            | Perform the enclosed command string n times.                                                        | (Section 3.12) |
| n;             | If n=0, jump out of the current iteration field.                                                    | (Section 3.12) |
| ;              | Jump out of the current iteration field, if the last search executed failed.                        | (Section 3.12) |
| !tag!          | Define a position in the command string with the name "tag".                                        | (Section 3.13) |
| Otag Ⓢ         | Jump to the position defined by !tag!.                                                              | (Section 3.13) |
| n'Ecommands'   | If n=0, execute the commands specified between 'E and '; otherwise, skip to the '.                  | (Section 3.13) |
| n'Ncommands'   | If n≠0, execute the enclosed commands.                                                              | (Section 3.13) |
| n'Lcommands'   | If n<0, execute the enclosed commands.                                                              | (Section 3.13) |
| n'Gcommands'   | If n>0, execute the enclosed commands.                                                              | (Section 3.13) |
| n-1'Lcommands' | If n≤0, execute the enclosed commands.                                                              | (Section 3.13) |
| n+1'Gcommands' | If n≥0, execute the enclosed commands.                                                              | (Section 3.13) |
| n'Ccommands'   | If n is the ASCII value (decimal) of a symbol constituent character, execute the enclosed commands. | (Section 3.13) |

Table C-1 (Cont)  
Command Description

| Command                           | Function                                                                                                                    | Reference      |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------|----------------|
| ITERATION AND FLOW CONTROL (Cont) |                                                                                                                             |                |
| n'Dcommands'                      | If n is a digit execute the enclosed commands.                                                                              | (Section 3.13) |
| n'Acommands'                      | If n is alphabetic, execute the enclosed commands.                                                                          | (Section 3.13) |
| n'Vcommands'                      | If n is lower case alphabetic, execute the enclosed commands.                                                               | (Section 3.13) |
| n'Wcommands'                      | If n is upper case alphabetic, execute the enclosed commands.                                                               | (Section 3.13) |
| n'Tcommands'                      | If n is true, execute the enclosed commands.                                                                                | (Section 3.13) |
| n'Fcommands'                      | If n is false, execute the enclosed commands.                                                                               | (Section 3.13) |
| n'Scommands'                      | If n is "successful", execute the enclosed commands.                                                                        | (Section 3.13) |
| n'Ucommands'                      | If n is "unsuccessful", execute the enclosed commands.                                                                      | (Section 3.13) |
| Q-REGISTER                        |                                                                                                                             |                |
| nUi                               | Store the integer n in Q-register i.                                                                                        | (Section 3.14) |
| Qi                                | Equal to the value stored in Q-register i.                                                                                  | (Section 3.14) |
| %i                                | Increment the value in Q-register i by 1 and return this value.                                                             | (Section 3.14) |
| nXi                               | Store, in Q-register i, all characters from the current pointer position to the beginning of the nth line from the pointer. | (Section 3.14) |
| m,nXi                             | Store the m+1st through nth characters in Q-register i.                                                                     | (Section 3.14) |
| Gi                                | Place the text in Q-register i at the current pointer position.                                                             | (Section 3.14) |
| Mi                                | Execute the text in Q-register i as a command string.                                                                       | (Section 3.14) |
| [i                                | Push the current contents of Q-register i onto the Q-register pushdown list.                                                | (Section 3.14) |
| ]i                                | Pop the last stored entry from the Q-register pushdown list into Q-register i.                                              | (Section 3.14) |
| *i                                | (As first command in a string.) Save the preceding command string in Q-register i.                                          | (Section 3.14) |

Table C-1 (Cont)  
Command Description

| Command       | Function                                                                                                                                                       | Reference      |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
|               | SPECIAL NUMERIC VALUES                                                                                                                                         |                |
| 1A            | The ASCII value (decimal) of the character following the pointer.                                                                                              | (Section 3.16) |
| (tE) or tE    | The form feed flag. Equals 0 if no form feed character was read on the last input, -1 otherwise.                                                               | (Section 3.16) |
| (tN) or tN    | The end-of-file flag; equals -1 if end of input file seen on last input. Otherwise equals 0.                                                                   | (Section 3.16) |
| tF or (tF)    | Decimal value of the console data switches.                                                                                                                    | (Section 3.16) |
| (tH) or tH    | The time of day in 60th's of a second.                                                                                                                         | (Section 3.16) |
| ET            | The value of the type-out mode switch. Equals 0 for normal type-out, -1 otherwise.                                                                             | (Section 3.16) |
| (tX) or tX    | Value of the search mode flag. (0=either case mode, -1= exact mode.)                                                                                           | (Section 3.11) |
| EU            | The value of the EU flag.<br>+1 = flag upper case characters.<br>0 = flag lower case characters,<br>-1 = no case flagging on typeout.                          | (Section 3.6)  |
| EO            | The value of the EO flag. 1= version 21A, 2= versions 22 and 23.                                                                                               | (Section 3.17) |
| EH            | The value of the EH flag. 1= code only, 2= code plus one line, 3= all of error message.                                                                        | (Section 5.2)  |
| (t↑) x or t↑x | Equivalent to the ASCII value (in decimal) of the arbitrary character x following t↑.                                                                          | (Section 3.16) |
| \             | Equivalent to the decimal value of the digit string following the pointer.                                                                                     | (Section 3.16) |
| (t↑) or t↑    | Stop command execution and then take on the ASCII value (in decimal) of the character typed in by the user.                                                    | (Section 3.16) |
|               | AIDS                                                                                                                                                           |                |
| /             | When used after an error message, this causes a more detailed explanation of the error to be typed.                                                            | (Section 5.2)  |
| *i            | When used at the beginning of a command string, this causes the entire command string (with one of the two concluding altmodes) to be moved into Q-register i. | (Section 5.2)  |

Table C-1 (Cont)  
Command Description

| AIDS (Cont)       |                                                                                                                                |                |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------|----------------|
| ?                 | When used after an error message, this causes the offending command to be typed out (with a few of the commands preceding it). | (Section 5.2)  |
| ?                 | Enter trace mode. A second ? command takes TECO out of trace mode.                                                             | (Section 3.17) |
| Ⓞ                 | Erase last character typed in the command string.                                                                              | (Section 5.1)  |
| Ⓞ Ⓞ               | Erase the entire command string.                                                                                               | (Section 5.1)  |
| Ⓞ                 | Erase everything typed in back to the last break character.                                                                    | (Section 5.1)  |
| Ⓞ <u>        </u> | Retype current line of command string.                                                                                         | (Section 5.1)  |
| 0EO               | Restore the EO value to standard.                                                                                              | (Section 3.17) |
| nEO (n≠0)         | Set the EO value to n.                                                                                                         | (Section 3.17) |
| 1EH               | Type only code part of error messages.                                                                                         | (Section 5.2)  |
| 2EH               | Type error code plus one line.                                                                                                 | (Section 5.2)  |
| 3EH               | Type all three parts of error.                                                                                                 | (Section 5.2)  |
| 0EH               | Equivalent to 2EH.                                                                                                             | (Section 5.2)  |

TECO

- 354 -

**decsystem10 LINED**  
**LINE EDITOR FOR DISK FILES**



1st Printing June 1971  
2nd Printing (Rev) July 1972

Copyright © 1971, 1972 by Digital Equipment Corporation

The material in this manual is for informational purposes and is subject to change without notice.

The following are trademarks of Digital Equipment Corporation, Maynard, Massachusetts:

|           |              |
|-----------|--------------|
| DEC       | PDP          |
| FLIP CHIP | FOCAL        |
| DIGITAL   | COMPUTER LAB |

CONTENTS

|     | Page                                            |   |
|-----|-------------------------------------------------|---|
| 1.0 | Monitor Commands                                | 1 |
| 1.1 | CREATE Command                                  | 1 |
| 1.2 | EDIT Command                                    | 1 |
| 2.0 | LINED Commands                                  | 1 |
| 2.1 | Inserting or Replacing a Line                   | 2 |
| 2.2 | Inserting Multiple Lines                        | 2 |
| 2.3 | Deleting a Line                                 | 2 |
| 2.4 | Deleting Multiple Lines                         | 3 |
| 2.5 | Printing a Line                                 | 3 |
| 2.6 | Printing Multiple Lines                         | 3 |
| 2.7 | Closing the Current File                        | 3 |
| 2.8 | Examples of Command Sequence                    | 3 |
| 3.0 | Auxiliary Commands                              | 4 |
| 3.1 | R LINED                                         | 4 |
| 3.2 | Initializing a File for Processing              | 4 |
| 4.0 | LINED Conventions and Restrictions              | 4 |
| 5.0 | Error Handling                                  | 5 |
| 6.0 | Implementation                                  | 7 |
| 7.0 | Standard for DECsystem-10 Line Sequence Numbers | 7 |

LINED

- 358 -

LINED  
A LINE EDITOR FOR DECsystem-10 FILES

LINED is a line editor for disk files. It is used to create and edit source program files which are written on disk in ASCII code with line sequence numbers appended. LINED has the ability to reference any line at any time without the user having to close and reopen the file. LINED is a reentrant program and loads in 2K pure and 2K impure segments of core.

NOTE

In this document, computer typeouts are indicated by underscoring. The symbol `)` represents the RETURN key. The symbol `Ⓢ` represents the ALTMODE key.

1.0 MONITOR COMMANDS

The MONITOR commands CREATE and EDIT may be used to select a file for editing with LINED. A temporary disk file, called `###EDT.TMP`, is created for passing the commands to LINED.

1.1 The CREATE Command

The CREATE command calls in LINED and opens the specified new disk file for editing. The CREATE command is of the form:

`._CREATE filename.ext )`

1.2 The EDIT Command

The EDIT command calls in LINED and opens the specified existing disk file for editing. The EDIT command is of the form:

`._EDIT filename.ext )`

2.0 LINED COMMANDS

LINED indicates its readiness to receive commands by typing an asterisk. At this time LINED is said to be in command mode. The user may then type in the following LINED commands.

### 2.1 Inserting or Replacing a Line

|                                                                    |                                                                                                                                                                                                                                                                                                                                                       |
|--------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> <u>*</u> Innnnn nnnn  aaaa.....a nnnxx  (\$) <u>*</u> </pre> | <p>Insert or replace the following typed line at line number nnnnn of the currently open file; nnnnn can be specified as a line sequence number or a point (.), or it can be omitted entirely. A point refers to the last line which was typed, or the last line deleted, or the last line inserted. If nnnnn is omitted, it is assumed to be 10.</p> |
|--------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

When LINED has typed a line sequence number, the program enters text mode. In the text mode, characters typed by the user are understood to be text for the insertion. Following the user's typein of the line to be inserted, LINED types out the next sequential line number (nnnnn+10) following which the user presses the ALTMODE key (sometimes labeled PREFIX or ESC) to terminate the insert process and return to LINED command level.

If there already exists a line at nnnnn, it will be replaced. A single quote following the line number indicates that insertion at this line number will cause the existing line to be replaced.

### 2.2 Inserting Multiple Lines

|                                                                                                 |                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> <u>*</u>Innnnn,iiii nnnnn  aaaaa.....a nnnxx  bbbbb.....b . . nnnyy  (\$) <u>*</u> </pre> | <p>Insert the following typed lines, beginning at line number nnnnn (which can be specified as either a line number or a point) of the currently open file. Each time a line is entered, nnnnn is increased by the specified increment, iiii. If iiii is omitted, it is assumed to be 10 (if iiii has never been specified previously), or the previous increment specified.</p> |
|-------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

If nnnnn is omitted, it is assumed to be 10, and the result becomes the line number of the next insertion. Type ALTMODE on the line following the last insertion to return to LINED command mode. LINED then awaits another command.

A double quote following a line number indicates that the increment specified for the current insert instruction has resulted in an existing line being skipped.

### 2.3 Deleting a Line

|                              |                                                                                                                                     |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <pre> <u>*</u> Dnnnnn </pre> | <p>Delete a line number nnnnn from the currently open file; nnnnn can be specified as either a line sequence number or a point.</p> |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|

2.4 Deleting Multiple Lines

\*Dmmmm,nnnn

Delete Lines mmmm through nnnn from the currently open file; mmmm must be less than nnnn. Either mmmm or nnnn may be specified as a point as long as mmmm is less than nnnn.

2.5 Printing a Line

\*Pnnnn

Print line nnnn on the user's Teletype; nnnn can be specified as either a line sequence number or a point. Typing ALTMODE following a typeout will cause the next sequential line to be printed.

2.6 Printing Multiple Lines

\*Pmmmm,nnnn

Print lines mmmm through nnnn of the currently open file; mmmm must be less than nnnn. Either mmmm or nnnn may be specified as a point as long as mmmm is less than nnnn.

2.7 Closing the Current File

E )

Closes the current file and returns to LINED command mode. At this point, the user may either open another file or type ↑C to return to Monitor level to assemble, list, and/or save his file on a permanent storage device (e.g., DECTape).

2.8 Examples of Command Sequence

Example 1

\_CREATE FILEA  
\*I10  
00010 THE PROGRAM  
00020 IS INSERTED  
00030 HERE  
:  
:  
:  
00350 (\$) \*E  
\*↑C  
:  
\_

RUN LINED AND OPEN FILE FILEA  
BEGIN INSERTING LINES AT LINE NUMBER  
10 INCREMENTING BY 10.  
  
RETURN CONTROL TO LINED COMMAND  
MODE BY TYPING (\$). CLOSE FILE FILEA  
BY TYPING AN E. TYPING A ↑C RETURNS  
TO THE MONITOR COMMAND LEVEL.

## Example 2

|                                                                                                                                                                                                                           |                                                                                                                                                                                                                                        |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> .<u>E</u>EDIT FILEA *<u>P</u>10,30 00010 THE PROGRAM 00020 IS INSERTED 00030 HERE *<u>I</u>20 00020 IS PLACED 00030 (\$) *<u>D</u>30 *<u>P</u> 10,30 00010 THE PROGRAM 00020 IS PLACED *<u>E</u> *<u>↑</u>C .</pre> | <pre> RUN LINED AND OPEN EXISTING FILE FILEA PRINT LINES 10 THROUGH 30 PRINTOUT  INSERT LINE 20  DELETE LINE 30 PRINT LINES 10 THROUGH 30 PRINTOUT  TYPE E TO CLOSE FILE FILEA TYPING A ↑C RETURNS JOB TO MONITOR CONTROL LEVEL.</pre> |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## 3.0 AUXILIARY COMMANDS

These Auxiliary Commands provide an alternate method of calling LINED and opening files. In most cases, auxiliary commands can be replaced by the monitor instructions CREATE and EDIT (Section 1).

## 3.1 R LINED

LINED can be called in from the system device by typing

```

.R LINED )
*
```

LINED responds with an asterisk to indicate its readiness to receive a command.

## 3.2 Initializing a File for Processing

|                                 |                                                                                   |
|---------------------------------|-----------------------------------------------------------------------------------|
| <pre> S filename.ext )</pre>    | <pre> Select an existing disk file, filename.ext, for editing.</pre>              |
| <pre> S filename.ext (\$)</pre> | <pre> Select (create) a new disk file for editing, calling it filename.ext.</pre> |

## 4.0 LINED CONVENTIONS AND RESTRICTIONS

The following conventions and restrictions should be noted.

- a. Files are written with the installation standard protection. See the DECsystem-10 Operating System Commands manual for explanation of protected files.

- b. When in insert mode, typing ALTMODE following the printout of the next insertion line sequence number causes a returned to LINED command level. Typing ALTMODE to terminate a line of text to be inserted causes the text line to be ignored.

```

00010   LINE OF TEXT
00020   ($)           Returns to LINED command level
*
    
```

```

00010   LINE OF TEXT ($) Line is ignored
*
    
```

- c. LINED assumes that all blocks in a disk file have an integral number of lines (i.e., each block begins with a sequence number and no line is split between blocks). This will always be the case with files which have been created and edited only with LINED; however, if sequence numbers have been removed, say by TECO, they may be restored by using PIP switch /S.
- d. LINED files can be resequenced using PIP switch /S.
- e. Line number 0 is illegal and cannot be used.
- f. Lines can be edited in any order; however, editing lines by ascending line numbers reduces file access time.

### 5.0 ERROR HANDLING

When an error is detected, LINED types a message and returns the user to LINED command level (indicated by the output of an \* on the Teletype). Some errors are fatal and cause control to return to the monitor. Error messages for LINED are given in Table 1.

Table 1  
LINED Error Messages

| Message                   | Meaning                                                                                                                                                |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| ?FILE NAME ALREADY IN USE | The filename specified in a CREATE or S command already exists on disk. Type the S command with a correct filename, followed by (\$).                  |
| ?FILE NOT SPECIFIED       | The user attempted to execute an editing command without first naming the file to be edited. Using an S command, name the file to be edited.           |
| ?ILLEGAL COMMAND          | The user attempted to use a letter that is not a command. Type the correct command letter.                                                             |
| ?INPUT FILE NOT FOUND     | The file named in an EDIT or S command cannot be found on disk. Either place the file on disk, or create the file with the S command followed by (\$). |

(continued on next page)

Table 1 (Cont)  
LINED Error Messages

| Message                                                                                                                                | Meaning                                                                                                                                                                                                                                                                                                                                                      |
|----------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>?LINE REFERENCED DOES NOT EXIST</p> <p>SYSTEM ERROR READING COMMAND FILE</p>                                                        | <p>A line referenced in a P or D command does not exist in the file. Either retype the command with the correct line number, or insert the line.</p> <p>A system error occurred while LINED was trying to read the CCL command file generated by a CREATE or EDIT command. Try to create or select the file using the appropriate form of the S command.</p> |
| <p>NOTE</p>                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                              |
| <p>The following are internal system errors and cause control to return to the monitor.</p>                                            |                                                                                                                                                                                                                                                                                                                                                              |
| <p>?CANNOT ACCESS DISK</p>                                                                                                             | <p>LINED cannot access the disk. This message can only occur at the beginning of operations. Notify the system manager.</p>                                                                                                                                                                                                                                  |
| <p>?CANNOT INIT TTY</p>                                                                                                                | <p>LINED cannot initialize the user's terminal. This message can only occur at program initialization. Notify the system manager.</p>                                                                                                                                                                                                                        |
| <p>?ERROR IN RENAME PROCESS<br/>INPUT FILE CLOSED WITH NAME<br/>    filnam.ext<br/>OUTPUT FILE CLOSED WITH NAME<br/>    ###LIN.TMP</p> | <p>An error occurred while LINED was renaming the output file. The input file should be renamed filnam.BAK and the output file should be renamed filnam.ext.</p>                                                                                                                                                                                             |
| <p>?ERROR IN RENAME PROCESS<br/>INPUT FILE CLOSED WITH NAME<br/>    ###TMP.TMP<br/>OUTPUT FILE CLOSED WITH NAME<br/>    ###LIN.TMP</p> | <p>An error occurred while LINED was renaming the files. The input file should be renamed filnam.BAK and the output file should be renamed filnam.ext.</p>                                                                                                                                                                                                   |
| <p>?ERROR IN RENAME PROCESS<br/>INPUT FILE CLOSED WITH NAME<br/>    ###TMP.TMP<br/>OUTPUT FILE CLOSED WITH NAME<br/>    filnam.ext</p> | <p>An error occurred while LINED was renaming the files. The input file should be renamed filnam.BAK.</p> <p>In the three messages above, ### is the user's job number and filnam.ext is the name of the file that he was editing.</p>                                                                                                                       |
| <p>?INPUT ERROR.<br/>INCOMPLETE OUTPUT FILE CLOSED<br/>WITH NAME ###LIN.TMP</p>                                                        | <p>A system error occurred on input. The output file is incomplete; thus, the user must start editing again with the backup file.</p>                                                                                                                                                                                                                        |
| <p>?NO CORE AVAILABLE FOR DATA SEGMENT</p>                                                                                             | <p>There is no core available for LINED to do editing on the user's file. This message can occur only during program initialization. Notify the system manager.</p>                                                                                                                                                                                          |
| <p>?OUTPUT ERROR.<br/>INCOMPLETE OUTPUT FILE CLOSED<br/>WITH NAME ###LIN.TMP</p>                                                       | <p>A system error occurred on output. The output file is incomplete; thus the user must start editing again with the backup file.</p>                                                                                                                                                                                                                        |

## 6.0 IMPLEMENTATION

The following explanation is intended to help the user to understand how LINED works so that he may use it more effectively.

Lines of text are stored in a 1000-word working buffer. Each line has a 1-word header containing two items. The left half contains the sequence number of the line, and the right half contains the number of words (including the word containing the line header) needed to store the line of text. Thus, to find the beginning of the next line of text, it is necessary to simply take the address of the current line header and add the word count of the current line.

Several pointer words are used to keep track of the lines in the working buffer. WRTLST contains the sequence number of the highest line in the buffer. SN contains the sequence number of the line currently being handled in a command.

When LINED discovers that SN is greater than WRTLST, it knows that the line being sought has already passed through the working buffer. This line is not directly accessible, because there is no way to read a disk file backwards. Consequently, it is necessary for LINED to close the file and then reopen it. This process of going from the current position of the file to the end of the file, from there to the beginning of the file, and finally to the line being sought is accomplished as follows:

- a. To close the file, all remaining text must be passed through the working buffer to the temporary output file (called ###LIN.TMP). This is done by giving the subroutine FNDLIN (which finds a line whose sequence number is SN) the highest possible sequence number - 99999.
- b. Next, the original file is renamed to ###TMP.TMP, the temporary output file is renamed to the original filename and the original file (###TMP.TMP) is renamed to name.BAK (same name as original with an extension of BAK).
- c. FNDLIN is then given the sequence number being sought, and LINED continues with the original command.

## 7.0 STANDARD FOR DECsystem-10 LINE SEQUENCE NUMBERS

ASCII data files containing line sequence numbers conform to the following rules.

- a. Each line must begin at a word boundary. Lines are padded at the end with nulls to fill an integral number of words.
- b. Every line must have a line sequence number.
- c. The line sequence number consists of five ASCII characters contained in the first word of the line.
- d. Bit 35 of the line sequence number word is set to 1.
- e. The line sequence number can contain only decimal digits. The characters preceding the first non-zero digit should be ASCII zeros. However, on input, leading spaces as well as leading zeros are accepted for compatibility with those data files that have leading spaces.

- f. The first character after the line sequence number is always a tab except in files created by BASIC. All compilers except BASIC ignore the character after the line sequence number. The utility programs (editors and PIP) automatically cause a tab to follow the line sequence number when they are creating new line sequence numbers. However, for compatibility with BASIC, the utility programs do not force a tab after the line sequence number when they are merely transferring existing line sequence numbers from an input file to an output file.
- g. Line blocking is optional.

**PDP-10**  
**PIP**  
**(PERIPHERAL INTERCHANGE PROGRAM)**  
**PROGRAMMER'S REFERENCE MANUAL**

1st Edition, October 1967  
2nd Edition (Rev) May, 1968  
3rd Edition (Rev) November, 1968  
4th Edition (Rev) November, 1969  
5th Edition (Rev) June, 1970  
6th Edition (Rev) March, 1972

Copyright © 1967, 1968, 1969, 1970, 1972 by Digital Equipment Corporation

The material in this manual is for information purposes and is subject to change without notice.

The following are trademarks of Digital Equipment Corporation, Maynard, Massachusetts:

|           |              |
|-----------|--------------|
| DEC       | PDP          |
| FLIP CHIP | FOCAL        |
| DIGITAL   | COMPUTER LAB |

PREFACE

The functions provided the user by the DECsystem-10 Peripheral Interchange Program (PIP) and their use are described in this manual.

NOTE

Monitor commands are available which perform the common PIP functions of copying, renaming, protecting and deleting files.

It was assumed in the preparation of this manual that the reader is familiar with or has access to the DECsystem-10 Monitor Calls manual and the DECsystem-10 Monitor Commands manual. These manuals as well as the PIP manual are available in the DECsystem-10 Software Notebook and in the following handbooks:

- a) DECsystem-10 User's Handbook (contains both PIP and the Monitor commands manuals).
- b) DECsystem-10 Assembly Language Handbook (contains Monitor calls manual).

PIP

- 370 -

CONTENTS

|                                                              | <u>Page</u> |
|--------------------------------------------------------------|-------------|
| SECTION 1. INTRODUCTION                                      |             |
| 1.1 INTRODUCTION                                             | 375         |
| 1.1.1 Controlling PIP Indirectly                             | 375         |
| 1.2 WRITING CONVENTIONS                                      | 376         |
| SECTION 2. PIP COMMAND STRING AND ITS BASIC ELEMENTS         |             |
| 2.1 COMMAND STRING                                           | 379         |
| 2.1.1 Command Format                                         | 379         |
| 2.1.2 File Specification                                     | 380         |
| 2.1.3 Command String Delimiters                              | 382         |
| 2.2 DEVICE NAMES                                             | 383         |
| 2.2.1 Physical Device Names                                  | 383         |
| 2.2.2 Logical Device Names                                   | 383         |
| 2.3 FILENAMES                                                | 384         |
| 2.3.1 Naming Files with Octal Constants                      | 385         |
| 2.3.2 Wildcard Characters                                    | 386         |
| 2.3.2.1 The Asterisk Symbol                                  | 386         |
| 2.3.2.2 The Question Mark Symbol                             | 386         |
| 2.3.2.3 Combining * and ? Wildcard Symbols                   | 386         |
| 2.4 DIRECTORY IDENTIFIER                                     | 387         |
| 2.4.1 UFD-Only Identifiers                                   | 388         |
| 2.4.2 SFD (Full Directory Path) Identifiers                  | 388         |
| 2.4.3 Specifying Default and Current [Directory] Identifiers | 389         |
| 2.5 FILE ACCESS PROTECTION CODES                             | 390         |
| 2.5.1 Digit Numeric Protection Code Values                   | 391         |
| 2.6 UFD AND SFD PROTECTION CODES                             | 392         |
| SECTION 3 STANDARD PIP SWITCHES                              |             |
| 3.1 OPTIONAL PIP FUNCTIONS                                   | 393         |
| 3.1.1 Adding Switches to PIP Commands                        | 393         |
| 3.2 BASIC TRANSFER FUNCTION                                  | 394         |
| 3.2.1 X-Switch Copy Files Without Combining                  | 394         |
| 3.2.1.1 Non-Directory to Directory Copy Operation            | 395         |
| 3.2.1.2 Assigning Names to DEctape Tapes                     | 397         |
| 3.2.2 DX-Switch, Copy All but Specified Files                | 397         |
| 3.2.3 Transfer Without X-Switch (Combine Files)              | 398         |
| 3.2.4 U-Switch, Copy DEctape Blocks $\emptyset$ , 1, and 2   | 398         |

## CONTENTS

|           | <u>Page</u>                                                                  |
|-----------|------------------------------------------------------------------------------|
| 3.3.1     | A-Switch, Integral Output Lines (Line Blocking) 399                          |
| 3.3.2     | C-Switch, Delete Trailing Spaces and Convert Multiple Spaces to Tabs 399     |
| 3.3.3     | E-Switch, Ignore Card Sequence Numbers 399                                   |
| 3.3.4     | N-Switch, Delete Sequence Number 399                                         |
| 3.3.5     | S-Switch, Insert Sequence Numbers 400                                        |
| 3.3.6     | O-Switch, Insert Sequence Numbers and Increment by One 400                   |
| 3.3.7     | P-Switch, Prepare FORTRAN Output for Line Printer Listing 400                |
| 3.3.7.1   | Copy FORTRAN Binary Files 401                                                |
| 3.3.8     | T-Switch, Delete Trailing Spaces 402                                         |
| 3.3.9     | W-Switch, Converts Tabs to Spaces 402                                        |
| 3.3.10    | V-Switch, Match Angle Brackets 402                                           |
| 3.3.11    | Y-Switch, DECTape to Paper Tape 403                                          |
| 3.4       | SET DATA MODE, SWITCHES B, H AND I 405                                       |
| 3.5       | FILE DIRECTORY SWITCHES 406                                                  |
| 3.5.1     | L-Switch, List Source Device Directory 406                                   |
| 3.5.2     | F-Switch, List Limited Source Directory 407                                  |
| 3.5.3     | R-Switch, Rename Source Files 407                                            |
| 3.5.3.1   | Changing Source UFD or SFD Protection Code Using the Rename (R) Function 408 |
| 3.5.4     | D-Switch, Delete Files 409                                                   |
| 3.5.5     | Z-Switch, Zero Directory 411                                                 |
| 3.5.6     | Q-Switch, Print Summary of PIP Functions 411                                 |
| 3.6       | PERMITTED SWITCH COMBINATIONS 413                                            |
| SECTION 4 | SPECIAL PIP SWITCHES                                                         |
| 4.1       | SPECIAL PIP FUNCTIONS 415                                                    |
| 4.2       | MAGNETIC TAPE SWITCHES 415                                                   |
| 4.2.1     | Switches for Setting Density and Parity Parameters 415                       |
| 4.2.2     | Switches for Positioning Magnetic Tape 416                                   |
| 4.2.2.1   | Backspace to Start of Current File 417                                       |
| 4.2.2.2   | Advance to End of Current File 417                                           |
| 4.3       | G-SWITCH, ERROR RECOVERY 417                                                 |
| 4.4       | J-SWITCH, CARD PUNCH 418                                                     |

CONTENTS

|            | <u>Page</u>                            |     |
|------------|----------------------------------------|-----|
| SECTION 5  | PIP ERROR REPORTING AND ERROR MESSAGES |     |
| 5.1        | ERROR MESSAGES                         | 419 |
| 5.2        | I/O ERROR MESSAGES                     | 419 |
| 5.3        | FILE REFERENCE ERRORS                  | 420 |
| 5.4        | PIP COMMAND ERRORS                     | 421 |
| 5.5        | Y-SWITCH ERRORS                        | 422 |
| 5.6        | GENERAL ERROR MESSAGES                 | 422 |
| 5.7        | TMPCOR (DEVICE TMP) ERROR MESSAGES     | 424 |
| APPENDIX A | STANDARD FILENAME EXTENSIONS           | 425 |



SECTION 1

INTRODUCTION

1.1 INTRODUCTION

PIP (Peripheral Interchange Program) transfers files between standard I/O devices and can be used to perform simple editing and magnetic tape control operations during those transfer operations.

To call PIP into core (1) from the Monitor level, the user types the command

```
.R PIP <CR>
```

When PIP is loaded and ready for input it prints the character \* at the console. The user may then enter the command string needed to perform the desired operations followed by a carriage return input. On completion of the operation or operations requested in a command string, PIP again prints the character \* to indicate that it is ready for the next command string input. To exit from PIP, the user types a Control C (↑C) command.

1.1.1 Controlling PIP Indirectly

PIP is normally controlled by commands entered via the console keyboard. PIP, however, is also capable of reading commands from a prepared file and executing these commands as if they had been just entered via the input console. PIP command files which are to be processed indirectly are identified by the addition of the symbol @ to their identifying file specification (see paragraph 2.1.2 for a description of file specifications). For example, the file specification FOO.CCL@ identifies the file FOO.CCL as an indirect command file. Any filename extension may be used in specifying an indirect command file, however, if none is given, the default extension .CCL is assumed.

An indirect PIP command file consists of one or more PIP commands structured as described in Section 2.

---

(1) The PIP program operates in 4K pure core plus a minimum of 1K of impure core in all DECsystem-10 systems.

Once PIP is in core, the user passes control of PIP to an indirect command file by entering the file's filename. For example, the input command sequence

```
.R PIP      <CR>
*FOO.CCL@  <CR>
```

loads PIP and initiates the execution of the indirect PIP command file FOO.CCL.

## 1.2 WRITING CONVENTIONS

The following symbols and abbreviations are used throughout this manual:

| <u>Symbol or Abbreviation</u> | <u>Meaning</u>                                                                                                                                                                                                                   |                     |              |                 |         |                  |         |
|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|--------------|-----------------|---------|------------------|---------|
| dev:                          | Any logical or physical device name, the colon must be included when it is used as part of a PIP command.                                                                                                                        |                     |              |                 |         |                  |         |
| file.ext                      | Any filename and filename extension.                                                                                                                                                                                             |                     |              |                 |         |                  |         |
| [directory]                   | Identifies the directory of a specific file storage area within the system; it may also specify the location of specific file within the identified storage area. (See paragraph 2.4 for a detailed description of [directory].) |                     |              |                 |         |                  |         |
|                               | When the input terminal used is either a Model 33 or 35 Teletype unit, the right and left brackets are input in the following manner:                                                                                            |                     |              |                 |         |                  |         |
|                               | <table> <thead> <tr> <th><u>To Obtain a:</u></th> <th><u>Type:</u></th> </tr> </thead> <tbody> <tr> <td>a) left bracket</td> <td>SHIFT K</td> </tr> <tr> <td>b) right bracket</td> <td>SHIFT M</td> </tr> </tbody> </table>      | <u>To Obtain a:</u> | <u>Type:</u> | a) left bracket | SHIFT K | b) right bracket | SHIFT M |
| <u>To Obtain a:</u>           | <u>Type:</u>                                                                                                                                                                                                                     |                     |              |                 |         |                  |         |
| a) left bracket               | SHIFT K                                                                                                                                                                                                                          |                     |              |                 |         |                  |         |
| b) right bracket              | SHIFT M                                                                                                                                                                                                                          |                     |              |                 |         |                  |         |
| ↑ch                           | A control character obtained by depressing the CTRL key and then the selected character key (e.g. ↑Z).                                                                                                                           |                     |              |                 |         |                  |         |
| =                             | An equals character is used in the PIP command to separate the destination and source command sections.                                                                                                                          |                     |              |                 |         |                  |         |

### NOTE

PIP will also accept the back arrow (SHIFT-O) entry. A SHIFT-O entry is echoed on the terminal printer as the symbol ←.

\* PIP's response to a command string to indicate that it is ready for the next input string.

| <u>Symbol or Abbreviation</u> | <u>Meaning</u>                                                                                                                                                                                                                                                                                                                                               |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                               | The Monitor's response to a command string to indicate that it is ready for the next command.                                                                                                                                                                                                                                                                |
| <CR>                          | This symbol represents a carriage return, line-feed operation. It is initiated by the entry of a RETURN keyboard input. A RETURN input is normally used to terminate each PIP input command.                                                                                                                                                                 |
| -----                         | Underscoring indicates computer typeout.                                                                                                                                                                                                                                                                                                                     |
| n                             | A number, either octal or decimal.                                                                                                                                                                                                                                                                                                                           |
| ↑                             | This up-arrow symbol indicates the use of a CTRL key entry. The up-arrow is used with other character key inputs to produce special control entries such as ↑C which requests that control be returned to the Monitor. Up-arrows are also used to enclose identifiers which may be assigned to DECTapes using the facilities provided by PIP (see 3.2.1.2.). |

PIP

- 378 -

SECTION 2

PIP COMMAND STRING AND ITS BASIC ELEMENTS

2.1 COMMAND STRING

PIP command strings may be of any length; both upper and lower case characters may be used. PIP commands are normally terminated and the requested operation is initiated by a RETURN keyboard entry (i.e., <CR>). However, an ALT MODE, line feed, vertical TAB or form feed keyboard entry can also be used as a command terminator.

2.1.1 Command Format

All PIP commands which involve the interchange (transfer) or data must have the following format:

DESTINATION=SOURCE <Terminator>

where:

- a. The DESTINATION portion of a PIP command describes the device and file(s) which are to receive the transferred data. This portion of a command consists of either one file specification or a subset of a file specification.
- b. The equals sign is a required delimiter in all PIP commands to separate the DESTINATION and SOURCE portions of the command.
- c. The SOURCE side of the command describes the device from which the transferred data is to be taken. This portion of a command may contain one or more file specifications or subsets of file specifications.
- d. A Terminator is required to end each PIP command. A RETURN entry (symbolized as <CR>) is normally used, however, any other paper-motion command may be used as a terminator.

PIP commands which do not require the transfer of information may be written using the form

DESTINATION=Terminator

The equals delimiter and a terminator are still required in commands

formatted in this manner despite the fact that only the destination portion of the command is used.

### 2.1.2 File Specification

A file specification contains all of the information needed to identify a file involved in a PIP function. It may consist of:

1. a device name;
2. a filename;
3. a directory identifier;
4. a protection code which is to be assigned to either a specified file, a User File Directory (UFD), or a SubFile Directory (SFD);
5. and an identifier to be assigned to the tape mounted on a specified DECTape unit.

The format of a PIP command containing all possible items of a file specification is:

```
dev:name.ext[directory]<nnn>↑ident↑=dev:name.ext[directory] <CR>
```

where:

1. DEV is either a physical device name (e.g., DSK, DTAL, etc.) or a logical device name (refer to paragraph 2.2).
2. NAME is a 1 to 6 alphameric character identification which is either to be assigned to a new file (NAME is on the destination side of the command) or which identifies an existing file (NAME is on the source side of the command). (Refer to paragraph 2.3 for a description of filenames.)
3. EXT is a 1 to 3- character extension assigned to the name of a file either by the user or by the system. (Refer to paragraph 2.3 for a description of filename extensions.)
4. [DIRECTORY] is the identifier of a specific directory (i.e., UFD or MFD) within the system. This identifier may consist of a project, programmer number pair and Sub File Directory (SFD) names. (See paragraph 2.4 for details.)
5. <nnn> is a 3-digit protection code which is to be assigned to either one or more destination files or to a specified User File Directory<sup>1</sup>. (Refer to paragraph 2.5 for a description of protection codes.)
6. ↑IDENT↑ is a 1 to 6 character name which is to be given to the contents of a DECTape reel mounted on a specified DECTape unit. (Refer to paragraph 3.2.1.2 for details.)

---

<sup>1</sup>A User File Directory (UFD) is contained by the system for each user permitted access to it. A user's UFD is identified by his project, programmer number; it contains the names of all files belonging to the user together with pointers to the actual location of each file.

The manner in which each of the possible elements of a file specification may be used in either the destination or source portions of a PIP command is described in the following table:

| <u>Element</u> | <u>Destination</u>                                                                      | <u>Source</u>                                                                 |
|----------------|-----------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| dev.           | Name of device onto which the specified file is to be written.                          | Name of device on which the specified file resides.                           |
| name           | Name to be assigned to the copied file.                                                 | Name of the file to be copied.                                                |
| .ext           | User-specified file-name extension.                                                     | Current filename extension.                                                   |
| [directory]    | Identification of the disk storage area which is to receive the file to be transferred. | Identification of the disk storage area which contains the file to be copied. |

NOTE

The [directory] identifier must include a full directory path specification whenever sub-file directories are involved. For example [proj,prog,SFDA...SFDn]. (See paragraph 2.4 for more details.)

|         |                                                                            |                                                  |
|---------|----------------------------------------------------------------------------|--------------------------------------------------|
| <nnn>   | Protection code to be assigned to either a copied file or a specified UFD. | NOT PERMITTED IN SOURCE PORTION OF PIP COMMANDS. |
| †ident† | Name to be assigned to the tape mounted on a specified DECTape unit.       | NOT PERMITTED IN SOURCE PORTION OF PIP COMMANDS. |

File specifications may be delimited by:

1. an equals character (=) if the specification is on the destination side of the command string (e.g. dev:name.ext=...<CR>).

NOTE

PIP will accept a back-arrow entry (+) in place of the equals character (=).

2. a comma (,) if the specification is on the source side of the command string and is one of a series of file specifications. For example  
dev=dev1:name.ext,dev2:name.ext,name.ext,..name.ext<CR>
3. a RETURN <CR> entry if it is the last item on the source side of a command. For example  
dev=dev1:name.ext,dev2:name.ext,..devn:name.ext<CR>

## 2.1.3 Command String Delimiters

The delimiters which may be used to separate the elements of a PIP command string are described in the following table.

## PIP COMMAND STRING DELIMITERS

| <u>Delimiter</u> | <u>Use and Description</u>                                                                                                                                                                                                                                                                                    |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| :                | The colon delimiter follows and identifies a device name. For example, the device DTAl is specified as DTAl: in PIP commands.                                                                                                                                                                                 |
| [ ]              | Square brackets are used to enclose the user DIRECTORY numbers and SFD names (if SFDs are used). For example [40,633] or [40,633,SFD1,SFD2,...SFDn] represent the manner in which DIRECTORY numbers can be written.                                                                                           |
| < >              | Angle brackets must be used to enclose a protection code (e.g. <Ø57> which is to be assigned to either a file or a user file directory (UFD).                                                                                                                                                                 |
| ,                | Commas are used to separate user project and programmer numbers, and file specification groups.<br>For example<br><br>dev:[4Ø,633]=dev:name.ext,name.ext<CR>                                                                                                                                                  |
| ↑↑               | A name to be assigned as an identifier to a DEC-tape is enclosed within a set of up-arrows (e.g. ↑MACFLS↑).                                                                                                                                                                                                   |
| .                | A period delimiter must be the first character of a filename extension. The form on an extension is .ext.                                                                                                                                                                                                     |
| #                | A number symbol is used as a flag to indicate the presence of an octal constant in a filename or a filename extension.                                                                                                                                                                                        |
| !                | An exclamation symbol may be used to delimit a file specification. When used, the ! symbol causes control to be returned to the Monitor from PIP and the specified file (or program) to be loaded and run. This function is provided as a user convenience to eliminate the need for several control entries. |
| =                | The equals character must be used to separate the destination and source portions of a PIP command.                                                                                                                                                                                                           |
| ( )              | Parentheses are used to enclose magnetic tape options, PIP control switches, and one or more PIP function switches. The form of a command employing parentheses to enclose a series of switches is:<br><br>dev:name.ext(sw1sw2..swn)=...<CR>                                                                  |

## 2.2 DEVICE NAMES

Both physical or logical device names may be used in PIP commands. The user must remember that a logical name takes precedence over a physical name when both are used in the same command.

### 2.2.1 Physical Device Names

Each standard DECsystem-10 peripheral device is assigned a specific device name consisting of a 3-character generic name plus either a unit number (0 to 777) or:

- 1) 3 characters,
- 2) 3 characters and a station number,
- 3) an abbreviated disk name or,
- 4) the name of a disk file structure.

A list of the generic physical device names is given below:

#### PERIPHERAL DEVICES

| <u>Device</u>        | <u>Generic Physical Device Name</u> |
|----------------------|-------------------------------------|
| Card Punch           | CDP                                 |
| Card Reader          | CDR                                 |
| Console TTY          | CTY                                 |
| DEctape              | DTA                                 |
| Disk                 | DSK                                 |
| Packs                | DPx                                 |
| Fixed-Head           | FHx                                 |
| Display              | DIS                                 |
| Line Printer         | LPT                                 |
| Magnetic Tape        | MTA                                 |
| Operator Terminal    | OPR                                 |
| Paper-tape Punch     | PTP                                 |
| Paper-tape Reader    | PTR                                 |
| Plotter              | PLT                                 |
| Pseudo-TTY           | PTY                                 |
| System Library       | SYS                                 |
| Terminal             | TTY                                 |
| Pseudo-device TMPCOR | TMP                                 |

### 2.2.2 Logical Device Names

A logical device name is a user-assigned designation which is employed in the preparation of a program in place of a specific physical device name. The use of logical device names permits the programmer to write programs which do not specify one particular device but may use, at run time, any available device which can perform the required function.

Logical device names may consist of from one to six alphanumeric characters of the user's choice.

### 2.3 FILENAMES

Filenames are file identifiers assigned either by the system (for system programs) or by the user. A filename may consist of a name field and an extension field but only a name field is required. Whenever both fields are used in a filename, it has the form name.ext. A period delimiter is required as the first character of the extension. Filename fields are defined as:

1. Name Field. Names of files may consist of from one to six alphanumeric characters or octal constants; in user-assigned names the characters may be arbitrarily selected by the user. Names generated by the user must be unique at least within the file structure in which the file is located.
2. Extension Field. Filename extensions may consist of up to three alphanumeric characters. Extensions are normally used to specify the type of data contained by the file identified by the filename field. Filename extensions which are recognized by the system and the type of data each specifies are given in Appendix A. In filenames, users may specify a standard extension (one recognized by the system), one which he has devised, or none at all. If no extension is given in a filename, the system may add one to the filename during PIP operations.

PIP utilizes the filename extension given in a file specification to determine whether the file is to be transferred in a binary or ASCII mode. If it is all possible, PIP will transfer files in a binary mode since it is faster.

In dealing with filename extensions PIP performs a specific series of tests in order to determine the mode which should be used during a requested transfer operation. The following mode determination tests are performed in succession until PIP obtains a firm indication as to the type of mode required:

- a) PIP tests for the presence of a data mode switch (see paragraph 3.4.). If no switch is found, PIP goes to the next test.
- b) PIP tests for the presence of a known (standard) filename extension which specifies a binary mode of transfer (see Appendix A). If no binary extensions are found, PIP goes to the next test.

- c) PIP tests both the input and output devices specified to determine if they are both capable of handling binary data. If either or both of the devices cannot handle binary, the transfer is made in the ASCII mode. If both devices can handle binary data, PIP goes to the next test.
- d) PIP tests for the presence of an X option switch (/X) in the command string; if it is found, the transfer is made in the binary mode. If an X option is not found, PIP goes to the next test.
- e) PIP tests for the presence of commas (non-delimiters) in the command string; if commas are found an ASCII mode is indicated. If no commas are found, the transfer is made in the binary mode.

### 2.3.1 Naming Files with Octal Constants

Octal constants may be used as either a part of or all of a filename. In either of the foregoing cases, the first constant of each group of octal constants which appear in a filename must be preceded by the symbol #, and each group is delimited by a non-octal digit or a character. For example, the filenames:

- 1. #124ABC.ext (constants are used as part of a filename)
- 2. #12AB#34.ext (constants are intermixed with other characters)
- 3. #124670.#123 (constants form the whole filename)

are all acceptable to PIP.

The symbol # is not regarded by PIP as part of the filename but is used only as a flag to PIP to indicate an octal constant.

The number of octal digits used in a filename or an extension should be even since two octal constants may be stored in a SIXBIT character. If an odd number of octal constants is given, PIP will add an extra 0 to the filename or extension. For example, the constant #123 would be expanded to #1230 by PIP.

Names comprised of octal constants are left-justified by PIP. The following are examples of the use of octal filenames:

DTA01:#124670.BIN=DSK:#1000000.BIN<CR>

### 2.3.2 Wildcard Characters

The two symbols \* and ? may be used in PIP to represent, respectively, complete fields and single characters. These symbols are referred to as wildcard characters; their use is described in the following paragraphs.

2.3.2.1 THE ASTERISK SYMBOL - The asterisk symbol \* may be used to replace a filename or extension:

1. name field (e.g. \*.ext),
2. extension field (e.g. name,\*),
3. both filename fields (e.g., \*.\*).

For example, the filename FILEA.MAC, which specifies the MACRO source language file named FILEA, may be altered by the use of the asterisk in the following manner:

1. \*.MAC specifies all files with the extension .MAC.
2. FILEA.\* specifies all files with the name FILEA, and,
3. \*.\* specifies all files.

2.3.2.2 THE QUESTION MARK SYMBOL - The character ? may be used to indicate a wild character in file names and extensions. The symbol ? replaces characters of a filename to mask out any or all of the characters of a name, extension or both the name and extension fields of a file. When PIP processes a filename which includes ? characters, it ignores the wildcard characters. This masking capability enables the user to specify, with one command, groups of files whose filenames have common characters identically positioned within their filenames. For example, assume that the device DTAL contains the files TEST1.BIN, TEST2.BIN, TEST3.BIN and TEST4.BIN; the user can specify all of these files with one file specification:

```
DTAL:TEST?.BIN
```

2.3.2.3 COMBINING \* AND ? WILDCARD SYMBOLS - The symbols \* and ? can be combined in filenames to specify specific groups of files which have common characteristics in either or both of their name or extension files.

For example, the file specification

ABC???.\*

specifies all files having the character group ABC as the first three characters of its filename. Again, the file specification

\*.??A

specifies all files having an extension which has the character A as its third character.

In combining the \* and ? symbols, the user should remember that for:

- a. filenames, \* is equivalent to ??????, and
- b. extensions, \* is equivalent to ???.

For example, the filenames \*.\* and ??????.??? are equivalent.

#### 2.4 DIRECTORY IDENTIFIER

The [directory] identifier is used in PIP commands to identify a specific:

- a) User File Directory (UFD),
- b. Sub File Directory (SFD), or
- c) a specific UFD-SFD directory path.

The item identified by a given [directory] identifier can be a directory or an item located within a directory which belongs to either the current user or, when the protection code scheme permits, to another user. (Refer to paragraph 2.5 for a description of protection codes.)

A [directory] identifier can consist of a project, programmer number pair (abbreviated as proj,prog) and the names of SFDs. The most expanded form of the [directory] identifier is:

[proj,prog,SFD1,SFD2,...SFDn]

As shown, a [directory] identifier is always enclosed within square brackets and its elements are delimited by commas.

### 2.4.1 UFD-Only Identifiers

Each UFD is identified in the system by the project, programmer number pair assigned to the user for whom the UFD was created. A [directory] identifier for a UFD has the form

[proj,prog]

UFD [directory] identifiers may be written without either one or both of the project, programmer numbers. In such cases, PIP assumes either a previously specified default number or the number assigned to the current user. For example, assume that the current user is logged in under the number pair [57,124] and that no default identifier has been specified. The current user can use [directory] identifiers having any of the following formats:

|    | The Format: | Which is Interpreted by PIP as: |
|----|-------------|---------------------------------|
| 1) | [ , ]       | [57,124]                        |
| 2) | [57, ]      | [57,124]                        |
| 3) | [ ,124]     | [57,124]                        |

### 2.4.2 SFD (Full Directory Path) Identifiers

A Sub File Directory (SFD) is identified by its user-assigned name plus the project, programmer number pair which identifies the UFD in which it is located. A [directory] identifier for an SFD then has the form

[proj,prog,SFDname]

Whenever an SFD is located in a UFD which has a multi-level directory arrangement, the UFD containing the desired SFD must be included in the [directory] identifier for the desired SFD. A [directory] identifier for an SFD in a multi-directory level UFD has the form

[proj,prog,SFD1,SFD2,...SFDn]

and is referred to as a full directory path identifier. For example, assuming that the current UFD is identified by the proj,prog number pair 57,124 and has the following directory organization:

|         |      |      |      |
|---------|------|------|------|
| Level 1 |      | UFD  |      |
| Level 2 |      | SFDA |      |
| Level 3 | SFD1 |      | SFDB |
| Level 4 | SFD2 |      | SFDC |

the [directory] identifier for SFD2 is written as

[57,124,SFDA,SFD1,SFD2]

The proj,prog number pairs in full directory path identifiers may be written using the format variations described in paragraph 2.4.2. However, when no proj,prog numbers are specified by the user, two commas must be used in the identifier in the following manner

[,,SFD1,...SFDn]

The first comma represents the delimiter between the proj,prog numbers; the second represents the delimiter between the last number (prog) and the first SFD name.

### 2.4.3 Specifying Default and Current [Directory] Identifiers

The position in which a [directory] identifier is given in a PIP command determines if it is viewed as a default identifier for all subsequent file specifications given in that command or is the current identifier for an individual file specification.

If a [Directory] identifier is given before one or more file specifications of a command it regarded as the DEFAULT identifier for those specifications. For example, in a command segment having the form:

[directory A] File Specification 1,File Specification 2

the identifier [directory A] is the default for both File Specifications 1 and 2.

If a [Directory] identifier is given after the filename within a File Specification it is viewed as the current identifier for that file specification and will override any given default [directory]. The form of a file specification with the current identifier specified is:

dev:filename.ext[directory]

Both default and current [directory] identifiers can be specified in the same PIP command. For example, the PIP command source segment:

```
=dev:[directory A]filename.ext,dev:filename.ext[directory B]<CR>
```

is valid. In the foregoing example, the identifier [directory A] is the default identifier for the first file specification; and will act as the default identifier for the second file specification if [directory B] is not given. When [directory B] is given, it overrides the default identifier and is accepted as the identifier for the second file specification.

## 2.5 FILE ACCESS PROTECTION CODES

Three-digit (octal) protection codes which specify the degree of access that each of three possible types of users may gain to a file can be specified in the destination side of a PIP command string. File access protection codes are written within angle brackets and must contain three digit positions (e.g., <nnn>). Each digit within a protection code specifies the type of access a specific type of user may have to the file or files involved. Considering the protection code <n1n2n3> the digits give the file access code for the following types of users:

- a. n1 = File OWNER
- b. n2 = project MEMBER, and
- c. n3 = OTHER system users.

The user types are defined as follows:

1. FILE OWNERS. Users who are logged in under either:
  - a. the same programmer number as that of the UFD which contains the file; or
  - b. the same project and programmer number as associated with the UFD which contains the file.

The decision as to which of the above items defines an OWNER is made at Monitor Generation time.

2. PROJECT MEMBER. Users who are logged in under the same project number as that which identifies the UFD containing the file.

- 3. OTHER USERS, any user of the system whose project and programmer number do not match those of the UFD containing the file in question.

File access protection codes are placed in PIP commands after the destination filename of the file involved. For example, the command

```
DPA3:FILEA.BIN<nnn>=DSK:SOURCE.BIN<CR>
```

copies the contents of file SOURCE.BIN onto disk pack device DPA3 under the name FILEA.BIN with an assigned file protection code of nnn.

### 2.5.1 Digit Numeric Protection Code Values

Each of the digits in a 3-digit file protection code may be assigned an encoded numeric value ranging from 0 to 7. The meaning of each octal value is:

| <u>Code Value</u> | <u>Permitted Operations</u>                                      |
|-------------------|------------------------------------------------------------------|
| 7                 | No access privileges. File may be looked up if the UFD permits.  |
| 6                 | Execute only.                                                    |
| 5                 | Read, execute.                                                   |
| 4                 | Append, read, execute.                                           |
| 3                 | Update, append, read, execute.                                   |
| 2                 | Write, update, append, read, execute.                            |
| 1                 | Rename, write, update, append, read, execute.                    |
| 0                 | Change protection, rename, write, update, append, read, execute. |

Files are afforded the greatest protection by the code value 7; the least protection by 0. It is always possible for the owner of a file to change the access protection associated with that file even if the owner-protection field is not set to 0; thus, the values 0 and 1 are equivalent for the owner. Files with their owner-protection field set to 1 are preserved (i.e., saved by .KJOB/K).

It is recommended that important files such as source files be assigned an owner-protection code of 2. This level of protection will prevent the file from being accidentally deleted by permitting them to be edited.

## 2.6 UFD AND SFD PROTECTION CODES

When a user directory (UFD or SFD) is created, it is assigned a 3-digit octal access protection code by either the owner of the file or, by default, the system. The 3-digit code specifies the type of access permitted to the directory by each of the three possible classes of users (i.e., OWNER, MEMBER, or OTHER). (Refer to paragraph 2.5 for a description of user classes.)

Once assigned, a directory access protection code may be changed by the owner and, if the protection code permits (i.e. CREATES allowed), by users other than the owner. (Refer to the description of the PIP rename option given in paragraph 3.5.3.1 for the procedure required to change directory protection codes.)

The access protection code assigned each user class may range from 0 through 7; the following table lists the codes and the operations which each permits.

| CODE | PERMITTED OPERATION(S)                                                          |
|------|---------------------------------------------------------------------------------|
| 0    | Access not permitted.                                                           |
| 1    | The directory may be read as a file.                                            |
| 2    | CREATES are permitted.                                                          |
| 3    | The directory may be read as a file and CREATES are permitted.                  |
| 4    | LOOKUPS are permitted.                                                          |
| 5    | The directory may be read as a file and LOOKUPS are permitted.                  |
| 6    | CREATES and LOOKUPS are both permitted.                                         |
| 7    | The directory may be read as a file and both CREATES and LOOKUPS are permitted. |

## SECTION 3

### STANDARD PIP SWITCHES

#### 3.1 OPTIONAL PIP FUNCTIONS

PIP provides the user with a group of optional functions which can be executed during the performance of the primary PIP transfer function.

Each optional function is assigned an identifier which, when added as a "switch" to a PIP command, initiates the execution of the identified function.

For the purposes of this manual, the PIP optional functions are divided into standard and special groups. The standard group of options described in this section consist of switches which:

1. determine which files are transferred;
2. edit all the data contained by each source file;
3. define the mode of transfer;
4. manipulate the directory of a directory-type device.

All optional functions which deal with non-directory devices and which perform functions other than those listed above are considered special and are described in Section 4.

##### 3.1.1 Adding Switches to PIP Commands

All switches in PIP commands must be preceded by a slash (i.e., /sw); for example, the optional function identified by the letter w is added to a PIP command:

```
*DTA1:DESTFL.BIN/w=DSK:FILEA.BIN,FILEB.BIN<CR>
```

When more than one switch is to be added to a command, they may be listed either separated by slashes (e.g., /B/X....) or enclosed in parentheses (e.g., (BX)).

### 3.2 BASIC TRANSFER FUNCTION

The basic function performed by PIP is the interchange (i.e., read/write transfer) of files or data blocks between devices. There are two types of transfer operations:

1. An optional X-switch transfer in which the source files or blocks are transferred as separate files to the destination device.
2. A non-X type in which all files or blocks transferred from the source device are combined (i.e., concatenated) into a single file on the destination device.

#### 3.2.1 X-Switch Copy Files Without Combining

The use of the X-switch enables the user to move (copy) a group of source files onto the destination device as individual files without changing their creation dates, time dates, filenames and filename extensions. The following are examples of how the X-switch is used in PIP:

1. To transfer all the user's disk files to a DECTape, type:

```
DTA1:/X=DSK:*. *<CR>
```

Assuming that there are three files on the user's disk area named FILEA, FILEB, FILEC.REL, these files will be transferred to DTA1 and can be referenced on DTA1 by those names.

One significant difference between the disk and all other devices is file protection. If the disk is the source device, PIP will by-pass those protected files to which the current user is not permitted access. A suitable message is then issued by PIP if the rest of the command string is successfully executed. Similar processing is described later for the L, Z and D switches. If none of these switches is given, a requested DSK file which is protected will cause termination of the request.

2. To transfer all the files from card reader to disk, type:

```
DSK:/X+CDR: *<CR>
```

When transferring files from the card reader with the \* command, the input files must either be wholly ASCII or wholly binary.

3. To transfer two specific files from user [11,7]'s disk area to a DECTape, type:

```
DTA2:/X=DSK:[11,7]FILEA,REL.FILEA.MAC<CR>
```

4. To copy files from a paper tape onto a directory-type device, the user may employ either:

- a. A copy command in which the number of files to be read are specified by adding a series of commas to the command after the source device name (i.e., PTR,,,,,,). The number of commas required is always one less than the total number of files to be transferred. For example, the command:

```
DSK:/X=PTR:,,,,,<CR>
```

specifies that five (5) files are to be copied from paper tape and written, individually, into the current user's disk area.

- b. A copy command in which all the files contained by a paper tape are to be copied onto a specified device. For example, the command

```
DSK:/X=PTR:*<CR>
```

specifies that all files contained on the paper tape loaded as PTR are to be copied into the current user's disk area. Whenever a command of this type is used, the last file on the paper tape must be followed by two consecutive end-of-file codes.

#### NOTE

In both the foregoing examples, PIP will generate any needed destination filenames. This function is described in paragraph 3.2.1.1.

Whenever the X-switch is used and is not combined with an editing option, PIP transfers any file involved as it appeared on the source device. X-switch operations are copy operations and are referred to as such.

3.2.1.1 NON-DIRECTORY TO DIRECTORY COPY OPERATION - In copying files from a non-directory device onto a directory-type device, PIP must perform special operations in naming the destination files. For example, a special case of source and destination filenames arises in the command:

```
DTA2:FNME.EXT/X=MTAØ:*<CR>
```

Here, every file is to be copied from a non-directory device (MTAØ) to a directory device (DTA2) without combining files (/X). Only one destination filename is given (i.e., FNME.EXT) but the source device (MTAØ) may contain more than one file. If more than one file is transferred, it is necessary for PIP to generate a unique filename for each copied file. PIP generates filenames by developing a 6-character name field in which the first three characters are either:

1. the first three characters of a given destination filename, or
2. the characters "XXX" if no destination filename is given in the command.

The second portion of the PIP-generated name field consists of the decimal numbers ØØ1 through 999 which are added, in sequence, to each filename developed during the /X copy operation.

For filename extensions, PIP uses either the extension of a given destination filename or a null field if no filename is given in the command.

For example, assuming that three files are present on MTAØ, the command:

```
DTA2:FNME.EXT/X=MTAØ:*<CR>
```

transfers the files to DTA2 and establishes the following names in the DECTape directory for the files copied:

1. FNMØØ1.EXT,
2. FNMØØ2.EXT,
3. FNMØØ3.EXT.

If, in the above example, the command given did not include a destination filename (i.e., DTA2:/X=MTAØ:\*<CR>) the copied files would have been named:

1. XXXØØ1
2. XXXØØ2
3. XXXØØ3

The use of the 3-digit decimal number for the last three characters of the filename name gives the user 999 possible input files from non-directory devices. If PIP finds more than 999 files on the source device it will terminate the transfer operation after the 999th file is copied and will issue the error message

?TERMINATE/X,MAX OF 999 FILES PROCESSED.

Any error messages referring to individual files named by PIP (either input or output) will use the generated filename.

3.2.1.2 ASSIGNING NAMES TO DECTAPE TAPES - A tape mounted on a specified DECTape unit can be assigned an identifier during copy operations. Identifiers are from 1 to 6 character names (any SIXBIT character - except ↑ - within the code range 40-137 can be used) which are added to the DECTape's directory (128th word). DECTape identifiers can be read by PIP, FILEX and DIRECT programs; the Monitor does not read identifiers. A DECTape identifier is assigned by adding the selected name to a PIP command when the DECTape to be named is mounted on the specified destination device.

The format required for a DECTape identifier is

↑name↑

A DECTape identifier is inserted into a PIP command following the given destination device name:

dev:↑name↑=source file specification(s)

For example, the command

\*DTA3:↑MYFILE↑/X=DTA1:\*. \*

specifies that the DECTape on device DTA3 be given the identifier "MYFILE" and receive copies of all the files contained by the tape on device DTA1.

### 3.2.2 DX-Switch, Copy All But Specified Files

When the DX-switch is added to a PIP command it causes all the files to be copied from the source device to the destination device except

those files which are named in the command string. If the source device is DSK, a maximum of 10 source-file specifications are allowed. Only directory-type devices are allowed as source devices; no check is made on the existence of the files which are not to be copied. Only one source device is permitted; for example, the command

```
DTA1:(ZDX)=DSK:*.LST,*.SAV,CREF.CRF<CR>
```

zeroes out the directory of DTA1 and transfers to DTA1, from the disk, all files except CREF.CRF and all files with either the extension .LST or .SAV.

### 3.2.3 Transfer Without X-Switch (Combine Files)

When the X-switch is not included in a PIP command all files or blocks transferred from the source device are combined into a single file on the destination device. For example:

1. To combine three paper tape files into one, type

```
PTP:=PTR:,,<CR>
```

2. To combine two files on DECTape into one on another DECTape, type

```
DTA3:FILCOM=DTA2:FILE,FILB<CR>
```

3. To combine files from two DECTapes into one on the user's disk area, type

```
DSK:DSKFIL=DTA2:ONE,DTA4:TWO.MAC<CR>
```

4. To combine all the files on MTAØ into one file on the user's disk area, type

```
DSK:TAPE.MAC=MTAØ:*<CR>
```

(This assumes that MTAØ is positioned at the Load Point).

### 3.2.4 U-Switch, Copy DECTape Blocks Ø, 1 and 2

The U-switch is used during DECTape-to-DECTape copy operation to specify that Blocks Ø, 1 and 2 of the source tape are to be copied onto the destination tape.

This switch is commonly used to transfer DTBOOT from one tape to another. For example, the command:

DTA1:/U=DTA5:<CR>

transfers blocks 0 through 2 of DTA5 to DTA1.

### 3.3.1 A-Switch, Integral Output Lines (Line Blocking)

The use of the A-switch (/A) in a PIP command specifies that each output buffer is to contain an integral number of lines, no lines are to be split between physical output buffers. Line blocking is required for FORTRAN ASCII input. Each line starts with a new word.

### 3.3.2 C-Switch, Delete Trailing Spaces and Convert Multiple Spaces to Tabs

The addition of a C-switch (/C) to a PIP command causes groups of multiple spaces in the material being copied to be replaced by one or more TAB codes; trailing spaces are deleted.

The conversion of the spaces to TAB codes is performed in relation to the standard line TAB "stop" positions located at 8-character intervals throughout the line. Only those groups of multiple spaces which precede a TAB "stop" will produce a TAB code. For example:

1. [space][stop]--will not produce a TAB code.
2. [space][space][stop]--will produce [TAB].
3. [space][space][stop][space][space]--will produce [TAB]  
[space][space]

A totally blank input line is replaced by one space when this switch is used. The C-switch is used to save space when storing card images in DSK file structures. The conversion of spaces to tabs must be done with care since it could alter Hollerith text.

### 3.3.3 E-Switch, Ignore Card Sequence Numbers

This switch, normally used when a card reader is the source device, causes characters (i.e., columns) 73 through 80 of each input line to be replaced by spaces.

### 3.3.4 N-Switch, Delete Sequence Number

This switch causes line sequence numbers to be deleted from any ASCII file being transferred. Line sequence numbers are recognized

as any word in the file in which bit 35 is a binary 1 and follows a carriage return, vertical TAB, form feed for start-of-file identification. Nulls used to fill the last word(s) of a line are ignored. If a line sequence number is followed by a TAB, the TAB is also deleted.

### 3.3.5 S-Switch, Insert Sequence Numbers

This switch causes a line sequence number to be computed and inserted as the output buffer at the start of each line. Sequence numbers are indicated by a 1 in bit 35 of a word following a carriage return, a vertical TAB or start-of-file indicator.

Sequence numbers assigned by PIP take the form nnnnn, starting at 00010 and ranging through 9990 in increments of 10. Approximately one-third of each output buffer is left blank to facilitate editing operations on the file (DTA only).

### 3.3.6 O-Switch, Insert Sequence Numbers and Increment By 1

This switch causes the same operations to be performed as those for switch S, (see 3.3.5) except that the assigned sequence numbers are incremented by 1 instead of 10.

### 3.3.7 P-Switch, Prepare FORTRAN Output for Line Printer Listing

This switch causes PIP to take output generated by a FORTRAN program, which was output on a device other than the line printer (LPT), for which it was intended, and performs the carriage control character interpretations needed when the data is sent to the LPT. The first character in each input line is interpreted by PIP according to the following table.

FORTRAN CARRIAGE CONTROL CHARACTER INTERPRETATION

| Carriage Control Character Produced by FORTRAN Program | ASCII Character(s) Substituted | Line Printer Action                                                        |
|--------------------------------------------------------|--------------------------------|----------------------------------------------------------------------------|
| space                                                  |                                | Skips to next line (single space) with a FORM FEED after every 6Ø lines.   |
| \                                                      |                                |                                                                            |
| *                                                      | Ø23                            | Skips to next line with no FORM FEED.                                      |
| +                                                      | Ø15                            | Precede line with a carriage return only (i.e., over-print previous line). |
| , (comma)                                              | Ø21                            | Skips to next 1/3Øth of page.                                              |
| =                                                      | Ø15,Ø12,Ø12                    | Skips two lines.                                                           |
| .                                                      | Ø22                            | Skips to next 1/2Øth of page.                                              |
| /                                                      | Ø24                            | Skips to next 1/6th of page.                                               |
| Ø                                                      | Ø15,Ø12                        | Skips 1 line (double space).                                               |
| 1                                                      | Ø14                            | Skips to top of next page (page eject).                                    |
| 2                                                      | Ø2Ø                            | Skips to next 1/2 page.                                                    |
| 3                                                      | Ø13                            | Skips to next 1/3 page (also vertical tab).                                |

3.3.7.1 COPY FORTRAN BINARY FILES - The binary mode switch (/B) can be combined with /P in a PIP command to enable the user to obtain a copy of a FORTRAN binary file. The /B/P switch combination is needed when copying FORTRAN binary file(s) from a DECTape source onto a Disk in order to insert a needed control word into each physical buffer. The /B/P switch combination is not needed if both the source and destination devices have the same buffer size. The format for a FORTRAN binary file copy command is

dev:name.ext/B/P=dev:name.ext...<CR>

### 3.3.8 T-Switch, Delete Trailing Spaces

This switch causes all trailing spaces to be deleted from the file being transferred. If a transfer line consists of nothing but spaces, then a single space and a line terminator will be retained in its place in the copied file.

### 3.3.9 W-Switch, Converts Tabs to Spaces

The addition of a W-switch (/W) to a PIP command causes each TAB code contained by the material being copied to be converted to one or more sequential spaces.

The number of spaces produced when a TAB code is converted is determined by the position of the TAB in relation to the standard line TAB "stops". Each line has TAB stops positioned at 8-character intervals throughout the length of the line. When a TAB is converted in a /W switch operation, only enough spaces are produced to reach the next sequential line TAB stop position. For example, the series

```
[stop]ABCD[TAB]
```

is converted to

```
[stop]ABCDspspspsp[stop]
```

where:

```
sp = space.
```

The use of the W-switch causes files previously edited by the use of a C-switch to be restored to their original form (less the deleted trailing spaces).

### 3.3.10 V-Switch, Match Angle Brackets

This switch is not a true edit switch, because the input file is not edited. The use of this switch generates an output file which contains the results of cumulative matching of angle brackets located in the input file. If a line in the input file contains brackets which are not needed to match earlier brackets and which match each other, no output occurs. In all other cases where brackets occur,

a cumulative total and the line currently considered are printed. The symbol > scores a negative count; the symbol < scores a positive count. A typical use for this switch is to check source input to the MACRO-1Ø Assembler; for example, assuming that the file A contains:

```
ONE<<>
TWO<
THREE>
FOUR<>>
FIVE<>
SIX>
```

The request

```
LPT:=DTA2:A/V<CR>
```

results in the Line Printer output:

```
1 ONE<<>
2 TWO<
1 THREE>
Ø FOUR<>>
-1 SIX>
```

From this general example, the most likely conclusion is that there is either a < missing or an extra > in this file. Line five (i.e., FIVE <>) was not printed because the brackets which it contained were matched.

### 3.3.11 Y-Switch, DECTape to Paper Tape

The Y-switch enables the user to transfer DECTape files having the filename extension .RMT, .RTB or .SAV onto SAVE-formatted RIM1Ø or RIM1ØØ paper tapes. The type and contents of the paper tape produced in a Y-transfer is determined by the source file filename extension. If the extension is:

1. .RMT, - A RIM1Ø paper tape (with terminating transfer word) is produced;
2. .RTB, - A RIM1ØØ paper tape (with RIM loader and terminating transfer word) is produced;
3. .SAV, - A RIM1ØB paper tape is produced (with neither RIM loader nor terminating transfer word).

For example, the command

```
PTP:/Y=DTA2:TESTI.RTB<CR>
```

will punch a RIM1ØB tape as described in item 1 of the foregoing description from DECTape file TESTI.RTB.

Switches D and X may be used in conjunction with the Y-switch.

It is assumed that .RTB, .RMT and .SAV files are all in the standard "save" file format. In particular, it is assumed that no block of an .RMT saved file overlaps a preceding one.

#### NOTE

Optional switch Y is obtained by setting RIMSW=1 at assembly time (see source file PIP.CTL.).

The functions performed by PIP during /Y transfers in response to each possible type of source file filename extension are:

1. An .RTB file causes PIP to:
  - a. Punch a RIM loader.
  - b. Punch an I/O word (-n,x) at the start of each data block. The variable n is the number of data words punched in each block and has the octal value 17, or less. The variable x is the starting address-1 for loading the following data. Successive values of x are derived from the pointer words in the DECTape blocks. The first value of x is the value of the right side of the first pointer word in the DECTape file.
  - c. The complete DECTape file is punched as described in item b.
  - d. The final block punched is followed by a block containing a transfer word. If the right half of .JBSA contains Ø then a halt is punched. If the right half of .JBSA contains a non-zero value, a jump to that address is punched.
2. A .SAV file is treated in the same way as one having .RTB extension except that no RIM loader and no transfer word are punched.
3. An .RTM file initiates PIP functions which are similar to those described for .RTB files but which have the following differences:
  - a. Only one IOWD is produced, (-n,x) where (n-1) data words and a transfer instruction follow.
  - b. The first of the (n-1) data words punched from the saved file is the first word of the logical block which contains location .JBDA (i.e., the first location after the end of the JOBDATA area).

- c. The variable x is then set to the starting address (address-1) of the first data word found. The effective program length is determined by the relationship  $n = (.JBFF) - x$ . Data is now transferred from (x+1) until (n-1) words have been punched.
- d. Zero fill is used if a pointer word in a source block indicates noncontinuous data. The transfer word, calculated as described for .RTB files terminates the output file.

### 3.4 SET DATA MODE, SWITCHES B, H AND I

The addition of optional data mode switches to a PIP command specifies the mode in which the file(s) involved must be transferred.

Data modes are device dependent; complete descriptions of their use and effect on different devices are given in the DECSYSTEM-10 Monitor Calls manual.

If both input and output devices can do binary I/O, no editing switches are in force and no concatenation is required. All files are transferred in binary mode (36-bit bytes). If an editing switch that requires PIP to do character processing is used, ASCII mode is used. The data mode switches are:

- 1. /B - initializes the input and output devices in binary mode.

#### NOTE

Since PIP recognizes the following as binary extension, /B is not required when these extensions are used in the PIP command.

#### Binary Extensions Recognized by PIP

|      |      |      |
|------|------|------|
| .BIN | .HGH | .RES |
| .CHN | .INI | .SAV |
| .CKP | .LOW | .SFD |
| .DAF | .QUC | .SHR |
| .DAT | .QUD | .SYS |
| .DCR | .QUE | .UFD |
| .DMP | .QUF |      |

- 2. /H - initializes the input and output devices in image binary mode.
- 3. /I - initializes the input and output devices in image mode.

## 3.5 FILE DIRECTORY SWITCHES

Optional PIP switches whose functions affect user file directories are described in paragraphs 3.5.1 through 3.5.6.

## 3.5.1 L-Switch, List Source Device Directory

## NOTE

The Monitor command DIRECT provides the user with more facilities for obtaining directory-type information than the PIP L-switch option (refer to the DECsystem-10 Monitor Command Manual for details).

This switch enables the user to obtain a listing of the source device directory. The type of output device used affects the directory listing as follows:

1. If the output device is TTY, the directory listing formats for directory-type devices are:
  - a. For DTA source (e.g., TTY:=DTA4:/L<CR>)
 

```

n FREE BLOCKS LEFT
filename.ext no. of blocks creation date
.
.
.
.

```
  - b. For DSK source (e.g., TTY:=DSK:/L<CR>)
 

```

DIRECTORY [directory] (CURRENT TIME) (TODAY'S DATE)
where [directory] is the project-programmer
number of the requested directory.

filename.ext<protection>no.of blocks creation date
.
.
.
.
Total Blks n

```

Asterisk or question mark wildcard symbols (refer to paragraph 2.3.2.2) can be used in either the specified filename or extension fields to cause only those files in the disk directory of a particular filename or extension to be listed. Thus, the command TTY:/L=DSK:\*.REL<CR> causes only those files with extension .REL to be printed in the directory listing.

2. If the output is not TTY, the directory listing is printed in one of the following formats:
  - a. For DTA, source format is as in paragraph 1.(a)
  - b. For DSK, source format is as in paragraph 1.(b) but includes access date and mode as well as the creation time and access date. If any disk file is protected, as much information as possible is given about it.

### 3.5.2 F-Switch, List Limited Source Directory

This switch performs, essentially, the same function as the L-switch; however, only the filenames and extensions of the files in the specified disk or DECTape directory are listed.

#### NOTE

The Monitor command DIRECT provides the user with more facilities for obtaining directory-type information than the PIP F-switch option (refer to the DECSYSTEM-10 Monitor Command Manual for details).

Only DSK: and DTAN: are permitted as source device; if no source device is given, DSK: is assumed.

For example, the command

```
TTY:/F=<CR>
```

lists the directory of the user's disk area as described. The /F switch may work in cases where /L will not because of file access protection.

### 3.5.3 R-Switch, Rename Source Files

The use of this switch causes PIP to rename the source file to the name given as the destination file name. Only one source file specification can be given. If more than one is given, the error message PIP COMMAND ERROR is printed and no action is taken. The destination file specification can take the following forms (protection can always be specified):

1. Filename.extension
2. Filename.\*
3. \*,Extension
4. \*.\*<protection>

5. Filename
6. ??????.ext
7. ??????.???
8. \*.???
9. ??????.\*

In fact, <protection> can always be specified but the request \*.\* (4) has no effect without it. If no protection is specified, the current file protection is not altered.

During a rename operation on device DSK, if PIP finds that the filename to be changed exists on more than one file structure, PIP will output the following message to the user's terminal:

?AMBIGUOUS[file structure list][filename.ext]

The following are examples of the proper use of the /R switch:

1. DSK:MONI.F4/R=MONI.MAC<CR>  
Rename the file MONI.MAC as MONI.F4.
2. DSK:MON2,\*/R=MONA.\*<CR>  
Rename all files of name MONA and any extension to retain the extensions but take the new name MON2.
3. DSK:\*.EXT/R=\*.MAC<CR>  
Rename all files of extension MAC to retain their own names but take the extension EXT.
4. DSK:\*. \*<Ø77>/R=\*.SAV<CR>  
Give all files of extension SAV the protection Ø77
5. DTAL:MON2/R=MONA.REL<CR>  
Rename the file MONA.REL to have the name MON2 and the null extension.

### 3.5.3.1 CHANGING SOURCE UFD OR SFD PROTECTION CODE USING THE RENAME (R) FUNCTION

- The access protection codes assigned to UFDs or SFDs can be changed using the PIP rename switch (/R) if the privileges assigned the current user permits the operation. (Refer to the DECSYSTEM-10 Monitor Calls manual for a detailed description of user UFD and SFD access privileges.) The owner of a directory is always permitted the use of the PIP rename function.

The command format required to change a directory access protection code is

```
*dev:[directory].UFD<nnn>/R=[directory].UFD<CR>
```

where:

1. <nnn> represents the desired (new) protection code.
2. [directory] must be the same on both sides of the command.
3. The user indicates to PIP that the protection code of the identified directory (UFD or SFD) is to be changed by specifying the extension .UFD without a filename. Note that the same extension, .UFD, is used when changing the access protection of an SFD as well as for changing the protection of a UFD.

The following examples illustrate the use of the /R switch in changing the access protection codes of directories.

1. The command:

```
DSKA:[57,123].UFD<222>/R=[57,123].UFD<CR>
```

changes the access code of the UFD identified by the number pair 57,123 to /222.

2. The command

```
DSKA:[57,123,AAA,BBB,111].UFD<222>/R=[57,123,AAA,BBB,111].UFD<CR>
```

changes the access code of the SFD named 111 to the value 222. Note that the last name given in the [directory] identifier is the SFD which is affected by the /R operation.

#### 3.5.4 D-Switch, Delete Files

This switch causes PIP to delete one or more specified files from the device given in the destination side of the PIP command. Only one device can be specified in a delete command; it is assumed that the source and destination devices are the same device.

For example, the following command

```
DSK:/D=FILEA,FILEB,FILEC.MAC,*.REL<CR>
```

causes PIP to delete from the user's disk area files FILEA, FILEB, FILEC.MAC and all files having the extension .REL.

If a nonexistent file is specified in a delete command, PIP prints the error message

```
%filename.ext FILE WAS NOT FOUND
```

and continues to process deletions of the existing specified files. If an existing file is found to be protected it will be skipped and the message

```
?filename.ext (2) PROTECTION FAILURE
```

is printed. If a user has the correct privileges he can delete files from other users' areas.

#### NOTE

An attempt to delete files from a DECTape that is write-locked results in the error message

```
DEVICE dev.name OPR operator station no.  
ACTION REQUESTED
```

being printed at the user's terminal. When a system operator has write-enabled the DECTape unit involved, he will start the requested action and cause the message

```
CONT BY OPER
```

to be printed at the user's terminal.

On completion of a disk delete operation, PIP lists the names of the files deleted and the total number of blocks freed by the deletion.

For example, assume that a file three blocks in length and named FILEA.MAC exists in the current UFD: the command for its deletion and the subsequent messages printed by PIP would appear as:

```
*DSK:/D=FILEA.MAC <CR>      (user command)
FILES DELETED:                (PIP response)
FILEA.MAC                     (PIP response)
3 BLOCKS FREED                (PIP response)
*
```

### 3.5.5 Z Switch, Zero Directory

The use of this switch causes PIP to zero out the directory of the destination's device; a source device does not have to be specified in the command. A Z-switch request is implemented before any other operation specified in the command string in which it occurs. Thus,

```
DTA2:CARDS/Z=CDR:<CR>
```

zeroes out the directory of DTA2 before transferring one file from CDR onto DTA2. The command

```
DTA2:/Z=<CR>
```

zeroes out the directory of DTA2.

If the destination device is the disk, an attempt is made to delete all the files whose names are found in the directory specified. If protection codes prohibit the deletion of some of the files, the request will terminate after as many files as possible have been deleted, and the message

```
?filename.ext(2)PROTECTION FAILURE
```

is printed. The user should then change the protection of the protected files and repeat his request if he wants all files deleted. For example, the command

```
DSK:FLOUT/Z=DTA2:CARY<CR>
```

zeroes out the directory of the user's disk area, transfers file CARY from DTA2 to the disk, and names the disk file FLOUT.

### 3.5.6 Q-Switch, Print Summary of PIP Functions

This switch causes PIP to print on a specified device the system device file SYS:PIP.HLP. This file contains an alphabetical list of all PIP switches and functions. For example, the command

```
LPT:/Q=<CR>
```

causes the following summary to be listed on the line printer:

| <u>PIP</u> | <u>Switches (Alphabetic order) Summary</u>                                                                                                                                                                                               |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A          | Line Blocking                                                                                                                                                                                                                            |
| B          | Binary Processing (Mode)                                                                                                                                                                                                                 |
| C          | Suppress Trailing Spaces, Convert Multiple Spaces to TABs                                                                                                                                                                                |
| D          | Delete File                                                                                                                                                                                                                              |
| E          | Treat (Card) Columns 73-80 as Spaces                                                                                                                                                                                                     |
| F          | List Disk or DTA Directory *FileNames and Ext. only)                                                                                                                                                                                     |
| G          | Ignore I/O Errors                                                                                                                                                                                                                        |
| H          | Image Binary Processing (Mode)                                                                                                                                                                                                           |
| I          | Image Processing (Mode)                                                                                                                                                                                                                  |
| J          | Punch Cards in 029 (Output Device must be CDP)                                                                                                                                                                                           |
| L          | List Directory                                                                                                                                                                                                                           |
| M          | See MTA Switches Below                                                                                                                                                                                                                   |
| N          | Delete Sequence Numbers                                                                                                                                                                                                                  |
| O          | Same as /S switch, except Increment is by 1                                                                                                                                                                                              |
| P          | FORTTRAN output Conversion assumed. Convert format control character for LPT listing. /B/P FORTTRAN Binary                                                                                                                               |
| Q          | Print (this) List of Switches and Meanings                                                                                                                                                                                               |
| R          | Rename File                                                                                                                                                                                                                              |
| S          | Resequence, or Add Sequence Number to File; increment is by 10                                                                                                                                                                           |
| T          | Suppress Trailing Spaces Only                                                                                                                                                                                                            |
| U          | Copy Block 0 (DTA)                                                                                                                                                                                                                       |
| V          | Match parentheses (<>)                                                                                                                                                                                                                   |
| W          | Convert TABs to Multiple Spaces                                                                                                                                                                                                          |
| X          | Copy Specified Files                                                                                                                                                                                                                     |
| *Y         | RIM, DTA to PIP if source extension is RTB Destination format is RIM Loader, RIM 100 file transfer. If source extension is SAV destination format is as RTB - RIM 10B file only. If source extension is RMT destination format is RIM10. |
| Z          | Zero Out Directory                                                                                                                                                                                                                       |

MTA switches:

Enclose in parentheses ( ).

M followed by 8 means select 800 B.P.I. Density  
 5 556 B.P.I. Density  
 2 200 B.P.I. Density  
 E Even Parity  
 A Advance MTAl File  
 D Advance MTAl Record  
 B Backspace MTAl File  
 P Backspace MTAl Record  
 W Rewind MTA or DTA  
 T Skip to Logical EOT  
 U Rewind and Unload MTA or DTA  
 F Mark EOF

(M#NA), (M#NB), (M#ND), (M#NP) mean advance or backspace MTAn files, or records.

\*This is an optional switch obtained by setting RIMSW=1 at assembly time.

3.6 PERMITTED SWITCH COMBINATIONS

The combinations of PIP's standard and special option switches which are permitted in PIP commands are illustrated in the following matrix.

|   | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Notes                |                    |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|----------------------|--------------------|
| A | # | ✓ | # | ✓ | # | ✓ | # | # | # | ? | # | ✓ | ✓ | ✓ | ? | ✓ | # | ✓ | ✓ |   |   | ? | ✓ | ✓ | # | ✓ | ASCII mode           |                    |
| B | # |   | # | # | # | # | ✓ | # | # | # | # | # | ✓ | # | # | * | ✓ | # | # | # |   |   | # | # | ✓ | # | ✓                    | Binary mode        |
| C | ✓ | # |   | # | ✓ | # | ✓ | # | # | ✓ |   |   | ✓ | ✓ | ✓ | ✓ | ✓ | # | ✓ | ✓ |   |   | ? | ✓ | ✓ | # | ✓                    | ASCII mode         |
| D |   |   |   |   |   |   | # |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | * | # | Delete only          |                    |
| E | ✓ | # | ✓ | # |   | # | ✓ | # | # | ✓ |   |   | ✓ | ✓ | ✓ | ✓ | ✓ | # | ✓ | ✓ |   |   | ✓ | ✓ | # | ✓ | ASCII mode           |                    |
| F |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | List Directory only  |                    |
| G | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |   | ✓ | ✓ | ✓ |   | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |   |   | ✓ | ✓ | ✓ | ✓ | Always legal         |                    |
| H | # | # | # | # | # | # | ✓ |   | # |   |   |   | ✓ | # | # | # | # | # | # |   |   |   |   | # | ✓ | ✓ | Binary mode          |                    |
| I | # | # | # | # | # | # | ✓ | # |   |   |   |   | ✓ | # | # | # | # | # | # |   |   |   |   | # | ✓ | ✓ | Binary mode          |                    |
| J | ✓ | # | ✓ | # | ✓ | # | ✓ | # | # |   |   |   | ✓ | ? | ? | ? | # | ? | ✓ | # |   |   | ✓ | ✓ | # | ? | ASCII mode           |                    |
| K |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | Unused               |                    |
| L |   |   |   |   |   | # |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | ✓ | List Directory only  |                    |
| M | ✓ | ✓ | ✓ | # | ✓ | # | ✓ | ✓ | ✓ | ? |   | ✓ |   | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |   |   | # | ✓ | ✓ | ✓ | ?                    | Magnetic Tape Only |
| N | ✓ | # | ✓ | # | ✓ | # | ✓ | # | # | ? |   |   | ✓ |   | ? | ? | ✓ | # | ? | ✓ |   |   |   | ✓ | ✓ | ✓ | ASCII mode           |                    |
| O | ✓ | # | ✓ | # | ✓ | # | ✓ | # | # | ? |   |   | ✓ | # |   | ? | ✓ | # | # | ✓ |   |   |   | ✓ | ✓ | ✓ | ASCII                |                    |
| P | * |   |   |   |   | # |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | ✓ | FORTTRAN             |                    |
| Q | ✓ | ✓ | ✓ | # | ✓ | # | ✓ | ✓ | ✓ |   |   | ? | ✓ | ✓ | ✓ |   | # | ✓ | ✓ |   |   |   | ✓ | ✓ | ✓ | ✓ | Prints file PIP.HLP  |                    |
| R |   |   |   |   |   | # |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | ✓ | # | Rename only          |                    |
| S | ✓ | # | ✓ |   |   | # |   |   |   |   |   |   |   |   |   |   |   |   |   | ✓ |   |   |   |   |   | ✓ | ASCII mode           |                    |
| T |   | # | ✓ |   |   | # |   |   |   |   |   |   |   |   |   |   |   |   |   | ✓ |   |   |   |   |   | ✓ | ASCII mode           |                    |
| U |   |   |   |   |   | # |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | ✓ | DTA only             |                    |
| V | ✓ | ? | ✓ | # | ✓ | # | ✓ | ? | ? | ✓ |   |   | ✓ | ✓ | ✓ | # | # | ✓ | ✓ |   |   |   | ✓ |   | ✓ | ✓ | ASCII mode           |                    |
| W |   | # | ✓ |   |   | # |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | ✓ | ASCII mode           |                    |
| X | ✓ | ✓ | ✓ | * | ✓ | # | ✓ | ✓ | ✓ | ✓ |   | # | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |   |   |   |   | ✓ | ✓ | ✓ | ASCII or Binary mode |                    |
| Y |   |   |   |   |   | # |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | ✓ | Binary mode          |                    |
| Z | ✓ | ✓ | ✓ | # | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |   | ✓ | ✓ | ✓ | ✓ | ✓ | # | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | Output only          |                    |

LEGEND:      Symbol                      Meaning

                 ✓                      A permitted combination

                 ?                      A permitted but unlikely combination

                 #                      Not permitted

                 \*                      Special purpose combination

                 Blank                      Untested or unused combination



SECTION 4

SPECIAL PIP SWITCHES

4.1 SPECIAL PIP FUNCTIONS

This section contains descriptions of optional PIP functions used in magnetic tape, error recovery and card punch operations.

4.2 MAGNETIC TAPE SWITCHES

When magnetic tape is used in a file transfer, PIP can set the tape parity and density parameters and position the tape reels. In PIP commands, magnetic tape switches apply to only one particular magnetic tape unit or file specification.

The optional PIP magnetic tape (MTA) switches are written enclosed in parentheses; the letter M is used as the first character of all optional switches or series of switches (e.g. (Msw) or (Mswlsw2..)).

MTA switches must appear within the command file specifications of the particular file to which they refer. Thus, MTA switches refer to a particular device and, except for density and parity selections, to a particular file specification of that device.

4.2.1 Switches for Setting Density and Parity Parameters

The default Monitor density of 800 bits-per-word (bpi) and odd parity are assumed unless either the Monitor SET DENSITY command was given or one of the following switches is included in the PIP command file specifications:

| <u>Switch</u> | <u>Meaning</u>                      |
|---------------|-------------------------------------|
| (M8)          | 800 bpi density (default value)     |
| (M5)          | 556 bpi density                     |
| (M2)          | 200 bpi density                     |
| (ME)          | Even parity (odd parity is default) |

The following command string causes PIP to transfer a file from MTAl to MTA2 at 200 bpi, with even parity (and in ASCII line mode)

MTA2: (M2E)=MTAl (ME2) <CR>

## 4.2.2 Switches for Positioning Magnetic Tape

The following switches are used in PIP command strings for magnetic tape handling:

| <u>Switch</u> | <u>Function Performed</u>       |
|---------------|---------------------------------|
| (MA)          | Advance tape reel one file.     |
| (MB)          | Backspace tape reel one file.   |
| (MD)          | Advance tape reel one record.   |
| (MP)          | Backspace tape reel one record. |
| (MW)          | Rewind tape reel.               |
| (MT)          | Skip to logical End-of-Tape.    |
| (MU)          | Rewind and unload.              |
| (MF)          | Mark End-of-File.               |

In PIP MTA commands, the source device need not be given. For example, to rewind MTAL:, type

```
MTAL:(MW)=<CR>
```

If a source device is specified in the command string, information transfer will occur, except when PIP is requested to rewind and unload a magnetic tape.

Several magnetic tape functions may be specified in a single command string. Density or parity, when changed, will appear in the file specification. In the following example, density is set to 2000 bpi, parity is even, the tape is to be rewound and the first, third, fourth and fifth files on that reel are to be printed on the line printer.

```
LPT:=MTAL:(M2EW),(MA),,<CR>
```

If multiple backspace, advance file or record movements are needed, the number of movements required is specified by #n (interpreted as decimal). All positioning switches are implemented before any related file transfers are made; thus MTAL:(M#3A)-PTR: will advance MTAL by three files before transferring a paper tape file to it.

1. If a backspace file (M#nB) request is given, after completion of "n+1" backspace files one advance file request is made unless the tape is at Load Point. In this way the tape is always initially positioned at the beginning of a file. Thus, the command:

```
MTAØ:(MB)=<CR>
```

will backspace MTAØ to the start of the previous file.

2. If the Load Point is reached before a backspace file or record request is completed, an error diagnostic will terminate the run and the following error message is printed

?LOAD POINT BEFORE END OF BACKSPACE REQUEST?

3. Only one MTA movement per file specification is allowed in a command string. Thus:

MTAØ: (MT#2B)=...<CR>

is illegal since it requests two distinct types of MTA movement.

4.2.2.1 BACKSPACE TO START OF CURRENT FILE - The specification of Ø as the value of n in a multiple backspace command (e.g., M#ØB) causes the tape to be backspaced to the start of the current file. The use of M#ØB is not the same as MB, switch MB is equivalent to M#1B.

4.2.2.2 ADVANCE TO END OF CURRENT FILE - The specification of Ø as the value of n in a multiple advance command (e.g., M#ØA) causes the tape to be moved to a point just before the EOF marker of the current file. The use of M#ØA is not the same as MA, switch MA is equivalent to M#1A.

#### NOTE

The advance and backspace record requests are available as a convenience for the knowledgeable user, and should be approached with caution. Always remember that PIP typically has multiple input and output buffers and the physical position of the tape need not correspond to the physical position of the record currently being processed.

#### 4.3 G-SWITCH, ERROR RECOVERY

If the error recovery switch /G is present in a command string, a specific set of I/O errors will be acknowledged by error messages. The I/O errors affected by the presence or absence of /G are listed in Section 5, paragraph 5.2, item 3 of the error messages, and are flagged by an asterisk (\*). Processing will continue after the error message is printed as though no error had occurred. Thus, must I/O errors occurring within a file may be overridden. However, if the same error condition occurs in each buffer of the file, the error

message is repeated for each buffer until either the end of file occurs or the error condition disappears. A disk directory is used as an input file if it is read to be either listed or searched and is obtained as a core image from the Monitor; therefore, it is not subject to the input errors which may be diagnosed by PIP. However, I/O errors can occur for DECTape directories and are diagnosed at the Monitor level when a directory is read or written. This is, typically, on a LOOKUP or RELEAS request. If the G-switch is not used, any I/O error will close the current output file and, after printing a suitable message, terminate the current request to PIP.

#### 4.4 J-SWITCH, CARD PUNCH

The J-switch causes cards to be punched in Ø29 mode. The output device specified by the command string must be the card punch (CDP).

SECTION 5

ERROR REPORTING AND ERROR MESSAGES

5.1 ERROR MESSAGES

This section describes the various types of error conditions and error messages that can occur during PIP operations.

The special treatment of recoverable error messages which prevent the current job being prematurely terminated when running under the Batch Processor is also described.

When an error message terminates a PIP run, both the input and output devices are released. This means that all files, fully or partly created, are available on the destination device.

NOTE

All error messages preceded by a question mark (?) indicate a fatal (non-recoverable) error.

5.2 I/O ERROR MESSAGES

I/O error messages are opened with a description of the relevant device and file; for example,

- 1. INPUT                    DEVICE DTA3:FILE FILNAM.EXT...
- 2. OUTPUT                  DEVICE DTA3:FILE FLNAM.EXT...
- 3. DISK DIRECTORY READ...

| <u>Device</u> | <u>Message</u>                     |
|---------------|------------------------------------|
| DTA,DKS,MTA   | WRITE (LOCK) ERROR                 |
| *CDR          | 7-9 PUNCH MISSING                  |
| *OTHER        | BINARY DATA INCOMPLETE             |
| *ALL DEVICES  | DEVICE ERROR                       |
| *ALL DEVICES  | CHECKSUM OR PARITY ERROR           |
| DTA           | BLOCK OR BLOCK NUMBER<br>TOO LARGE |
| *OTHER        | INPUT BUFFER OVERFLOW              |
| *MTA          | PHYSICAL EOT                       |

\_\_\_\_\_  
\*Recoverable error if a G-switch is used, read paragraph 4.3 for a description of /G.

Thus, for the command DTA4:CON.REL=DTA3:CON.REL, if DTA4 is WRITE LOCKed, PIP prints the error message:

```
?OUTPUT DEVICE DTA4:FILE CON,REL WRITE(LOCK)ERROR
```

Other messages for devices are:

1. ?DEVICE dev DOES NOT EXIST (DEVCHR request)
2. ?DEVICE dev NOT AVAILABLE (INIT request)

### 5.3 FILE REFERENCE ERRORS

The following error messages can occur during a LOOKUP, RENAME or ENTER request on disk.

message:?(filename.ext) then one of the following:

- 
- (Ø) FILE WAS NOT FOUND or (Ø) ILLEGAL FILE NAME (used for enter errors only)
  - (1) NO DIRECTORY FOR PROJECT-PROGRAMMER NUMBER
  - (2) PROTECTION FAILURE
  - (3) FILE WAS BEING MODIFIED
  - (4) RENAME FILE NAME ALREADY EXISTS
  - (5) ILLEGAL SEQUENCE OF UUOS
  - (6) BAD UFD OR BAD RIB
  - (7) NOT A SAV FILE
  - (1Ø) NOT ENOUGH CORE
  - (11) DEVICE NOT AVAILABLE
  - (12) NO SUCH DEVICE
  - (13) NOT TWO RELOC REG. CAPABILITY
  - (14) NO ROOM OR QUOTA EXCEEDED
  - (15) WRITE LOCK ERROR
  - (16) NOT ENOUGH MONITOR TABLE SPACE
  - (17) PARTIAL ALLOCATION ONLY
  - (2Ø) BLOCK NOT FREE ON ALLOCATION
  - (21) CAN'T SUPERSEDE (ENTER) AN EXISTING DIRECTORY
  - (22) CAN'T DELETE (RENAME) A NON-EMPTY DIRECTORY
  - (23) SFD NOT FOUND
  - (24) SEARCH LIST EMPTY
  - (25) SFD NESTED TOO DEEPLY
  - (26) NO-CREATE ON FOR SPECIFIED SFD PATH

If the error code (V) is greater than 26<sub>8</sub>, the error message:

```
?(V) LOOKUP,ENTER, OR RENAME ERROR
```

is printed.

Error values are used by the UUO's LOOKUP, ENTER and RENAME. Refer to the DECsystem-10 Monitor Calls manual for complete descriptions of these UUO's.

The following error messages may be given on a reject to an ENTER request on DECTape:

1. The error message printed if there is no room for an entry in a DECTape directory is

?DIRECTORY FULL:

2. The error message printed if a zero filename is given for a DECTape output file is

?ILLEGAL FILE NAME:

The following message is given if a filename is not found in a directory search of disk or DECTape

?NO FILE NAMED filename.ext

#### 5.4 PIP COMMAND ERRORS

The following error messages are output by PIP on the detection of errors in the user command string:

1. ?PIP COMMAND ERROR

Some of the possible causes of this type of error are:

- a. an illegal format for a command string,
- b. a nonexistent switch was requested,
- c. a filename other than \* or \*.\* was given for a non-directory (source) device.

2. ?INCORRECT PROJECT-PROGRAMMER NUMBER:

The project-programmer number must be in the form

[number,number]

where  $\emptyset$ <number<777777<sub>8</sub>, a full path specification must be made if SFD's are involved.

3. ?SFD LIST TOO LONG:

Too many SFD's were listed in the full directory path. A maximum of five levels (not including the UFD) is permitted in a directory path specification.

4. ?ILLEGAL PROTECTION:

The protection number must be in the form <number>, where:  $\emptyset$ <=number<=777<sub>8</sub>.

## 5. ?NO BLOCK Ø COPY

The /U switch was specified, but PIP was not assembled to allow this.

## 6. ?TOO MANY REQUESTS FOR...(magnetic tape)

Conflicting density and/or parity requests were given.

## 5.5 Y-SWITCH ERRORS

The following error messages occur only when the Y-switch is included in the PIP command string:

## 1. ?DTA to PTP ONLY:

Only DECTape input and paper tape output are permitted.

## 2. ?/Y SWITCH NOT AVAILABLE THIS ASSEMBLY:

The /Y switch was specified, but PIP was not assembled to allow this.

## 3. FILE filename.ext ILLEGAL EXTENSION:

The extensions of the filenames given must be .RMT, .RTB or .SAV.

## 4. Filename.ext ILLEGAL FORMAT:

The reasons for getting the diagnostic ILLEGAL FORMAT are:

- a. a zero length file was found,
- b. the required job data information was not available,
- c. a block overlapped a previous block (RIM lØ),
- d. an EOF was found when data was expected,
- e. a pointer word expected but not found in the source file.

## 5.6 GENERAL ERROR MESSAGES

The following is a list of the PIP error messages which are not included in any of the preceding categories:

## 1. ?DISK OR DECTAPE INPUT REQUIRED:

This message is printed when a non-directory source device is specified for a PIP function which requires a directory-type source device.

2. ?filename.ext ILLEGAL FILE NAME:  

This message is output if an attempt is made to ENTER without giving a filename.
3. Errors found during /X, /Z, /D, and /R operations result in error messages which pertain to the specific error found. Error messages for these operations are printed only if no other fatal error occurs before the command string is processed. If another error does occur, its diagnostic takes precedence over the diagnostics for the above switch functions.
4. ?4K NEEDED:  

4K not currently available but is needed (for non-reentrant disk system).
5. ?DECTAPE I/O ONLY:  

The I/O device for a block Ø copy (/U switch) must be a DECTape.
6. ?TERMINATE /X.MAX. OF 999 FILES PROCESSED:  

PIP, during a /X copy function from a non-directory device, has processed 999 files. This is the maximum number of files which such a /X request can handle.
7. ?TOO MANY INPUT DEVICES:  

This error is for the /D and /DX functions; only one input device is allowed when these switches are used. If more than one device is specified in a /D command and the first device given is DSK, the disk files are deleted when this diagnostic is given.
8. ?NO FILE NAMED PIP.HLP:  

The data file requested by a PIP Q-switch is not available on the system device.
9. ?LINE TOO LONG:  

During an ASCII mode file transfer a line containing more than 18Ø characters was detected. This occurs only when switches entailing line processing are given (i.e., /A or /S).
10. ?LOAD POINT BEFORE END OF BACKSPACE REQUEST:  

This diagnostic occurs only if either the MTA (M#nB) or (M#nP) switch is used. If the Load Point is sensed before the "n" backspace files or records function is completed, an error is assumed to have been made by the user.

## 5.7 TMPCOR (DEVICE TMP) ERROR MESSAGES

If the temporary storage facilities provided by the UWO TMPCOR are used or are attempted to be used during PIP operations, the following error messages can occur:

1. ?TMPCOR NOT AVAILABLE:
2. ?NOT ENOUGH ROOM IN TMPCOR:
3. ?COMMAND NOT YET SUPPORTED FOR TMPCOR:
4. nn TMPCOR WORDS FREE

Number of word locations free in the TMPCOR storage area.

Refer to the DECsystem-10 Monitor Calls manual for a description of the UWO TMPCOR.

APPENDIX A

STANDARD FILENAME EXTENSIONS

Table A-1  
Filename Extensions

| <u>Filename Extension</u> | <u>Type of File</u> | <u>Meaning</u>                                                                         |
|---------------------------|---------------------|----------------------------------------------------------------------------------------|
| AID                       | Source              | Source file in AID language.                                                           |
| ALG                       | Source              | Source file in ALGOL language.                                                         |
| ALP                       | ASCII               | Printer forms alignment.                                                               |
| BAC                       | Object              | Output from the BASIC Compiler.                                                        |
| BAK                       | Source              | Backup file from TECO or LINED.                                                        |
| BAS                       | Source              | Source file in BASIC language.                                                         |
| BIN                       | Object              | Binary file.                                                                           |
| BLB                       | ASCII               | Blurb file.                                                                            |
| BLI                       | Source              | Source file in BLISS language.                                                         |
| BNC                       | ASCII               | BINCOM output.                                                                         |
| BUG                       | Object              | Saved to show a program error.                                                         |
| CAL                       | Object              | CAL data and program files.                                                            |
| CBL                       | Source              | Source file in COBOL language.                                                         |
| CCL                       | ASCII               | Alternate convention for command file (@ construction for programs other than COMPIL). |
| CCO                       | ASCII               | Listing of modifications to non-resident software.                                     |
| CKP                       | Binary              | Checkpoint core image file created by COBOL operating system.                          |
| CHN                       | Object              | CHAIN file.                                                                            |
| CMD                       | ASCII               | Command file for indirect commands (@ construction for COMPIL).                        |
| CMP                       | ASCII               | Complaint file by GRIPE.                                                               |
| COR                       | ASCII               | Correction file for SOUP.                                                              |
| CRF                       | ASCII               | CREF (cross-reference) input file.                                                     |

Table A-1  
 Filename Extensions (Cont'd)

| <u>Filename<br/>Extension</u> | <u>Type of<br/>File</u> | <u>Meaning</u>                                                       |
|-------------------------------|-------------------------|----------------------------------------------------------------------|
| CTL                           | ASCII                   | MP batch control file.                                               |
| DAE                           | Binary                  | Default output for DAEMON-taken core dump.                           |
| DAT                           | ASCII,<br>Binary        | Data (FORTRAN) file.                                                 |
| DCR                           | Binary                  | Core image save (DCORE).                                             |
| DDT                           | ASCII                   | Input file to FILDDT.                                                |
| DIR                           | ASCII                   | Directory from FILE command or DIRECT program.                       |
| DMP                           | PDP-6                   | PDP-6 format for a file created by a SAVE command.                   |
| DOC                           | ASCII                   | Listing of modifications to the most recent version of the software. |
| ERR                           | ASCII                   | Error message file.                                                  |
| F4                            | Source                  | Source file in FORTRAN language.                                     |
| FLO                           | ASCII                   | English language flowchart.                                          |
| FRM                           | ASCII                   | Form.                                                                |
| FUD                           | ASCII                   | FUDGE2 listing output.                                               |
| HGH                           | Object                  | Nonsharable high segment of a two-segment program.                   |
| HLP                           | ASCII                   | Help files containing switch explanations, etc.                      |
| INI                           | ASCII,<br>Binary        | Initialization file.                                                 |
| LOG                           | ASCII                   | MP batch log file.                                                   |
| LOW                           | Object                  | Low segment of a two-segment program.                                |
| LSD                           | ASCII                   | Default output for DUMP program.                                     |
| LSQ                           | ASCII                   | Queue listing.                                                       |
| LST                           | ASCII                   | Listing data.                                                        |
| MAC                           | Source                  | Source file in MACRO language.                                       |
| MAN                           | ASCII                   | Manual (documentation) file.                                         |

Table A-1  
Filename Extensions (Cont'd)

| <u>Filename Extensions</u> | <u>Type of File</u> | <u>Meaning</u>                                       |
|----------------------------|---------------------|------------------------------------------------------|
| MAP                        | ASCII               | Loader map file.                                     |
| MEM                        | ASCII               | Memorandum file.                                     |
| MSB                        | Object              | Music compiler binary output.                        |
| MUS                        | Source              | Music compiler input.                                |
| OLD                        | Source              | Backup source program.                               |
| OPR                        | ASCII               | Installation and assembly instructions.              |
| PAL                        | Source              | Source file in PAL 1Ø (PDP-8 assembler).             |
| PBT                        | ASCII               | P-batch control file.                                |
| PLG                        | ASCII               | P-batch log file.                                    |
| QUC                        | Binary              | Queue change request file.                           |
| QUD                        | ASCII,<br>Binary    | Queued data file.                                    |
| QUE                        | Binary              | Queue request file.                                  |
| QUF                        | Binary              | Master queue and request file.                       |
| REL                        | Object              | Relocatable binary file.                             |
| RIM                        | Object              | RIM loader file.                                     |
| RMT                        | Object              | Read-In mode (RIM) format file (PIP).                |
| RNC                        | ASCII               | RUNOFF input for producing a .CCO file.              |
| RND                        | ASCII               | RUNOFF input for producing a .DOC file.              |
| RNO                        | ASCII               | Programming specifications in RUNOFF input.          |
| RNP                        | ASCII               | RUNOFF input for producing a .OPR file.              |
| RSP                        | ASCII               | Script response time log file.                       |
| RTB                        | Object              | Read-In mode (RIM1ØB) format file (PIP).             |
| SAV                        | Object              | Low segment from a one-segment program.              |
| SCP                        | ASCII               | SCRIPT control file.                                 |
| SFD                        | Binary              | Sub-file directory (restricted usage).               |
| SHR                        | Object              | Sharable high segment file of a two-segment program. |

Table A-1  
 Filename Extensions (Cont'd)

| <u>Filename<br/>Extension</u> | <u>Type of<br/>File</u> | <u>Meaning</u>                          |
|-------------------------------|-------------------------|-----------------------------------------|
| SNO                           | Source                  | Source file in SNOBOL language.         |
| SNP                           | ASCII                   | Snapshot of disk by DSKLST.             |
| SRC                           | ASCII                   | SRCCOM output.                          |
| SVE                           | Object                  | .SAVed file from a single user Monitor. |
| SYS                           | Binary                  | Special System files.                   |
| TEC                           | ASCII                   | TECO macro.                             |
| TMP                           | ASCII,<br>Binary        | Temporary files.                        |
| TXT                           | ASCII                   | Text file.                              |
| UFD                           | Binary                  | User file directory (restricted usage). |
| UPD                           | ASCII                   | Updates flagged in margin (SRCCOM).     |
| WCH                           | ASCII                   | SCRIPT Monitor (WATCH) file.            |
| XPN                           | Object                  | Expanded save file (FILEX).             |

# **dec**system10

## OPERATING SYSTEM COMMANDS

This manual reflects the software associated with the 5.05 monitor. For individual system program version numbers, refer to Page iii.

1st Printing May 1971  
2nd Printing (Rev) December 1971  
3rd Printing (Rev) June 1972

Copyright © 1971, 1972 by Digital Equipment Corporation

The material in this manual is for informational purposes and is subject to change without notice.

The following are trademarks of Digital Equipment Corporation, Maynard, Massachusetts:

|           |              |
|-----------|--------------|
| DEC       | PDP          |
| FLIP CHIP | FOCAL        |
| DIGITAL   | COMPUTER LAB |

SOFTWARE VERSION NUMBERS

The following versions of the software are discussed in this manual.

|        |                         |         |                         |
|--------|-------------------------|---------|-------------------------|
| ALCFIL | Version 7               | INITIA  | Version 3               |
| BACKUP | Preliminary Information | KJOB    | Version 47              |
| BATCON | Version 6               | LINED   | Version 13              |
| CDRSTK | Version 11              | LOGIN   | Version 53              |
| COMPIL | Version 20              | Monitor | 5.05                    |
| COPY   | Version 6               | OMOUNT  | Version 22              |
| CREF   | Version 46              | OPSER   | Version 4               |
| DAEMON | Version 6               | PIP     | Version 32              |
| DIRECT | Version 2               | PLEASE  | Version 11              |
| DUMP   | Version 4               | QUEUE   | Version 3               |
| FILCOM | Version 16              | QUOLST  | Version 4               |
| FILEX  | Version 15              | REATA   | Version 3               |
| FUDGE2 | Version 14              | RESTORE | Preliminary Information |
| GLOB   | Version 5               | SETSRC  | Version 11              |
| GRIPE  | Version 3               | SYSTAT  | Version 467             |
| HELP   | Version 3               | TECO    | Version 23              |
|        |                         | UMOUNT  | Version 20              |



CONTENTS

|                                        | Page                                         |
|----------------------------------------|----------------------------------------------|
| CHAPTER 1 INTRODUCTION                 |                                              |
| 1.1                                    | Jobs 443                                     |
| 1.2                                    | Monitor Mode and User Mode 444               |
| 1.3                                    | Command Interpreters 445                     |
| 1.3.1                                  | Monitor Command Language Interpreter 445     |
| 1.3.1.1                                | Special Characters 446                       |
| 1.3.2                                  | Batch Command Interpreter 447                |
| 1.4                                    | Command Formats 447                          |
| 1.4.1                                  | Command Names 447                            |
| 1.4.2                                  | Command Arguments 448                        |
| 1.4.2.1                                | Project-Programmer Numbers and Passwords 448 |
| 1.4.2.2                                | Device Names 448                             |
| 1.4.2.3                                | File Structure Names 451                     |
| 1.4.2.4                                | File Specifications 451                      |
| 1.5                                    | COMPILE-Class Commands 453                   |
| 1.5.1                                  | Indirect Commands (@ Construction) 454       |
| 1.5.2                                  | The + Construction 455                       |
| 1.5.3                                  | The = Construction 456                       |
| 1.5.4                                  | The < > Construction 456                     |
| 1.5.5                                  | Compile Switches 457                         |
| 1.5.6                                  | Standard Processor 457                       |
| 1.5.7                                  | Processor Switches 458                       |
| 1.5.8                                  | LOADER Switches 460                          |
| CHAPTER 2 SYSTEM COMMANDS AND PROGRAMS |                                              |
| 2.1                                    | Commands by Functional Groups 463            |
| 2.1.1                                  | Job Initialization Commands 463              |
| 2.1.2                                  | Facility Allocation Commands 464             |
| 2.1.3                                  | Source File Preparation Commands 465         |
| 2.1.4                                  | File Manipulation Commands 465               |
| 2.1.5                                  | Object Program Preparation Commands 465      |
| 2.1.6                                  | Object Program Control Commands 465          |
| 2.1.7                                  | Object Program Examination Commands 466      |
| 2.1.8                                  | Multiple Job Control Commands 466            |

## CONTENTS (Cont)

|        |                             | Page |
|--------|-----------------------------|------|
| 2.1.9  | Job Termination Command     | 466  |
| 2.1.10 | Sending Messages            | 466  |
| 2.1.11 | Job Information Commands    | 467  |
| 2.1.12 | System Information Commands | 467  |
|        | ALCFIL program              | 468  |
|        | ASSIGN command              | 470  |
|        | ATTACH command              | 472  |
|        | BACKSPACE command           | 474  |
|        | BACKUP program              | 475  |
|        | CLOSE command               | 479  |
|        | COMPILE command             | 480  |
|        | CONTINUE command            | 485  |
|        | COPY command                | 486  |
|        | COPY program                | 488  |
|        | CORE command                | 491  |
|        | CPUNCH command              | 493  |
|        | CREATE command              | 498  |
|        | CREF command                | 499  |
|        | CSTART command              |      |
|        | CCONTINUE command           | 500  |
|        | D (deposit) command         | 502  |
|        | DAYTIME command             | 504  |
|        | DCORE command               | 505  |
|        | DDT command                 | 510  |
|        | DEASSIGN command            | 512  |
|        | DEBUG command               | 513  |
|        | DELETE command              | 517  |
|        | DETACH command              | 519  |
|        | DIRECT command              | 520  |
|        | DISMOUNT command            | 525  |
|        | DSK command                 | 527  |
|        | DUMP command                | 529  |
|        | DUMP program                | 530  |
|        | E (examine) command         | 536  |

CONTENTS (Cont)

|                   | Page |
|-------------------|------|
| EDIT command      | 537  |
| EOF command       | 538  |
| EXECUTE command   | 539  |
| FILCOM program    | 543  |
| FILE command      | 553  |
| FILEX program     | 557  |
| FINISH command    | 560  |
| FUDGE command     | 562  |
| FUDGE2 program    | 563  |
| GET command       | 568  |
| GLOB program      | 569  |
| GRIPE program     | 573  |
| HALT command      | 574  |
| HELP command      | 575  |
| INITIA command    | 577  |
| JCONTINUE command | 578  |
| KJOB command      | 579  |
| LIST command      | 584  |
| LOAD command      | 585  |
| LOCATE command    | 589  |
| LOGIN command     | 590  |
| MAKE command      | 592  |
| MOUNT command     | 593  |
| OPSER program     | 597  |
| PJOB command      | 601  |
| PLEASE command    | 602  |
| PLOT command      | 604  |
| PRESERVE command  | 609  |
| PRINT command     | 610  |
| PROTECT command   | 616  |
| QUEUE command     | 618  |
| QUOLST program    | 631  |
| R command         | 632  |
| REASSIGN command  | 633  |

## CONTENTS (Cont)

|                        | Page |
|------------------------|------|
| REATA program          | 635  |
| REENTER command        | 637  |
| RENAME command         | 638  |
| RESOURCES command      | 640  |
| RESTORE program        | 641  |
| REWIND command         | 645  |
| RUN command            | 646  |
| SAVE command           | 648  |
| SCHED command          | 650  |
| SEND command           | 651  |
| SET BLOCKSIZE command  | 653  |
| SET CDR command        | 654  |
| SET CPU command        | 655  |
| SET DENSITY command    | 657  |
| SET DSKPRI command     | 658  |
| SET HPQ command        | 659  |
| SET SPOOL command      | 660  |
| SETSRC program         | 662  |
| SET TIME command       | 666  |
| SET TTY or TTY command | 668  |
| SFT WATCH command      | 672  |
| SKIP command           | 675  |
| SSAVE command          | 677  |
| START command          | 679  |
| SUBMIT command         | 680  |
| SYSTAT command         | 686  |
| TECO command           | 693  |
| TIME command           | 694  |
| TPIJNCH command        | 696  |
| TYPE command           | 702  |
| UNLOAD command         | 703  |
| VERSION command        | 704  |
| WHERE command          | 706  |
| ZERO command           | 707  |

CONTENTS (Cont)

|                                                                    | Page |
|--------------------------------------------------------------------|------|
| CHAPTER 3 BATCH SYSTEM COMMANDS                                    |      |
| 3.1 Batch Components                                               | 709  |
| 3.1.1 The Stacker                                                  | 709  |
| 3.1.2 The Queue Manager                                            | 710  |
| 3.1.3 The Batch Controller                                         | 710  |
| 3.1.4 The Output Spoolers                                          | 712  |
| 3.2 Submitting Jobs                                                | 712  |
| 3.2.1 Submitting a Job with Cards                                  | 713  |
| 3.2.1.1 The \$JOB Card                                             | 713  |
| 3.2.1.2 The \$PASSWORD Card                                        | 713  |
| 3.2.1.3 The \$FORTRAN Card                                         | 713  |
| 3.2.1.4 The \$DATA Card                                            | 714  |
| 3.2.1.5 The End of File Card                                       | 714  |
| 3.2.1.6 Output                                                     | 714  |
| 3.2.2 Submitting a Job with a File                                 | 714  |
| 3.2.2.1 Image of the \$JOB Card                                    | 714  |
| 3.2.2.2 Image of the \$FORTRAN Card                                | 715  |
| 3.2.2.3 Image of the \$EOD Card                                    | 715  |
| 3.2.2.4 Image of the \$DATA Card                                   | 715  |
| 3.2.2.5 Running CDRSTK                                             | 715  |
| 3.2.3 Submitting a Job with a Control File to the Batch Controller | 716  |
| 3.2.4 Interjob Dependency                                          | 716  |
| 3.3 CDRSTK Control Cards                                           | 717  |
| 3.3.1 \$ALGOL                                                      | 718  |
| 3.3.2 \$COBOL                                                      | 719  |
| 3.3.3 \$DATA                                                       | 721  |
| 3.3.4 \$DECK                                                       | 723  |
| 3.3.5 \$DUMP                                                       | 723  |
| 3.3.6 \$EOD                                                        | 724  |
| 3.3.7 \$ERROR<br>\$NOERROR                                         | 724  |
| 3.3.8 \$FORTRAN or \$F40                                           | 725  |
| 3.3.9 \$JOB                                                        | 727  |

## CONTENTS (Cont)

|                                                          | Page                                 |
|----------------------------------------------------------|--------------------------------------|
| 3.3.10                                                   | \$MACRO 729                          |
| 3.3.11                                                   | \$MODE 730                           |
| 3.3.12                                                   | \$PASSWORD 731                       |
| 3.3.13                                                   | \$RELOCATABLE 731                    |
| 3.3.14                                                   | \$SEQUENCE 732                       |
| 3.4                                                      | BATCON Control File Commands 732     |
| 3.4.1                                                    | .BACKTO 734                          |
| 3.4.2                                                    | .CHKPNT 735                          |
| 3.4.3                                                    | .ERROR 735                           |
| 3.4.4                                                    | .GOTO 735                            |
| 3.4.5                                                    | .IF 736                              |
| 3.4.6                                                    | .NOERROR 737                         |
| 3.4.7                                                    | .NOOPERATOR 737                      |
| 3.4.8                                                    | .OPERATOR 737                        |
| 3.4.9                                                    | .REQUEUE 738                         |
| 3.4.10                                                   | .REVIVE 739                          |
| 3.4.11                                                   | .SILENCE 739                         |
| 3.5                                                      | Job Output 739                       |
| 3.5.1                                                    | The Log File 740                     |
| 3.5.1.1                                                  | CDRSTK Messages 740                  |
| 3.5.1.2                                                  | CDRSTK Error Reporting 741           |
| 3.5.1.3                                                  | Batch Controller Messages 742        |
| 3.5.1.4                                                  | Batch Controller Error Reporting 742 |
| 3.6                                                      | Sample Jobs 743                      |
| <br>CHAPTER 4 SYSTEM DIAGNOSTIC MESSAGES AND ERROR CODES |                                      |
| 4.1                                                      | System Diagnostic Messages 748       |
| 4.2                                                      | Error Codes 780                      |
| <br>APPENDIX A STANDARD FILENAME EXTENSIONS 783          |                                      |
| <br>APPENDIX B CARD CODES 787                            |                                      |
| <br>APPENDIX C TEMPORARY FILES 791                       |                                      |
| <br>APPENDIX D SAVE AND SSAVE COMMANDS 795               |                                      |

ILLUSTRATIONS

| Figure No. | Title                | Page |
|------------|----------------------|------|
| 3-1        | Typical Job on Cards | 713  |
| 3-2        | Sample Job #1        | 743  |
| 3-3        | Sample Job #2        | 744  |
| 3-4        | Sample Job #3        | 744  |
| 3-5        | Sample Job #4        | 745  |
| 3-6        | Sample Job #5        | 746  |

TABLES

| Table No. | Title               | Page |
|-----------|---------------------|------|
| 1-1       | System Devices      | 449  |
| 1-2       | Processor Switches  | 459  |
| 4-1       | Error Codes         | 780  |
| A-1       | Filename Extensions | 783  |
| B-1       | ASCII Card Codes    | 787  |
| B-2       | DEC-029 Card Codes  | 789  |
| B-3       | DEC-026 Card Codes  | 790  |
| C-1       | Temporary Files     | 791  |



## FOREWORD

DECsystem-10 Operating System Commands is a complete reference document describing the commands available in the DECsystem-10 operating system. The information presented in this manual reflects the 5.05 release of the monitor and other related programs. Commands to both the monitor command language interpreter and the programs in the Batch system are grouped in alphabetical order for easy reference to the command repertoire.

DECsystem-10 Operating System Commands does not include reference material on assembly language programming. This information can be found in DECsystem-10 Monitor Calls (DEC-10-MRRC-D), which is intended for the experienced assembly language programmer. Included in DECsystem-10 Monitor Calls are discussions of the monitor programmed operators and the various I/O devices connected to the system. The two manuals, DECsystem-10 Operating System Commands and DECsystem-10 Monitor Calls, supersede the Timesharing Monitors Programmer's Reference Manual (DEC-T9-MTSD-D) and all of its updates.

A third manual, Introduction to DECsystem-10 Software (DEC-10-MZDA-D), is a general overview of the DECsystem-10. It is written for the person, not necessarily a programmer, who knows computers and computing concepts and who desires to know the relationship between the various components of the DECsystem-10. This manual is not intended to be a programmer's reference manual, and therefore, it is recommended that it be read at least once before reading the above-mentioned reference documents.

### SYNOPSIS OF DECsystem-10 OPERATING SYSTEM COMMANDS

Chapter 1 presents all of the commands available to the user and introduces the various components of the operating system that interface with the user. Chapter 2 is a detailed description of the commands processed by the monitor command language interpreter. Presented in Chapter 3 are the commands to the Batch system and a discussion of the programs in this system. The DECsystem-10 system error messages and error codes are listed in Chapter 4 along with descriptive information on how to correct the errors. The appendices contain supplementary reference material and tables.

### CONVENTIONS USED IN DECsystem-10 OPERATING SYSTEM COMMANDS

The following conventions have been used throughout this manual:

|             |                                                                                                                                                                                                                                                                                               |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dev:        | Any logical or physical device name. The colon must be included when a device is used as part of a file specification.                                                                                                                                                                        |
| list        | A single file specification or a string of file specifications. A file specification consists of a filename (with or without a filename extension), a device name if the file is not on disk, a project-programmer number, if the file is not in the user's disk area, and a protection code. |
| arg         | A pair of file specifications or a string of pairs of file specifications.                                                                                                                                                                                                                    |
| jobn        | A job number assigned by the monitor.                                                                                                                                                                                                                                                         |
| file.ext    | Any legal filename and filename extension.                                                                                                                                                                                                                                                    |
| core        | Decimal number of 1K blocks of core.                                                                                                                                                                                                                                                          |
| adr         | An octal address.                                                                                                                                                                                                                                                                             |
| C(adr)      | The contents of an octal address.                                                                                                                                                                                                                                                             |
| [proj,prog] | Project-programmer numbers; the square brackets must be included in the command string.                                                                                                                                                                                                       |
| fs          | Any legal file structure name or abbreviation.                                                                                                                                                                                                                                                |
| Ⓢ           | The symbol used to indicate an altmode.                                                                                                                                                                                                                                                       |
| ↑x          | A control character obtained by depressing the CTRL key and then the character key x.                                                                                                                                                                                                         |
| ←           | A back arrow used in command strings to separate the input and output file specifications.                                                                                                                                                                                                    |
| *           | The system program response to a command string.                                                                                                                                                                                                                                              |
| .           | The monitor response to a command string.                                                                                                                                                                                                                                                     |
| ↵           | The symbol used to indicate that the user should depress the RETURN key. This key must be used to terminate every command to the Monitor Command Language Interpreter.                                                                                                                        |
| —           | Underscoring used to indicate computer typeout.                                                                                                                                                                                                                                               |
| n           | A decimal number.                                                                                                                                                                                                                                                                             |
| =           | An equal sign used in command strings to separate the input and output file specifications.                                                                                                                                                                                                   |



## CHAPTER 1 INTRODUCTION

The DECSYSTEM-10 Operating System is the interface between the user and the actual machine. The operating system, or monitor, has many functions, some of which are:

1. scheduling multiple and simultaneous use of the system,
2. protecting users of the system from one another,
3. allowing access to system resources including peripheral devices,
4. providing a comprehensive disk file system,
5. directing data flow between peripheral devices and the user's program,
6. controlling non-interactive jobs, and
7. overlapping input-output operations with computations for high system efficiency.

The user communicates with the operating system by means of the monitor command language. With the command language he may access all available resources of the computing system and obtain all the services provided by the operating system.

### 1.1 JOBS

The DECSYSTEM-10 computing system is a multiprogramming system; that is, control is transferred rapidly among a number of jobs in such a way that all jobs appear to be running simultaneously. The term job refers to the entire sequence of steps, from beginning to end, that the user initiates from his interactive terminal or card deck or that the operator initiates from his operator's console. When a user initiates a job from his interactive terminal, the beginning of the job is designated by the LOGIN command and the end by the KJOB command. If a user initiates a job with a card deck, the beginning of the job is the \$JOB card and the end is the end-of-file card. Operator jobs usually begin when the system is initialized and end when the system goes down.

Jobs, which may be timesharing, batch, or real-time in nature, may be initiated at the central computer site or at remote locations connected by the telephone system. Once a user initiates a job, it is possible for him to initiate another job without killing the first one. For example, a user can initiate a timesharing job and by using the SUBMIT monitor command submit a second job for batch processing

(refer to Chapter 2). He may then wait for the results from this batch job, or have the results automatically output while he continues his timesharing job.

In configuring and loading the DECsystem-10, the system administrator sets the maximum number of jobs that his system can simultaneously handle. This number may be up to 127 jobs if the system has enough memory, disk storage, processor capacity, and terminals to handle this load.

### 1.2 MONITOR MODE AND USER MODE

From the timesharing user's point of view, his terminal is in either monitor mode or user mode. In monitor mode, each line the user types in is sent to the monitor command language interpreter. The execution of certain commands (as noted in the following examples) places the terminal in user mode. When the terminal is in user mode, it becomes simply an I/O device for that user. In addition, user programs use the terminal for two purposes. The user program will either accept user command strings from the terminal (user mode) or use the terminal as a direct I/O device (data mode).

Example (terminal dialogue):

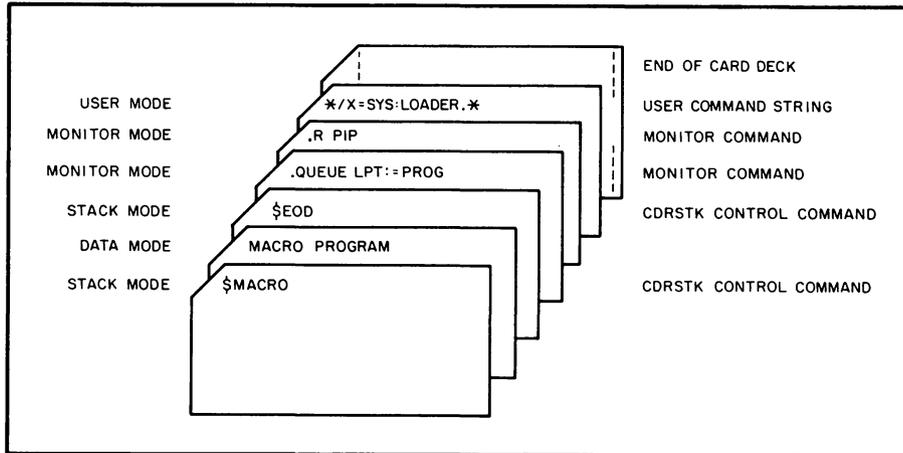
|              |                                      |                                                 |
|--------------|--------------------------------------|-------------------------------------------------|
| monitor mode | <u>.R PIP</u>                        | monitor command                                 |
| user mode    | * <u>DSK:PROG1.MAC</u> ~TTY:~        | user command string                             |
| data mode    | THIS IS FILE 1 ^Z                    | user program using terminal as input device     |
| monitor mode | ^ <u>↑C</u><br><u>.R MACRO</u>       | monitor command                                 |
| user mode    | * <u>TTY:~DSK:PROG1</u>              | user command string                             |
| data mode    | :<br>:<br>assembly listing<br>:<br>: | user program using terminal as an output device |

The special character ^C (produced by typing C with the CONTROL key depressed) is used by a time-sharing user to stop a user program and return the terminal to monitor mode. If the user program is waiting for input from the terminal, the user needs to type only one ^C to return the terminal to monitor mode; otherwise, he must type two ^C's. Because of this procedure, the user knows that his program is not waiting for input if there is no response from the monitor after one ^C. Certain commands cause the user program to start running or to continue (as noted in the following chapter) but leave the terminal in monitor mode.

When the system is started, each terminal is in monitor mode ready for users to log in. However, if the system becomes fully loaded (i.e., the maximum number of jobs that the system is set to handle has been initiated), then any unused terminals from which access is requested will receive the message JOB CAPACITY EXCEEDED.

The card-oriented Batch user can think of his cards as being in stack mode, monitor mode, or user mode. When the card is in stack mode, it contains a control command beginning with a \$ (refer to Chapter 3) and is sent to the Stacker, CDRSTK. CDRSTK interprets these commands and performs various actions to create a control file for the Batch Controller. When the card is in monitor mode, it contains a monitor command preceded by a period and is copied by CDRSTK into the control file. When the card is in user mode, it contains a user-level program command preceded by an asterisk or an equal sign and is also copied by CDRSTK into the control file. As each line in the control file is executed, the Batch Controller passes the monitor-level line to the monitor command language interpreter and the user-level line to the user program.

Example (sample card deck):



10-0897

### 1.3 COMMAND INTERPRETERS

#### 1.3.1 Monitor Command Language Interpreter

When the terminal is in monitor mode, the user communicates with the monitor command language interpreter. By means of commands to this interpreter, the user may initialize jobs, allocate facilities, prepare source files, manipulate files, prepare, control, and examine object programs, control job sequences and multiple jobs, terminate jobs, send messages, and obtain job and system information. The commands described in Chapter 2 are processed by this interpreter.

Most commands are processed without delay. However, a command may be momentarily delayed if a job is swapped out to the disk and the command requires that the job be resident in core; the command is executed when the job is swapped into core. The completion of each command is signaled by the output of a carriage return, line feed sequence. If the terminal is left in monitor mode, a period follows the carriage return, line feed. If the terminal is left in user mode, any response other than a carriage return, line feed comes from the user's program. For example, most standard system programs immediately send an asterisk(\*) to the user's terminal to indicate their readiness to accept user command strings.

The type-ahead technique may be employed by the experienced timesharing user at a terminal. This means that the user does not have to wait for the completion of one command before he can begin another. For example, if two operations are desired from the monitor, the request for the second operation can be typed before receiving the period after completion of the first.

The command interpreter makes several checks before processing commands from users. On disk systems, if a user who has not logged in types a command that requires him to be logged in, the system responds with

?LOGIN PLEASE

and the user's command is not executed. The commands discussed in Chapter 2 all require login except where explicitly stated otherwise. When a command is recognized that requires the job to have core and the job has no core allocated, the command interpreter responds with

?NO CORE ASSIGNED

and the user's command is not executed.

1.3.1.1 Special Characters - There are several special characters recognized by the monitor command language interpreter that cause specific functions to be performed. As noted previously, control-C (↑C) interrupts the program that is currently running and returns the terminal to monitor mode. This character causes the input line back to the last break character (e.g., carriage return, line feed) to be deleted (equivalent to the action of a ↑U). Two control-C's are necessary if the user program is not requesting input from the terminal (i.e., the program is in the middle of execution).

The RUBOUT key on the terminal generates a character that causes the last character typed to be deleted. This permits correction of typing errors. Depressing the RUBOUT key n times causes the last n characters typed to be deleted. The deleted characters are echoed on the terminal enclosed in backslashes (\). Characters beyond the last break character or characters already processed by the user program are not deleted.

Control-U (↑U) causes the deletion of the current line, back to the last break character. The system responds with a carriage return, line feed so that the line may be typed again. Once a break character has been typed, line-editing features (↑U and RUBOUT) can no longer be used on that line, except when running TECO.

Control-O (↑O) suppresses output to the terminal. The system responds with a carriage return, line feed sequence. A subsequent control-O re-enables output to the terminal. At remote stations, the effect of the ↑O may be somewhat delayed.

### 1.3.2 Batch Command Interpreter

The monitor command language interpreter is used for all monitor commands submitted via the Batch system. In addition, the Batch user issues commands that are only used by the Batch programs, Stacker (CDRSTK) and Batch Controller (BATCON). Control commands, discussed in Chapter 3, are processed by the Stacker and, by means of these commands, the user can create a control file, a log file, and data files; can enter jobs into the Batch input queue; and can insert monitor commands into the control file. An additional interpretation is done by the Batch Controller. When the job is executed, the Batch Controller processes the control file to pass monitor commands to the monitor command language interpreter and user-level commands to the appropriate programs.

## 1.4 COMMAND FORMATS

Each command is a line of ASCII characters in upper and/or lower case. Spaces and TABs preceding the command name are ignored. Comments may be typed on the same line as the command by preceding the comment with a semicolon. The monitor and batch command language interpreters do not interpret or execute a line of comments. Every command to the monitor command interpreter should be terminated by pressing the RETURN key on the console. In examples in this manual, the symbol ↵ is used to indicate that the user should depress the RETURN key. If the command is in error, the command up to the error is typed out by the monitor preceded and followed by a ?, and the terminal remains in monitor mode.

### 1.4.1 Command Names

Commands to the monitor command interpreter are alphabetic strings of one to six characters; characters after the sixth are ignored. Only enough characters to uniquely identify the command need be typed. It is recommended that a Batch job use the full command name since the abbreviations may change with the addition of new commands.

Installations choosing to implement additional commands should take care to preserve the uniqueness of the first three letters of existing commands.

Control commands to the Stacker in the multiprogramming batch system must have a dollar sign (\$) in the first column of the card or the line and an alphabetic character in the second column. Only the first part of the command name need be specified; as long as the specified command name is unique, it is accepted. The first three characters of the command name are generally sufficient to ensure uniqueness.

#### 1.4.2 Command Arguments

Arguments follow the command name and are separated from it by a space or TAB. If the monitor command interpreter recognizes a command name, but a necessary argument is missing, the monitor responds with

?TOO FEW ARGUMENTS

Extra arguments are ignored.

1.4.2.1 Project-Programmer Numbers and Passwords - Access to the DECsystem-10 is limited to authorized users. The system administrator provides each authorized user with a project number, a programmer number, and a password. The project numbers range from 1 to 377777 octal (numbers 1 to 10 are reserved for DEC) and the programmer numbers range from 1 to 777777 octal (numbers 1 to 7 are reserved for DEC and numbers 400000 to 777777 are reserved for special purposes)<sup>1</sup>. These numbers identify the user and his file storage area on a file structure. In a command string, the project and programmer numbers are separated with a comma and must be enclosed in square brackets, e.g., [10,7].

The password is from one to six SIXBIT characters and is only used when logging on the computing system. To maintain password security, the monitor does not echo the password. On terminals with local copy (refer to DECsystem-10 Monitor Calls), a mask is typed to make the password unreadable.

1.4.2.2 Device Names - Associated with each system device controlled by the computing system is a physical device name. This name consists of three letters, zero to three numerals specifying the unit number, and a colon. Table 1-1 lists the generic physical device names associated with the various system devices.

---

<sup>1</sup>When the programmer number is from 1 to 7, all project numbers are reserved for DEC.

Table 1-1  
System Devices

| Device                                                                                                                                                                                                                                                                                                          | Generic Physical Device Name                                                                                                                                    |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| All Disks<br>Card Punch<br>Card Reader<br>Console TTY<br>DECTape<br>Disk<br>Packs<br>Fixed-Head<br>Display<br>Experimental System Library<br>Help Library<br>Line Printer<br>Magnetic Tape<br>Operator Terminal<br>Paper-tape Punch<br>Paper-tape Reader<br>Plotter<br>Pseudo-TTY<br>System Library<br>Terminal | ALL:<br>CDP:<br>CDR:<br>CTY:<br>DTx:†<br>DSK:<br>DPx:†<br>FHx:†<br>DIS:<br>NEW:<br>HLP:<br>LPT:<br>MTA:<br>OPR:<br>PTP:<br>PTR:<br>PLT:<br>PTY:<br>SYS:<br>TTY: |
| †X represents A,B,..., indicating the first controller, second controller, etc.                                                                                                                                                                                                                                 |                                                                                                                                                                 |

The user may also assign a logical device name to a physical device. The logical name is from one to six alphanumeric characters of the user's choice, followed by a colon, and is used synonymously with a physical device name in all references to the device. Logical device names allow the user, when writing his program, to use arbitrarily selected device names, which he assigns to the most convenient physical devices at run time. However, care should be exercised when assigning logical device names because these names have priority over physical device names. For example, if a DECTape is assigned the logical name DSK, then all of the user's programs attempting to use the disk via the device name DSK will use the DECTape instead.

Except for disk devices, only one logical device name can be associated at any one time with a physical device. The same logical name can be used for a second physical device by disassociating it from the first device and associating it with the second device via the ASSIGN command. Logical device

names are disassociated from all devices with the DEASSIGN command (refer to Chapter 2). Subsequent ASSIGN commands (refer to Chapter 2) to all devices except disk devices replace the old logical name with the new one.

The following is an example of the use of physical and logical device names. Underscoring is used to indicate computer typeout.

|                                                           |                                                                                                                                                                               |
|-----------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><u>._ASSIGN DTA: ABC:)</u></p>                         | <p>User requests a DECTape drive with the logical name ABC.</p>                                                                                                               |
| <p><u>DEVICE DTA6 ASSIGNED</u></p>                        | <p>Monitor has given the user drive DTA6. The user mounts a DECTape on drive DTA6.</p>                                                                                        |
| <p><u>._ASSIGN PTP: ABC:)</u></p>                         | <p>User requests the paper-tape punch with the logical name ABC.</p>                                                                                                          |
| <p><u>% LOGICAL NAME WAS IN USE,<br/>PTP ASSIGNED</u></p> | <p>Paper-tape punch is reserved, and ABC now refers to the PTP.</p>                                                                                                           |
| <p><u>._R PIP)</u></p>                                    | <p>User requests the system program PIP (Peripheral Interchange Program).</p>                                                                                                 |
| <p><u>*ABC:~DTA6:FILEA)</u></p>                           | <p>User issues a command string to PIP asking that file FILEA be transferred from device DTA6 to logical device ABC (physical device PTP: which is assigned to the user).</p> |
| <p><u>*+C</u></p>                                         | <p>User returns to monitor mode.</p>                                                                                                                                          |
| <p><u>._ASSIGN DTA: DEF:)</u></p>                         | <p>User requests another DECTape drive with logical name DEF.</p>                                                                                                             |
| <p><u>._ASSIGNED TO JOB N1,N2,...</u></p>                 | <p>All drives are in use by the specified jobs. No DECTape drive is assigned, and no logical assignment is made.</p>                                                          |
| <p><u>._ASSIGN DTA6: DEF:)</u></p>                        | <p>User requests drive DTA6 (which he already has) with logical name DEF. The copy of the directory currently in core is cleared.</p>                                         |
| <p><u>DEVICE DTA6 ASSIGNED</u></p>                        | <p>User mounts a new DECTape on the previously assigned drive. The new DECTape directory is read into core when next accessed.</p>                                            |
| <p><u>._DEASSIGN PTP:)</u></p>                            | <p>User deassigns PTP, thereby clearing the logical name ABC.</p>                                                                                                             |
| <p><u>._R PIP)</u></p>                                    | <p>User requests PIP.</p>                                                                                                                                                     |
| <p><u>*ABC:~DEF:FILEB)</u></p>                            | <p>User requests that file FILEB be transferred from device DEF to device ABC.</p>                                                                                            |
| <p><u>?DFVICE ABC DOES NOT EXIST</u></p>                  | <p>The logical device name ABC is no longer assigned.</p>                                                                                                                     |

(continued on next page)

\*+C

User returns to monitor mode.

.ASSIGN DTA6: XYZ:)

User requests drive DTA6 again with logical name XYZ. The logical name DEF is no longer associated with DTA6. The old directory is cleared from core.

DEVICE DTA6 ASSIGNED

User mounts a new DECtape. The new directory is read into core when next accessed.

1.4.2.3 File Structure Names - Disk devices are grouped according to file structures, which are logical arrangements of 128-word blocks on one or more disk units of the same type. Examples of types of disk units are: an RP02 disk pack or an RM10B drum. Although a file structure can exist on exactly one disk unit, it can be distributed over several disk units of the same type and designated by a single name. However, two file structures cannot exist on the same unit. Each file structure has a SIXBIT name specified by the operator at structure definition time. This name can consist of five or less alphanumeric characters and must not duplicate a physical device name, a unit name, or an existing file structure name. The recommended names for public file structures are DSKA, DSKB, ..., DSKN in order of decreasing speed.

1.4.2.4 File Specifications - All information (programs and data) in the system is stored as named files. Each named file has associated with it a file specification which consists of

1. the physical device name or file structure name,
2. the filename,
3. the filename extension,
4. the ordered list of directory names, and
5. the access protection code.

The first four items of the file specification are necessary to uniquely identify a disk file. File specifications are ignored when given for devices other than DECtape or disk since these two devices are the only directory-oriented devices. In addition, items 4 and 5 do not apply to DECtapes.

The physical device name used for DECtape or the file structure name used for disk may be any legal device name discussed in the foregoing sections. A colon should always follow the device name; e.g., DTA3:. The filename is from one to six SIXBIT characters; all characters after the sixth are ignored. The filename extension is a period following by zero to three characters and is used to indicate the type of information in the file. (Refer to Appendix A for a list of standard filename extensions.) It is recommended that only the standard extensions be used even though other extensions are valid. Most programs only recognize filenames and extensions consisting of letters and digits. The ordered list of directory names identifies the disk area in which the file is stored. This list can be a user file directory (UFD)

represented by the project-programmer number of the owner of the files in the directory or can be a user file directory followed by one or more sub-file directories (SFDs). (Refer to the DECsystem-10 Monitor Calls for a description of SFDs.) The directory name must be enclosed in square brackets. The access protection of the file is a three-digit code designating which users can read or write the file and must be enclosed in angle brackets. The protection code is specified only for output files. For a given file, the users are divided into three groups: the owner of the file, the users with the same project number as the owner, and the rest of the users. The standard protection code is 057 which allows users in the owner's project to read and execute the file and prevents access by all other users. (For a complete description of access protection, refer to DECsystem-10 Monitor Calls.) The standard protection code can be redefined by the various installations.

In command strings, the filename, the device name if the file is not on disk, and the directory name if the file is not in the user's disk area, are required. The filename extension, the device name if the file is on the disk, the directory name if the file is in the user's disk area, and the protection code are optional. The following are examples of file specifications:

|                            |                                                 |
|----------------------------|-------------------------------------------------|
| TEXT.MAC                   | filename and extension                          |
| DTA3:FILEA                 | device and filename                             |
| DSK:PROG2.CBL [10,16]      | device, filename, extension, and directory name |
| DSKA:MAIN.F4[27,235] <057> | complete file specification                     |

Many command strings allow the wildcard construction to be used. This means that the filename, the extension, or the directory name may be replaced totally with an asterisk or partially with a question mark to designate certain filenames, extensions, or directories. The asterisk is used as a wild field to designate the entire filename, extension, or directory name. For example,

|                 |                                                                              |
|-----------------|------------------------------------------------------------------------------|
| filename.*      | All files with this filename and any extension.                              |
| *.ext           | All files with this extension and any filename.                              |
| *.*             | All files.                                                                   |
| *.* [project,*] | All files in directories with this project number and any programmer number. |

The question mark is used as a wild character to designate part of the filename, extension, or directory name. A question mark is used for each character that is to be matched; i.e., PR?? matches on four characters or less. For example,

|               |                                                                                       |
|---------------|---------------------------------------------------------------------------------------|
| filename .M?? | All files with this filename and any extension beginning with M.                      |
| TES?? .ext    | All files with this extension and any filename up to 5 characters beginning with TES. |
| ?? .???       | All files with filenames of two characters or less.                                   |

(continued on next page)

|          |                                                                                         |
|----------|-----------------------------------------------------------------------------------------|
| [25,5??] | All files in directories with the project number 25 and the programmer numbers 500-577. |
|----------|-----------------------------------------------------------------------------------------|

The asterisk and the question mark can be specified together in the same construction.

|      |                                                     |
|------|-----------------------------------------------------|
| ??.* | All files with filenames of two characters or less. |
|------|-----------------------------------------------------|

In addition, the directory name can be specified with the project number, the programmer number, or both numbers missing. The following are examples of the various ways of representing a particular directory.

|         |                                                                                                                                           |
|---------|-------------------------------------------------------------------------------------------------------------------------------------------|
| [15,23] | The UFD [15,23].                                                                                                                          |
| [,30]   | The UFD that has the user's project number and the specified programmer number (30).                                                      |
| [36,]   | The UFD that has the specified project number (36) and the user's programmer number.                                                      |
| [,]     | The user's UFD.                                                                                                                           |
| [-]     | The user's default directory which may be different from his UFD (refer to the <u>DECsystem-10 Monitor Calls</u> and the SETSRC program). |

The number sign can be used to represent a filename or extension that contains characters that cannot be typed because they have special meanings in the system. For example, if a file with the name \*.MAC were typed in a command string, the user would be referencing all files with the extension .MAC since the \* designates all files with the specified extension MAC. To allow a filename or extension to be typed that is composed of special characters, the user employs the number sign followed by the octal representation of the SIXBIT filename or extension. For example, #120000000000 represents the file named \*. If letters or digits are part of the filename or the extension containing the special characters, the octal representation of the letters or digits must also appear following the number sign. In other words, the number sign must be placed at the beginning of the filename and all characters following must be represented in octal. Furthermore, this construction can be used to read the contents of a UFD. For example, #000010000073.UFD [1,1] represents the file named 10,73.UFD in the directory [1,1].

The programs that recognize the number sign are DUMP, DIRECT, PIP, and QUEUE.

### 1.5 COMPIL-CLASS COMMANDS

Certain monitor commands simplify communication between the user and the system programs of the DECsystem-10 by allowing the user to type a short, concise monitor command string that causes a series of operations to be performed. These commands are known as COMPIL-class commands and are described in detail in Chapter 2. These commands cause the monitor to run the COMPIL program, which

deciphers the command and constructs new command strings for the system program (e.g., TECO, PIP, LINED, FORTRAN) that actually processes the command. Each time CREATE, MAKE, EDIT, or TECO is executed, the command with its arguments is written as a temporary file in core or on the disk. Therefore, the file specification last edited may be recalled for the next edit without specifying the arguments again. (This is an exception to the requirement that the filename must always be specified.) For example, if the command

```
.CREATE PROGX.MAC
```

is executed, then the user may later type the command

```
.EDIT
```

instead of

```
.EDIT PROGX.MAC
```

assuming no other EDIT-class command that changed the filename was used in the interim.

The COMPIL, LOAD, EXECUTE, and DEBUG commands with their arguments are also written in a temporary file so that the file specification given last may be recalled without specifying the arguments again.

The temporary files containing these file specifications have filenames of the following form:

```
nnnxxx.TMP
```

where nnn is the user's job number in decimal, with leading zeros to make three digits, and xxx specifies the use of the file. Refer to Appendix C for a list of the temporary files.

### 1.5.1 Indirect Commands (@ Construction)

When there are many program names and switches, they can be put into a file and do not have to be typed in for each compilation. This is accomplished by the use of the @ file construction, which may be combined with any COMPIL-class command.

The @ file may appear at any point after the first word in the command. In this construction, the word file must be a filename, which may have an extension and a project-programmer number. If the extension is omitted, a search is made for the command file with a null extension and then for a command file with the extension .CMD. The information in the specified command file is then put into the command string to replace the characters @ file.

For example, if the file FLIST contains the string

```
FILEB,FILEC/LIST,FILED
```

then the command

```
.COMPILE FILEA,FILEB,FILEC/LIST,FILED,FILEZ
```

could be replaced by

```
.COMPILE FILEA,@FLIST,FILEZ
```

Command files may contain the @ file construction to a depth of nine levels.<sup>1</sup> If this process of indirection results in files pointing in a loop, the maximum depth is rapidly exceeded and an error message is produced.

The following rules apply in handling format characters in a command file.

- a. Spaces are used to delimit words but are otherwise ignored. Similarly, the characters TAB, VTAB, and FORM are treated like spaces.
- b. To allow long command strings, command terminators (CARRIAGE RETURN, LINE FEED, ALTMODE) are ignored if the first nonblank character after a sequence of command terminators is a comma. Otherwise, they are treated either as commas by the COMPILE, LOAD, EXECUTE, and DEBUG commands or as command terminators by all other COMPIL-class commands.
- c. Blank lines are completely ignored because strings of returns and line feeds are considered together.
- d. Comments may be included in command files by preceding the comment with a semicolon. All text from the semicolon to the line feed is ignored.
- e. If command files are sequenced, the sequence numbers are ignored.

### 1.5.2 The + Construction<sup>2</sup>

A single relocatable binary file may be produced from a collection of input source files by the "+" construction. For example: a user may wish to compile the parameter file, PAR.MAC, the switch file, SWIT.MAC, and the file that is the body of the program, MAIN.MAC. This is specified by the following command:

```
.COMPILE PAR+SWIT+MAIN
```

---

<sup>1</sup> However, if BLISS, SNOBOL, and MACX11 (the PDP-11 assembler for the PDP-10) are added as processors, one less level of indirection for each processor is obtained. These processors will be recognized only when the appropriate assembly switches are set. These assembly switch settings are not supported.

<sup>2</sup> Use in COMPILE, LOAD, EXECUTE, and DEBUG commands only.

The name of the last input file in the string is given to any output (.REL, .CRF, and/or .LST) files (e.g., MAIN in the preceding example). The source files in the "+" construction may each contain device and extension information and project-programmer numbers.

### 1.5.3 The = Construction<sup>1</sup>

Usually the filename of the relocatable binary file is the same as that of the source file, with the extension specifying the difference. This can be changed by the "=" construction, which allows a filename other than the source filename to be given to the associated output files. For example: if a binary file named BINARY.REL is desired from a source program named SOURCE.MAC, the following command is used.

```
.COMPILE BINARY=SOURCE
```

This technique may be used to specify an output name to a file produced by the use of "+" construction. To give the name WHOLE.REL to the binary file produced by PART1.MAC and PART2.MAC, the following is typed.

```
.COMPILE WHOLE=PART1+PART2
```

Although the most common use of the "=" construction is to change the filename of the output files, this technique may be used to change any of the other default conditions. The default condition for processor output is DSK:source.REL [self]. For example: if the output is desired on DTA3 with the filename FILEX, the following command may be used:

```
EXECUTE DTA3:FILEX=FILE1.F4
```

### 1.5.4 The <> Construction<sup>1</sup>

The <> construction causes the programs within the angle brackets to be assembled with the same parameter file. If + is used, it must appear before the <> construction. For example to assemble the files LPTSER.MAC, PTPSER.MAC, and PTRSER.MAC, each with the parameter file PAR.MAC, the user could type

```
.COMPILE PAR+LPTSER, PAR+PTPSER, PAR+PTRSER
```

With the angle brackets, however, the command becomes

```
.COMPILE PAR+<LPTSER, PTPSER, PTRSER >
```

However, the following command is invalid:

```
.COMPILE <LPTSER, PTPSER, PTRSER >+PAR
```

<sup>1</sup>Used in COMPILE, LOAD, EXECUTE, and DEBUG commands only.

### 1.5.5 Compile Switches

The COMPILE, LOAD, EXECUTE, and DEBUG commands can be modified by including switches in the command string. These switches can be used to indicate the processor to be used, to force a compilation, to generate listings, to create libraries, to search user libraries, and to obtain loader maps. Each switch is preceded by a slash and terminated with a non-alphanumeric character, usually a space or a comma. The switch used can be abbreviated if the abbreviation uniquely identifies the switch.

The switches used with these four commands are either temporary or permanent. A temporary switch applies only to the file immediately preceding it. An intervening space or comma cannot separate the filename and the switch. For example,

```
.COMPILE PROG,TEST/MACRO,SUBLET
```

The /MACRO switch applies only to the file named TEST.

A permanent switch applies to all files following it until modified by a subsequent switch. It is separated from the file by spaces, commas, or a combination of both. For example,

```
.COMPILE PROG /MACRO TEST,SUBLET  
.COMPILE PROG,/MACRO,TEST,SUBLET  
.COMPILE PROG,/MACRO TEST,SUBLET  
.COMPILE PROG /MACRO,TEST,SUBLET
```

In all four examples, the /MACRO switch applies to the files named TEST and SUBLET.

The switches that can be used with the COMPILE, LOAD, EXECUTE, and DEBUG commands are described in the individual command explanations in Chapter 2.

### 1.5.6 Standard Processor

Files with recognizable processor extensions (e.g., .MAC, .CBL, .F4, .ALG) are always translated by the processor implied by the extension.<sup>1</sup> For example, a file named DATPRO.CBL will be processed by the COBOL compiler. Files without a recognizable processor extension are compiled or assembled according to the standard processor, which is normally FORTRAN at the beginning of the command string. The user can control the setting of the standard processor by including switches in the COMPILE, LOAD, EXECUTE, or DEBUG command string. Refer to the appropriate command descriptions in Chapter 2 for the switches used to change the standard processor.

---

<sup>1</sup> By setting the appropriate assembly switches, SNOBOL, BLISS, and MACX11 (the PDP-11 assembler for the PDP-10) will be recognized as processors. However, these assembly switch settings are not supported.

In the following examples, the installation has chosen FORTRAN as the standard processor. The command

```
.COMPILE NOEXT
```

causes the file named NOEXT (with a null extension) to be compiled by FORTRAN. The command

```
.COMPILE FILEZ.MIN
```

also compiles the file with FORTRAN since .MIN is not recognized as a processor extension. The command

```
.COMPILE APART,DATA/COBOL, TEST
```

causes the files APART and TEST to be compiled by FORTRAN and the file DATA by COBOL.

The switches used to change the standard processor can be temporary or permanent switches (refer to Paragraph 1.5.5). For example,

```
.COMPILE APART, /COBOL DATA, TEST
```

causes APART to be compiled by FORTRAN, and DATA and TEST to be compiled by COBOL.

Note that if source files are specified with the appropriate extensions, the subject of the standard processor can be disregarded, since files with processor extensions are always translated by the processor implied.

### 1.5.7 Processor Switches

Occasionally it is necessary to pass switches to the assembler or compiler being used in a COMPILE, LOAD, EXECUTE, or DEBUG command. For each translation (assembly or compilation), the COMPIL program sends a command string to the translator containing three parts: the source files, a binary output file, and a listing file. To include switches with these files, the user must:

- a. If the + construction is used, group the switches according to each related source file.
- b. Group the switches according to the three types of files (source, binary, and listing) for each file.
- c. For each source file, separate the groups of switches by commas.
- d. Enclose all the switches for each source file within one set of parentheses.
 

|                  |                                                   |
|------------------|---------------------------------------------------|
| (SSSS)           | Only source switches are present.                 |
| (SSSS,BBBB)      | Source and binary switches are present.           |
| (SSSS,BBBB,LLLL) | Source, binary, and listing switches are present. |
- e. Place each parenthesized string immediately after the source file to which it refers.

The processor switches are listed in Table 1-2, along with their meanings and the types of files to which they apply.

Table 1-2  
Processor Switches

| Processor | Source | Binary | Listing     | Meaning                                                                                                                                                                                                                                                                       |
|-----------|--------|--------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ALGOL     |        | D      |             | Set dynamic storage region for own arrays (known as the heap).                                                                                                                                                                                                                |
|           | E      |        |             | The source file has line numbers in columns 73-80.                                                                                                                                                                                                                            |
|           | L      |        | N           | List the source program.                                                                                                                                                                                                                                                      |
|           | Q      |        |             | Suppress the error print out on the terminal.                                                                                                                                                                                                                                 |
|           | S      |        |             | Delimit the words in quotes.<br>Suppress the listing of the source program.                                                                                                                                                                                                   |
| COBOL     | A      | A      | A           | Allow the listing of code generated.                                                                                                                                                                                                                                          |
|           |        |        | C           | Produce a cross-referenced listing of all user-defined items in the source program.                                                                                                                                                                                           |
|           | E      | E      | E           | Check the program for errors but do not generate code.                                                                                                                                                                                                                        |
|           | L      |        |             | Use the preceding file descriptor as a library file whenever the COPY verb is encountered.                                                                                                                                                                                    |
|           | M      | M      | M           | Print a map showing the parameters of the user-defined item.                                                                                                                                                                                                                  |
|           |        |        | N           | Suppress output of source errors on the terminal.                                                                                                                                                                                                                             |
|           |        |        | P<br>R      | Do not generate trace calls and symbols.<br>Produce a two-segment object program. The high segment contains the resident sections of the Procedure division; the low segment contains everything else. When the object program is loaded, LIBOL is added to the high segment. |
|           | S      | S      | S           | The source file has sequence numbers in columns 1-6 and comments starting in column 73.                                                                                                                                                                                       |
|           | W      | W<br>Z | W<br>Z      | Rewind the magnetic tape.<br>Zero the DECTape directory.                                                                                                                                                                                                                      |
| FORTRAN   | A      | A      | A           | Advance magnetic tape reel by one file.                                                                                                                                                                                                                                       |
|           | B      | B      | B           | Backspace magnetic tape reel by one file.                                                                                                                                                                                                                                     |
|           |        |        | C           | Generate a CREF-type cross-reference listing.                                                                                                                                                                                                                                 |
|           |        |        | D           | List error message codes only.                                                                                                                                                                                                                                                |
|           |        |        | E           | Print an octal listing of the binary program in addition to the symbolic listing. Must be accompanied by /M.                                                                                                                                                                  |
|           |        |        | I           | Translate the letter D in column 1 as a space and treat the line as a normal FORTRAN statement.                                                                                                                                                                               |
|           |        |        | M           | Include MACRO coding in output listing.                                                                                                                                                                                                                                       |
|           |        |        | N           | Suppress output of error messages on the terminal.                                                                                                                                                                                                                            |
|           |        |        | S           | Produce code for execution on the KA10 if running on the KI10, and vice-versa.                                                                                                                                                                                                |
|           |        | T<br>W | T<br>W<br>Z | T<br>W<br>Z                                                                                                                                                                                                                                                                   |

(continued on next page)

Table 1-2 (Cont)  
Processor Switches

| Processor | Source                    | Binary                             | Listing | Meaning                                                       |
|-----------|---------------------------|------------------------------------|---------|---------------------------------------------------------------|
| MACRO     | A<br>B                    | A<br>B                             | A       | Advance magnetic tape reel by one file.                       |
|           |                           |                                    | B       | Backspace magnetic tape reel by one file.                     |
|           |                           |                                    | C       | Produce listing file in a format acceptable as input to CREF. |
|           |                           |                                    | E       | List macro expansions.                                        |
|           |                           |                                    | F       | Byte sizes match the format of the instruction.               |
|           |                           |                                    | G       | Byte sizes are two 18-bit fields.                             |
|           |                           |                                    | L       | Reinstate listing (used after list suppression by S switch).  |
|           | O<br>P<br>Q<br><br>T<br>W | O<br>P<br>Q<br><br>T<br>W<br><br>Z | M       | Suppress ASCII text in macro and repeat expansion (SALL).     |
|           |                           |                                    | N       | Suppress error printouts on the terminal.                     |
|           |                           |                                    | O       | Allow literals to occupy only one line.                       |
|           |                           |                                    | P       | Increase the size of the pushdown list.                       |
|           |                           |                                    | Q       | Suppress questionable (Q) error indications on the listing.   |
|           |                           |                                    | S       | Suppress listing.                                             |
|           |                           |                                    | T       | Skip to the logical end of magnetic tape.                     |
| W         | Rewind the magnetic tape. |                                    |         |                                                               |
|           | X                         | Suppress all macro expansions.     |         |                                                               |
|           | Z                         | Zero the DECTape directory.        |         |                                                               |

Examples:

- DEBUG TEST(N) Suppress typeout of errors during assembly.
- COMPILE OUTPUT=MTA0:(W,S,M)/L Rewind the magtape (W), compile the first file, produce binary output for the K10(S), and include the MACRO coding in the output listing (M). Output files are given the names OUTPUT.REL and OUTPUT.LST.
- COMPILE/MACRO A=MTA0:(W,,Q)/L Rewind the magtape (W), compile the first file, and suppress Q (questionable) error indications on the listing. Note that when a binary switch is not present, the delimiting comma must appear.
- COMPILE /MACRO A=MTA0:(,,Q)/L Compile file at current position of the tape and suppress Q error indications on the listing. Note that when the source and binary switches are not present, the delimiting commas must appear.

1.5.8 LOADER Switches

In complex loading processes, it may be necessary to pass switches to the LOADER to direct its operation. This is accomplished by the % character. The % has the same meaning as that of the / in the

LOADER'S command string (refer to the LOADER documentation). Also, like the /, the % takes a leading sign (+ or -) and one letter (or a sequence of digits and one letter) following it. Therefore, to set a program origin of 6000 for program C, the user types

```
.LOAD A,B,%6000OC,D
```

The COMPIL program allows more than one LOADER switch to be specified. For example:

```
.LOAD PROG %F/MAP
```

Refer to the LOAD command in Chapter 2 for a description of /MAP.

The most commonly used LOADER switches are:

- a. %S Load with symbols.
- b. %nO Set program origin to n.
- c. %F Cause early search of the default libraries.
- d. %P Prevent search of the default libraries.



## CHAPTER 2 SYSTEM COMMANDS AND PROGRAMS

Although there is one operating system for all configurations of the DECsystem-10, some commands may not be included in each DECsystem-10. This is especially true of the DECsystem-1040, the basic system intended for small installations that do not want all of the system's features because of a constraint on core. Commands are deleted from the DECsystem-1040 by feature test switches (recognized by the beginning characters FT) defined at MONGEN time. In the standard DECsystem-1040, many of these switches are not set and, therefore, the corresponding commands are not available. This saves core but limits various features of the operating system. In the command descriptions that follow, the Characteristics section indicates if the switch is normally off in the DECsystem-1040. If not stated, the command is available on all DECsystem-10s.

In many cases, there are two commands to run a program. For example, the indirect command MAKE and the direct command R TECO both run the TECO program. In the DECsystem-1040, the switch implementing the indirect command may not be set but the switch implementing the direct command is always set. Therefore, it is always possible to run a program with the .R or .RUN command, even if the switch implementing the corresponding indirect command is off.

### 2.1 COMMANDS BY FUNCTIONAL GROUPS

Although the commands are arranged in alphabetical order for ease of reference, they can be divided into functional groups for ease of learning. These groups with their associated commands are as follows.

#### 2.1.1 Job Initialization Commands

Since the system is limited to authorized persons, these commands protect the system from unauthorized use.

INITIA  
LOGIN

### 2.1.2 Facility Allocation Commands

The monitor allocates peripheral devices, file structure storage, and core memory to users on request and protects these allocated facilities from interference by other users. Software provisions are incorporated in the monitor to differentiate the central station from the remote stations. Certain monitor commands, for example, ASSIGN and PLEASE, include station identification arguments to allow both user-access and allocation of system resources at any station. This feature gives the user considerable flexibility in allocating system facilities and directing input and output to the station of his choice. For example, by specifying a station number, the user can assign devices and input data from a peripheral device at a station other than his own. In addition, by using the LOCATE command, he can logically establish his job at a station other than his physical station. If the station identification argument is not included in a command, the system automatically directs input and output to the user's logical station. The user's logical station is the same as his physical station if he has not issued the LOCATE command.

When a nonsharable device is assigned to a job, it is removed from the monitor's pool of available resources. Any attempt by another user to reference or assign the device fails. Thus, a user should never leave the system without first returning his allocated facilities to the monitor pool. Allocated facilities are automatically returned to the monitor pool when the user deassigns them or kills his job. Until a user returns these facilities, no other users may utilize them except through operator intervention.

Assignable devices (i.e., nondisk and nonspooled devices) in the monitor's pool of available resources are designated as being either unrestricted or restricted devices. An unrestricted device can be assigned (ASSIGN command or INIT UO) by any user. A restricted device can be assigned only by a privileged job (i.e., a job logged in under [1,2] or running with the JACCT bit set). However, a nonprivileged user can have a restricted device assigned to him via the MOUNT command. This command allows operator intervention for the selection or denial of a particular device; thus the operator can control the use of the assignable devices. This is particularly useful when there are multiprogramming batch and interactive jobs competing for the same devices. The restricted status of a device is set or removed by the OPSER commands :RESTRICT and :UNRESTRICT.

The facility allocation commands are as follows:

|          |          |               |                |
|----------|----------|---------------|----------------|
| ASSIGN   | DISMOUNT | REASSIGN      | SET DENSITY    |
| CLOSE    | FINISH   | SET BLOCKSIZE | SET DSKPRI     |
| CORE     | LOCATE   | SET CDR       | SET HPQ        |
| DEASSIGN | MOUNT    | SET CPU       | SET SPOOL      |
|          |          |               | SET TTY or TTY |

2.1.3 Source File Preparation Commands

These commands call the system editing programs in order to create or edit a specified text file. The system editing programs available are LINED (a line-oriented editor) and TECO (a character-oriented editor). In general, the editor used to create the file should be used for editing, since LINED requires line-blocked files and TECO does not.

CREATE  
EDIT  
MAKE  
TECO

2.1.4 File Manipulation Commands

The commands in this group allow the user to manipulate his files to any desired extent. He can list source files, and DECTape and disk directories on the terminal or the line printer, possibly via the spooling mechanism. He can delete or rename files from disk and DECTape. In addition, the user can transfer files between standard I/O devices, perform conversion between various core image formats, and read and write various directory formats. Disk space can be either allocated for a new file or re-allocated for an existing file. Finally, the user can place files in the system queues and obtain listings of entries in those queues.

|           |        |          |        |
|-----------|--------|----------|--------|
| ALCFIL    | DIRECT | PRESERVE | REWIND |
| BACKSPACE | EOF    | PRINT    | SKIP   |
| BACKUP    | FILE   | PROTECT  | SUBMIT |
| COPY      | FILEX  | QUEUE    | TPUNCH |
| CPUNCH    | LIST   | RENAME   | TYPE   |
| DELETE    | PLOT   | RESTORE  | UNLOAD |
|           |        |          | ZERO   |

2.1.5 Object Program Preparation Commands

The commands in this group are used to prepare object programs and save the user's core area as one or two files.

|         |         |       |
|---------|---------|-------|
| COMPILE | EXECUTE | LOAD  |
| CREF    | FUDGE   | SAVE  |
| DEBUG   | FUDGE2  | SSAVE |

2.1.6 Object Program Control Commands

By using the commands in this group, the user can load core image files from retrievable storage devices (i.e., disk, DECTape, magnetic tape). These files can be retrieved and controlled from the user's terminal. Files stored on disk and DECTape are addressable by name. Files on magnetic tape

require the user to pre-position the tape to the beginning of the file. Refer to DECsystem-10 Monitor Calls, Chapter 1, for a description of the job data area locations referenced by the command descriptions in this group.

|              |       |                |
|--------------|-------|----------------|
| CONT (CCONT) | HALT  | REENTER        |
| DDT          | JCONT | RUN            |
| GET          | R     | START (CSTART) |

### 2.1.7 Object Program Examination Commands

The commands in this group aid the user in examining and analyzing his object program. Dumps of the user's core area can be taken and later processed by the system program DUMP according to the arguments specified by the user.

D (deposit)  
DCORE  
DUMP  
E (examine)

### 2.1.8 Multiple Job Control Commands

There is not necessarily a one-to-one relationship between jobs and terminals. A terminal must initiate a job, but the user or operator may issue commands to permit a job to float in a detached state where it is not associated with a particular terminal. Thus, more than one job may be controlled from the same terminal.

|        |        |       |
|--------|--------|-------|
| ATTACH | CSTART | OPSER |
| CCONT  | DETACH | REATA |

### 2.1.9 Job Termination Command

When the user leaves the system, all facilities allocated to his job must be returned to the monitor facility pool so that they are available to other users.

KJOB

### 2.1.10 Sending Messages

The commands in this group allow the user interconsole communication with other users of the system or with operators at any station. In addition, the user may record information in a disk file to be read by the operations staff at a later time.

GRIPE  
PLEASE  
SEND

### 2.1.11 Job Information Commands

The user can obtain various job-related information with this group of commands. This information includes the number of his job, the quotas for each file structure associated with his job, and the running time and disk space that his job has used. In addition, the user may type or modify his file structure search list.

DSK  
PJOB

QUOLST  
SETSRC

SET TIME  
SET WATCH  
TIME

### 2.1.12 System Information Commands

With the commands in this group, the user is able to obtain system status information, including the time of day, the list of available devices, file structures, and physical units not in file structures, the scheduled use of the system, and the location of a specific peripheral device.

DAYTIME  
RESOURCES

SCHED  
SYSTAT

VERSION  
WHERE

## ALCFIL program

### Function

The ALCFIL program enables the user to allocate space for a new file or reallocate space for an existing file in one contiguous region on the disk. The size of the region is restricted by the size of the cluster count field (usually 512) times the cluster size of the file structure times the number of pointers in a disk device data block (not less than 10).

### Command Format

R ALCFIL

The ALCFIL program responds with

```
/H FOR HELP
FILE?
```

The user may respond with

```
dev:file.ext [proj,prog]
or /H (for help)
or /X (to exit)
```

where dev: is a file structure or physical unit name. If dev: is omitted, DSK is assumed. If one of the other arguments is omitted, 0 is assumed. If a filename is specified, the number of blocks presently allocated, if nonzero, is typed. ALCFIL responds with

```
ALLOCATE?
```

User may type N or N,M (decimal numbers)

```
N = total number of blocks to be allocated for the file.
M = logical block within the file structure or unit (depending on dev:)
   where the allocation is to begin.
```

If the total number of blocks requested cannot be allocated (because of disk quotas), a partial allocation is given and the message

```
PARTIAL ALLOCATION ONLY
```

is typed. The user can issue the DIRECT command with the ALLOCATE switch to determine the number of blocks allocated. If the new blocks can be allocated, the message

```
ALLOCATED
```

is typed.

Since an extended ENTER (refer to DECsystem-10 Monitor Calls) is executed to allocate the new blocks, the file need not exist before the blocks are allocated.

**ALCFIL program (Cont)**

Characteristics

The R ALCFIL command:

- Places the terminal in user mode.
- Runs the ALCFIL program, thereby destroying the user's core image.

Associated Messages

Refer to Chapter 4.

Example

```

.R ALCFIL )
/H FOR HELP
FILE? TEST4.TST )
ALLOCATE? 2000 )

ALLOCATED
FILE? TEST5.TST )
ALLOCATE? 1000 )

ALLOCATED
FILE? TEST5.TST )
1000 BLOCKS ALREADY ALLOCATED
ALLOCATE? 500 )

ALLOCATED
FILE? DSKB:FILEA )
ALLOCATE? 3000 )

PARTIAL ALLOCATION ONLY
FILE? /X

EXIT

.DIR/ALLOC )

FILEA          175  <057>   14-APR-72   DSKB:

:

```

## ASSIGN command

### Function

The ASSIGN command allocates an I/O device to the user's job for the duration of the job or until a DEASSIGN command is given. This command, applied to DECTapes, clears the copy of the directory currently in core, forcing any directory references to read a new copy from the tape. (Refer to DECsystem-10 Monitor Calls for further details.)

Although DECTape is the only device that should be ASSIGNED before use, to ensure that the monitor has a copy of the proper DECTape directory in its core area, it is wise to ASSIGN all devices, such as magnetic tape, before use.

### Command Formats

#### 1. ASSIGN phys-devn log-dev

phys-devn = any physical device listed in Table 1-1 in Paragraph 1.4.2.2, followed by a 1-to-3 digit number representing a specific unit, or any file structure name. This argument is required. With this command format, the monitor attempts to assign the device specifically requested. If unable to assign the device, the monitor types an appropriate message (refer to Chapter 4).

log-dev = a logical name assigned by the user. This argument is optional. Except for disk devices, only one logical name can be assigned to a physical device. Subsequent ASSIGN commands to all devices except disk devices replace the old logical name with the new one. Logical names are disassociated from all devices by the DEASSIGN command.

#### 2. ASSIGN phys-devSnn log-dev

phys-devSnn = any physical device followed by the letter S and a 1 or 2 digit number representing a specific station, or any file structure name. This argument is required. With this command format, the monitor attempts to assign a device at the requested station. An appropriate message is typed if the device cannot be assigned (refer to Chapter 4).

log-dev = same as above.

#### 3. ASSIGN phys-dev log-dev

phys-dev = any physical device followed by a null argument implying any device of the designated type, or any file structure name. This argument is required. With this command format, the monitor attempts to assign the requested device at the user's logical station. If this type of device does not exist at the user's logical station, the monitor attempts to assign the device at the central station. If unable to assign the device, the monitor types an appropriate message (refer to Chapter 4).

log-dev = same as above.

**ASSIGN command (Cont)**

Characteristics

The ASSIGN command:

Leaves the terminal in monitor mode.

Restrictions

A comma may not be used to separate the logical and physical device names. If a comma is used, the monitor terminates its scan at the comma; therefore, the logical name is not assigned.

Non-privileged jobs (i.e., jobs not logged in as [1,2] or running with JACCT set) can only use this command to allocate unrestricted I/O devices. Restricted devices can be obtained by non-privileged jobs via the MOUNT command. The ASSIGN when issued by a privileged job allocates both restricted and unrestricted devices.

Associated Messages

Refer to Chapter 4.

Examples

.ASSIGN LPT2:)  
LPT2 ASSIGNED

The user assigns a specific line printer (LPT2).

.AS CDRS2:)  
CDR4 ASSIGNED

The user assigns any available card reader at station 2.

.ASSIGN TTY1:LPT:)  
TTY1 ASSIGNED

The user assigns TTY 1 and gives it logical name LPT.

.AS LPT:)  
LPT4 ASSIGNED

The user assigns any available line printer. The LPT chosen is at either the user's station or the central station.

.ASSIGN DTA2:)  
?DEVICE NOT ASSIGNABLE

A non-privileged user attempted to allocate a restricted device (DTA2).

.ASSIGN DTA:)  
DTA4 ASSIGNED

The user then uses the generic device name (DTA) to obtain the device. He could have used the MOUNT command to assign the restricted device DTA2.

**ATTACH command**Function

The ATTACH command detaches the current job, if any, and connects the terminal to a detached job.

Command Format

ATTACH job [proj,prog]

job = the job number of the job to which the terminal is to be attached. This argument is required.

[proj,prog] = the project-programmer number of the originator of the desired job. This argument may be omitted if it is the same as the job to which the terminal is currently attached. The operator (device OPR) or a user logged-in under [1,2] may always attach to a job although another terminal is attached, provided he specifies the proper [proj,prog].

To prevent users from attaching the jobs without knowing the PASSWORD associated with the job, a new job is temporarily created when the [proj,prog] argument is specified. This temporary job runs LOGIN to check the password. This can result in the current job not being able to attach to the specified job if the job capacity of the system would be exceeded with the creation of the temporary job. However, the current job is still detached even if there are no available jobs. The operator or any job logged-in as [1,2] can always attach to another job since they do not require the creation of a temporary job.

Characteristics

The ATTACH command:

Leaves the terminal in monitor mode.

Does not require LOGIN.

Depends on FTATTACH which is normally absent in the DECsystem-1040.

Restrictions

Remote users cannot attach to jobs with a project number of 1. Batch users cannot issue this command.

Associated Messages

Refer to Chapter 4.

**ATTACH command (Cont)**

Examples

1. .ATT 1)  
FROM JOB 5

The user attaches to job 1 from job 5. The two jobs have the same [proj,prog] and therefore, the argument is not required.

.

2. .LOG 27,235)  
JOB 7 5S04 TTY25  
PASSWORD:  
1634 23-FEB-72 WED

The user logs-in and is given job 7. TTY25 is now attached to job 7.

.ATTACH 36 [50,27])  
FROM JOB 7  
PASSWORD:

The user attaches to an existing job (36) and thereby detaches his current job (7). Since the [proj, prog] associated with job 36 is different than the user's, he must specify the [proj,prog] of the desired job. The system then requests the PASSWORD. If the given PASSWORD is correct, the terminal is attached to job 36.

.

The terminal is attached to job 36.

.ATTACH 7)  
?CAN'T ATT TO JOB

The user attempts to attach to job 7. The command fails because the [proj,prog] of job 7 is not the same as the [proj,prog] of job 36. The terminal is still attached to job 36.

.K/F)  
JOB 36,USER [50,27] LOGGED OFF TTY25 1635 23-FEB-72  
RUNTIME 0MIN,00.34SEC

The user killed job 36. The terminal is currently not attached to any job.

.ATTACH 7)  
?CAN'T ATT TO JOB

Since the terminal is currently not attached to a job, the command fails because there is no [proj, prog] to compare with the [proj,prog] of job 7.

.ATTACH 7 [27,235])  
PASSWORD:

The command is accepted and the PASSWORD is requested. The message FROM JOBn is not output since the terminal was not attached to a job.

.

The terminal is attached to job 7.

**BACKSPACE command <sup>1</sup>**Function

The BACKSPACE command spaces a magnetic tape backward a specified number of files or physical records. This command, depending on its arguments, is equivalent to the following PIP command strings:

MTAn: (M #nB) ←  
MTAn: (M #nP) ←

Command Formats

1. BACKSPACE MTAn: x FILES  
skips backward x files.
2. BACKSPACE MTAn: x RECORDS  
skips backward x records.

Characteristics

The BACKSPACE command:

Leaves the terminal in monitor mode.  
Runs the PIP program.  
Depends on FTCCLX which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

Examples

```
._BAC MTA2: 7 RECORDS )  
._BACKSP MTA3: 11 FILES )
```

---

<sup>1</sup>This command runs the COMPIL program, which interprets the command before running the PIP program.

## **BACKUP program**

### Function

The BACKUP program enables the user to save disk files on magnetic tape (MTA), DECtape (DTA), or disk (DSK). The save can be of the entire disk or selected subsets of the disk.

The BACKUP program places data in the following files:

1. BACKUP SET file

This file contains the data saved on the backup medium (MTA, DSK, DTA) with one BACKUP command. When data is saved on disk, the BACKUP SET file is one file with file delimiting control words. When written on magnetic tape, it is several files written in buffered binary mode.

The BACKUP SET file is composed of a BACKUP header, a user set, and a BACKUP trailer. The BACKUP header is one block in length and contains information concerning the creation of the BACKUP SET. This information includes the name and date of creation, the name of the system, and the user identification.

The USER SET contains the directories and the files associated with the directories for all user areas. Even if a user has files on more than one file structure, his files will be saved together. For example, all files on all file structures for user 1,2 will be stored before the files for user 1,3. In other words, the user set is ordered according to project-programmer number, not according to file structures. However, within individual project-programmer numbers, all files on one file structure are saved before files on another.

The BACKUP trailer contains information about the user set that was just created. This information includes the time, the date, and the length of the user set. The BACKUP trailer immediately follows the user set.

2. INDEX file

This file contains the directories and the filenames of all the disk areas that have been saved on the backup medium. In addition, it contains the relative block number in the BACKUP SET file where each element (file) begins. The index file is the last file written on the backup medium and is separated from the BACKUP SET file. It can be saved on DECtape and can be listed, if desired.

3. COMMAND RECOVERY file

This file contains information that indicates how much of the user's command has been processed and how much of the command remains. It is updated at every check point in order to make crash recovery possible.

4. Log File

The user is given a log file to aid him in error analysis. This file contains a record of either all actions performed by the BACKUP and RESTORE programs or only errors encountered in processing. The log file is a listing file.

**BACKUP program (Cont)**

Command Format

R BACKUP

The user may type any of the following commands after the slash output by the BACKUP program. These commands are stored in core and are processed only when a START command is given by the user. All commands are terminated with a carriage return. Multiple files may be specified in one command string by separating the filenames with commas. The full wildcard construction may be used to replace the filename, the extension, or the directory (refer to Paragraph 1.4.2.4).

| <u>Command</u>                                              | <u>Explanation</u>                                                                                                                                                                                                                                                                                                                                                      |       |         |       |         |       |         |
|-------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|-------|---------|-------|---------|
| BACKSPACE FILE                                              | Backspaces the magnetic tape to a user file header and positions the tape immediately before the header.                                                                                                                                                                                                                                                                |       |         |       |         |       |         |
| BACKSPACE SET                                               | Backspaces the magnetic tape to a BACKUP header and positions the tape either immediately before the header or to the beginning of the tape if there is no BACKUP header (i.e., there is only one BACKUP SET on the tape).                                                                                                                                              |       |         |       |         |       |         |
| BACKSPACE UFD                                               | Backspaces the magnetic tape to a UFD header and positions the tape immediately before the header.                                                                                                                                                                                                                                                                      |       |         |       |         |       |         |
| BACKUP dev2: file descriptor +<br>dev1: [proj,prog] /switch | Writes the designated file on dev2 from dev1. The user may specify files to be taken from and/or written to a user's area other than his own, provided that he has access privileges to the user areas. The device arguments are required.<br><br>/switch = /EXCEPT file descriptor<br><br>Indicates the files and/or areas that should not be written as BACKUP files. |       |         |       |         |       |         |
| DELETE dev: file.ext                                        | Deletes the file named from the specified device. The specified device must be the device for which a BACKUP has been taken.                                                                                                                                                                                                                                            |       |         |       |         |       |         |
| DENSITY MTAn:x                                              | Sets the magnetic tape density as specified by x.<br><br><table border="0" style="margin-left: 100px;"> <tr> <td>x = 2</td> <td>200 bpi</td> </tr> <tr> <td>x = 5</td> <td>556 bpi</td> </tr> <tr> <td>x = 8</td> <td>800 bpi</td> </tr> </table><br>The default is the system standard defined at MONGEN time.                                                         | x = 2 | 200 bpi | x = 5 | 556 bpi | x = 8 | 800 bpi |
| x = 2                                                       | 200 bpi                                                                                                                                                                                                                                                                                                                                                                 |       |         |       |         |       |         |
| x = 5                                                       | 556 bpi                                                                                                                                                                                                                                                                                                                                                                 |       |         |       |         |       |         |
| x = 8                                                       | 800 bpi                                                                                                                                                                                                                                                                                                                                                                 |       |         |       |         |       |         |
| DUMP ON dev: file.ext                                       | Dump the contents of the BACKUP set file beginning at the present position and ending at the next file control word. All types of errors are ignored. The device on which the dump is to be written may not be a listing device.                                                                                                                                        |       |         |       |         |       |         |
| ERROR DUMP dev:                                             | Returns to the last file control word and dumps the file on the device specified if a transmission error occurs during the backup of any file.                                                                                                                                                                                                                          |       |         |       |         |       |         |

(continued on next page)

**BACKUP program (Cont)**

| <u>Command</u>            | <u>Explanation</u>                                                                                                                                                                                                                                       |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| INDEX dev: file.ext       | Writes the index file with the designated filename on the device named. The index file is also written on the disk and saved as the last file on the backup medium. The default is DSK:MTnnnn.BKP where nnnn corresponds to the tape sequence number.    |
| LOG dev: file.ext /switch | Writes a file on the specified device which contains a record of the operations performed. The default is DSK:BACKUP.LOG.<br>/Switch = /ERROR<br>Logs only the errors. This switch is optional. If omitted, all operations are recorded.                 |
| PARITY dev: ODD or EVEN   | Specifies the parity on magnetic tape as odd or even. The default is odd.                                                                                                                                                                                |
| REWIND dev:               | On magnetic tape, closes the backup set and rewinds the tape. On disk, closes the backup set.                                                                                                                                                            |
| START                     | Begins execution of a series of commands sent previously. Commands are not processed until a START command is given. If there are no commands to be processed when the START is executed, the command recovery file is searched for executable commands. |
| UNLOAD dev:               | Performs a rewind and unload to magnetic tape.                                                                                                                                                                                                           |

The user may restart the BACKUP program at any time. By issuing a ↑C ↑C START sequence, the user can cancel present operations and specify new commands. A ↑C START sequence deletes the command recovery file if the next command given to the BACKUP program is not START. If START is the next command to BACKUP, the command recovery file is scanned, and the BACKUP program continues according to the information in the file. A ↑C CONT sequence does not delete the command recovery file, but completes the current requests. After all requests have been completed, the BACKUP program closes out the log file, and types BACKUP COMPLETED and an asterisk on the user's terminal indicating that it is ready for more requests.

MTA rewinds due to the magnetic tape being filled are actually rewind and unload operations to insure that the magnetic tape is not overwritten. When the BACKUP program reaches completion, the magnetic tape last written on remains in position unless a REWIND command is given.

Characteristics

The R BACKUP command:

Runs the BACKUP program, thereby destroying the user's core image.

**BACKUP program (Cont)**Associated Messages

Refer to Chapter 4.

Examples

```
._R BACKUP )  
/BACKUP MTA1:-DSKB:(10.2251*.* )  
/INDEX DSKC:BAKFIL.LST )  
/START )
```

```
._$BACKUP COMPLETED AT 16:45:22  
/C
```

```
._
```

**CLOSE command**

Function

The CLOSE command terminates any input or output currently in progress on the specified device, and automatically performs the CLOSE UUO (refer to DECsystem-10 Monitor Calls). Files are CLOSEd, but not RELEASEd, and logical names and device assignments are preserved. Since most programs CLOSE files when they finish performing a command string, the CLOSE command is provided for the occurrence of a program not terminating or a program being debugged. This command causes any disk files being written to be entered into the user's UFD. If a CLOSE is not done, the next RESET by a command (R, RUN, GET) or program will delete the partially written file.

Command Format

CLOSE dev

dev = the logical or physical name of the device on which I/O is to be terminated. This argument is optional.

If dev is omitted, I/O is terminated on all devices, except for the job's controlling terminal, and all files are CLOSEd.

Characteristics

The CLOSE command:

- Leaves the terminal in monitor mode.
- Requires core.
- Depends on FTFINISH which is normally absent in the DECsystem-1040.

Restrictions

The user cannot continue, but can start at the beginning or enter DDT.

Associated Messages

Refer to Chapter 4.

Examples

```

.CLOSE PTR;)
.CLOSE DEVA;)
.CLOSE)
.

```

**COMPILE command <sup>1</sup>**

Function

The COMPILE command produces relocatable binary files (.REL files) and/or compilation listings for the specified source program files. The assembler or compiler used is determined by the source file extension or by switches in the command string. If no switches appear in the command string, the following translators are used:

| <u>Source File Extension</u> | <u>Translator Used</u>                                                                                          |
|------------------------------|-----------------------------------------------------------------------------------------------------------------|
| .ALG                         | ALGOL compiler                                                                                                  |
| .BLI                         | BLISS compiler <sup>2</sup>                                                                                     |
| .CBL                         | COBOL compiler                                                                                                  |
| .F4                          | FORTRAN compiler (F40)                                                                                          |
| .MAC                         | MACRO assembler                                                                                                 |
| .P11                         | MACX11 assembler <sup>2</sup>                                                                                   |
| .SNO                         | SNOBOL compiler <sup>2</sup>                                                                                    |
| Other than above, or null    | Standard processor, which is usually FORTRAN at the beginning of the command string (refer to Paragraph 1.5.6). |

NOTE

If a source file has a recognizable processor extension (see above), the processor cannot be changed with a switch. The only time that a processor can be specified with a switch is when the source file has a non-recognizable processor extension or a null extension.

Normally the source file is translated if there is no corresponding binary (.REL) file or if the source file's date and time is later than or equal to the binary file's date and time. If the binary file is newer than the source file, the source file is not translated and the current .REL file is used. However, switches can be used to override this action.

Each time the COMPILE, LOAD, EXECUTE, or DEBUG command is executed, the command with its arguments is remembered in a temporary file on disk, or in core if the monitor has the TPCOR feature. Therefore, the filename used last can be recalled for the next command without specifying the arguments again (refer to Paragraph 1.5).

The COMPILE command accepts several command constructions: the @ construction (indirect commands), the + construction, the = construction, and the < > construction. Refer to Paragraph 1.5 for a complete description of each of these constructions.

---

<sup>1</sup> This command runs the COMPIL program, which interprets the command before running the appropriate processor.

<sup>2</sup> SNOBOL, BLISS, and MACX11 (the PDP-11 assembler for the PDP-10) will be recognized as processors only if the appropriate assembly switches are set. However, these assembly switch settings are not supported.

**COMPILE command (Cont)**

Command Format

COMPILE list

list = a single file specification, or a string of file specifications separated by commas. A file specification consists of a device name, a filename with or without an extension, and a directory name.

The following switches can be used to modify the command string. These switches can be temporary or permanent switches (refer to Paragraph 1.5.5). Note that all the switches allowed with the LOAD, EXECUTE, and DEBUG commands can be used with the COMPILE command. However, only the switches pertinent to COMPILE are listed below; the others are ignored.

- /ALGOL            Compile the file with ALGOL. Assumed for files with the extension of .ALG.
- /BLISS<sup>1</sup>         Compile the file with BLISS. Assumed for files with the extension of .BLI.
- /COBOL           Compile the file with COBOL. Assumed for files with the extension of .CBL.
- /COMPILE         Force a compilation on this file even though a binary file exists with a newer date and time than the source file. This switch is used to obtain an extra compilation (e.g., in order to obtain a listing of the compilation) since normally compilation is not performed if the binary file is newer than the source file.
- /CREF             Produce a cross-reference listing file on the disk for each file compiled for later processing by the CREF program. These files have the filename of the source file and the extension of .CRF. The file can then be listed with the CREF command. However, with COBOL files, the cross-referenced listing is always appended to the listing file. No additional command need be given to obtain the listing.
- /FORTRAN         Compile the file with FORTRAN. Assumed for files with the extension of .F4 and all files with non-recognizable processor extensions (if FORTRAN is the standard processor).

---

<sup>1</sup>BLISS will be recognized as a processor only if the appropriate assembly switch is set. However, this assembly switch setting is not supported.

**COMPILE command (Cont)**Command Format (cont)

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /FUDGE               | <p>Create a disk file containing the names of the .REL files produced by the command string. When the FUDGE command is given, PIP reads this file in order to generate a library REL file. Arguments to this switch are:</p> <p style="padding-left: 40px;">/FUDGE:dev:file.ext [proj,prog]</p> <p>dev: - the device on which to write the file. DSK: is assumed.</p> <p>file.ext - the name of the library file. The filename is required. If the extension is omitted, it is assumed to be .REL.</p> <p>[proj,prog] - the directory in which to place the file. The user's directory is assumed if none is given.</p> <p>This switch is permanent in the sense that it pertains to all .REL files generated by the command string.</p> |
| /LIST                | <p>Generate a disk listing file, for each file compiled, with the filename of the source file and the extension of .LST. These files can be listed later with the LIST command. Unless this switch is specified, listing files are not generated except in COBOL; COBOL listings are always generated.</p>                                                                                                                                                                                                                                                                                                                                                                                                                               |
| /MACRO               | <p>Assemble the file with MACRO. Assumed for files with extension of .MAC.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| /MACX11 <sup>1</sup> | <p>Assemble the file with MACX11. Assumed for files with extension of .P11.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| /NOCOMPIL            | <p>Complement the /COMPILE switch by not forcing a compilation on a source file whose date is not as recent as the date on the binary file. /NOCOMPIL is the default action.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| /NOLIST              | <p>Do not generate listing files. This is the default action except for COBOL files; COBOL listings are always generated.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| /SNOBOL <sup>1</sup> | <p>Compile the file with SNOBOL. Assumed for files with an extension of .SNO.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

---

<sup>1</sup>SNOBOL and MACX11 (the PDP-11 assembler for the PDP-10) will be recognized as processors only if the appropriate assembly switches are set. However, these assembly switch settings are not supported.

**COMPILE command (Cont)**

Characteristics

The COMPILE command:

- Leaves the terminal in monitor mode.
- Runs the appropriate processor.

Restrictions

The wildcard construction cannot be used.

Associated Messages

Refer to Chapter 4.

Examples

COMPILE PROG,TEST,MAC,MANAGE/COBOL )

Compiles PROG (with null extension) with FORTRAN, TEST.MAC with MACRO, and MANAGE (with null extension) with COBOL only if REL files do not exist with later dates. A listing file is generated only for MANAGE. The files generated are PROG.REL, TEST.REL, MANAGE.REL, and MANAGE.LST.

COMPILE /LIST SIGN,MAC,TABLES/NOLIST,MULTI.ALG )

Compiles SIGN.MAC with MACRO, TABLES (with null extension) with FORTRAN, and MULTI.ALG with ALGOL. Listing files are generated for SIGN.MAC and MULTI.ALG.

COMPILE/CREF/COMPILE DIVIDE,SUBTRC,ADD )

Forces a compilation of the source files although current .REL files exist and generates cross-referenced listing files. The files created are DIVIDE.CRF, DIVIDE.REL, SUBTRC.CRF, SUBTRC.REL, ADD.CRF, and ADD.REL.

COMPILE /FUDGE:MONITR.REL@LIBALL )

Compiles the files contained in the command file LIBALL and enters the names of all the REL files generated in a temporary disk file. When the FUDGE command is given, PIP generates the library REL file with name MONITR.REL. The library is created with the REL files in the same order as they were specified in the command file.

**COMPILE command (Cont)**Examples (cont)

```
._COMPILE OUTPUT=MTA0:(W,S,M)/L_
```

Rewinds the magnetic tape (W), compiles the first file with FORTRAN, produces binary output for the KA10 (S), and includes the MACRO coding in the output listing (M). These switches are processor switches (refer to Paragraph 1.5.7). A listing file is generated with the name OUTPUT.LST, along with the file OUTPUT.REL.

**CONTINUE command**

Function

The CONTINUE command starts the program at the saved program counter address stored in .JBPC by a HALT command (tC) or a HALT instruction. Refer to DECsystem-10 Monitor Calls for a description of the job data area.

Command Format

CONTINUE

Characteristics

The CONTINUE command:

- Places the terminal in user mode.
- Requires core.
- Does not require LOGIN.

Associated Messages

Refer to Chapter 4.

Example

```
.RUN LOOP)  
tC  
tC  
.DAYTIME)  
23-FEB-72 16:33:10  
_CONT)
```

Run a program called LOOP in your disk area.

Stop the program.  
Check the time of day.

Continue the program.

**COPY command <sup>1</sup>**Function

The COPY command transfers files from one standard I/O device to another. The command string can contain one device output specification and any number of input specifications. The equal sign separates the destination (output) side from the source (input) side. This command runs PIP and performs the basic PIP function of transferring files.

Command Format

COPY dev: file.ext [proj,prog] <nnn> = dev: file.ext [proj,prog], file.ext [proj,prog], ...

dev: = a physical or logical device name. If the device name is omitted, DSK: is assumed.

file.ext = the name of the file(s) to be used on input or for output. If the output filename is omitted, the input filename is assumed. PIP combines the files if many input files are being transferred to one output file. If many input files are being transferred to the same number of output files, PIP uses the /X switch to keep the files separate. The wildcard construction is allowed.

[proj,prog] = the disk area in which either the files are to be read or written. If this argument is omitted, the user's default disk area is assumed. The user may transfer files to or from another area only if he has access to the area.

<nnn> = the protection code to be given to the output file(s). If omitted, the system standard is assigned.

Switches can be passed to PIP by enclosing them in parentheses in the COPY command string. When COMPIL interprets the command string, it passes the switches on to PIP.

Characteristics

The COPY command:

- Leaves the terminal in monitor mode.
- Runs the PIP program, thereby destroying the user's core image.
- Depends on FTCCLX which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

---

<sup>1</sup>This command runs the COMPIL program, which interprets the command before running PIP.

**COPY command (Cont)**

Examples

.COPY=DTA3: FILNAM.MAC, MANY.CBL, COMMON.ALG )

The three files from DTA3 are transferred to the user's disk area with the same filenames.

.COPY DTA3: OUTPUT=\*.\*)

All files in the user's disk are transferred to one file on DTA3 with the name OUTPUT.

.COPY FILEA.\*=DTA1: SOURCE.\*)

The input files on DTA1 named SOURCE with any extension are transferred to DSK with the filename FILEA and the same extension. The number of output files equal the number of input files.

.COPY YOURS.CBL [20, 17] = MINE.CBL )

The file MINE.CBL from the user's disk area is transferred to [20,17] disk area with the name YOURS.CBL. The user must have privileges to write in area [20,17].

## COPY program

### Function

The COPY program is a DECTape copy routine that allows the user to

1. Copy the entire contents of an input DECTape to an output DECTape.
2. Zero all blocks on an output DECTape and clear the directory.
3. Perform a word-by-word comparison of two DECTapes.
4. Load a bootstrap loader and write it in blocks 0, 1, and 2 of the output DECTape.

### Command Format

```

_R COPY
*_output DTA:=input DTA: /switches

```

/switches = one or more of the following switches. Switches are preceded by a slash or enclosed in parentheses and can appear anywhere in the command string.

- |    |                                                                                                                                                                                                                        |
|----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /C | Copy all blocks from the input DECTape to the output DECTape.                                                                                                                                                          |
| /G | Do not restart the program after a parity error. Output an error message and continue the program.                                                                                                                     |
| /H | Type the available switches and their meanings.                                                                                                                                                                        |
| /L | Load the bootstrap loader into a core buffer. COPY expects the loader to be on logical device PTR in the file named BSLDR.REL. Note that COPY must be SAVED if the loader is to be preserved with the COPY core image. |
| /N | Suppress the directory listing.                                                                                                                                                                                        |
| /T | Write the bootstrap loader in blocks 0, 1, and 2 of the output DECTape. This switch accepts, as input from the terminal, a core bank or offset. The loader is offset and then written on the tape.                     |
|    | core bank = nnnK (16K to 256K)<br>offset = 1000 to 777600 octal                                                                                                                                                        |
| /V | Verify the similarities of the two DECTapes by performing a word-by-word comparison and typing on the terminal the number of discrepancies discovered.                                                                 |
| /Z | Zero all blocks of the output DECTape and clear the directory.                                                                                                                                                         |
| /6 | Look for the directory in PDP-6 format (i.e., in block one instead of block 144).                                                                                                                                      |

**COPY program (Cont)**

Command Format (cont)

If no switches are specified, /C (copy) and /V (verify) are assumed by default. Note that upon completion, the directory in core may not agree with the directory of the output DECTape. The output DECTape should be reassigned to guarantee that the directory in core is up-to-date.

Characteristics

The R COPY command:

- Places the terminal in user mode.
- Runs the COPY program, thereby destroying the user's core image.

Associated Messages

Refer to Chapter 4.

Examples

|                                                                                                                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> _R COPY ) *DTA7: = DTA3: )  *DTA2: /Z= ) *+C _ASSIGN DSK:PTR: ) _RENAME BSLDR.REL=DTBGOT.REL )  _R COPY ) */L ) *+C _SAVE DSK:COPY ) </pre> | <pre> Run COPY  Copy the contents of DTA3 to DTA7 and determine if the two DECTapes are the same (default condi- tion). If the DECTapes disagree, the number of discrepancies is typed on the terminal.  Zero all blocks and clear the directory on DTA2.  Return to monitor mode.  The bootstrap loader must be on logical device PTR.  COPY expects the bootstrap loader to be named BSLDR.  Run COPY  Load the bootstrap loader into a core buffer.  Return to monitor mode.  Save COPY so that the bootstrap loader is preserved with the COPY core image. </pre> |
|---------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**COPY program (Cont)**

Examples (cont)

|                                             |                                                                      |
|---------------------------------------------|----------------------------------------------------------------------|
| <u>.START</u> ↵                             | Start the COPY program.                                              |
| *DTA5: /T= ↵                                | Write the bootstrap loader in blocks 0, 1, and 2 of DTA5.            |
| <u>TYPE COKE BANK AND OFFSET FOR DTBOOT</u> | Respond with size of core bank or offset.                            |
| 64K ↵                                       | Size of core bank (64K core bank = 177000 offset, top of core -1000) |
| *↑C                                         | Return to monitor mode.                                              |

**CORE command**

Function

The CORE command types or modifies the amount of core assigned to the user's job. Because programs usually allocate core, the user generally does not need this command. It is included for completeness and is used more frequently in non-swapping systems than in swapping systems.

If the job is locked into core, this command with a nonzero argument cannot be satisfied and therefore gives an erroneous return.

Command Format

CORE n

n = a decimal number. This argument is optional.

If n is omitted, the monitor types out the amount of core used and does not change the core assignment.

If n = 0, the low and high segments disappear from the virtual addressing space of the job.

If n > 0, n represents the total number of blocks of core to be assigned to the job from this point on.

If n is less than high plus minimum low segment size, n plus high segment size is assumed.

Core arguments can be specified in units of 1024 words or 512 words (a page) by following n with the letter K or P, respectively. For example, 3P represents 3 pages or 1536 words. If K or P is not specified, K (1024 words) is assumed.

On systems with the KA10 processor (DECsystem-1040, 1050, or 1055), 1024 words is the minimum unit of allocation and therefore, all arguments are rounded up to the nearest multiple of 1024 words. For example, 3P on the KA10 is treated the same as 2K.

Characteristics

The CORE command:

Leaves the terminal in monitor mode.

Does not operate when a device is currently transmitting data.

**CORE command (Cont)**

Associated Messages

Refer to Chapter 4.

Examples

```

.CORE 5)
.CORE)
5+0/46K CCRE
VIR.CORE LEFT = 274
    
```

```

.CORE 10P)
.CORE)
10+0/93P CCRE
VIR.CORE LEFT = 549P
    
```

## CPUNCH command

### Function

The CPUNCH command is used to place entries into the card punch output queue. This command is equivalent to the following form of the QUEUE command:

QUEUE CDP: job name = list of input specifications

### Command Format

CPUNCH jobname = list of input specifications

jobname = name of the job being entered into the queue. The default is the name of the first file in the request, not the name of the first file given. These differ when the first file given does not yet exist.

input specifications = a single file specification or a string of file specifications, separated by commas, for the disk files being processed. A file specification is in the form dev:file.ext [proj,prog].

dev: = any file structure to which CDPSPL will have access; the default is DSK:.

file.ext = names of the files. The filename is optional. The default for the first filename is \*, the default for subsequent files is the last filename used. The extension can be omitted; the default is .CDP.

[proj,prog] = a directory to which the user has access; the user's directory is assumed if none is specified.

Note that if all arguments to the command are omitted (i.e., only the command name is given), the listing of all entries in the card punch queue for all jobs of all users is output.

The wildcard construction can be used for the input specifications. Switches that aid in constructing the queue entry can also appear as part of the input specifications. These switches are divided into three categories:

1. Queue-operation - Only one of these switches can be placed in the command string because they define the type of queue request. The switch used can appear anywhere in the command string.
2. General - Each switch in this category can appear only once in the command string because they affect the entire request. The switch used can appear anywhere in the command string.
3. File control - Any number of these switches can appear in the command string because they are specific to individual files within the request. The switch used must be adjacent to the file to which it applies. If the switch precedes the filename, it becomes the default for subsequent files.

**CPUNCH command (Cont)**

Command Format (cont)

The following switches can be used with the CPUNCH command.

| <u>Switch</u>     | <u>Explanation</u>                                                                                                                                                                                                                                                                                   | <u>Category</u> |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| /AFTER:tt         | Process the request after the specified time; tt is either in the form of hhmm (time of day) or +hhmm (time later than the current time). The resulting AFTER time must be less than the DEADLINE time. If the switch, or the value of the switch, is omitted, no AFTER constraints are assumed.     | General         |
| /BEFORE:t         | Queue only the files with a creation date before time t where t is in the form dd-mmm-yy hhmm. If the switch, or the value of the switch, is omitted, no BEFORE constraints are assumed.                                                                                                             | General         |
| /BEGIN:n          | Start the output on the nth card. The default is to begin output on the first card.                                                                                                                                                                                                                  | File Control    |
| /COPIES:n         | Repeat the output the specified number of times. N must be less than 64. If more than 63 copies are needed, two separate requests must be made. If this switch is not specified, the default is 1.                                                                                                   | File Control    |
| /CREATE           | Make a new entry into the card punch output queue. This switch is the default for the queue-operation switches.                                                                                                                                                                                      | Queue Operation |
| /DEADLINE:tt      | Process the request before the specified time; tt is either in the form hhmm (time of day) or +hhmm (time later than the current time). The resulting DEADLINE time must be greater than the AFTER time. If the switch, or the value of the switch, is omitted, no DEADLINE constraints are assumed. | General         |
| /DISPOSE:DELETE   | Delete the file after spooling.                                                                                                                                                                                                                                                                      | File Control    |
| /DISPOSE:PRESERVE | Save the file after spooling. This is the default for all files except files with extensions .LST, .TMP, or .CDP (if protection of .CDP is 0xx).                                                                                                                                                     | File Control    |

**CPUNCH command (Cont)**

Command Format (cont)

| <u>Switch</u>   | <u>Explanation</u>                                                                                                                                                                                                                  | <u>Category</u> |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| /DISPOSE:RENAME | Rename the file from the specified directory immediately, remove it from the logged-out quota, and delete it after spooling. This is the default for files with extensions of .LST, .TMP, and, if the protection is 0xx, .CDP.      | File Control    |
| /F              | List the entries in the card punch queue, but do not update the queues. Therefore, the list may not be an up-to-date listing but the listing will be faster than with /LIST.                                                        | Queue Operation |
| /FORMS:a        | Place the output on the specified form. The argument to the switch must be six alphabetic characters. The default is that normal forms are used.                                                                                    | General         |
| /KILL           | Remove the specified entry from the card punch queue. This switch can be used for deleting a previously-submitted request as long as the request has not been started by the Spoolers.                                              | Queue Operation |
| /LIMIT:n        | Limit the output to the specified number of cards.                                                                                                                                                                                  | General         |
| /LIST           | After updating the queues, list the entries in the card punch queue; if this switch, along with all other switches, is omitted, all entries for all jobs of all users are listed.                                                   | Queue Operation |
| /MODIFY         | Alter the specified parameters in the job. This switch requires that the user have access rights to the job. It can be used for altering a previously submitted request as long as the request has not been started by the Spooler. | Queue Operation |

**CPUNCH command (Cont)**Command Format (cont)

| <u>Switch</u> | <u>Explanation</u>                                                                                                                                                                                                     | <u>Category</u> |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| /NEW          | Accept the request even if the file does not yet exist. An appropriate error message is given if the file does not exist by the time the request is processed by the spooler.                                          | File Control    |
| /NULL         | Accept the request even if there is nothing in the request (i.e., create a queue entry to be later modified). No error message is given if there are no files in the request.                                          | General         |
| /OKNONE       | Do not output message if no files match the wildcard construction. This is assumed at KJOB time.                                                                                                                       | File Control    |
| /PHYSICAL     | Suppress logical device name assignments for the device specified.                                                                                                                                                     | File Control    |
| /PRIORITY:n   | Assign the specified external priority (n=0 to 62) to the request. The larger the number, the greater priority the job has. The default is 10 if no switch is given and 20 if the switch is specified without a value. | General         |
| /PROTECT:nnn  | Assign the protection nnn (octal) to the job. If the switch, or the value of the switch, is omitted, the standard protection is assumed.                                                                               | General         |
| /PUNCH:026    | Punch the files in 026 Hollerith code. If a /PUNCH: switch is not given the files are punched according to the data mode specified in the file.                                                                        | File Control    |
| /PUNCH:ASCII  | Punch the files in ASCII card code. If a /PUNCH: switch is not given, the files are punched according to the data mode specified in the file.                                                                          | File Control    |
| /PUNCH:BINARY | Punch the files in binary card code. If a /PUNCH: switch is not given, the files are punched according to the data mode specified in the file.                                                                         | File Control    |
| /PUNCH:D029   | Punch the files in DEC029 card code. If a /PUNCH: switch is not given, the files are punched according to the data mode specified in the file.                                                                         | File Control    |
| /PUNCH:IMAGE  | Punch the files in image card code. If a /PUNCH: switch is not given, the files are punched according to the data mode specified in the file.                                                                          | File Control    |
| /REMOVE       | Remove the file from the queue. This switch is valid only with the /MODIFY and can be used to remove a previously submitted file as long as CDPSPL has not started processing the request.                             | File Control    |

**CPUNCH command (Cont)**

Command Format (cont)

| <u>Switch</u> | <u>Explanation</u>                                                                                                                        | <u>Category</u> |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| /SEQ:n        | Specify a sequence number to help in identifying a request to be modified or deleted.                                                     | General         |
| /SINCE:t      | Queue only the files with creation dates after the specified time. t is in the form dd-mmm-yy hhmm.                                       | General         |
| /START:n      | Begin on the nth line of the file. If the switch, or the value of the switch, is omitted, the spooler starts with the first line.         | File Control    |
| /STRS         | Search for the file on all file structures in the search list and take each occurrence. The default is to take just the first occurrence. | File Control    |
| /UNPRESERVED  | Output the files only if they are not preserved (i.e., the first digit is 0). This switch avoids redundant punching.                      | General         |

Characteristics

The CPUNCH command:

- Leaves the terminal in monitor mode.
- Runs the QUEUE program, thereby destroying the user's core image.
- Depends on FTQCOM which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

Examples

.CPUNCH SYSTAT.MAC/PUNCH:ASCII)      Punch the file SYSTAT.MAC in ASCII format.

.CPUNCH SYSTAT.REL/PUNCH:BINARy/AFTER:1700)      Punch the file SYSTAT.REL in binary format, but do not begin punching it until after 5:00 pm.

**CREATE command <sup>1</sup>**Function

The CREATE command runs LINED (Line Editor for disk) and opens a new file on disk for creation. Refer to the LINED writeup in the DECsystem-10 Software Notebooks.

Command Format

CREATE file.ext

file.ext = any legal filename and filename extension. The filename is required; the filename extension is optional.

Characteristics

The CREATE command:

- Places the terminal in user mode.
- Runs the LINED program, thereby destroying the user's core image.
- Depends on FTCCLX which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

Example

```
._CREATE TEST1.F4 )  
_*
```

---

<sup>1</sup>This command runs the COMPIL program, which interprets the commands before running LINED.

**CREF command**

Function

The CREF command runs CREF and lists on the line printer (LPT) any cross-reference listing files generated by previous COMPILE, LOAD, EXECUTE, and DEBUG commands, using the /CREF switch, since the job was initiated. The file containing the names of these CREF-listing files is then deleted so that subsequent CREF commands will not list them again. The output goes either to LPT immediately or to the disk to be spooled later to LPT. When the logical device name LPT is assigned to a device other than the line printer, the CREF files are stored on that device with the same filename and the extension .LST.

Command Format

CREF

Characteristics

The CREF command:

- Leaves the terminal in monitor mode.
- Runs the CREF program, thereby destroying the user's core image.
- Depends on FTCCLX which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

Example

.COMPILE/CREF#PROMAC )

Compile the files contained in the command file PROMAC and produce CREF input compatible cross-reference listing files on the disk.

.CREF )

Process and list the cross-reference listing files produced by the COMPILE command.

.LOAD/C/MAP:NAME@CONALL )

Compile and load the files contained in the command file CONALL. Produce a loader map with the filename NAME and CREF input compatible cross-reference listing files on the disk.

.ASSIGN MTA1 LPT )

Assign the logical name LPT to MTA1.

.CREF )

Store the CREF files on MTA1 to be listed at a later time.

|                                                   |
|---------------------------------------------------|
| <b>CSTART command</b><br><b>CCONTINUE command</b> |
|---------------------------------------------------|

Function

The CSTART and CCONTINUE commands are identical to the START and CONTINUE commands, respectively, except that the terminal is left in the monitor mode.

Command Format

CSTART adr  
CCONTINUE

adr = the address at which execution is to begin if other than the location specified within the file (.JBSA). If adr is not specified, the starting address comes from .JBSA. An explicit starting address of 0 may be specified for adr.

To use:

1. Begin the program with the terminal in user mode.
2. Type control information to the program, then type 1C to halt the job with the terminal in monitor mode.
3. Type CCONTINUE to allow job to continue running and leave the terminal in monitor mode.
4. Additional monitor commands can now be entered from the terminal.

Characteristics

The CSTART and CCONTINUE commands:

Leave the terminal in monitor mode.

Require core.

Depend on FTATTACH which is normally absent in the DECsystem-1040.

Restrictions

These commands should not be used when the user program (which is continuing to run) is also requesting input from the terminal. These commands are not available to Batch users.

Associated Messages

Refer to Chapter 4.

CSTART command  
CCONTINUE command (Cont)

Example

```
.TYPE LOOP.F4)
  ACCEPT 10,I
10 FORMAT (I)
  DO 20 J=1,I
20 CONTINUE
  END
```

The user types his source program .

```
.EXECUTE LOOP)
```

The user compiles, loads, and executes the program .

```
FORTRAN:LOOP.F4
LOADING
LOOP 2K CORE
EXECUTION
1000000
```

The user indicates that the program should loop 1000000 times .

```
*C
*C
```

The user stops the program .

```
.CCONT)
```

The user continues the program but keeps the terminal in monitor mode .

```
.TIME)
```

The user times the program .

```
0.95
19.62
KILO-CORE-SEC=133
```

The program is still running .

```
.TIME)
1.45
23.07
KILO-CORE-SEC=157
```

```
.SYSTAT)
PLEASE TYPE AC FIRST
```

SYSTAT would cause the program to terminate .

```
.TIME)
0.00
23.07
KILO-CORE-SEC=157
.*C
```

The program appears to have finished because the runtime has stopped incrementing . The program will not output until the CONT command is given .

Return to the monitor .

```
.CONT)
```

Continue the program so it can complete its typing .

```
CPU TIME:5.55 ELAPSED TIME:1:5.73
NO EXECUTION ERRORS DETECTED
```

```
EXIT
```

```
:
```

**D (deposit) command**Function

The D command deposits information in the user's core area (high or low segment). When debugging a sharable program with the D command, the SAVE command should be used rather than the SSAVE command (refer to Appendix D).

Command Format

D lh rh adr

lh = the octal value to be deposited in the left half of the location. This argument is required.

rh = the octal value to be deposited in the right half of the location. This argument is required.

adr = the address of the location into which the information is to be deposited. This argument is optional.

If adr is omitted, the data is deposited in the location following the last D adr or in the location of the last E adr (whichever was last).

Characteristics

The D command:

Leaves the terminal in monitor mode.  
Requires core.

Associated Messages

Refer to Chapter 4.

**D command (Cont)**

Example

.D 266000 2616 141

Deposit in location 141.

.E 140  
000140/ 047000 000000

Examine location 140.

.D 47000 1

Since adr is omitted, the deposit is in the location of the last E command.

.E  
000140/ 047000 000001

.

The examine is of the location of the previous D command.

**DAYTIME command**Function

The DAYTIME command types the date followed by the time of day. The date and time are typed in the following format:

dd-mmm-yy hh:mm:ss

where

dd = day  
mmm = month  
yy = year  
hh = hours  
mm = minutes  
ss = seconds to nearest hundredth.

Command Format

DAYTIME

Characteristics

The DAYTIME command

Leaves the terminal in monitor mode.  
Does not require LOGIN.  
Does not destroy the user's core area.

Example

```
.DAY )  
11-SEP-70 22:36:34
```

```
.DA )  
15-DEC-71 :47:02
```

:

**DCORE command**

Function

The DCORE command causes the DAEMON program to write a core-image file of the user's core area that includes all accumulators and all relevant job tables. The job can continue to run; i.e., the DCORE command does not destroy the user's core area. The file produced may be later processed by the DUMP program, if the user so desires.

The DAEMON-written file consists of four categories: JOB, CONFIGURATION, DDB, and CORE. Each category has a two-word header, the first word contains the category number and the second word contains the number of data words in the category. The categories are as follows:

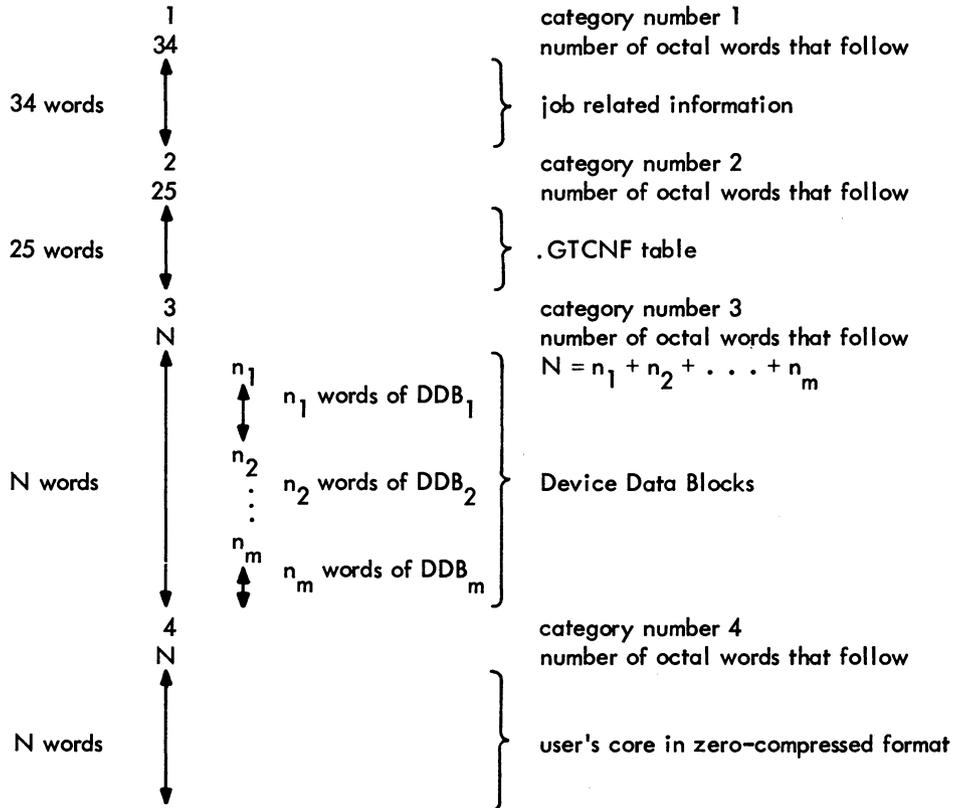
| <u>Mnemonic</u> | <u>Category Number</u> | <u>Description</u>                                                                                 |
|-----------------|------------------------|----------------------------------------------------------------------------------------------------|
| JOB             | 1                      | Job related information.                                                                           |
| CONFIGURATION   | 2                      | The Configuration Table (.GTCNF) from the GETTAB UUC.                                              |
| DDB             | 3                      | The device data blocks (DDB) assigned to this job.                                                 |
| CORE            | 4                      | The user's core area, both low and high segments, in zero-compressed format (refer to Appendix D). |

The third word of each category begins the data for that category. DAEMON treats each category as a file and the addresses within the category start at zero. The user cannot examine the category header nor can he read past the end of one category into the next category.

**DCORE command (Cont)**

Function (cont)

The DAEMON-written file appears as follows:



Category 1 presently contains the following information, but may expand as more GETTAB entries appear.

- Word 1                      Version of DAEMON that wrote the file.
- DATE the file was written in standard system format.
- TIME in milliseconds that the file was written.
- JOB NUMBER in left half, HIGH SEG number (or 0) in right half.

(continued on next page)

**DCORE command (Cont)**

Function (cont)

- Word 5            LH is reserved, TTY LINE NUMBER in right half.
  - .GTSTS (job status word) for job.
  - .GTSTS for high segment.
  - .GTPPN (project-programmer number) for job.
  - .GTPPN for high segment.
  - .GTPRG (user program name) for job.
  - .GTPRG for high segment.
  - .GTTIM (total time used) for job.
  - .GTKCT (kilo-core-ticks) for job.
  - .GTPRV (privilege bits) for job.
  - .GTSWP (swapping parameters) for job.
- Word 20            .GTSWP for high segment.
  - .GTRCT (disk blocks read) for job.
  - .GTWCT (disk blocks written) for job.
  - .GTTDB (time of day of last disk allocation, number of disk blocks allocated) for job.
  - .GTDEV (device or file structure name) for high segment.
  - .GTNM1 (first half of user's name) for job.
  - .GTNM2 (last half of user's name) for job.
  - .GTCNO (charge number) for job.
  - .GTTMP (TMPCOR pointers) for job.
  - .GTWCH (WATCH bits) for job.
  - .GTSPL (spooling control bits) for job.
  - .GTRTD (real-time status word) for job.
- Word 34            .GTLIM (time limit in jiffies) for job.

Category 2 presently contains the following information, but may expand if more .GTCNF entries are added.

- %CNFG0            Name of system in ASCIZ.
- ↓
- %CNFG4
- %CNDT0            Date of system in ASCIZ.
- %CNDT1
- %CNTAP            Name of system device in SIXBIT.
- %CNTIM            Time of day in jiffies.
- %CNDAT            Today's date.
- %CNSIZ            Highest location in monitor +1.
- %CNOPR            Name of OPR TTY console.

continued on next page)

**DCORE command (Cont)**Function (cont)

|        |                                                            |
|--------|------------------------------------------------------------|
| %CNDEV | LH = beginning of DDB chain.                               |
| %CNSGT | LH = -# of high segments, RH = +# of jobs.                 |
| %CNTWR | Non zero if system has two register hardware and software. |
| %CNSTS | LH = feature switches, RH = current state of switches.     |
| %CNSER | Serial number of processor.                                |
| %CNNSM | # of nanoseconds per memory cycle.                         |
| %CNPTY | PTY parameters for Batch.                                  |
| %CNFRE | AOBJN word to use bit map in monitor.                      |
| %CNLOC | LH = 0, RH = address of free 4-word core block area.       |
| %CNSTB | Link to STB chain for remote Batch.                        |

Category 3 contains the device data blocks currently in use for this job. Each DDB is preceded by a word containing the length of the DDB.

Category 4 is a compressed core image of both the high and low segments, i.e., it contains only nonzero words.

Command Format

DCORE dev:name.ext [proj,prog]

dev: = a disk-like device on which the core-image file is to be written. If omitted, DSK is assumed.

name.ext = the name of the file to be written. The default filename is nnnDAE, where nnn is the job number in decimal, and the default extension is .TMP. If a filename is specified, the default extension is .DAE.

[proj,prog] = the disk area other than that of the user. If omitted, the user's disk area (the number under which he is logged in) is assumed.

Characteristics

The DCORE command:

- Leaves the terminal in monitor mode.
- Runs the DAEMON program.
- Can continue after command.
- Depends on FTDAEM which is normally absent in the DECsystem-1040.

**DCORE command (Cont)**

Associated Messages

Refer to Chapter 4.

Examples

`._DCORE)`

The core image file is written on the user's area of the disk with the name nnnDAE.TMP where nnn is the user's job number.

`._DCORE DSKB:FILEC)`

The core image file is written in the user's area on DSKB with name FILEC.DAE.

`._`

**DDT command**Function

The DDT command copies the saved program counter value from .JBPC into .JBOPC and starts the program at an alternate entry point specified in .JBDDT (beginning address of DDT as set by Linking Loader). DDT contains commands to allow the user to start or resume at any desired address. Refer to DECsystem-10 Monitor Calls for a description of the job data area locations.

If the job was executing a UUO when interrupted (i.e., it was in exec mode and not in TTY input wait or SLEEP mode), the monitor sets a status bit (UTRP) and continues the job at the location at which it was interrupted. When the UUO processing is completed, the monitor clears the status bit, sets .JBOPC to the address following the UUO, and then traps to the DDT address found in .JBDDT. If the job is in exec mode and in TTY input wait or SLEEP mode, the trap to the DDT address occurs immediately and .JBOPC contains the address of the UUO. If the job is in user mode, the trap also occurs immediately. Therefore, it is always possible to continue the interrupted program after trapping to DDT by executing a JRSTF @ .JBOPC.

For additional information on the DDT program, refer to the DDT Programmer's Reference Manual in the DECsystem-10 Software Notebooks.

Command Format

DDT

Characteristics

The DDT command:

- Places the terminal in user mode.
- Requires core.
- Requires the user to have a job number.

Associated Messages

Refer to Chapter 4.

**DDT command (Cont)**

Examples

```
.TYPE LOOP.MAC)  
LOOP: JRST LOOP  
END LOOP
```

Type an undebugged program.

```
.DEBUG LOOP  
MACRO: .MAIN  
LOADING
```

Assemble and load the program with DDT.

```
LOOP 3K CORE  
DDT EXECUTION
```

```
!Z  
.SAVE)  
JOB SAVED
```

Save the program.

```
.START
```

Start the program.

```
!C  
!C
```

Stop it.

```
.DDT)
```

Enter DDT.

```
LOOP/JRST LOOP CALLI12  
JRSTF @.JBOPC$X
```

Fix the program.

```
EXIT
```

```
.
```

## DEASSIGN command

### Function

The DEASSIGN command returns one or more devices currently ASSIGNED to the user's job back to the monitor pool of available devices and clears any logical names. Restricted devices are returned to the restricted pool, and unrestricted devices to the unrestricted pool. Note that an INITed device is not returned to the monitor pool unless a RELEASE UUC is done, only the logical name is cleared. Therefore, this command is provided for programs that are not terminating or programs that are being debugged. The command, applied to DECtapes, clears the copy of the directly currently in core, forcing the next directory reference to read a new copy from the tape. (Refer to DECsystem-10 Monitor Calls for further details.)

### Command Format

DEASSIGN dev

dev = either the logical or physical device name. This argument is optional. If it is not specified, all devices assigned to the user's job, except the job's controlling terminal, are deassigned, and the logical name of the controlling terminal is cleared.

### Characteristics

The DEASSIGN command:

Leaves the terminal in monitor mode.

### Associated Messages

Refer to Chapter 4.

### Examples

```

.DEASSIGN LPT: )
.

```

The line printer is returned to the monitor's pool of available resources.

```

.DEASSIGN )
.

```

All devices assigned to the job are returned.

**DEBUG command <sup>1</sup>**

Function

The DEBUG command translates the specified source files if necessary (function of the COMPILER command), loads the REL files generated (function of the LOAD command), and prepares for debugging. DDT (the Dynamic Debugging Technique program) is loaded first, followed by the user's program with local symbols. Upon completion of loading, control is transferred to the DDT program. This program is used to check programs section by section by allowing the user to examine and modify the contents of any location either before execution or during breakpoints. Refer to the DDT documentation for a description of the DDT commands.

Each time a COMPILER, LOAD, EXECUTE, or DEBUG command is executed, the command with its arguments is remembered in a temporary file on disk, or in core if the monitor has the TMAPCOR feature. Therefore, the last filename used can be recalled for the next command without specifying the arguments again (refer to Paragraph 1.5).

The DEBUG command accepts several command constructions: the @ construction (indirect commands), the + construction, the = construction, and the <> construction. Refer to Paragraph 1.5 for a complete description of each of these constructions.

Command Format

DEBUG list

list = a single file specification, or a string of file specifications separated by commas. A file specification consists of a device name, a filename with or without an extension, and a directory name.

The following switches can be used to modify the command string. These switches can be temporary or permanent (refer to Paragraph 1.5.5).

- /ALGOL                    Compile the file with ALGOL. Assumed for files with the extension of .ALG.
- /BLISS<sup>2</sup>                    Compile the file with BLISS. Assumed for files with the extension of .BLI.
- /COBOL                    Compile the file with COBOL. Assumed for files with the extension of .CBL.

(continued on next page)

---

<sup>1</sup> This command runs the COMPIL program, which interprets the command before running the appropriate processor, the LOADER, and DDT.

<sup>2</sup> BLISS will be recognized as a processor only if the appropriate assembly switch is set. However, this assembly switch setting is not supported.

**DEBUG command (Cont)**

Command Format (cont)

- `/COMPILE`            Force a compilation on this file even though a binary file exists with a newer date and time than the source file. This switch is used to obtain an extra compilation (e.g., in order to obtain a listing of the compilation) since normally compilation is not performed if the binary file is newer than the source file.
- `/CREF`                Produce a cross-reference listing file on the disk for each file compiled for later processing by the CREF program. These files have the filename of the source file and the extension of .CRF. The file can then be listed with the CREF command. However, with COBOL files, the cross-reference listing is always appended to the listing file. No additional command need be given to obtain the listing.
- `/FORTRAN`            Compile the file with FORTRAN. Assumed for files with the extension of .F4 and all files with non-recognizable processor extensions (if FORTRAN is the standard processor).
- `/FUDGE`              Create a disk file containing the names of the .REL files produced by the command string. When the FUDGE command is given, PIP reads this file in order to generate a library REL file. Arguments to this switch are:

`/FUDGE:dev:file.ext [proj,prog]`

dev: - the device on which to write the file. DSK: is assumed.

file.ext - the name of the library file. The filename is required. If the extension is omitted, it is assumed to be .REL.

[proj,prog] - the directory in which to place the file. The user's directory is assumed if none is given.

This switch is permanent in the sense that it pertains to all REL files generated by the command string.
- `/LIBRARY`            Load the files in library search mode. This mode causes a program file in a special library file to be loaded only if one or more of its declared entry symbols satisfies an undefined global request in the source file. The system libraries are always searched. Refer to the LOADER documentation.

(continued on next page)

**DEBUG command (Cont)**

Command Format (cont)

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /LIST                | Generate a disk listing file, for each file compiled, with the filename of the source file and the extension.LST. These files can be listed later with the LIST command. Unless this switch is specified, listing files are not generated except in COBOL; COBOL listings are always generated.                                                                                                                                                                                     |
| /LMAP                | Produce a loader map during the loading process (same action as /MAP) containing the local symbols.                                                                                                                                                                                                                                                                                                                                                                                 |
| /MACRO               | Assemble the file with MACRO. Assumed for files with extensions of .MAC.                                                                                                                                                                                                                                                                                                                                                                                                            |
| /MAP                 | Produce a loader map during the loading process. When this switch is encountered, a loader map is requested from the loader. After the library search of the system libraries, the map is written in the user's disk area with either the filename specified by the user (e.g., /MAP:file) or the default filename MAP.MAP. This switch is an exception to the permanent switch rule in that it causes only one map to be produced even though it may appear as a permanent switch. |
| /MACX11 <sup>1</sup> | Assemble the file with MACX11. Assumed for files with the extension .P11.                                                                                                                                                                                                                                                                                                                                                                                                           |
| /NOCOMPIL            | Complement the /COMPILE switch by not forcing a compilation on a source file whose date is not as recent as the date on the binary file. Note that this switch is not the same as the /REL switch, which turns off all compilation, even if the source file is newer than the REL file. /NOCOMPIL is the default action.                                                                                                                                                            |
| /NOLIST              | Do not generate listing files. This is the default action except for COBOL files; COBOL listings are always generated.                                                                                                                                                                                                                                                                                                                                                              |
| /NOSEARCH            | Loads all routines of the file whether the routines are referenced or not. Since this is the default action, this switch is used only to turn off library search mode (/LIBRARY). This switch is not equivalent to the /P switch of the LOADER, which does not search any libraries. The /NOSEARCH default is to search the system libraries.                                                                                                                                       |

(continued on next page)

---

<sup>1</sup> MACX11 (the PDP-11 assembler for the PDP-10) will be recognized as a processor only if the appropriate assembly switch is set. However, this assembly switch setting is not supported.

**DEBUG command (Cont)**

Command Format (cont)

|                      |                                                                          |
|----------------------|--------------------------------------------------------------------------|
| /REL                 | Use the existing REL files although newer source files may be present.   |
| /SNOBOL <sup>1</sup> | Compile the file with SNOBOL. Assumed for files with extensions of .SNO. |

Characteristics

The DEBUG command:

Places the terminal in user mode.  
Runs the appropriate processor, the LOADER, and DDT.

Associated Messages

Refer to Chapter 4.

Examples

DEBUG/L FILEA,FILEB,FILEC/N,FILED )      **Generate listings for FILEA, FILEB, and FILED**

DEBUG TEST )  
MACRO: TEST  
LOADING

LOADER 2K CORE  
DDT EXECUTION

:/      BLT 15,0(16)

---

<sup>1</sup> SNOBOL will be recognized as a processor only if the appropriate assembly switch is set. However, this assembly switch setting is not supported.

**DELETE command <sup>1</sup>**

Function

The DELETE command deletes one or more files from disk or DECtape.

Command Format

DELETE list

list = a single file specification or a string of file specifications separated by commas. The full wildcard construction (\* and ?) can be used.

If a device or file structure name is specified, it remains in effect until changed or until the end of command string is reached.

Characteristics

The DELETE command:

- Leaves the terminal in monitor mode.
- Runs the PIP program, thereby destroying the user's core image.
- Depends on FTCCLX which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

Examples

```
.DEL *.MAC )
FILES DELETED:
T1.MAC
T2.MAC
T3.MAC
14 BLOCKS FREED
```

```
.DEL TEST1.MAC )
FILES DELETED:
TEST1.MAC
3 BLOCKS FREED
```

:

<sup>1</sup>This command runs the COMPIL program, which interprets the command before running PIP.

**DELETE command (Cont)**Examples (cont)

```
.DEL TEST?.F4  
FILES DELETED:  
TEST.F4  
TESTS.F4  
TEST03.F4  
TESTZ.F4  
23 BLOCKS FREED
```

**DETACH command**

Function

The DETACH command disconnects the terminal from the user's job without affecting the status of the job. The user terminal is now free to control another job, either by initiating a new job or attaching to a currently running detached job.

Command Format

DETACH

Characteristics

The DETACH command:

Detaches the terminal.  
Depends on FTATTACH which is normally absent in the DECsystem-1040.

Restrictions

This command is not available to Batch users.

Associated Messages

Refer to Chapter 4.

Example

```
.DETACH )  
FROM JOB 1
```

⋮

## DIRECT command

### Function

The DIRECT command lists the directory entries specified by the argument list. The standard output consists of the following columns: filename, filename extension, length in blocks written, protection, creation date, version number, structure.name, and directory name.

### Command Format

DIRECT output specification = list of input specifications

list = A single file specification, or a string of files specifications separated by commas or plus signs. The devices used on input can be DSK:, DTA:, MTA:, and TMP: (TMPCOR). If the device is a magnetic tape, the tape is rewound before and after the listing operation and analyzed to determine if it is a FAILSAFE or BACKUP tape. The default input specification is DSK:\*.\*, and the user's directories in all file structures defined by the job's search list are listed. Generally, a device name, an extension, or a directory name that precedes the filename becomes the default for all succeeding files in the list. The full wildcard construction can be used.

output specification = This argument (and the equal sign) is optional. If the entire output specification is omitted, the default is TTY:. If an output filename is given, the default device is DSK:. If an output filename is not given, and one is needed, the filename is generated from the time of day as hhmmss. The default output extension is .DIR. The wildcard construction cannot be used in the output specification.

The following switches may be used in the command string. Switches that precede the filename become the default for all succeeding files. Switches can be abbreviated as long as the abbreviation is unique.

|           |                                                                                                                                                                                                                                                                                                         |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /ACCESS:n | Update the access date to the current date for any file of n blocks or less accessed by the DIRECT program. n is the written length unless the ALLOC switch is used and is a decimal number. If /ACCESS is omitted, the date is not changed. If /ACCESS is specified but :n is omitted, n=5 is assumed. |
| /ALLOC    | List the allocated length of the file instead of the written length. The allocated length is used by LOGOUT in checking quotas. (Disk and magnetic tape only.)                                                                                                                                          |

(continued on next page)

**DIRECT command (Cont)**

Command Format (cont)

- `/CHECKSUM` Compute and print an 18-bit checksum for each file. This checksum is computed by rotating the result left one bit before adding each word. (Disk and magnetic tape only.)
- `/DENSITY:n` Use the specified density when reading a magnetic tape. N is 200, 556, or 800 bpi. The default is installation dependent and is modified by the SET DENSITY command.
- `/DETAIL` Print all nonzero words in the LOOKUP block. The protection and data mode are also listed, even if they are zero. The author is not listed if it is the same as the owner of the directory. (Disk and magnetic tape only.)
- `/FAST` List short form of directory (i.e., filename, extension, structure name, and directory name). Equivalent to /F.
- `/HELP` Help text which indicates some of the switches available and how to use them. Equivalent to /H.
- `/HELP:S` List all switches (S) without their explanations. An asterisk prefixes those switches which have a single-letter abbreviation.
- `/LIST` List the output on device LPT:. Equivalent to /L.
- `/MARKS` Indicate each tape mark and UFD when reading a magnetic tape.
- `/OKNONE` Suppress the error message if no files match the wildcard construction.
- `/PARITY:ODD`  
`/PARITY:EVEN` Specify the parity to be used when reading a magnetic tape. The default is ODD.
- `/PHYSICAL` Ignore logical names used for device names.
- `/PROTECTION:nnn` Give the output file the protection nnn (octal).
- `/RUN:file spec` Run the specified program when this command is finished.
- `/RUNOFFSET:n` Run the program specified with /RUN with an offset of n. If the switch is omitted, the default is 0; if the switch is given without a value, the default is 1.
- `/SLOW` Output a full listing that includes the filename, extension, length in blocks written, protection, creation time, access date, structure name, and directory name. Equivalent to /S. (Disk and magnetic tape only.)

(continued on next page)

**DIRECT command (Cont)**

Command Format (cont)

|          |                                                                                                                                                                                                                                          |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /SORT    | List the file structure name and directory name on each line instead of only on the first line in which they change. Multiple spaces are output instead of TABs. This switch is used to prepare a file to be sorted by the SORT program. |
| /SUMMARY | Output only the summary line which indicates the total number of blocks and files. Note a /F /SUMMARY lists a /F listing followed by the summary.                                                                                        |
| /TITLES  | Cause a heading to be output on each page consisting of a label for each column, date, time, and page number. Standard output to the line printer has this heading.                                                                      |
| /UNITS   | List the name of the actual disk unit instead of the file structure name.                                                                                                                                                                |
| /WIDTH:n | Output several entries on a single line to make the output n columns wide. For example, if /F is specified for output to the scope, four filenames appear per line. The default for n is 64 columns.                                     |
| /WORDS   | Output the length of the file in words instead of blocks.                                                                                                                                                                                |

Characteristics

The DIRECT command:

- Leaves the terminal in monitor mode.
- Runs the DIRECT program, thereby destroying the user's core image.
- Depends on FTCCLX which is normally absent in the DECsystem-1040.

Examples

|                       |                                                                                            |
|-----------------------|--------------------------------------------------------------------------------------------|
| .DIR DTA3: ↵          | Lists all files on DTA3.                                                                   |
| .DIR *.MAC ↵          | Lists all files with MAC filename extension in all file structures in the job search list. |
| .DIR TFST.F4[27,60] ↵ | Lists the directory entry for file TEST.F4 in user area 27, 60.                            |

(continued on next page)

DIRECT command (Cont)

Examples (cont)

\_DIRECT)

|        |     |    |       |           |        |          |
|--------|-----|----|-------|-----------|--------|----------|
| SAMPL  | CTL | 1  | <155> | 4-MAY-71  | DSKC:  | [27,235] |
| PIP    | DAE | 0  | <055> | 25-FEB-72 |        |          |
| G16DAE | TMP | 22 | <055> | 25-FEB-72 |        |          |
| WEIRD  |     | 4  | <055> | 25-FEB-72 |        |          |
| WEIRD  | SAV | 21 | <055> | 25-FEB-72 | 34(70) |          |

TOTAL OF 48 BLOCKS IN 5 FILES ON DSKC: [27,235]

\_DIRECT/ALLOC

|        |     |    |       |           |        |          |
|--------|-----|----|-------|-----------|--------|----------|
| SAMPL  | CTL | 3  | <155> | 4-MAY-71  | DSKC:  | [27,235] |
| PIP    | DAE | 2  | <055> | 25-FEB-72 |        |          |
| G16DAE | TMP | 24 | <055> | 25-FEB-72 |        |          |
| WEIRD  |     | 6  | <055> | 25-FEB-72 |        |          |
| WEIRD  | SAV | 23 | <055> | 25-FEB-72 | 34(70) |          |

TOTAL OF 58 BLOCKS IN 5 FILES ON DSKC: [27,235]

\_DIRECT/DETAIL)

DSKC0:SAMPL,CTL [27,235]  
 ACCESS DATE: 5-JAN-72  
 CREATION TIME, DATE: 16:51 4-MAY-71  
 ACCESS PROTECTION: 155  
 MODE: 14  
 WORDS WRITTEN: 54.  
 ESTIMATED LENGTH: 5.  
 BLOCKS ALLOCATED: 3.  
 DATA BLOCK IN DIRECTORY: 303.

DSKC0:PIP,DAE [27,235]  
 ACCESS DATE: 25-FEB-72  
 CREATION TIME, DATE: 13:47 25-FEB-72  
 ACCESS PROTECTION: 055  
 MODE: 17  
 BLOCKS ALLOCATED: 2.  
 AUTHOR: 1,2  
 DATA BLOCK IN DIRECTORY: 303.

(continued on next page)

**DIRECT command (Cont)**Examples (cont)

```
DSKC0:G16DAE.TMP [27,235]
ACCESS DATE: 25-FEB-72
CREATION TIME, DATE: 14:10 25-FEB-72
ACCESS PROTECTION: 055
MODE: 17
WORDS WRITTEN: 2816.
BLOCKS ALLOCATED: 24.
AUTHOR:1,2
DATA BLOCK IN DIRECTORY: 303.
```

```
DSKC0:WEIRD.[27,235]
ACCESS DATE: 25-FEB-72
CREATION TIME, DATE: 14:88 25-FEB-72
ACCESS PROTECTION: 055
MODE: 1
WORDS WRITTEN: 471.
BLOCKS ALLOCATED: 6.
DATA BLOCK IN DIRECTORY: 303.
```

```
DSKC0:WEIRD.SAV [27,235]
ACCESS DATE: 25-FEB-72
CREATION TIME, DATE: 14:09 25-FEB-72
ACCESS PROTECTION: 055
MODE: 10
WORDS WRITTEN: 2566.
VERSION:34(70)
BLOCKS ALLOCATED: 23.
DATA BLOCK IN DIRECTORY: 303.
```

TOTAL OF 48 BLOCKS IN 5 FILES ON DSKC: [27,235]

\_DIRECT(40,+1)

Lists the directory entries for user with project number 40 and the user's programmer number.

**DISMOUNT command**

Function

The DISMOUNT command allows a user to return devices to the monitor pool of available resources and to remove a file structure from his search list. Restricted devices are returned to the restricted pool and unrestricted devices to the unrestricted pool. The command applied to non-file structures is identical to the DEASSIGN command if the user waits for completion of the operator action. If the user does not wait for completion (e.g., he types a control-C after the message OPERATOR NOTIFIED), the device is not deassigned, but the request to the operator is still queued for the purpose of removing the media. The user must then issue the DEASSIGN command to release the device. This command applied to file structures enforces logged-out quotas (if necessary), allows physical removal of disk packs (if there are no other users of the pack), and removes the file structure name from the job's search list.

The UMOUNT program runs privileged in the user's core area when the DISMOUNT command is typed. This program scans the user's command string, checks its validity, and performs as much of the requested action as possible. The UMOUNT program can complete all actions requested by the DISMOUNT command except for the action of physically removing packs, tapes, or cards. When operator action is required, the UMOUNT program writes a command file on 3,3 disk area and notifies the OMOUNT program (running on the operator's terminal) to perform the action. When the operator action has been completed, OMOUNT deletes the command file and notifies UMOUNT (if UMOUNT is waiting) to inform the user of completion.

Command Format

DISMOUNT dev: switches

dev: = any previously ASSIGNED or MOUNTed device or file structure name. This argument is required.

switches = the following switches are optional and only enough characters to make the switch unique are required.

- /CHECK            Check and list pending requests.
- /HELP            Type this list.
- /PAUSE            Notify the user before requesting operator action. The user can then abort the command if desired.

(continued on next page)

**DISMOUNT command (Cont)**

Command Format (cont)

/REMOVE

Notify operator to physically remove disk packs, tape, or cards. A file structure is removed from the system only if no other users are using it. A request to remove the pack is queued to the operator and the message WAITING . . . is typed to the user. If the user does not want to wait for confirmation of the operator action, he may type control-C. This switch must be specified to notify the operator to remove the pack, even if no other jobs are using it.

Characteristics

The DISMOUNT command:

Places the terminal in user mode.

Runs the UMOUNT program, thereby destroying the user's core image.

Depends on FTCCLX and FTMOUN which are normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

Examples

```

.DISMOUNT DSKA:)
DSKA DISMOUNTED
    
```

The user dismounts the file structure DSKA. This does not require an operator action.

```

.DISMOUNT DTA4:/R)
OPERATOR NOTIFIED
    
```

The user asks the operator to deassign DTA4 and remove the tape.

```

WAITING...
    
```

The command is waiting for completion of the operator action.

```

^C
    
```

The user does not wish to wait for confirmation of removal.

```

.DISMOUNT/CHECK)
NONE PENDING
0,COMMANDS IN QUEUE
.
    
```

The user checks for completion and determines that his request is finished.

**DSK command**

Function

The DSK command types disk usage for the combined structures of the job, since the last DSK command, followed by the total disk usage since the job was initialized (logged in). Disk usage is typed in the following format:

RD, WT=I, J  
RD, WT=M, N

where I and J are the incremental number of 128-word blocks read and written since the last DSK command, and M and N are the total number of 128-word blocks read and written since the job was initialized.

NOTE

I and J are kept modulo 4096. If automatic READ or WRITE print outs have been enabled using the SET WATCH command, I and J are usually zero, since the SET WATCH output also resets these values.

Command Format

DSK job

job = the job number of the job for which the disk usage is desired. This argument is optional.

If job is omitted, the job to which the terminal is attached is assumed.

If job is supplied (whether the job of this user or another user) the incremental quantities are not reset to zero.

Characteristics

The DSK command:

Leaves the terminal in monitor mode.

Associated Messages

Refer to Chapter 4.

**DSK command (Cont)**

Example

.DSK)  
RD,WT=12,0  
RD,WT=475,243

.

**DUMP command**

Function

The DUMP command calls the DAEMON program to write a core image file (function of the DCORE command) and then invokes the DUMP program to analyze the file written and to provide printable output. The core image file is named nnnDAE.TMP where nnn is the user's job number. This file is described in detail in the DCORE command description.

Command Formats

1. DUMP /command /command /command ...
2. DUMP @ dev:file.ext [proj,prog]
3. DUMP

The commands that appear in the DUMP command string are passed to the DUMP program and therefore are described in the DUMP program description. A DUMP command using a command file can also specify these commands. A DUMP command without any arguments prints a short dump of the user's core area via the command file QUIKDM.CCL which resides on device SYS:.

Characteristics

The DUMP command:

- Leaves the terminal in monitor mode.
- Runs the DAEMON program.
- Depends on FTDAEM which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

Examples

```
.DUMP/OUT:TTY:/MODE:ASCII,SIXBIT/WIDTH:7,10/JUST:L,R-  
7RIGHTMA:26/D[3000 & 3004
```

This command string writes a core image file named nnnDAE.TMP and invokes the DUMP program to perform the output. The output goes to the terminal and the modes used on output are ASCII and SIXBIT. The ASCII field is 7 characters long, left justified and the SIXBIT field is 10 characters long, right justified. The right margin of the output is 26 characters. The dump consists of the contents of word 3000 to word 3004. The hyphen is used to continue the command string onto the next line.

**DUMP program**

Function

The DUMP program provides printable dumps of arbitrary data files in modes and forms specified by the user. The DUMP program accepts any data file as input and produces an ASCII file suitable for listing by PIP, the output spoolers, or other listing programs. For example, the DUMP program takes core image files prepared by the DAEMON program or SAVED files produced by the monitor. For a description of the DAEMON-written file, refer to the DCORE command.

Command Formats

1. R DUMP ↵  
/command ↵
2. R DUMP ↵  
/@ dev:file.ext [proj ,prog]

NOTE

DUMP indicates its readiness by typing a slash (/) instead of an asterisk.

The commands with their arguments are as follows. Lines can be continued by typing a hyphen followed by a carriage return.

| <u>Command</u> | <u>Argument</u>                                                         | <u>Meaning</u>                                                                                                                                                                                                        |
|----------------|-------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ADDRESS        | ON or OFF                                                               | Specifies if the address is to be dumped along with its contents. The default is ON.                                                                                                                                  |
| ALL            |                                                                         | Dumps the entire file. If the file is a DAEMON core image file, the entire category is dumped.                                                                                                                        |
| APPEND         |                                                                         | Appends the output to the output file. The existing output file is not overwritten. This command is the default; its complement is SUPERSEDE.                                                                         |
| AUTOFORMAT     | ON or OFF                                                               | Attempts to format output with line feeds, form feeds, and titles, if ON. If OFF, the user is responsible for all formatting. The default is ON.                                                                      |
| CATEGORY       | mnemonic for name of category. Can be JOB, CONFIGURATION, DDB, or CORE. | Selects the category of the DAEMON dump file to be used. Addressing begins with 0 at the beginning of each category. The default category is CORE. If the input file is not a DAEMON file, this switch has no effect. |

(continued on next page)

**DUMP program (Cont)**

Command Formats (cont)

| <u>Command</u> | <u>Argument</u>                         | <u>Meaning</u>                                                                                                                                                                                                                                                                                                                                                  |
|----------------|-----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CLOSE          |                                         | Closes the output file. After this command is given, another OUT command should be given before the next command which does any output.                                                                                                                                                                                                                         |
| DUMP           | dump descriptor,<br>dump descriptor,... | Dumps the specified bytes in the current modes.                                                                                                                                                                                                                                                                                                                 |
| EJECT          |                                         | Starts a new page in the output file.                                                                                                                                                                                                                                                                                                                           |
| EXIT           |                                         | Closes all files and returns control to the monitor (↑Z has the same effect).                                                                                                                                                                                                                                                                                   |
| INPUT          | <file descriptor>                       | Specifies the input file. The defaults are: DSK:nnnDAE.TMP where nnn is the job number; the user's directory. If the filename is specified, it determines the extension from the set .TMP, .DAE, .SHR, .SAV, .HGH, .LOW, .XPN, and .DMP in that order. If an extension is specified with no filename, the extension determines the filename.                    |
| IRADIX         | decimal number                          | Specifies radix for numbers for input. This command uses decimal to compute the argument. The default is 10 for decimal. The argument must be numeric. If numeral is 0 or is missing, the input radix is set back to its default value.                                                                                                                         |
| JUSTIFY        | LEFT, CENTER, or<br>RIGHT               | Specifies the justification of the output in the output field. If the output overflows the output field, the entire output appears; it is not truncated. This switch is used in a one-to-one relationship with the MODE and WIDTH commands. If there are more MODE commands, an argument of LEFT is used. If there are more JUSTIFY commands, they are ignored. |
| LEFTMARGIN     | expression                              | Sets the left margin of the output file. The default is 0.                                                                                                                                                                                                                                                                                                      |
| LINEPAGE       | expression                              | Specifies the number of lines per output page. The default is 50.                                                                                                                                                                                                                                                                                               |

(continued on next page)

**DUMP program (Cont)**

Command Formats (cont)

| <u>Command</u> | <u>Argument</u>                                                             | <u>Meaning</u>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------|-----------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MODES          | ASCII, DECIMAL, NULL, NUMERIC, OCTAL, RADIX50, SIXBIT, SOCTAL, or SYMBOLIC. | <p>Selects the type of output. ASCII dumps the word as a single right justified character if bits 0-28 are zero or as 5 ASCII characters if bits 0-28 are non-zero. Non printing characters print as a space. DECIMAL dumps as a signed decimal number. NULL declares that nothing is to be dumped. NUMERIC dumps as a signed number in the current output radix. OCTAL dumps as half-words separated by a comma (default) and takes 13 positions. RADIX50 dumps in RADIX50. SIXBIT dumps as one SIXBIT character if bits 0-24 are zero, or 6 SIXBIT characters if bits 0-24 are nonzero. SOCTAL dumps as signed octal and suppresses leading zeroes. SYMBOLIC dumps as a symbolic instruction.</p> <p>Any mode specification can appear more than once in the command string. The output is in the same order as the MODE list.</p> |
| NUMPAGE        | expression                                                                  | <p>Specifies that pages are to be numbered. If expression is 0, page numbering is turned off. If expression is not 0, page numbering begins at page = &lt;expression&gt;. If command is omitted, numbering starts at the first page.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| ORADIX         | decimal number                                                              | <p>Specifies radix for numbers for output. The default is 10 for decimal. If number is 0, the standard is used. The argument to this command is decimal and must not be an expression.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| OUTPUT         | <file descriptor>                                                           | <p>Specifies the output file. The defaults are: LPT:, the filename of the input file; the extension .LSD; the user's directory.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| RIGHTMARGIN    | expression                                                                  | <p>Sets the right margin of the output file. A field may overflow the right margin if it will not fit between the left and right margins. If ADDRESS is ON, the new line will have an address typed. If a page overflow occurs, a title line may also be printed.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| SUPERSEDE      |                                                                             | <p>Specifies that the output is to supersede an existing file of the same name, if there is one. The complement of this command is APPEND, which is the default.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

(continued on next page)

**DUMP program (Cont)**

Command Formats (cont)

| <u>Command</u>    | <u>Argument</u>                         | <u>Meaning</u>                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------------|-----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SYFILE            | <file descriptor>                       | Specifies the file to take symbols from if XTRACT command is specified. Defaults are: DSK:, the filename of the input file; one of the saved file extensions; the user's directory area.                                                                                                                                                                                                                                                                        |
| TDUMP             | dump descriptor,<br>dump descriptor,... | Dumps specified bytes to both output file and TTY.                                                                                                                                                                                                                                                                                                                                                                                                              |
| TITLE             | <string of characters>                  | Specifies a title to be included in the subsequent page headings. If no argument is specified, titling is turned off.<br><br>After this command, an EJECT command should be given to skip to a new page.                                                                                                                                                                                                                                                        |
| TYPE <sup>1</sup> | DAE                                     | Specifies that the input file is separated by DAEMON.                                                                                                                                                                                                                                                                                                                                                                                                           |
| TYPE              | DAT                                     | Specifies that the input file is a data file (i.e., no special format; therefore, no special processing is done).                                                                                                                                                                                                                                                                                                                                               |
| TYPE              | HGH                                     | Specifies that the input file is in .HGH file format.                                                                                                                                                                                                                                                                                                                                                                                                           |
| TYPE              | LOW                                     | Specifies that the input file is in .LOW file format.                                                                                                                                                                                                                                                                                                                                                                                                           |
| TYPE              | SAV                                     | Specifies that the input file is in .SAV file format.                                                                                                                                                                                                                                                                                                                                                                                                           |
| TYPE              | SHR                                     | Specifies that the input file is in .SHR file format.                                                                                                                                                                                                                                                                                                                                                                                                           |
| TYPE              | XPN                                     | Specifies that the input file is in .XPN file format.                                                                                                                                                                                                                                                                                                                                                                                                           |
| WIDTH             | expression                              | Selects the width of each output mode (see the MODE and JUSTIFY commands). If a MODE command is specified without a corresponding WIDTH, the byte is dumped in exactly the number of positions required followed by 3 blanks. If a WIDTH command is specified, no free blanks are output. If a MODE specification overflows its WIDTH specification, the entire output is given without justification. If expression is omitted, a null list will be generated. |
| XTRACT            |                                         | Uses the file specified in the last SY FILE command as a core image and extracts the symbol table.                                                                                                                                                                                                                                                                                                                                                              |

---

<sup>1</sup> If TYPE is not specified, the extension of the input file is used to determine the type of file being produced. If the extension is not one recognized in the TYPE command, TYPE DAE is assumed.

**DUMP program (Cont)**Command Formats (cont)

An expression is an octal or decimal number, a symbol, arithmetic operations using expressions (+, -, \*, /, and † grouped with parentheses), or contents operators ([, \, and @). A symbol is a string of SIXBIT characters, or program symbol, where program defines the program containing symbol.

A text string is a string of characters enclosed in single quotes. Special characters are represented by patterns of graphic characters. To override these special patterns, a double quote indicates that the next character is to be accepted as is, without including it as part of a special pattern. The following patterns represent non-graphic characters and are replaced in output strings by the characters represented unless a double quote appears.

|           |                        |
|-----------|------------------------|
| <EL>      | - end line, <CR-LF>    |
| <VT>      | - vertical tab         |
| <FF>      | - form feed            |
| <AL>      | - altmode              |
| <HT>      | - horizontal tab       |
| †<letter> | - control character    |
| \<letter> | - lower case character |

A byte descriptor is the description of the byte in the input file to be dumped. The format is:

WORDS <POS, SIZE>

where

WORDS = the address of the word desired.

POS = the position of the byte within the word. It specifies the bit number of the leftmost bit in the byte.

SIZE = the number of bits in the byte. It may be any size and can cross word or block boundaries.

A dump descriptor has the form

<FROM byte-descriptor> & <TO byte descriptor>

signifying everything from the first byte descriptor to the second.

**DUMP program (Cont)**

Characteristics

The R DUMP command:

- Places the terminal in user mode.
- Is used with disk monitors only.
- Runs the DUMP program.

Associated Messages

Refer to Chapter 4.

**E (examine) command**

Function

The E command examines a core location in the user's area (high or low segment).

Command Format

E adr

adr is required the first time the E or D command is used. If adr is specified, the contents of the location are typed out in half-word octal mode.

If adr is not specified, the contents of the location following the previously specified E adr or the location of the previous D adr (whichever was last) are typed out.

Characteristics

The E command:

- Leaves the terminal in monitor mode.
- Requires core.

Associated Messages

Refer to Chapter 4.

Example

```
.E 140 )
000140/ 264000 002616 .E
000141/ 000000 000000 .E
000142/ 000000 000000 .
```

**EDIT command <sup>1</sup>**

Function

The EDIT command runs LINED (Line Editor for disk) and opens an already existing line sequence-numbered file on disk for editing. Refer to the LINED writeup in the DECsystem-10 Software Notebooks.

Command Format

EDIT file.ext

file.ext = a filename and filename extension of an existing file. This argument is optional if a CREATE or EDIT command has been given since the initialization of the job, because the arguments of the EDIT-class commands are remembered in temporary files on the disk or in core if the monitor has the TMPCOR feature.

Characteristics

The EDIT command:

Places the terminal in user mode.  
Runs the LINED program, thereby destroying the user's core image.  
Depends on FTCCLX which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

Example

```
.EDIT TEST.F4 )  
*
```

---

<sup>1</sup> This command runs the COMPIL program, which interprets the command before running LINED.

**EOF command <sup>1</sup>**Function

The EOF command writes an end of file mark on the specified magnetic tape. This command is equivalent to the following PIP command string:

MTAn: (MF) ←

Command Format

EOF MTAn:

Characteristics

The EOF command:

Leaves the terminal in monitor mode.

Runs the PIP program, thereby destroying the user's core image.

Depends on FTCLX which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

Examples

.\_EOF MTA3:.)

⋮

---

<sup>1</sup>This command runs the COMPIL program, which interprets the command before running the PIP program.

**EXECUTE command <sup>1</sup>**

Function

The EXECUTE command translates the specific source files if necessary (function of COMPILE command), loads the REL files generated into a core image (function of LOAD command), and begins execution of the program. The assembler or compiler used is determined from the source file extensions or from switches in the command string (refer to the COMPILE command). If a REL file already exists with a newer date than that of the source file, compilation is not performed (unless requested explicitly via a switch).

This command is equivalent to a LOAD and START sequence of commands.

Each time a COMPILE, LOAD, EXECUTE, or DEBUG command is executed, the command with its arguments is remembered in a temporary file on disk, or in core if the monitor has the TMPCOR feature. Therefore, the last filename used can be recalled for the next command without specifying the arguments again (refer to Paragraph 1.5).

The EXECUTE command accepts several command constructions: the @ construction (indirect commands), the + construction, the = construction, and the <> construction. Refer to Paragraph 1.5 for a complete description of each of these constructions.

Command Format

EXECUTE list

list = a single file specification, or a string of file specifications separated by commas. A file specification consists of a device name, a filename with or without an extension, and a directory name.

The following switches can be used to modify the command string. These switches can be temporary or permanent switches (refer to Paragraph 1.5.5).

- /ALGOL            Compile the file with ALGOL. Assumed for files with the extension of .ALG.
- /BLISS<sup>2</sup>         Compile the file with BLISS. Assumed for files with the extension of .BLI.
- /COBOL           Compile the file with COBOL. Assumed for files with the extension of .CBL.

(continued on next page)

<sup>1</sup>This command runs the COMPIL program, which interprets the command before running the appropriate processor and the LOADER.

<sup>2</sup>BLISS will be recognized as a processor only if the appropriate assembly switch is set. However, this assembly switch setting is not supported.

**EXECUTE command (Cont)**Command Format (cont)

- /COMPILE** Force a compilation on this file even though a binary file exists with a newer date and time than the source file. This switch is used to obtain an extra compilation (e.g., in order to obtain a listing of the compilation) since normally compilation is not performed if the binary file is newer than the source file.
- /CREF** Produce a cross-reference listing file on the disk for each file compiled for later processing by the CREF program. These files have the filename of the source file and the extension of .CRF. The files can then be listed with the CREF command. However, with COBOL files, the cross-referenced listing is appended to the listing file. No additional command need be given to obtain the listing.
- /FORTRAN** Compile the file with FORTRAN. Assumed for files with the extension of .F4 and all files with nonrecognizable processor extensions (if FORTRAN is the standard processor).
- /FUDGE** Create a disk file containing the names of the .REL files produced by the command string. When the FUDGE command is given, PIP reads this file in order to generate a library REL file. Arguments to this switch are:
- /FUDGE:dev:file.ext[proj,prog]**
- dev: - the device on which to write the file. DSK: is assumed.
- file.ext - the name of the library file. The filename is required. If the extension is omitted, it is assumed to be .REL.
- [proj,prog] - the directory in which to place the file. The user's directory is assumed if none is given.
- This switch is permanent in the sense that it pertains to all REL files generated by the command string.
- /LIBRARY** Load the files in library search mode. This mode causes a program file in a special library file to be loaded only if one or more of its declared entry symbols satisfies an undefined global request in the source file. The system libraries are always searched. Refer to the LOADER documentation.
- /LIST** Generate a disk listing file, for each file compiled, with the filename of the source file and the extension of .LST. These files can be listed later with the LIST command. Unless this switch is specified, listing files are not generated except in COBOL; COBOL listings are always generated.

(continued on next page)

## EXECUTE command (Cont)

Command Format (cont)

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /LMAP                | Produce a loader map during the loading process (same action as /MAP) containing the local symbols.                                                                                                                                                                                                                                                                                                                                                                             |
| /MACRO               | Assemble the file with MACRO. Assumed for files with extensions of .MAC.                                                                                                                                                                                                                                                                                                                                                                                                        |
| /MACX11 <sup>1</sup> | Assemble the file with MACX11. Assumed for files with extensions of .P11.                                                                                                                                                                                                                                                                                                                                                                                                       |
| /MAP                 | Produce loader maps during the loading process. When this switch is encountered, a loader map is requested from the loader. After the library search of the system libraries, the map is written in the user's disk area with either the filename specified by the user (e.g., /MAP:file) or the default filename MAP.MAP. This switch is an exception to the permanent switch rule in that it causes only one map to be produced even though it appears as a permanent switch. |
| /NOCOMPILE           | Complement the /COMPILE switch by not forcing a compilation on a source file whose date is not as recent as the date on the binary file. Note that this switch is not the same as the /REL switch, which turns off all compilation, even if the source file is newer than the REL file. /NOCOMPILE is the default action.                                                                                                                                                       |
| /NOLIST              | Do not generate listing files. This is the default action except for COBOL files; COBOL listings are always generated.                                                                                                                                                                                                                                                                                                                                                          |
| /NOSEARCH            | Loads all routines of the file whether the routines are referenced or not. Since this is the default action, this switch is used only to turn off library search mode (/LIBRARY). This is not equivalent to the /P LOADER switch, which does not search any libraries; the /NOSEARCH switch scans the system libraries.                                                                                                                                                         |
| /REL                 | Use the existing REL files although newer source files may be present.                                                                                                                                                                                                                                                                                                                                                                                                          |
| /SNOBOL <sup>2</sup> | Compile the file with SNOBOL. Assumed for files with an extension of .SNO.                                                                                                                                                                                                                                                                                                                                                                                                      |

---

<sup>1</sup>MACX11, the PDP-11 assembler for the PDP-10, will be recognized as a processor only if the appropriate assembly switch is set. However, this assembly switch setting is not supported.

<sup>2</sup>SNOBOL will be recognized as a processor only if the appropriate assembly switch is set. However, this assembly switch setting is not supported.

**EXECUTE command (Cont)**Characteristics

The EXECUTE command:

- Places the terminal in user mode.
- Runs the appropriate processor and the LOADER.
- Starts the execution of the compiled and loaded program.

Associated Messages

Refer to Chapter 4.

Examples

```
.EXECUTE TEST
MACRO: TEST
LOADING

LOADER 2K CORE
EXECUTION
```

**FILCOM program**

Function

The FILCOM program is used to compare two versions of a file and to output any differences. Generally, this comparison is line by line for ASCII files or word by word for binary files. FILCOM determines the type of comparison to use by examining either the switches specified in the command string or the extensions of the files. Switches always take precedence over file extensions.

Command Format

```
.R FILCOM )  
*output dev:file.ext [proj,prog] = input dev1:file.ext [proj,prog],  
input dev2:file.ext [proj,prog]
```

output dev: = the device on which the differences are to be output.

input dev: = the device on which an input file resides.

Defaults

1. If the entire output specification is omitted (including the equal sign), the output device is assumed to be TTY.
2. If an output filename is specified, the default output device is DSK.
3. If the output filename is omitted, the second input filename is used, unless it is null. In this case, the filename FILCOM is used.
4. If the output extension is omitted, .SCM is used on a source compare and .BCM is used on a binary compare.
5. If the [proj,prog] is omitted (input or output side), the user's default directory is assumed.
6. If an input device is omitted, it is assumed to be DSK.
7. If the filename and/or extension of the second input file is omitted, it is taken from the first input file.
8. A dot following the filename of the second input is necessary to explicitly indicate a null extension, if the extension of the first input file is not null.

**FILCOM program (Cont)**Command Format (cont)

9. The second input file specification cannot be null unless a binary compare is being performed. In a binary compare, if the first input file is not followed by a comma and a second input file descriptor, the input file is compared to a zero file and is output in its entirety. This gives the user a method of listing a binary file. Refer to Example 4.

## Switches

The following switches can appear in the command string, depending on whether a source compare or a binary source compare is being performed.

Binary Compare

- |     |                                                                                                                                                                                                |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /H  | Type list of switches available (help text).                                                                                                                                                   |
| /nL | Specify the lower limit for a partial binary compare (n is an octal number). This switch, when used with the /nU switch, allows a binary file to be compared only within the specified limits. |
| /nU | Specify the upper limit for a partial binary compare (n is an octal number). This switch, when used with the /nL switch, allows a binary file to be compared only within the specified limits. |
| /W  | Compare files in binary mode without expanding the files first (refer to Appendix D). This switch is used to compare two binary files with ASCII extensions.                                   |
| /X  | Expand SAV files before comparing them in binary mode. This action removes differences resulting from zero compression (refer to Appendix D).                                                  |

Source Compare

- |    |                                                                                                                                                                                                                 |
|----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /A | Compare files in ASCII mode. This switch is used to force a source compare on two ASCII files with binary extensions.                                                                                           |
| /B | Compare blank lines. Without this switch, blank lines are ignored.                                                                                                                                              |
| /C | Ignore comments (all text on a line following a semicolon) and spacing (spaces and tabs). This switch does not cause a line consisting entirely of a comment to become a blank line, which is normally ignored. |

(continued on next page)

FILCOM program (Cont)

Command Format (cont)

Source Compare (cont)

- /H Type list of switches available (help text).
- /nL Specify the number of lines that determine a match (n is an octal number). A match means that n successive lines in each input file have been found identical. When a match is found, all differences occurring before the match and after the previous match are output. In addition, the first line of the current match is output after the differences to aid in locating the place within each file at which the differences occurred. The default value for n is 3.
- /S Ignore spaces and tabs.
- /U Compare in update mode. This means that the output file consists of the second input file with vertical bars next to the lines that differ from the first input file. This feature is useful when updating a document because the changes made to the latest edition are flagged with change bars in the left margin. The latest edition of the document is the second input file.

If switches are not specified in the command string, the files are compared in the mode implied by the extension. The following extensions are recognized as binary and cause a binary compare if one or both of the input files have one of the extensions.

- |      |      |      |
|------|------|------|
| .BAC | .HGH | .RMT |
| .BIN | .LOW | .RTB |
| .BUG | .MSB | .SAV |
| .CAL | .OVR | .SFD |
| .CHN | .QUE | .SHR |
| .DAE | .QUF | .SVE |
| .DCR | .REL | .SYS |
| .DMP | .RIM | .UFD |
|      |      | .XPN |

Binary files are compared word by word starting at word 0 except for the following two cases:

1. Files with extensions .SHR and .HGH are assumed to be high segment files. Since the word count starts at 400000, upper and lower limits, if used, must be greater than (or equal to in the case of the lower limit) 400000.
2. Files with extensions .SAV, .LOW, and .SVE are assumed to be compressed core image files and are expanded before comparing.

**FILCOM program (Cont)**

Command Format (cont)

Conflicts are resolved by switches or defaults. If a conflict arises in the absence of switches, the files are assumed to be ordinary binary files.

**Output**

In most cases, headers consisting of the device, filename, extension, and creation date of each input file are listed before the differences are output. However, headers do not appear on output from the /U switch (update mode on source compare).

Source compare output - After the headers are listed, the following notation appears in the left column of the output

n)m

where

n is the number of the input file, and  
 m is the page number of the input file (see examples).

The right column lists the differences occurring between matches in the input files. Following the list of differences, a line identical to each file is output for reference purposes.

The output from the /U switch differs from the above-described output in that the output file created is the second input file with vertical bars in the left column next to the lines that are different from the first input file.

Binary compare output - When a difference is encountered between the two input files, a line in the following format appears on the output device:

octal loc.      first file-word      second file-word      XOR of both words

If the exclusive OR (XOR) of the two words differs only in the right half, the third word output is the absolute value of the difference of the two right halves. This usually indicates an address that changed.

If one input file is shorter than the other, after the end of file is encountered on the shorter file, the remainder of the longer file is output.

**FILCOM program (Cont)**

Characteristics

The R FILCOM command:

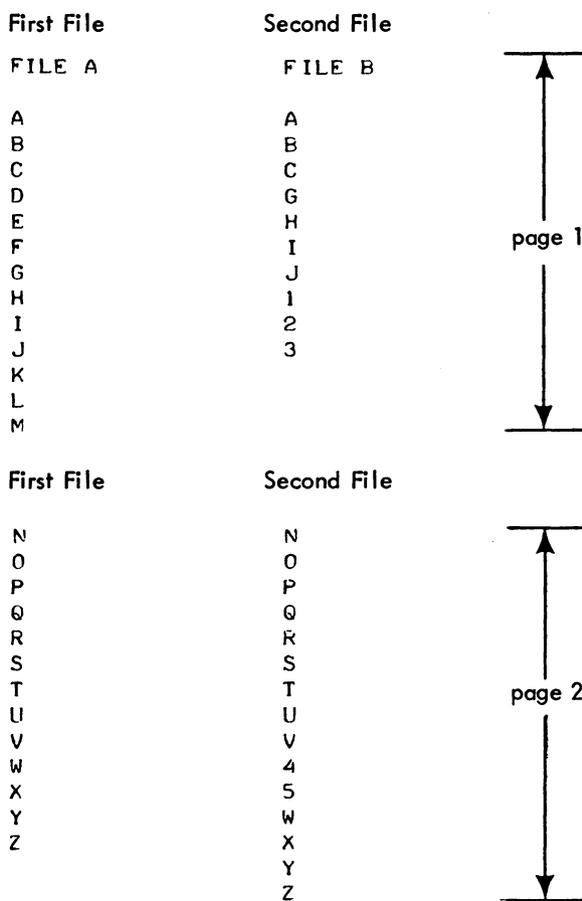
- Places the terminal in user mode.
- Runs the FILCOM program, thereby destroying the user's core image.

Associated Messages

Refer to Chapter 4.

Examples

1. The user has the following two ASCII files on disk:



(continued on next page)

**FILCOM program (Cont)**

Examples (cont)

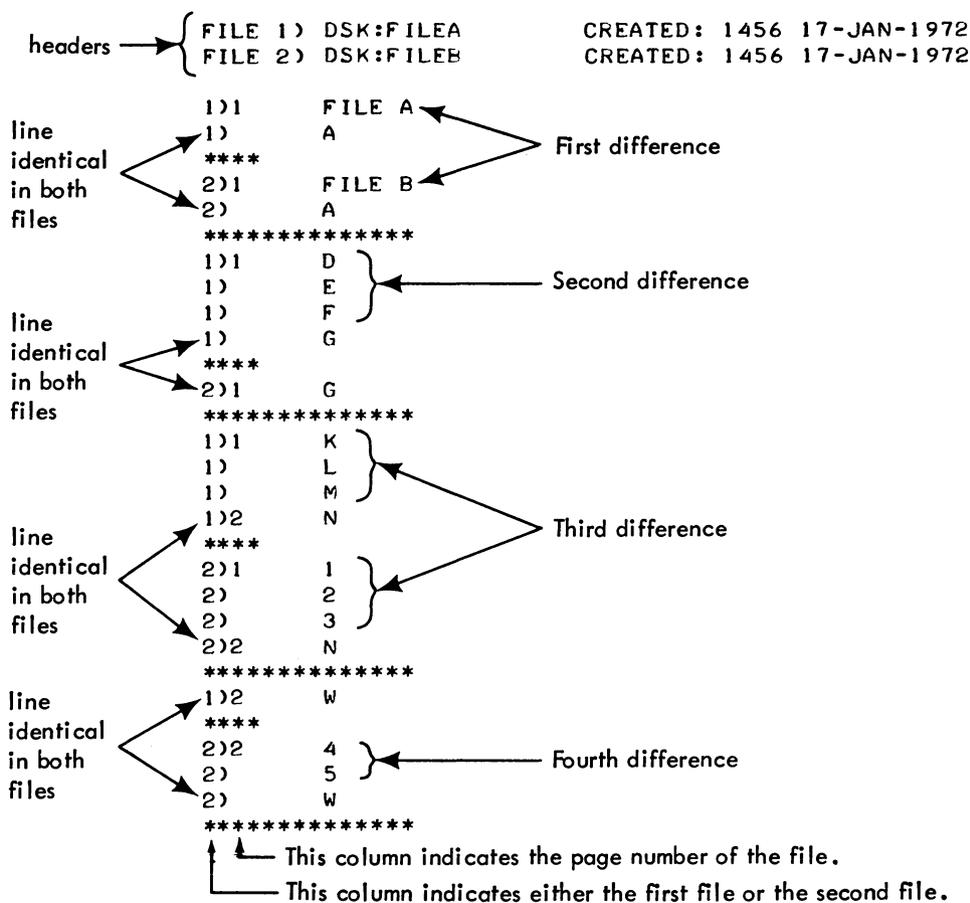
To compare the two files and output the differences on the terminal, the following sequence is used:

```

.R FILCOM )
.FILEA,FILEB )
    
```

Run the FILCOM program.

Compare the two files on disk and output the differences on the terminal. By default, three consecutive identical lines determine a match.



(continued on next page)

**FILCOM program (Cont)**

Examples (cont)

To compare the two files and output the differences on the line printer, the following commands are used. Note that in this example the number of successive lines that determines a match has been set to 4 with the /4L switch.

```
._R FILCOM)
*_LPT:/4L=FILEA,FILEB)
```

```
FILE 1) DSK:FILEA          CREATED: 1456 17-JAN-1972
FILE 2) DSK:FILEB          CREATED: 1456 17-JAN-1972
```

```
1)1  FILE A
1)   A
1)   B
1)   C
1)   D
1)   E
1)   F
1)   G
****
2)1  FILE B
2)   A
2)   B
2)   C
2)   G
```

These lines are listed as being different because the /4L switch specifies that 4 consecutive lines must be found identical in the two files before they are considered as a match.

```
*****
1)1  K
1)   L
1)   M
1)2  N
****
2)1  1
2)   2
2)   3
2)2  N
*****
1)2  W
****
2)2  4
2)   5
2)   W
*****
```

(continued on next page)

**FILCOM program (Cont)**

Examples (cont)

To compare the two files so that the second input file is output with vertical bars in the left column next to the lines that differ from the first input file, use the following command sequence.

```

.R FILCOM )
*LPT: /U=FILEA,FILEB )
    
```

FILE B

A  
B  
C  
G  
H  
I  
J  
I  
2  
3  
N  
O  
P  
Q  
R  
S  
T  
U  
V  
4  
5  
W  
X  
Y  
Z

The lines with vertical bars indicate the differences between the two files.

The lines with vertical bars indicate the differences between the two files.

2. To compare two binary files on the disk and output the differences on the terminal, use the following command sequence.

```

.R FILCOM )
*TTY: +DSK:DIAL.REL,DIAL2 )
FILE 1) DSK:DIAL.REL      CREATED: 0000 23-DEC-1971
FILE 2) DSK:DIAL2.REL    CREATED: 0000 12-AUG-1971

000000 000004 000001 000004 000060 000057
000002 000000 054716 000311 372712 000311 326004
000003 000006 000001 017573 510354 017575 510355
000004 000000 000000 017573 513216 017573 513216
    
```

**FILCOM program (Cont)**

Examples (cont)

- 3. To compare two high segment files, the command sequence below is used. Note that the locations begin at 400000.

```

.R FILCOM )
*TTY:~SYS:TABLE.SHR, TABLE.SHR )
FILE 1) SYS:TABLE.SHR      CREATED: 2020 24-JAN-1972
FILE 2) DSK:TABLE.SHR      CREATED: 1829 30-NOV-1971

400000 001611 400010      001630 407157      000021 007147
400003 006675 000000      015024 407670      013651 407670
400004 005600 000070      004700 000113      001100 000163
400005 545741 444562      554143 625700      011602 261262
400010 634000 000000      260740 403516      454740 403516
400011 474000 000000      200000 414036      674000 414036
400012 402000 000156      202000 000720      600000 000676
400013 200040 406354      201000 000472      001040 406726

```

- 4. To list a binary file, use the following command sequence.

```

.R FILCOM )
*TTY:~SYS:DOT.REL )
000000 000004 000001
000001 000000 000000
000002 000000 054716
000003 000006 000001
000004 000000 000000
000005 000007 517716
000006 000001 000002
000007 000000 000000
.
.
.

```

Note that the following sequence will not work because of the terminating comma.

```

*TTY:~SYS:DOT.REL, )
?COMMAND ERROR

```

**FILCOM program (Cont)**

Examples (cont)

5. To compare two binary files between locations 150-160 (octal).

```

.R FILCOM )
*TTY:/150L/160U+SYS:SYSTAT.SAV,SYS:SYSDPY.SAV )
FILE 1) SYS:SYSTAT.SAV   CREATED: 0818 30-NOV-1971
FILE 2) SYS:SYSDPY.SAV   CREATED: 1642 29-NOV-1971

000150 200400 000137    200740 003217    000340 003320
000151 260740 004226    404500 004242    664240 000064
000152 260740 004253    661500 002000    401240 006253
000153 200040 005011    260740 002723    060700 007732
000154 260740 004063    200040 004243    060700 000220
000155 201041 777777    202040 003241    003001 774536
000156 047040 000042    200040 004241    247000 004203
000157 254000 000174    251040 004142    005040 004036
000160 476000 006774    211040 000144    667040 006630
    
```

6. To compare two .SAV files. Note that the files are expanded before the comparison.

```

.R FILCOM )
*TTY:+SYS:TRY1.SAV,SYS:TRY.SAV )
FILE 1) SYS:TRY1.SAV     CREATED: 2043 05-JAN-1972
FILE 2) SYS:TRY.SAV      CREATED: 0818 30-NOV-1971

000114 004000 000140    000000 000000    004000 000140
000116 777536 005536    000000 000000    777536 005536
000117 000000 005536    000000 000000    000000 005536
000120 006000 000140    007222 000140    001222 000000
000121 000000 006000    000000 007222    001222 001222
000130 010000 000005    000000 000000    010000 000005
000133 003727 005777    006643 007777    005164 002000
000137 003400 000070    046700 000004    045300 000074
000140 264000 001454    047000 000000    223000 001454
000141 260040 001773    200040 005075    060000 004706
000142 201240 001447    402000 006644    603240 007203
000143 542240 001634    251040 007221    713200 006415
000144 260040 002774    403000 000015    663040 002761
000145 621000 000010    476000 006715    257000 006705
000146 200240 003504    200740 006606    000500 005302
000147 251240 000012    051140 005076    200300 005064
000150 402000 003613    200400 000137    602400 003724
000151 201040 003730    260740 004226    061700 007516
000152 200260 003632    260740 004253    060520 007461
000153 321240 000164    200040 005011    121200 005175
    
```

**FILE command**

Function

The FILE command provides remote control of DECTape-to-disk and disk-to-DECTape transfers on operator-handled DECTapes.

Command Formats

1. FILE C

Checks the queue of FILE commands to be read to determine if any of the user's requests are still pending. No argument is required. Pending requests will be listed.

2. FILE D, id, file.ext, file.ext, ...

Deletes the specified files from DECTape. Requires Tape ID and list of filenames as arguments. The tape ID is any alphanumeric name of 6 characters or less that is used to identify the tape. Upon completion, an automatic FILE L is performed.

3. FILE F, id, file.ext, file.ext, ...

Files information onto a DECTape. Requires Tape ID and list of filenames as arguments. Upon completion, an automatic FILE L is performed.

4. FILE L, id

Reads the directory of a DECTape and places it in the user's disk area as an ASCII file with filename id.DIR. id is any alphanumeric name of 6 characters or less that is used to identify the tape. It is the only argument. The user may then read the directory with a monitor command string. (See Examples).

5. FILE R, id, file.ext, file.ext, ...

Recalls (transfers) information from the user's DECTape to the disk. Requires Tape id and list of filenames as arguments. If the specified files already exist, they are superseded with the ones from the DECTape. If the specified files do not exist, they will be created on the first file structure in the job's search list for which creation is allowed. After the files are transferred, an automatic FILE L is performed.

**FILE command (Cont)**Command Formats (cont)

## 6. FILE W

Waits until all of the user's pending requests are processed before continuing. If there are pending requests, the message WAITING . . . is typed to the user. Control returns when all requests have been processed. The user may type control-C if he decides not to wait.

## 7. FILE Z, id, file.ext, file.ext, . . .

Zeros the directory of the DECTape before the files are copied and then performs the same operations as the F option. Requires Tape id and may have a list of filenames as arguments. After the files are copied, an automatic FILE L is performed.

The C and W functions are the only requests that are performed immediately. The other requests are placed in a queue to be performed whenever possible. The user's terminal and job are free to proceed before the request is completed. The function argument is optional. If the function argument is not specified, a brief dialogue is performed.

In most cases the user does not need to specify which file structures the files are on because UMount determines this (with LOOKUPs) and passes the information to OMount.

However, file structure names may be specified in file descriptors. When no structure name is explicitly typed, the default is initially the first file structure in the user's search list (implied by DSK:) on which he is allowed to create files. Refer to the description of the SETSRC program. When a file structure name is typed or implied, it becomes the new default.

The asterisk construction may be used, but care should be taken when generic DSK: is typed. Because DSK: may define many file structures, the single file structure is chosen as follows:

When the asterisk construction is used for the filename or extension, the first structure on which the user may create files in his search list is used. This is called the user's standard file structure.

If the asterisk construction is not used and the file exists, the first file structure in the search list that contains the specified file is used, unless overridden by a default. (See Examples.) If the file does not exist, the standard structure is used.

**WARNING**

If the user has a search list with multiple file structures, the asterisk construction when used with the FILE R command can cause files to be created rather than superseded.

**FILE command (Cont)**

Characteristics

The FILE command:

- Leaves the terminal in monitor mode.
- Runs the UMount program, thereby destroys the user's core image.
- Depends on FTCCLX which is normally absent in the DECsystem-1040.

Restrictions

The project-programmer number may not be specified in file descriptors.

Associated Messages

Refer to Chapter 4.

Examples

.FILE R,MINE,MAIN.F4,SUBFIL.MAC)

The files MAIN.F4 and SUBFIL.MAC are taken from the user's DECtape labeled MINE and placed on the first file structure in the user's search list for which creation is allowed. There are two commands in the queue (counting this one).

REQUEST STORED  
2.COMMANDS IN QUEUE

.FILE C)  
2. R JOB24 TTY5 27,235 MINE DSKB:,DSKB:MAIN.F4,SUBFIL.MAC)  
3. COMMANDS IN QUEUE

The user checks to see if his request is still waiting to be processed. The first line of the output indicates that the user's request is second in the queue (2.), that the request made is a RECALL (R), that the user's job is 24 (JOB24), that the user is on terminal 5 (TTY5) under the project-programmer number of [27,235] (27,235), that the tape is identified by the name MINE (MINE), and that the files will be written in the directory on DSKB: (DSKB:). The second line indicates that there are 3 commands in the queue.

(continued on next page)

**FILE command (Cont)**

Examples (cont)

.FILE L, 4)

The user wants the directory on the DECtape labeled 4 to be placed in his disk area as an ASCII file.

.TYPE 4.DIR)

The user then reads the directory file with the TYPE command.

If the user's search list is as follows:

DSKA/N, DSKB, DSKC

with file A on DSKA, file B on DSKB, and file C on DSKC, the following commands are equivalent:

The user types:

.FILE F,2,A,B,C

The user could have typed:

.FILE F,2,DSK:A,DSK:B,  
DSK:C

The command as passed to OMOUNT:

.FILE F,2,DSKA:A, DSKB:B,  
DSKC:C

The first file structure that contains each file is used.

.FILE R,3,A,DSKB:B,C

.FILE R,3,DSK:A,DSK:B,  
DSKB:C

.FILE R,3,DSKA:A,DSK:B,  
DSKB:C

The user changes the default to DSKB and even though file C exists on DSKC, file C is created on DSKB; files A and B are superseded.

.FILE F,1,\*. \*

.FILE F,1,DSK:\*. \*

.FILE F,1,DSKB:\*. \*

Because the asterisk convention was used, the first file structure on which the user may create files (DSKB) is used.

.FILE R,2,A,C.\*,  
DSKB:B.\*

.FILE R,2,DSK:A,  
DSKB:C.\*,DSKC:B.\*

.FILE R,2,DSKA:A,  
DSKB:C.\*,DSKC:B.\*

Because of the asterisk convention, DSKB is used for file C (even though file C exists on DSKC). The user explicitly typed a structure name for file B; therefore, DSKC is used even though file B is on DSKB. File A is superseded.

**FILEX program**

Function

The FILEX program is a general file transfer program intended to convert between various core image formats, and to read and write various directory formats. Files are transferred as 36-bit data. The only processing on the data is that necessary to convert between various core image representations.

Command Format

.R FILEX )

\*dev:ofile.ext [proj,prog] <nnn>/switches = dev:ifile.ext [proj,prog] /switches

If the project-programmer and/or the switches appear after the device name, they apply to all the following files. If they appear after the filename, the specifiers apply only to the preceding file. The input filename or extension may be \* in which case the usual processing of the \* construction occurs (refer to the TYPE command). The output filename and extension may be \* in which case the filename and extension of the input file is copied. If the output filename or extension is missing, the same procedure occurs as with the \* construction, except that all core image files are written with the default extension and format appropriate to the output device (unless overridden by switches).

If a protection <nnn> is not specified, files are written with the system standard protection unless the files are being written on SYS. On SYS, files are written with protection <155>, except for files with extension .SYS. These files have the default protection of <157>.

Meaning of Switches:

Help text

/H - to obtain an explanation of the command string and individual switches.

DECtape Format Specifiers

- /F - PDP-15 DECtape format
- /M - MIT project MAC PDP-6/10 DECtape format
- /O - Old DEC PDP-6 DECtape format
- /T - normal PDP-10 directory format
- /V - PDP-11 DECtape format (Note that PDP-11 contiguous files are not supported by FILEX.)

**FILEX program (Cont)**Command Format (cont)

## File Format Specifiers

- /A - ASCII processing; meaningful only for PDP-11 and PDP-15 tapes.
- /B - binary processing; overrides default extension.
- /C - compressed; save file format. This format is assumed for files with extensions .SAV, .LOW, .SVE. The default output extension is .SAV unless the input extension is .LOW or .SVE, in which case the extension remains unchanged.
- /D - dump format. This format is assumed for files with extension .DMP.
- /E - expanded core image files (used by FILDDT). This format is assumed for files with extension .XPN. The default output extension is .XPN.
- /I - image processing; meaningful only for PDP-11 and PDP-15 tapes.
- /S - simple block (SBLK) format, project MAC's equivalent of .SAV format. The default output extension is .BIN.

## DECtape Processing Specifiers

- /G - (go on), ignores read errors on input device. FILEX checks the always-bad-checksum bit in the 5-series monitor, so this switch is not needed for files with .RPABC on (e.g., CRASH.SAV).
- /L - (list), causes a directory on an input DECtape file to be typed on the terminal, or causes a directory listing of the output DECtape at the end (i.e., after the output).
- /P - (preserved), causes quick processing (/Q) and preserves the scratch file after processing for use by another command.
- /Q - (quick), causes an input or output DECtape to be processed quickly via a scratch file.
- /R - (reuse), reuses a scratch file preserved by a /P in a previous command.
- /Z - (zero), causes the appropriate format of a zeroed directory to be written on a DECtape output file. If TAPEID appears in the output specifier, then TAPEID is written as the tape identifier in the directory. TAPEID is preceded by a up arrow (^) and may be 6 characters on a PDP-10 tape, 3 characters on a project MAC tape, and is not present on a PDP-6 tape.

**FILEX program (Cont)**

Characteristics

The R FILEX command:

Runs the FILEX program, thereby destroying the user's core image.

Examples

```
.R FILEX )  
*DSK:+DTA1:TEST.DMP/C
```

The dump format file is compressed and written as TEST.SAV.

```
.R FILEX  
*DSK:SER105.XPN[10,1]+DSK:CRASH.SAV[1,4]
```

Copy CRASH.SAV to an expanded format file for FILDDT to examine.

**FINISH command**Function

The FINISH command terminates any input or output currently in progress on the specified device and automatically performs the RELEASE UJO (which CLOSES the files) and DEASSIGN command, thus making the device available to another user. This command is preferred over the DEASSIGN command because it completely disassociates an INITed device from the user's job, thereby preventing the user from continuing his program. If the user wishes to continue his program, he should use the DEASSIGN command.

Command Format

FINISH dev

dev = the logical or physical name of the device on which I/O is to be terminated.  
This argument is optional.

If dev is omitted, I/O is terminated on all devices, except the job's controlling terminal and the logical name of the controlling terminal is cleared.

Characteristics

The FINISH command:

Leaves the terminal in monitor mode.

Requires core.

Depends on FTFINISH which is normally absent in the DECsystem-1040.

Restrictions

The user cannot continue his program if the device was INITed, but he can start at the beginning or enter DDT.

Associated Messages

Refer to Chapter 4.

**FINISH command (Cont)**

Examples

```
.FINISH CDR:)
.
.FINISH DTA7:)
.
.FINISH LPT:)
.
```

**FUDGE command <sup>1</sup>**Function

The FUDGE command causes PIP to read a temporary file generated by a previous COMPIL, LOAD, EXECUTE, or DEBUG command using the /FUDGE switch and to create a library REL file. The library is created with the REL files in the same order in which they were specified in the command string containing the /FUDGE switch.

## NOTE

Since the COMPIL program sorts out files by compilers, mixed FORTRAN and MACRO programs are sorted so that all FORTRAN programs are compiled first and MACRO programs second. However, the /FUDGE switch combines them in the order in which the COMPIL program encountered them.

Command Format

FUDGE

Characteristics

The FUDGE command:

- Leaves the terminal in monitor mode.
- Runs the PIP program, thereby destroying the user's core image.
- Depends on FTCCLX which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

Examples

```
._COMPIL/FUDGE:LIBRARY/MACRO TEST,MATH,DATPRO,CBL,SCIENC.F4.)
```

Create a disk file named LIBRARY which contains the names of all the REL files produced.

```
._FUDGE.)
```

Create the library file and call it LIBRARY. This file contains the following: TEST.REL, MATH.REL, DATPRO.REL, and SCIENC.REL.

---

<sup>1</sup> This command runs the COMPIL program, which interprets the command before running the PIP program.

**FUDGE2 program**

Function

The FUDGE2 program is used to update files containing one or more relocatable binary programs and to manipulate programs within program files. Three files are used in the updating process:

1. A master file containing the file to be updated.
2. A transaction file containing the file of programs to be used when updating.
3. An output file containing the updated file.

All three files can be on the same device if the device is DSK. The two input files can be on the same DECtape.

The desired function of FUDGE2 is specified by a command code at the end of the command string. Only one command code can be specified in each command string. The command string is then terminated with an ALTmode, represented in this manual by a dollar sign (\$). Switches can also be used to manipulate file directories and to position a magnetic tape.

Command Format

`.R FUDGE2 )`  
`output dev:file.ext=master dev:file.ext<programs>, transaction dev:file.ext<programs>`  
`(command)$`

output dev: = the device on which the updated file is written. If omitted, DSK is assumed.

master dev: = the device containing the file to be updated. If omitted, the default is DSK.

transaction dev: = the device containing the files of programs to be used in the updating process. When more than one file is transferred from magnetic tape or paper tape, a colon must follow the device name for each file. For example,

MTA: : : Transfer 3 files

If the device is omitted, DSK is assumed.

file.ext = the filename and extension of each file. Filenames must be specified for directory devices, but the extension can be omitted. If the extension is not given, it is assumed to be .REL unless the /L switch appears in the command string. In this case, the output extension .LST is assumed.

(continued on next page)

**FUDGE2 program (Cont)**

Command Format (cont)

- `file.ext (cont)`      Project-programmer numbers appearing after a filename apply to that file only. If the project-programmer number appears before the filename, it applies to all subsequent files until another device is specified.
- The asterisk convention can be used with the input files (refer to Paragraph 1.4.2.4).
- `<programs>`              = Names of programs (on DSK or DTA only) to be used in the updating process. They are grouped within angle brackets in the same order as they appear in the file and are separated by commas. When manipulating all the programs within a file, only the filename need be specified. Program names cannot appear for the output file.
- `(command)`              = Code for the function to be performed. This code can be either preceded by a slash or enclosed in parentheses and must appear at the end of the command string. Each command results in the updated file being output to the output device. The command codes are as follows:
- A      Append the specified programs in the transaction file(s) to the master file.
  - C      Delete local symbols from the master file.
  - D      Delete the specified programs from the master file.
  - E      Extract the specified files and/or programs from the input files. The entire file is extracted if program names are not specified.
  - H      Type the commands and switches available.
  - I      Insert programs from the specified transaction files into the master file. The programs from the transaction files are inserted immediately before the specified programs in the master file.
  - L      List the names and lengths of all relocatable programs within a file. The length is in one of two forms:
    - low segment break, high segment break or
    - program break, absolute break
- The length of FORTRAN programs is not output.

(continued on next page)

**FUDGE2 program (Cont)**

Command Format (cont)

- |                  |   |                                                                                                                                                                                                |
|------------------|---|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| (command) (cont) | R | Replace the specified programs in the master file with the specified programs in the transaction file. The number of replacing programs must be the same as the number of programs to replace. |
|                  | S | List all the entry points within a program. These entry points are listed across the page.                                                                                                     |
|                  | X | Write index blocks into a library file. Index blocks are used in a direct access library search (refer to the LOADER documentation). This command implies a C command.                         |

File directories can be manipulated and magnetic tapes positioned by including switches in the command string. These switches can appear anywhere in the command string and are preceded by a slash or enclosed in parentheses. The following switches are available:

- |    |                                                     |
|----|-----------------------------------------------------|
| /B | Backspace a magnetic tape one file.                 |
| /K | Advance a magnetic tape one file.                   |
| /T | Skip to the logical end of tape on a magnetic tape. |
| /W | Rewind a magnetic tape.                             |
| /Z | Clear the directory of the output DECtape.          |

Characteristics

The R FUDGE2 command:

- Places the terminal in user mode.
- Runs the FUDGE2 program, thereby destroying the user's core image.

Associated Messages

Refer to Chapter 4.

**FUDGE2 program (Cont)**

Examples

```

.R FUDGE2
*LPT:=DTA1:LIB40(L)$
*DSK:LIB4BB=DTA2:LIB4AA <EXP.3,EXP.3C>
DTA1:F4<EXP.3A,EXP.3B>(R)$
*DTA1:NFILE=DSK:MFILE <M1,M2,M3,M4>
DTA3:TFILEA<TA1,TA2>
DTA4:TFILEB<TB1,TB2>/I$
*DTA1:NFILE=DSK:MFILE <M1,M2,M3,M4>
DTA3:TFILEA
DTA4:TFILEB/I$
*DTA2:TESTA=MTA1:(WK),MTA2: :(ZA)$
*OUTPUT=LIBRARY,DTA1:LIBRARY<FILEY,FILEZ>/A$

```

List all relocatable programs (.REL) from the file LIB40, located on DTA1, on the line printer.

Replace programs EXP.3 and EXP.3C located in file LIB4AA on DTA2, with programs EXP.3A and EXP.3B in File F4 on DTA1; write out the new LIB4AA file on disk and call it LIB4BB.

Insert into MFILE the programs TA1 and TA2 from TFILEA, and TB1 and TB2 from TFILEB. Create NFILE with the following order:

TA1,M1,TA2,M2,TB1,M3,TB2,M4

Insertion is on a one-to-one basis. If there are more programs to be inserted than specified programs before which they are to be inserted, the extra files are ignored.

However, in this example (where TFILEA and TFILEB contain the programs TA1 and TA2 and TB1 and TB2, respectively) create an NFILE with the following order:

TA1,TA2,M1,TB1,TB2,M2,M3,M4

Clear the directory of DTA2; rewind MTA1 and advance the tape one file; append the first two program files from MTA2 to the second file on MTA1 and write out the resultant file on disk, calling it TESTA.

Append the programs FILEY and FILEZ contained in the file LIBRARY on DTA1 to the end of the file LIBRARY on disk. Write the new file on disk and call it OUTPUT.

(continued on next page)

**FUDGE2 program (Cont)**

Examples (cont)

```
*NEWFIL=OLDFIL<TEST,SUBTRC,MULTI>,BASFIL<PROG,ROUTIN,ANSWER>,SUBFIL<MATH>(E)$  
*!C
```

Extract the specified programs from the files  
OLDFIL, BASFIL, and SUBFIL and create a new  
output file called NEWFIL. The order of the pro-  
grams in NEWFIL is as follows: TEST, SUBTRC,  
MULTI, PROG, ROUTIN, ANSWER, MATH.

Return to the monitor.

**GET command****Function**

The GET command loads a core image from a retrievable storage device but does not begin execution.

This command clears all of user core. However, programs should not count on this action and should explicitly clear those areas of core that are expected to contain zeroes (i.e., programs should be self-initializing). This action allows programs to be restarted by a tC, START sequence without having to do another GET command.

On magnetic tape, if the low or high segment is missing, a null record is output before the EOF for the missing segment so that two EOFs cannot occur consecutively. Therefore, a saved null segment does not appear as a logical EOT (2 EOFs in a row).

**Command Format**

GET dev:file.ext [proj,prog] core

The arguments and the defaults are the same as in the RUN command.

The extension applies to the low file, not the high file. An extension of .SHR, then .HGH, is assumed for the high file. If the user types an extension of .SHR or .HGH, the extension is treated as a null extension since .SHR and .HGH are confusing as low file extensions.

**Characteristics**

The GET command:

Leaves the terminal in monitor mode.

Does not operate when the device is currently transmitting data.

**Associated Messages**

Refer to Chapter 4.

**Example**

```
.GET SYS:PIP )  
JOB SETUP
```

```
.GET TEST )  
JOB SETUP
```

**GLOB program**

Function

The GLOB program reads multiple binary program files and produces an alphabetical cross-referenced list of all the global symbols (symbols accessible to other programs) encountered. This program also searches files in library search mode, checking for globals, if the program file was loaded by the LOADER in library search mode (refer to the LOADER documentation).

The GLOB program has two phases of operation; the first phase is to scan the input files and build an internal symbol table, and the second, to produce output based on the symbol table. Because of these phases, the user can input commands to GLOB in one of two ways. The first way is to specify one command string containing both the output and input specifications. (This is the command string format most system programs accept.) The second is to separate the command string into a series of input commands and output commands.

Command Formats

1. R GLOB

outdev:file.ext [proj,prog] = input dev:file.ext [proj,prog] ,file.ext ,... ,dev:file.ext [proj,prog] (\$)

2. R GLOB

followed by one or more input commands in the form

dev:file.ext [proj,prog] ,file.ext [proj,prog] ,... ,dev:file.ext [proj,prog] ,... )

and then one or more output commands in the form

outdev:file.ext [proj,prog] = (\$)

When the user separates his input to GLOB into input commands and output commands (Command Format #2), the input commands contain only input specifications and the output commands, only output specifications. Each output command causes a listing to be generated; any number of listings can be printed from the symbol table generated from the current input files as long as no input commands occur after the first output command. When an input command is encountered after output has been generated, the current symbol table is destroyed and a new one begun.

Defaults

- 1. If the device is omitted, it is assumed to be DSK. However, if the entire output specification is omitted, the output device is TTY.

(continued on next page)

**GLOB program (Cont)**Command Format (cont)Defaults (cont)

2. If the output filename is omitted, it is the name of the last input file. The input filenames are required.
3. If the output extension is omitted, .GLB is used. If the input extension is omitted, it is assumed to be .REL unless the null extension is explicitly specified by a dot following the filename.
4. If the project-programmer number [proj,prog] is omitted, the user's default directory is used.
5. An ALTmode terminates the command input and signals GLOB to output the cross-referenced listing. In other words, a listing is not output until GLOB encounters an ALTmode. The ALTmode appears at the end of the command string shown in Command Format #1 or at the end of each output command shown in Command Format #2.

Switches

Switches control the types of global listings to be output. Each switch can be preceded by a slash, or several switches can be enclosed in parentheses. Only the most recently specified switch (except for L, M, P, Q, and X, which are always in effect) is in effect at any given time. If no switches are specified, all global symbols are output. The following switches are available.

- |    |                                                                                                                                                                                                                                |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /A | Output all global symbols. This is the default if no switches are specified.                                                                                                                                                   |
| /E | List only multiple defined or undefined (erroneous) symbols.                                                                                                                                                                   |
| /F | List nonrelocatable (fixed) symbols only.                                                                                                                                                                                      |
| /H | List the switches available (help text).                                                                                                                                                                                       |
| /L | Scan programs only if they contain globals previously defined and not yet satisfied (library search mode).                                                                                                                     |
| /M | Turn off library search mode scanning resulting from a /L switch.                                                                                                                                                              |
| /N | List only symbols which are never referenced.                                                                                                                                                                                  |
| /P | List all routines that define a symbol to have the same value. The routine that defines the symbol first is listed followed by a plus (+) sign. Subsequent routines that define the symbol are listed preceded by a plus sign. |
| /Q | Suppress the listing of subsequent definers that result from the /P switch.                                                                                                                                                    |

(continued on next page)

**GLOB program (Cont)**

Command Format (cont)

Switches (cont)

- /R List only relocatable symbols.
- /S List symbols with non-conflicting values that are defined in more than one program.
- /X Do not print listing header when output device is not the terminal, and include listing header when it is the terminal. Without this switch, the header is printed on all devices except the terminal. The listing header is in the following format:

FLAGS SYMBOL OCTAL VALUE DEFINED IN REFERENCED IN

Symbols listed are in alphabetical order according to their ASCII code values. The octal value is followed by a prime (') if the symbol is relocatable. The value is then relative to the beginning of the program in which the symbol is defined. Flags preceding the symbol are shown below.

- M Multiply defined symbol (all values are shown).
- N Never referred to (i.e., was not declared external in any of the binary programs).
- S Multiply specified symbol (i.e., defined in more than one program but with non-conflicting values). The name of the first program in which the symbol was encountered is followed by a plus sign.
- U Undefined symbol.

Characteristics

The R GLOB command:

- Places the terminal in user mode.
- Runs the GLOB program, thereby destroying the user's core image.

Associated Messages

Refer to Chapter 4.



**GRIFE program**

Function

The GRIFE program accepts text from a user and records it in a disk file, thereby enabling users to record comments and complaints to be read at a later time by the operations staff.

Command Format

R GRIFE

When the GRIFE program responds with a YES?, type the text, using as many lines as necessary, terminated with an ESCAPE. The text is written as a file with <157>protection and includes a header with the date, time, and project-programmer number of the user writing the comment. Therefore, the user does not need to identify himself.

Characteristics

The R GRIFE command:

Places the terminal in user mode.  
Runs the GRIFE program, thereby destroying the user's core image.

Associated Messages

Refer to Chapter 4.

Example

.R GRIFE )

YES? (TYPE ESCAPE WHEN THROUGH) THIS CONSOLE IS  
ALMOST OUT OF PAPER\$  
THANK YOU

.

**HALT command**Function

The HALT (†C) command transmits a HALT command to the monitor command interpreter. It stops the job and stores the program counter in the job data area (.JBPC). Refer to the DECsystem-10 Monitor Calls for a description of the job data area.

Command Format

HALT (†C)

Characteristics

The HALT (†C) command:

Places the terminal in monitor mode.  
Does not require LOGIN.

Example

†C

⋮

**HELP command**

Function

The HELP command is used to obtain useful documentation on various system features.

Command Formats

1. HELP

outputs the instructions for receiving information.

2. HELP \*

outputs the names of features that have available documentation.

3. HELP name

outputs the information on the named feature.

Only the first six characters of the argument to the command are scanned. These characters must be A through Z, 0 through 9, or asterisk (\*).

Characteristics

The HELP command:

Leaves the terminal in monitor mode.  
Does not require LOGIN.

Associated Messages

Refer to Chapter 4.

**HELP command (Cont)**Examples

\_HELP \*)

HELP IS AVAILABLE FOR THE FOLLOWING:  
BATCON CDPSP L CDRSTK DIRECT FAILSA FGEN  
HELP LPTSP L MATCH MTCOPY PIP PLTSP L  
PTPSP L SOUP TECO

\_HELP DIRECT)

TYPE OUT=INPUT+INPUT+...  
/ACCESS:N ■ ACCESS ALL LISTED FILES UNDER N BLOCKS LONG  
/ALLOCATED ■ GIVE ALLOCATED LENGTH  
/CHECKSUM ■ COMPUTE CHECKSUM OF EACH FILE  
/DETAIL ■ EVERYTHING FROM EXTENDED LOOKUP  
/F ■ FAST MODE  
/H ■ THIS TEXT  
/L ■ OUT TO LPT  
/PHYSICAL ■ DO PHYSICAL OPENS  
/S ■ SLOW MODE  
/SORT ■ PREPARE FOR SORTING  
/SUMMARY ■ JUST PRINT SUMMARY LINE  
/TITLES ■ INCLUDE TITLES  
/UNITS ■ GIVE SPECIFIC UNIT  
/WIDTH:N ■ TRY TO FILL PAPER WIDTH OF N COLUMNS  
/WORDS ■ OUTPUT LENGTHS IN WORDS

\* IS WILD NAME, ETC.  
? IS WILD LETTER OF NAME, ETC.  
"OUT=" MAY BE OMITTED  
DEFAULT IS TTY: .DIR=DSK:\*. \* [MY DIRECTORY]

**INITIA command**

Function

The INITIA command performs standard system initialization for the terminal issuing the command. This command is issued automatically at system startup and at the 400 series restart at certain designated terminals, but may be re-issued at any time by the user. This command is used to initiate specific system programs, such as the operator service program, OPSER, on a particular console.

The INITIA command runs SYS:INITIA.SAV which, depending upon the system configuration and the number of the TTY from which it is typed, may cause any of a number of events to occur. For more information, refer to the INITIA specification in Notebook 7 of the DECsystem-10 Software Notebooks.

Command Format

INITIA

Characteristics

The INITIA command:

- Leaves the terminal in monitor mode.
- Runs a specific system program.
- Does not require LOGIN.
- Depends on FTCCLX which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

Examples

```
.INITIA )  
SS04 SYS #2 22:12:17 TTY24  
:
```

**JCONTINUE command**Function

The JCONTINUE command forces a continue of the specified job if the job was in a tC state because of a call to the device error message routine (HNGSTP).

Command Format

JCONTINUE n

n = the number of the job to be continued. This argument is required.

Characteristics

The JCONTINUE command:

Places the terminal in monitor mode.

Does not require LOGIN.

Depends on FTJCON which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

Example

JCONT 14)

**KJOB command**

Function

In multiprogramming systems, the KJOB command:

- Stops all assigned I/O devices and returns them to the monitor pool.
- Returns all allocated core to the monitor pool.
- Returns the job number to the pool.
- Leaves the console in the monitor mode.
- Performs an automatic TIME command.

In swapping systems, the KJOB command performs all the above procedures. In addition, the command responds with

CONFIRM:

The user may type !C to abort logout, or type an optional file structure name (or list of file structure names) preceded by one of the following:

- F ) to logout immediately saving all files (including temporary files) as they are. Identical to R LOGOUT, or RUN UJO to LOGOUT.
- D ) to delete all files on the specified file structures. Responds with ARE YOU SURE? Type Y or D for YES, any other character for NO.
- K ) to delete all unprotected files (i.e., files with 0xx protection code) on the specified file structures. If project 1 or other jobs are logged-in with the same project-programmer number, responds with ARE YOU SURE? Type Y or K for YES, any other character for NO.
- P ) to save and protect (i.e., assign a protection code of 1 in the owner's field) all but temporary files (TMP, CRF, LST) on the specified file structures. If project 1 or other jobs are logged-in with the same project-programmer number, responds with ARE YOU SURE? Type Y or P for YES, any other character for NO.
- S ) to save without protecting all but temporary files on the specified file structures. If project 1 or other jobs are logged-in with the same project-programmer number, responds with ARE YOU SURE? Type Y or S for YES, any other character for NO.
- L ) to list the directories of the specified file structures.
- I ) to individually determine what to do with all files on the specified file structure as follows:

(continued on next page)

**KJOB command (Cont)**Function (cont)

After each filename is listed, type

- P ) to protect the file.
- S ) to save the file.
- K ) to delete the file.
- Q ) to learn if over logged-out quota on this file structure. If not over quota, nothing is typed, and the same filename is repeated.
- E ) to skip to next file structure and save this file if below logged-out quota for this file structure. If not below logged-out quota, a message is typed and the same filename is repeated.
- H ) to list responses and meanings.
- U ) to individually determine what to do with all but protected files. Protected files are always preserved.
- B ) to delete no files except when user is over the logged-out quota, then delete enough files to be below quota. The files are deleted in the following order: 1) unprotected files according to the category of the file, 2) spooled files not previously queued, and 3) protected files according to the category of the file. The categories of files are as follows: 1) temporary files, 2) relocatable files, 3) backup files, 4) save files, and 5) all other files.
- Q ) to learn if over logged-out quota on the specified file structures.
- H ) to list the KJOB options and their meanings.
- W ) to list the names of the files that are deleted.
- X ) to turn off the listing of the names of the files that are deleted. Complement of W.

If no file structure names are specified, the responses are for all file structure names in the job search list. If file structure names are specified, the responses apply to those file structures, and CONFIRM is retyped. The KJOB command ignores all logical assignments.

The user has the option of going through the CONFIRM dialogue, even if other jobs are logged-in under the same project-programmer number or if he is logged-in under project 1. (However, if sufficient responses are included on the KJOB command line or in a temporary file entered through an alternate entry point, CONFIRM is not typed.) By responding to a CONFIRM message, the user has an opportunity to organize his disk area by deleting or preserving specific files.

The KJOB program calls the QUEUE program to perform the queuing of files which have been deferred to logout time. This includes all spooled output unless the user has specifically queued output spooling earlier. Queuing may be suppressed with the /Z response (see below).

**KJOB command (Cont)**

Command Formats

1. KJOB

CONFIRM:

When the CONFIRM: response is given, the user may type any of the above-described letters followed by an optional file structure name or list of file structure names separated by commas. The user may type one of the above-described letters, followed by optional file structure names, on the same line as the KJOB command, and the CONFIRM: message will not be typed.

2. KJOB <log file descriptor> = / <letter> <list of file structure names> / <letter> <list of file structure names> etc.

<log file descriptor> has the following form: <dev:file.ext [proj,prog]>. If the log file is not a disk or spooled device, TTY is used.

<letter> = any letter from the above-described set. In addition, the following responses are available to any jobs using this command format:

/Z:n specifies the degree of queuing desired:

- n = 0 suppresses all normal queuing done at LOGOUT time.
- n = 1 queues the log file only.
- n = 2 queues the log file and spooled output (\*.LPT, etc.)
- n = 3 queues the log file, spooled output, and \*.LST.
- n = 4 queues the log file, spooled output, \*.LST, and any requests deferred to LOGOUT time (deferred requests are not yet implemented).

If Z is given without a value or if there are no spool bits set for job, Z:0 is assumed. Otherwise, /Z:2 is assumed.

/VL:n specifies that the limit of pages for LPT files is to be n (decimal).

/VC:n specifies that the limit of cards for CDP files is to be n (decimal).

/VT:n specifies that the limit of feet of paper tape for PTP files is to be n (decimal).

/VP:n specifies that the limit of minutes for PLT files is to be n (decimal).

/VR:n specifies that the priority of the queue request is to be n; n is from 0 through 62. /VR:62 is the standard.

(continued on next page)

**KJOB command (Cont)**Command Formats (cont)

$\wedge$ S:n specifies that the sequence number for the queue request is to be n.

$\wedge$ D:v specifies that the file disposition of the log file is to be v.

v = D deletes the log file after printing.

v = P preserves the log file after printing.

v = R renames the log file before printing to the queue area and deletes it after printing.

Default is  $\wedge$ D:R.

If a value to the above switches is not specified, the value is equivalent to 0 (e.g.,  $\wedge$ D is equivalent to  $\wedge$ D:0). For the  $\wedge$ x switches, a value of 0 is equivalent to the standard (e.g.,  $\wedge$ D =  $\wedge$ D:0 =  $\wedge$ D:R).

The letters must appear on the input side of the command string. If the log file is specified, all TTY output is appended to the log file. If no log file is specified or if the log file is not a disk or spooled device, the default is TTY. In addition, if responses to CONFIRM are required and are not specified on the KJOB command line, these responses will then be read from TTY. Therefore, users should be careful when employing this command format.

3. The KJOB program may be entered at the CCL entry point through the RUN UUO. When this is done, TMPCOR file KJO or disk file nnnKJO.TMP, where nnn is the user's job number in decimal, is used instead of the TTY input. This temporary file has the following format:

KJOB <log file descriptor> = / <list of file structure names> / <letter>  
<list of file structure names> etc.

Characteristics

The KJOB command:

Detaches the terminal.

Stops all assigned I/O devices since it does not operate when a device is currently transmitting data.

Runs the KJOB and LOGOUT programs.

Does not require LOGIN.

Associated Messages

Refer to Chapter 4.

**KJOB command (Cont)**

Examples

1. An example of the CONFIRM dialogue.

```

.K )
CONFIRM: I )
DSKB:
TEST4 .TST <055> 2000. BLKS : K )
TEST5 .TST <055> 505. BLKS : P )
T11 .BAK <055> 5. BLKS : K )
T2 .BAK <055> 5. BLKS : K )
T3 .BAK <055> 5. BLKS : K )
TEST .BAK <055> 5. BLKS : K )
TEST .REL <055> 5. BLKS : S )
TEST .MAC <055> 5. BLKS : P )
TEST .SHR <055> 30. BLKS : S )
JOB 5, USER [10,63] LOGGED OFF TTY24 AT 2309 11-MAY-71
DELETED 5 FILES
SAVED 4 FILES 2565 TOTAL BLOCKS USED
RUNTIME 0 MIN, 00.60 SEC

```

2. An example of the user bypassing the CONFIRM dialogue.

```

.K/F )
JOB 9, USER [10,110] LOGGED OFF TTY3 1349 18-MAR-71
SAVED ALL 23 FILES (630. DISK BLOCKS)
RUNTIME 1 MIN, 51.52 SEC

```

3. An example of the command when used in the Batch system. The output appears in the log file.

```

12:20:51 MONTR K DSKB0:MUM.LOG[10,110]=/W/B/VL:200
12:21:02 LGOUT JOB 12, USER [10,110] LOGGED OFF TTY50 1221 18-MAR-71
12:21:02 LGOUT SAVED ALL 38 FILES (1275. DISK BLOCKS)
12:21:02 LGOUT ANOTHER JOB STILL LOGGED IN UNDER [10,110]
12:21:02 LGOUT RUNTIME 0 MIN, 00.65 SEC

```

4. An example of the User specifying two switches.

```

.K/W/B )
DELETED:
MYFILE
JOB 14, USER [20,275] LOGGED OFF TTY35 1454 13-APR-72
DELETED 1 FILES (1022. DISK BLOCKS)
SAVED 52 FILES (1735. DISK BLOCKS)
RUNTIME 0 MIN, 02.60 SEC

```

**LIST command <sup>1</sup>**Function

The LIST command directs PIP to list the contents of named source file(s) on the line printer (LPT). The output goes either to LPT immediately or to the disk to be spooled to LPT if it is being spooled for this job. Refer to the QUEUE and PRINT commands. If the LPT is being spooled, the QUEUE program should always be used since it saves time and disk accesses.

Command Format

LIST list

list = a single file specification or a string of file specifications separated by commas. A file specification consists of a device name, a filename and extension, and a directory name. This argument is required.

Switches can be passed to PIP by enclosing them in parentheses in the LIST command string. When COMPIL interprets the command string, it passes the switches on to PIP.

Characteristics

The LIST command:

Leaves the terminal in monitor mode.

Runs the PIP program, thereby destroying the user's core image.

Depends on FTCCCLX which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

Examples

```
.LIST TEST.* )  
-LIST *.MAC )  
-LIST DTA4:A,B,C )
```

---

<sup>1</sup>This command runs the COMPIL program, which interprets the command before running PIP.

**LOAD command <sup>1</sup>**

Function

The LOAD command translates the specified source files if necessary (function of COMPILE command), runs the LOADER, and loads the .REL files generated. The assembler or compiler used is determined by the source file extension or by switches in the command string (refer to the COMPILE command). If a REL file already exists with a more recent date than that of the source file, compilation is not performed (unless requested via a switch).

This command generates a core image but does not begin execution. At this point, the user can start his program or save the core image for future execution.

Each time the COMPILE, LOAD, EXECUTE, or DEBUG command is executed, the command with its arguments is remembered in a temporary file on disk, or in core if the monitor has the TPCOR feature. Therefore, the filename used last can be recalled for the next command without specifying the arguments again (refer to Paragraph 1.5).

The LOAD command accepts several command constructions: the @ construction (indirect commands), the + construction, the = construction, and the < > construction. Refer to Paragraph 1.5 for a complete description of each of these constructions.

Command Format

LOAD list

list = a single file specification, or a string of file specifications separated by commas. A file specification consists of a device name, a filename with or without an extension, and a directory name.

The following switches can be used to modify the command string. These switches can be temporary or permanent switches (refer to Paragraph 1.5.5).

- /ALGOL            Compile the file with ALGOL. Assumed for files with the extension of .ALG.
- /BLISS<sup>2</sup>         Compile the file with BLISS. Assumed for files with the extension of .BLI.
- /COBOL           Compile the file with COBOL. Assumed for files with the extension of .CBL.

(continued on next page)

<sup>1</sup>This command runs the COMPIL program, which interprets the command before running the appropriate processor and the LOADER.

<sup>2</sup>BLISS will be recognized as a processor only if the appropriate assembly switch is set. However, this assembly switch setting is not supported.

**LOAD command (Cont)**Command Format (cont)

- /COMPILE** Force a compilation of this file even though a binary file exists with a newer date and time than the source file. This switch is used to obtain an extra compilation (e.g., in order to obtain a listing of the compilation) since normally compilation is not performed if the binary file is newer than the source file.
- /CREF** Produce a cross-reference listing file on the disk for each file compiled for later processing by the CREF program. These files have the filename of the source file and the extension of .CRF. The files can then be listed with the CREF command. However, with COBOL files, the cross-referenced listing is always appended to the listing file. No additional command need be given to obtain the listing.
- /FORTRAN** Compile the file with FORTRAN. Assumed for files with the extension of .F4 and all files with nonrecognizable processor extensions (if FORTRAN is the standard processor).
- /FUDGE** Create a disk file containing the names of the .REL files produced by the command string. When the FUDGE command is given, PIP reads this file in order to generate a library REL file. Arguments to this switch are:
- /FUDGE:dev:file.ext [proj,prog]**
- dev: - the device on which to write the file. DSK: is assumed.
- file.ext - the name of the library file. The filename is required. If the extension is omitted, it is assumed to be .REL.
- [proj,prog] - the directory in which to place the file. The user's directory is assumed if none is given.
- This switch is permanent in the sense that it pertains to all REL files generated by the command string.
- /LIBRARY** Load the files in library search mode. This mode causes a program file in a special library file to be loaded only if one or more of its declared entry symbols satisfies an undefined global request in the source file. The default libraries are always searched. Refer to the LOADER documentation.
- /LIST** Generate a disk listing file, for each file compiled, with the filename of the source file and the extension of .LST. These files can be listed later with the LIST command. Unless this switch is specified, listing files are not generated except in COBOL; COBOL listings are always generated.

(continued on next page)

## LOAD command (Cont)

Command Format (cont)

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /LMAP                | Produce a loader map during the loading process (same action as /MAP) containing the local symbols.                                                                                                                                                                                                                                                                                                                                                                                     |
| /MACRO               | Assemble the file with MACRO. Assumed for files with extensions of .MAC.                                                                                                                                                                                                                                                                                                                                                                                                                |
| /MACX11 <sup>1</sup> | Assemble the file with MACX11. Assumed for files with an extension of .P11.                                                                                                                                                                                                                                                                                                                                                                                                             |
| /MAP                 | Produce a loader map during the loading process. When this switch is encountered, a loader map is requested from the loader. After the library search of the default libraries, the map is written with the filename specified by the user (e.g., /MAP:file) or with the default filename MAP.MAP in the user's disk area. This switch is an exception to the permanent compile switch rule in that it causes only one map to be produced although it may appear as a permanent switch. |
| /NOCOMPILE           | Complement the /COMPILE switch by not forcing a compilation on a source file whose date is not as recent as the date on the binary file. Note that this switch is not the same as the /REL switch, which turns off all compilation, even if the source file is newer than the REL file. /NOCOMPILE is the default action.                                                                                                                                                               |
| /NOLIST              | Do not generate listing files. This is the default action except for COBOL files; COBOL listings are always generated.                                                                                                                                                                                                                                                                                                                                                                  |
| /NOSEARCH            | Load all routines of the file whether the routines are referenced or not. Since this is the default action, this switch is used only to turn off library search mode (/LIBRARY). This is not equivalent to the /P LOADER switch because /P does not search any libraries where /NOSEARCH will scan the default libraries.                                                                                                                                                               |
| /REL                 | Use the existing .REL files although a newer source file may be present.                                                                                                                                                                                                                                                                                                                                                                                                                |
| /SNOBOL <sup>2</sup> | Compile the file with SNOBOL. Assumed for files with an extension of .SNO.                                                                                                                                                                                                                                                                                                                                                                                                              |

Characteristics

The LOAD command:

- Leaves the terminal in monitor mode.
- Runs the appropriate processor and the LOADER.

---

<sup>1</sup>MACX11 (the PDP-11 assembler for the PDP-10) will be recognized as a processor only if the appropriate assembly switch is set. However, this assembly switch setting is not supported.

<sup>2</sup>SNOBOL will be recognized as a processor only if the appropriate assembly switch is set. However, this assembly switch setting is not supported.

**LOAD command (Cont)**Associated Messages

Refer to Chapter 4.

Examples

```
LOAD TEST
MACRO: TEST
LOADING

LOADER 2K CORE

EXIT

.
```

**LOCATE command**

Function

The LOCATE command logically establishes the user's job at a specified station. When the job is initiated, the user's logical station corresponds to his physical station. Therefore, this command is needed only if the user desires to change his logical station.

Command Format

LOCATE nn

nn = the station number.

An argument of 0 denotes the central station. A null argument implies the station of the user's terminal, i.e., his physical station.

Characteristics

The LOCATE command:

Leaves the terminal in monitor mode.

Depends on FTREM which is normally absent in the DECsystem-1040.

Restrictions

The LOCATE command must specify a station that is currently in contact with the central station.

Associated Messages

Refer to Chapter 4.

Examples

```
._LOCATE 2 )  
.  
._LOC 0 )  
.  
.
```

**LOGIN command**Function

The LOGIN command is used to gain access to the system. This command loads a Monitor Support program which accepts the user's LOGIN data. The user types in his project and programmer numbers followed by his password. To login successfully, the project and programmer numbers and the password typed in by the user must match the project and programmer numbers and password stored in the system accounting file (SYS:ACCT.SYS).

Command Format

LOGIN proj,prog

proj,prog = the user's project-programmer number. The project and programmer numbers may be separated by either a comma or a slash. If a slash is used, the message of the day is not output to the user unless the date on the file containing the message (NOTICE.TXT) is later than the last time the user logged-in. If this is true, the message is typed only once, whereas, when the comma is used, the message is output every time the user logs in. This argument may be typed on the same line as the LOGIN command, or on the following line after LOGIN types out the number sign.

Characteristics

The LOGIN command:

- Returns the terminal to monitor mode or starts a program running if specified in ACCT.SYS entry for proj,prog.
- Runs the LOGIN program.
- Does not require LOGIN.

Associated Messages

Refer to Chapter 4.

**LOGIN command (Cont)**

Example

The following is the procedure used to gain access to the system.

.LOGIN 27,235  
JOB 21 5S0417A TTY23

LOGIN types the job number assigned to user (job number 21), followed by monitor name, version number, and console line number. If the user does not type his project-programmer number on the same line as the LOGIN command, LOGIN outputs a number sign indicating that the user should type in his project-programmer number.

PASSWORD:

System requests user to type his password. User types password followed by carriage return (refer to Paragraph 1.4.2.1). To maintain password security, the monitor does not echo the password. On terminals with local-copy (refer to DECsystem-10 Monitor Calls), a mask is typed to make the password unreadable.

1135 8-JUN-71 THUR  
TYPE SYS:SCHED FOR NEXT  
WEEKS SCHEDULE  
:

If user entries are correct, the system responds with time, date, day of the week, the message of the day (if any), and a period, indicating readiness to accept another command.

## MAKE command <sup>1</sup>

### Function

The MAKE command runs TECO (Text Editor and Corrector) and creates a new file on the disk. If a file already exists with the same name, a warning message is given and the file is superseded. Refer to the TECO manual in Notebook 6 of the DECsystem-10 Software Notebooks.

### Command Format

MAKE dev:file.ext [proj,prog]

dev: = the device or file structure name on which the file is to be created. If omitted, DSK: is assumed.

file.ext = any legal filename and filename extension. The filename is required; the filename extension is optional.

[proj,prog] = the directory in which the file is to be created. If omitted, the user's default directory is assumed. Note that the default directory may be an SFD or some other UFD.

### Characteristics

The MAKE command:

Places the terminal in user mode.

Runs the TECO program, thereby destroying the user's core image.

Depends on FTCCCLX which is normally absent in the DECsystem-1040.

### Associated Messages

Refer to Chapter 4.

### Example

```
._MAKE TEST3.MAC )
*
```

---

<sup>1</sup> This command runs the COMPIL program, which interprets the commands before running TECO.

**MOUNT command**

Function

The MOUNT command allows the user to request assignment of a device via the operator. This command is similar to the ASSIGN command, but, whereas the ASSIGN command operates without operator communication, the MOUNT command requests operator interaction when necessary. For example, if a Batch user requests a DECTape drive and all drives are in use, then the operator can free one for the user, if he wishes. The user can request devices from the restricted pool of devices.

The MOUNT command gives the operator greater control over assignment of devices on the system. When a user requests a device via this command, the operator has the option of either selecting a specific unit (e.g., DTA5) or cancelling the request completely (all units of this type are in use and the operator does not want to free one for this user). The operator may also mount the media for the requested unit if the media is sufficiently identified (e.g., a deck of cards in the card reader or an identified DECTape on a specific drive).

When the MOUNT command is used to gain access to a file structure, it allows the user to specify a particular drive, places the file structure name at the end of the job's search list, and waits for completion of operator action, if desired. Each file structure can have an administrative file, QUOTA.SYS, which contains a list of quotas for all users allowed access to the structure. When the file structure is mounted, a UFD is created for the user if he has an entry in QUOTA.SYS on the file structure.

The MOUNT command runs the UMOUNT program in the user's core area. UMOUNT scans the command string and completes as much of the command as possible without operator intervention. When operator intervention is required, UMOUNT queues a request to the OMOUNT program by writing a command file on the 3,3 disk area. OMOUNT examines these command files and interacts with the operator. When the command file is deleted, the operator action has been completed. UMOUNT waits for this completion of operator action unless the user types a control-C. When a control-C is typed, the user does not receive a message of confirmation, but can later use the /CHECK switch to see if his request is still pending (see Examples).

Command Format

MOUNT dev: log-dev /switches (drives)

dev: = one of the following: (1) a physical device name (e.g., DTA3, CDR, MTA), (2) a logical name previously associated with a physical device by either a MOUNT or ASSIGN command, or (3) a file structure name (one that is already mounted or one whose name appears in STRLST.SYS). This argument is required.

(continued on next page)

**MOUNT command (Cont)**Command Format (cont)

log-dev = any SIXBIT name that is not the same as dev:. In other words, it may not be a physical device name or logical name that is currently being used or has previously been used as dev:. This argument is optional.

switches = The following switches are optional and only enough characters to make the switch unique are required. The unique names are underlined below.

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>C</u> HECK    | Check and list pending requests.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <u>H</u> ELP     | Type this list.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <u>M</u> ULTI    | Multi-access, disk only, complement of /SINGLE, default condition.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <u>P</u> AUSE    | Notify the user before sending the message to the operator for a request. The user can then abort the command if desired.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <u>R</u> ONLY    | Read only, same as /WLOCK.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <u>S</u> INGLE   | Only this job can access files on the structure (single access), file protection is enforced for him, disk only.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <u>V</u> ID:name | A visual identification passed to the operator as a comment to assist him in identifying a particular unit to mount. The argument can be in one of two forms: 1) any string of up to 25 characters containing only letters, digits, periods, and hyphens, or 2) any string of up to 25 characters enclosed in single quotes. However, break characters and single quotes are not allowed in the string. The naming and use of this switch is relevant only to the extent that the installation operator knows what it means. The PLEASE command should be used for any complex procedures or long communications with the operator. |
| <u>W</u> ENABL   | Write enable for this job, complement of /WLOCK, default condition.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

(continued on next page)

**MOUNT command (Cont)**

Command Format (cont)

`/WLOCK` Write locked for this job. This job cannot write on this file structure and the monitor will not update BAT blocks or the access date. If `/SINGLE` is given, the operator may set hardware write lock to ensure that nothing is written.

(drives) = the physical drives on which the units are to be mounted. A drive argument may be used only when mounting file structures. The drives must be in the logical unit order within the file structure. Drive names are separated by commas. Leading and embedded drives that are not specified must be represented by null names (, ,DPA3). Unspecified trailing drives may be omitted. Drive names are as follows:

Blank, null - unspecified. UMount finds one of proper type.

Two letters - controller class (e.g., DP).

Three letters - specific controller (e.g., DPA). UMount finds a drive on that controller.

Three letters and one or two digits - specific drive (e.g., DPA0, DPA1).

The user, by specifying a drive list, may force the packs to be mounted on specific drives or controllers. If no drive (or incomplete) specification is given, an available drive of the proper type is found.

Characteristics

The MOUNT command:

Places the terminal in user mode.

Runs the UMount program, thereby destroying the user's core image.

Depends on FTCLX and FTMOUN which are normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

**MOUNT command (Cont)**

Examples

.MOUNT PRIV:)  
PRIV MOUNTED

Asks the operator to mount the file structure PRIV.

.MOUNT PAY:(DPA,,DPB)/S

Requests that the first unit of file structure PAY be mounted on Controller A, the second unit on any controller, the third unit on controller B, and any remaining units on any drives. The structure will be single access (i.e., available only to this job).

.MOUNT MINE:)  
OPERATOR NOTIFIED  
WAITING...  
↑C  
.  
.  
.

Mount the file structure MINE.  
The request is queued to the operator.  
UMOUNT is waiting for the request to be completed.  
The user does not wait for confirmation.

.MOUNT/CHECK)

The user wants to know if his request has been processed.

NONE PENDING  
.R SETSRC)  
\*T  
DSKA,DSKB,PRIV,PAY,MINE,FENCE  
\*↑C  
.MOUNT DTA INPUT

The request has been processed.  
The user wants to know if the file structure is in his search list.  
The file structure has been added to his search list.

OPERATOR NOTIFIED  
WAITING...  
INPUT (DTA5) MOUNTED  
.MOUNT INPUT/VID:325

The user wants the operator to select a DECtape drive and assign it with logical name INPUT.  
  
The request is queued to the operator.  
UMOUNT is waiting for the request to be completed.  
The operator has selected DTA5.  
The user asks the operator to mount the DECtape labeled 325. He may use either DTA5 or INPUT to refer to his device. For example, the Batch user would use INPUT since he would not know what DECtape drive he is assigned.

OPERATOR NOTIFIED  
WAITING...  
INPUT (DTA5) MOUNTED  
.MOUNT INPUT OUTPUT  
  
OUTPUT (DTA5) MOUNTED

The request is queued.  
UMOUNT is waiting for confirmation.  
The mount is successful.  
The user changes the logical name to OUTPUT.  
The logical name INPUT is no longer valid.  
The mount is successful.

**OPSER program**

Function

The OPSER program facilitates multiple job control from a single operator terminal by allowing the operator to run several jobs called subjobs from his terminal. The OPSER program acts as the supervisor of the various subjobs by allowing monitor level or user level commands to be passed to all of the subjobs or to selected subjobs. Output from the various subjobs may be retrieved by OPSER.

The subjobs of OPSER run on pseudo-TTYs (refer to DECsystem-10 Monitor Calls) and all initializations of the pseudo-TTYs are performed by OPSER. The operator needs only to provide the subjob name, either an OPSER-provided subjob number or an operator-assigned name. System programs that require a dedicated terminal can be run as subjobs of OPSER. By running system jobs on pseudo-TTYs, OPSER is able to maintain an I/O link between the running jobs and the operator. In addition, the output from the various subjobs is concentrated on one terminal instead of many, as was the case when each system program required its own terminal.

Refer to the MPB Operator's Manual in the DECsystem-10 Software Notebooks for complete information on OPSER.

Command Format

R OPSER

OPSER signifies its readiness to process commands by typing an asterisk if no subjobs are in use or subjobs are in a wait for an operator action. OPSER responds with an exclamation point when a subjob is running. Commands may be entered whenever OPSER is operating. Each command is preceded by a colon and must be typed to sufficient length to make it unique.

OPSER Commands

- :AUTO/hh:mm filespec      Process the specified file as an automatic startup file. The file is terminated by an end-of-file or the typing of a line on the console by the operator. This is the normal way that the standard subjobs are started by the operator. The time argument is optional; when it is given, the AUTO file is run at the specified time.
- :BATMAX n<sup>1</sup>      Specify the maximum number of batch jobs allowed.
- :BATMIN n<sup>1</sup>      Specify the minimum number of batch jobs guaranteed.

(continued on next page)

<sup>1</sup>Not yet implemented.

**OPSER program (Cont)**

Command Format (cont)

|                   |                                                                                                                                                                                                                       |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| :CLOSE            | Close the disk log file without opening a new one.                                                                                                                                                                    |
| :CURRENT          | Type the number of the current subjob (the last one typed into). Output from another subjob does not affect current subjob.                                                                                           |
| :DAYTIME          | Obtain the current date and time.                                                                                                                                                                                     |
| :DEFINE xxx=n     | Associate the symbol xxx as the mnemonic for subjob number n. The symbol B is reserved for the subjob running BATCON.                                                                                                 |
| :DEVICE nam:log:n | Assign the device with the physical name nam and logical name log to subjob n. The logical name is optional but a null field must be typed if the name is omitted, e.g., :DEVICE CDR::3.                              |
| :ERROR n          | Report only error messages (that is, ignore nonerror messages from subjob n). Message reporting is resumed with the :REVIVE command.                                                                                  |
| :EXIT             | Exit to the monitor if no subjobs are in use; otherwise give a list of those that are running. This should be used instead of ↑C, since EXIT does not return the job to monitor mode if there are any active subjobs. |
| :FREE             | Type the first free subjob number.                                                                                                                                                                                    |
| :HELP             | Type a text which briefly explains the command.                                                                                                                                                                       |
| :JCONT n          | Continue the specified stopped job.                                                                                                                                                                                   |
| :KJOB, n,m,p      | Kill the specified subjobs saving all files. Causes /Z:0 to be included to KJOB so spooled files are not queued.                                                                                                      |
| :KILL n,m,p       | Kill the specified subjobs. This is identical to :KJOB.                                                                                                                                                               |
| :KSYS hhmm        | Stop all timesharing at the time specified by hhmm.                                                                                                                                                                   |
| :LOGIN proj,prog  | Login a new subjob. If no project-programmer number is typed, assume OPSER's project-programmer number.                                                                                                               |
| :MSGLVL 0         | Cause the response to the :WHAT command to include the JOBSTS bits.                                                                                                                                                   |

(continued on next page)

**OPSER program (Cont)**

Command Format (cont)

|                              |                                                                                                                                                                                                                                                  |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| :MSGLVL 1                    | Cause the response to :WHAT command to eliminate the JOBSTS bits.                                                                                                                                                                                |
| :QUEUE <line>                | Initiate the first free subjob and send the typed-in line to the system queue manager.                                                                                                                                                           |
| :RESOURCES                   | Type the list of the available system resources.                                                                                                                                                                                                 |
| :RESTRICT dev <sup>1</sup>   | Make the specified device a restricted device (i.e., one that is assignable only by a privileged job or the MOUNT program).                                                                                                                      |
| :REVIVE n                    | Resume normal echoing of output from subjob n.                                                                                                                                                                                                   |
| :SEND <line>                 | Simulate the SEND monitor command.                                                                                                                                                                                                               |
| :SET a                       | Simulate a SET monitor command. Valid SET monitor commands are SET CORMAX, SET CORMIN, SET DATE, SET DAYTIME, SET LOGMAX, SET OPR TTY, SET SCHED, and SET TTY.                                                                                   |
| :SET RUN CPU <sub>n</sub>    | Add CPU <sub>n</sub> to the pool of CPUs to be used for running jobs.                                                                                                                                                                            |
| :SET RUN NO CPU <sub>n</sub> | Remove CPU <sub>n</sub> from the pool of CPUs to be used for running jobs.                                                                                                                                                                       |
| :SILENCE n                   | Ignore all output from subjob n.                                                                                                                                                                                                                 |
| :SLOGIN proj,prog            | LOGIN one subjob but suppress its response. If proj, prog is omitted, OPSER uses its own.                                                                                                                                                        |
| :STOP n                      | Put the specified subjob in monitor mode. This is equivalent to inputting two control-C's in interactive mode.                                                                                                                                   |
| :SYSTAT xx                   | Run SYSTAT with optional argument xx over the first free subjob.                                                                                                                                                                                 |
| :TLOG filespec               | Create a disk log file with the specified name. If the file already exists, a message is typed to determine whether the existing file should be superseded. If not, the file is appended to the existing one. Default for filespec is OPSER.LOG. |
| :TTYTST                      | Test this terminal by typing all the ASCII characters between octal 40 and 174, inclusive.                                                                                                                                                       |

(continued on next page)

<sup>1</sup>Not yet implemented.

**OPSER program (Cont)**Command Format (cont)

|  |                              |                                                                                                                                                                                                |
|--|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | :UNRESTRICT dev <sup>1</sup> | Make the specified device a unrestricted device (i.e., one that is assignable by both privileged and non-privileged jobs).                                                                     |
|  | :WHAT n,m,p                  | Type the status of the specified subjobs on the terminal. The status includes a SYSTAT with the time, the time of the last input and the last output, and a linear listing of the JOBSTS bits. |

When a subjob number or name is required in a command string, the subjob may be specified in one of four ways. It can be omitted, in which case the last subjob typed into is used. The mnemonic ALL may be used, in which case all active subjobs are implied. A decimal number can be used from zero to the limit OPSER is generated for. Finally, a mnemonic can be assigned to the subjob with the :DEFINE command.

Examples

```
.R OPSER)
*:AUTO CTY.ATO
```

To start an automatic startup file.

```
:MSGLVL 0
:TLOG
:SLOG
:DEFINE DAE=
DAE-R DAEMON
:SLOG
:DEFINE M=
M-R OMOUNT
M-START
:SLOG
:DEFINE L=
L-R LPTSPL
L-START
:SLOG
:DEFINE B=
B-MJOB 5
B-R BATCON
B-START
```

An example of an automatic startup file.

<sup>1</sup>Not yet implemented.

**PJOB command**

Function

The PJOB command causes the monitor to respond with the job to which the user's terminal is attached.

Command Format

PJOB

Characteristics

The PJOB command:

Leaves the terminal in monitor mode.

Associated Messages

Refer to Chapter 4.

Example

```
•PJOB)
1
:
```

## PLEASE command

### Function

The PLEASE command allows the user non-conflicting two-way communication with the designated station operator.

### Command Format

PLEASE dev: prog! text

dev = any terminal not assigned to a job (i.e., is not a job's controlling terminal) with which the user wishes to communicate, including:

- a. TTYn: directs the text to a specific terminal unit. The default is TTY0.
- b. OPRnn: directs the text to the operator's terminal station nn.
- c. (null argument) directs the text to TTY0 at the central station.

prog! = the name of the system program to be run automatically when the message is completed. This argument may appear before or after the device argument and must be concluded with an exclamation point. If PLEASE is entered at the CCL entry point, it reads file nnnPLS.TMP. This file is sent to the designated device. After the operator terminates the request, the specified program will be run at its CCL entry point. Neither the dev: or prog! argument can be used from a Batch control file.

text = the user's message. The argument is required. Characters are not transmitted until the RETURN, vertical tab, or form feed key is depressed, at which point the entire line is transmitted.

When the user depresses the RETURN, vertical tab, or form feed key, a message informing the operator of the caller's station number, proj-prog number or user's name if monitor job tables are available, and text message is printed on dev:. An ESCAPE or control-C on either the user's terminal or dev: causes communication to terminate and the user's TTY to be left in monitor mode. Note that when the line terminates with an ESCAPE, the line is typed but the operator response is not waited for. Messages may be typed in both directions without retyping the command.

### Characteristics

The PLEASE command:

- Places the terminal in user mode until ESCAPE is typed.
- Runs a system program except when used with Batch.
- Depends on FTCCCLX which is normally absent in the DECsystem-1040.

**PLEASE command (Cont)**

Restrictions

For Batch users, the PLEASE command is trapped by the Batch Controller and only PLEASE text is allowed. It can be used to request operator action while in the Batch mode. The line of text can only be one line terminated with an ESCAPE.

Associated Messages

Refer to Chapter 4.

Example

```
._PLEASE TELL ME WHEN DTA3 WILL BE FREE.)  
OPERATOR HAS BEEN NOTIFIED  
IN HALF AN HOUR  
THANKS  
↑C  
  
:
```

## PLOT command

### Function

The PLOT command is used to place entries in the plotter output queue. This command is equivalent to the following form of the QUEUE command:

QUEUE PLT:jobname = list of input specifications

### Command Format

PLOT jobname = list of input specifications

jobname = name of the job being entered into the queue. The default is the name of the first file in the request, not the name of the first file given. These differ when the first file given does not yet exist.

input specifications = a single file specification or a string of file specifications, separated by commas, for the disk files being processed. A file specification is in the form dev:file.ext[proj,prog].

dev: = any file structure to which PLTSPL will have access; the default is DSK:.

file.ext = names of the files. The filename is optional. The default for the first filename is \*, the default for subsequent files is the last filename used. The extension can be omitted; the default is .PLT.

[proj,prog] = a directory to which the user has access; the user's directory is assumed if none is specified.

If no arguments are given with the command (i.e., only the command name is given), the entries for all jobs of all users are output. The asterisk convention can be used for the input specifications. Switches that aid in constructing the queue entry can appear as part of the input specifications. These switches are divided into three categories:

1. Queue-operation - Only one of these switches can be placed in the command string because they define the type of queue request. The switch used can appear anywhere in the command string.
2. General - Each switch in this category can appear only once in the command string because they affect the entire request. The switch used can appear anywhere in the command string.

(continued on next page)

**PLOT command (Cont)**

Command Format (cont)

- 3. File control - Any number of these switches can appear in the command string because they are specific to individual files within the request. The switch used must be adjacent to the file to which it applies. If the switch precedes the filename, it becomes the default for subsequent files.

The following switches can be used with the PLOT command:

| <u>Switch</u>     | <u>Explanation</u>                                                                                                                                                                                                                                                                                   | <u>Category</u> |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| /AFTER:tt         | Process the request after the specified time; it is either in the form of hhmm (time of day) or +hhmm (time later than the current time). The resulting AFTER time must be less than the DEADLINE time. If the switch, or the value of the switch, is omitted, no AFTER constraints are assumed.     | General         |
| /BEFORE:t         | Queue only the files with creation dates before t where t is in the form dd-mmm-yy hhmm.                                                                                                                                                                                                             | General         |
| /BEGIN:n          | Start the output after n feet. The default is to start output at the beginning.                                                                                                                                                                                                                      | File Control    |
| /COPIES:n         | Repeat the output the specified number of times. n must be less than 64. If more than 63 copies are needed, two separate requests must be made. If the switch is omitted, single copies are output.                                                                                                  | File Control    |
| /CREATE           | Make a new entry into the plotter output queue. This switch is the default for the queue-operation switches.                                                                                                                                                                                         | Queue Operation |
| /DEADLINE:tt      | Process the request before the specified time; tt is either in the form hhmm (time of day) or +hhmm (time later than the current time). The resulting DEADLINE time must be greater than the AFTER time. If the switch, or the value of the switch, is omitted, no DEADLINE constraints are assumed. | General         |
| /DISPOSE:DELETE   | Delete the file after spooling.                                                                                                                                                                                                                                                                      | File Control    |
| /DISPOSE:PRESERVE | Save the file after spooling. This is the default for all files except files with extensions .LST, .TMP, and, if the protection is 0xx, .PLT.                                                                                                                                                        | File Control    |

(continued on next-page)

**PLOT command (Cont)**

Command Format (cont)

| <u>Switch</u>   | <u>Explanation</u>                                                                                                                                                                                                                   | <u>Category</u> |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| /DISPOSE:RENAME | Rename the file from the specified directory immediately, remove it from the logged-out quota and delete it after spooling. This is the default for files with extensions .LST, .TMP, and if the protection is 0xx, .PLT.            | File Control    |
| /F              | List the entries in the plotter queue, but do not update the queues. Therefore, the list may not be an up-to-date listing but the listing will be faster than with /LIST.                                                            | Queue Operation |
| /FORMS:a        | Place the output on the specified form. The argument to the switch must be six alphabetic characters. The default is that normal forms are used.                                                                                     | General         |
| /KILL           | Remove the specified entry from the plotter queue. This switch can be used for deleting a previously submitted request as long as the request has not been started by the spoolers.                                                  | Queue Operation |
| /LIMIT:n        | Limit the output to the specified number of pages.                                                                                                                                                                                   | General         |
| /LIST           | List the entries in the plotter queue; if the switch, along with all other switches, is omitted, all entries for all jobs of all users are listed.                                                                                   | General         |
| /MODIFY         | Alter the specified parameters in the job. This switch requires that the user have access rights to the job. It can be used for altering a previously submitted request as long as the request has not been started by the spoolers. | Queue Operation |
| /NEW            | Accept the request even if the file does not yet exist.                                                                                                                                                                              | File Control    |

(continued on next page)

**PLOT command (Cont)**

Command Format (cont)

| <u>Switch</u> | <u>Explanation</u>                                                                                                                                                                                                     | <u>Category</u> |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| /NOTE:a       | Plot the specified text (a) in the output.                                                                                                                                                                             | File Control    |
| /NULL         | Accept the request even if there is nothing in the request. No error message is given.                                                                                                                                 | General         |
| /OKNONE       | Do not output message if no files match the wildcard construction. This is assumed at KJOB time.                                                                                                                       | File Control    |
| /PHYSICAL     | Suppress logical device name assignments for the device specified.                                                                                                                                                     | File Control    |
| /PLOT:ASCII   | Plot the file in ASCII mode. If the /PLOT: switch is omitted, the file is plotted in the data mode specified in the file.                                                                                              | File Control    |
| /PLOT:BINARy  | Plot the file in binary mode. If the /PLOT: switch is omitted, the file is plotted in the data mode specified in the file.                                                                                             | File Control    |
| /PLOT:IMAGE   | Plot the file in image mode. If /PLOT: switch is omitted, the file is plotted in the data mode specified in the file.                                                                                                  | File Control    |
| /PRIORITY:n   | Assign the specified external priority (n=0 to 62) to the request. The larger the number, the greater priority the job has. The default is 10 if no switch is given and 20 if the switch is specified without a value. | General         |
| /PROTECT:nnn  | Assign the protection nnn (octal) to the job. If the switch or the value of the switch is omitted, the standard protection is assumed.                                                                                 | General         |
| /REMOVE       | Remove the file from the queue. This switch is valid only with /MODIFY and can be used to remove a previously submitted file as long as the spoolers have not started processing the request.                          | File Control    |
| /SEQ:n        | Specify a sequence number to help in identifying a request to be modified or deleted.                                                                                                                                  | General         |
| /SINCE:t      | Queue only the files with creation dates after the specified time t where t is in the form dd-mmm-yy hhmm.                                                                                                             | General         |

(continued on next page)

**PLOT command (Cont)**

Command Format (cont)

| <u>Switch</u> | <u>Explanation</u>                                                                                                                          | <u>Category</u> |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| /START:n      | Start on the nth line of the file. If the switch, or the value of the switch is omitted, the first line is assumed.                         | File Control    |
| /STRS         | Search for the file on all file structures in the search list and take each occurrence. The default is to take just the first occurrence.   | File Control    |
| /UNPRESERVED  | Output the files only if they are not preserved (i.e., the first digit of the protection code is 0). This switch avoids redundant plotting. | General         |

Characteristics

The PLOT command:

- Leaves the terminal in monitor mode.
- Runs the QUEUE program, thereby destroying the user's core image.
- Depends on FTQCOM which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

Examples

`.PLOT *.PLT/FORMS:PLAIN)`

Cause all files with the extension PLT in the user's area to be plotted. Because these are spooled files (i.e., have the extension .PLT), the files are renamed out of the user's area immediately, and deleted after plotting. The operator is asked to put PLAIN paper on the plotter.

**PRESERVE command <sup>1</sup>**

Function

The PRESERVE command renames the specified files with the standard protection inclusively; ORed with 100 (usually 155 or 157). The files are then preserved and KJOB will not delete them unless requested to. This command has the same action as the P argument to the KJOB command when individually determining what to do with each file.

Command Format

PRESERVE file1.ext, file2.ext, file3.ext, ...

The full wildcard construction can be used for either the filename or the extension.

Characteristics

The PRESERVE command:

- Leaves the terminal in monitor mode.
- Runs the PIP program, thereby destroying the user's core image.
- Depends on FTCCLX which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

Example

```
.PRESERVE TEST.MAC )  
.PRE PROG, COLE.F4, NAME.* )
```

---

<sup>1</sup>This command runs the COMPIL program, which interprets the command before running PIP.

**PRINT command**Function

The PRINT command is used to place entries into the line printer output queue. This command is equivalent to the following form of the QUEUE command:

QUEUE LPT:jobname = list of input specifications

Command Format

PRINT jobname = list of input specifications

jobname = name of the job being entered into the queue. The default is the name of the first file in the request, not the name of the first file given. These differ when the first file given does not yet exist.

input specifications = a single file specification or a string of file specifications, separated by commas, for the disk files being processed. A file specification is in the form dev:file.ext[proj,prog].

dev: = any file structure to which LPTSPL will have access; the default is DSK:.

file.ext = names of the files. The filename is optional. The default for the first filename is \*, the default for subsequent files in the last filename used. The extension can be omitted; the default is .LPT.

[proj,prog] = a directory to which the user has access; the user's directory is assumed if none is specified.

If no arguments are given with the command (i.e., only the command name is given), the entries for all jobs for all users are output.

The asterisk convention can be used for the input specifications. Switches that aid in constructing the queue entry can appear as part of the input specifications. These switches are divided into three categories:

1. Queue-operation - Only one of these switches can be placed in the command string because they define the type of queue request. The switch used can appear anywhere in the command string.
2. General - Each switch in this category can appear only once in the command string because they affect the entire request. The switch used can appear anywhere in the command string.

(continued on next page)

**PRINT command (Cont)**

Command Format (cont)

3. File control - Any number of these switches can appear in the command string because they are specific to individual files within the request. The switch used must be adjacent to the file to which it applies. If the switch precedes the filename, it becomes the default for subsequent files.

The following switches can be used with the PRINT command:

| <u>Switch</u>     | <u>Explanation</u>                                                                                                                                                                                                                                                                                   | <u>Category</u> |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| /AFTER:tt         | Process the request after the specified time; tt is either in the form of hhmm (time of day) or +hhmm (time later than the current time). The resulting AFTER time must be less than the DEADLINE time. If the switch, or the value of the switch, is omitted, no AFTER constraints are assumed.     | General         |
| /BEFORE:t         | Queue only the files with a creation date before time t, where t is in the form dd-mmm-yy hhmm. If this switch is omitted, no BEFORE constraints are assumed.                                                                                                                                        | General         |
| /BEGIN:n          | Start the output on the nth page. The default is to begin output on the first page.                                                                                                                                                                                                                  | File Control    |
| /COPIES:n         | Repeat the output the specified number of times. n must be less than 64. If more than 63 copies are needed, two separate requests must be made. If this switch is omitted, one copy is given.                                                                                                        | File Control    |
| /CREATE           | Make a new entry into the line printer output queue. This switch is the default for the queue-operation switches.                                                                                                                                                                                    | Queue Operation |
| /DEADLINE:tt      | Process the request before the specified time; tt is either in the form hhmm (time of day) or +hhmm (time later than the current time). The resulting DEADLINE time must be greater than the AFTER time. If the switch, or the value of the switch, is omitted, no DEADLINE constraints are assumed. | General         |
| /DISPOSE:DELETE   | Delete the file after spooling.                                                                                                                                                                                                                                                                      | File Control    |
| /DISPOSE:PRESERVE | Save the file after spooling. This is the default for all files except files with extensions of .LST, .TMP, and, if the protection is 0xx, .LPT.                                                                                                                                                     | File Control    |

(continued on next page)

**PRINT command (Cont)**

Command Format (cont)

| Switch          | Explanation                                                                                                                                                                                                                 | Category        |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| /DISPOSE:RENAME | Rename the file from the specified directory immediately, remove it from the logged-out quota, and delete it after spooling. This is the default for files with extensions .LST, .TMP, and, if the protection is 0xx, .LPT. | File Control    |
| /F              | List the entries in the line printer queue, but do not update the queues. Therefore, the list may not be an up-to-date listing but the listing will be faster than with /LIST.                                              | Queue Operation |
| /FILE:ASCII     | Indicate that the input file format is to be interpreted as ASCII text. This is assumed for all files with extensions other than .DAT.                                                                                      | File Control    |
| /FILE:COBOL     | Indicate that the input file format is to be interpreted as COBOL SIXBIT text.                                                                                                                                              | File Control    |
| /FILE:FORTTRAN  | Indicate that the input file format is to be interpreted FORTRAN ASCII text (obeys FORTRAN carriage control characters). This is assumed for files with the extension of .DAT.                                              | File Control    |
| /FORMS:a        | Place the output on the specified form. The argument to the switch must be six alphabetic characters. The default is that normal forms are used.                                                                            | General         |
| /HEADER:0 or 1  | Output block headers at the beginning of the file, if 1 (default). Do not output headers, if 0.                                                                                                                             | File Control    |
| /KILL           | Remove the specified entry from the Batch input queue. This switch can be used for deleting a previously submitted request as long as the request has not been started by the spooler.                                      | Queue Operation |
| /LIMIT:n        | Limit the output to the specified number of pages.                                                                                                                                                                          | General         |
| /LIST           | List the entries in the line printer queue; if the switch, along with all other switches, is omitted, all entries for all jobs of all users are listed.                                                                     | Queue Operation |
| /LOG            | Define the file that the spoolers will use to record their process. The default is jobname .LOG.                                                                                                                            | File Control    |

(continued on next page)

**PRINT command (Cont)**

Command Format (cont)

| <u>Switch</u>   | <u>Explanation</u>                                                                                                                                                                                                                  | <u>Category</u> |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| /MODIFY         | Alter the specified parameters in the job. This switch requires that the user have access rights to the job. It can be used for altering a previously submitted request as long as the request has not been started by the spooler. | Queue Operation |
| /NEW            | Accept the request even if the file does not yet exist.                                                                                                                                                                             | File Control    |
| /NOTE:a         | Print the specified text (a) in the output.                                                                                                                                                                                         | File Control    |
| /NULL           | Accept the request even if there is nothing in the request. No error message is given.                                                                                                                                              | General         |
| /OKBINARY       | Print files whose extensions imply binary information. Normally files with extensions .SAV, .SHR, .LOW, .REL, and .HGH will not appear in the print queue.                                                                          | File Control    |
| /OKNONE         | Do not output message if no files match the wildcard construction. This is assumed at KJOB time.                                                                                                                                    | File Control    |
| /PHYSICAL       | Suppress logical device name assignments for the device specified.                                                                                                                                                                  | File Control    |
| /PRINT:ARROW    | Convert all control characters to up-arrow format except 011-015 and 020-024. This is the default.                                                                                                                                  | File Control    |
| /PRINT:ASCII    | Send the file to the line printer with no changes.                                                                                                                                                                                  | File Control    |
| /PRINT:OCTAL    | Perform an octal dump of the file.                                                                                                                                                                                                  | File Control    |
| /PRINT:SUPPRESS | Suppress all carriage-control characters except for ASCII code characters LF and CR; this switch implies the use of the /PRINT:ARROW and is equivalent to the operator command to the spooler (SUPPRESS).                           | File Control    |
| /PRIORITY:n     | Assign the specified external priority (n=0 to 62) to the request. The larger the number, the greater priority the job has. The default is 10 if no switch is given and 20 if the switch is specified without a value.              | General         |
| /PROTECT:nnn    | Assign the protection nnn (octal) to the job. If the switch, or the value of the switch, is omitted, the standard protection is assumed.                                                                                            | General         |

(continued on next page)

**PRINT command (Cont)**

Command Format (cont)

| <u>Switch</u>   | <u>Explanation</u>                                                                                                                                                                          | <u>Category</u> |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| /REMOVE         | Remove the file from the queue. This switch is valid only with /MODIFY and can be used to remove a previously submitted file as long as the spooler has not started processing the request. | File Control    |
| /REPORT:code    | Print the specified report within a COBOL report file. Code can be up to 12 characters in length.                                                                                           | File Control    |
| /SEQ:n          | Specify a sequence number to help in identifying a request to be modified or deleted.                                                                                                       | General         |
| /SINCE:t        | Queue only the files with creation dates after the specified time t where t is in the form dd-mmm-yy hhmm.                                                                                  | General         |
| /SPACING:DOUBLE | Double-space the output lines.                                                                                                                                                              | File Control    |
| /SPACING:SINGLE | Single-space the output lines. This is the default if no /SPACING switch is used.                                                                                                           | File Control    |
| /SPACING:TRIPLE | Triple-space the output lines.                                                                                                                                                              | File Control    |
| /START:n        | Start on the nth line of the file. If the switch, or the value of the switch, is omitted, the first line is assumed.                                                                        | File Control    |
| /STRS           | Search for the file on all file structures in the search list and take each occurrence. The default is to take just the first occurrence.                                                   | File Control    |
| /UNPRESERVED    | Output the files only if they are not preserved (i.e., the first digit of the protection code is 0). This switch avoids redundant printing.                                                 | General         |

Characteristics

The PRINT command:

- Leaves the terminal in monitor mode.
- Runs the QUEUE program, thereby destroying the user's core image.
- Depends on FTQCOM which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

**PRINT command (Cont)**

Examples

- `._PRINT NOTICE.TXT)`                      Print the file DSK:NOTICE.TXT.
- `._PRINT SYSTAT.SCM/ISP:REN/COP:2)`                      Print two copies of the file DSK:SYSTAT.SCM from the user's default area. Rename the file out of the user's area immediately and delete it after spooling.
- `._PRINT *.TXT/HEAD:0/FORMS:2PART)`                      Print all files in the user's area which have the extension .TXT. Do not print file headers between the files. Print the files on forms known to the operator as 2PART.
- `._PRINT /SEQ:356/KILL)`                      Remove the request with sequence number 356 from the LPT queue. This is accepted only if the spooler has not started processing the request.
- `._PRINT LOADER.SAV/OKBINARY/PRINT:SUPPRESS)`                      Print a file known to be a binary file and suppress all carriage control characters except CR and LF.
- `._PRINT PRGMAC.REL/PRINT:OCTAL)`                      Print an octal dump of the file PRGMAC.REL.

**PROTECT command <sup>1</sup>**Function

The PROTECT command renames the specified files with the requested protection. The action of this command is similar to the R switch in PIP.

The protection of a file is indicated by three octal digits. Each digit represents a particular class of user. The first digit represents the owner of the file, the second represents users with the same project number of the owner, and the third represents all of the other users. Each number in the three digit code can be one of the following:

|   |                                                                               |
|---|-------------------------------------------------------------------------------|
| 7 | No access privileges                                                          |
| 6 | Execute the file only                                                         |
| 5 | Read and execute the file                                                     |
| 4 | Append, read, and execute the file                                            |
| 3 | Update, append, read, and execute the file                                    |
| 2 | Write, update, append, read, and execute the file                             |
| 1 | Rename, write, update, append, read, and execute the file                     |
| 0 | Change protection, rename, write, update, append, read, and execute the file. |

The standard protection is normally 057 which means the owner has all privileges (0), users in the owner's project can read and execute the file (5), and all other users cannot access the file (7). However, the system standard may be changed by the individual installations.

Command Format

PROTECT file1 <nnn>, file2 <nnn>, file3 <nnn>, ...

The protection can be specified before the filename in which case it is the default for subsequent files until changed. The full wildcard construction can be used for either the filename or the extension.

Characteristics

The PROTECT command:

- Leaves the terminal in monitor mode.
- Runs the PIP program, thereby destroying the user's core image.
- Depends on FTCLX which is normally absent in the DECsystem-1040.

---

<sup>1</sup>This command runs the COMPIL program, which interprets the command before running PIP.

**PROTECT command (Cont)**

Associated Messages

Refer to Chapter 4.

Examples

```
.PROTECT FORM.*<157> )  
.PRO MAIN.MAC<123>, <456>EQUIL.CBL,ADD.ALG )
```

## QUEUE command

### Function

The QUEUE command allows the user to make entries in several system queues - the input queue for the Batch system, and the output spooling queues for the line printer, the card punch, the paper-tape punch, and the plotter. The QUEUE command also provides the means of obtaining listings of the entries in the queues.

### Command Formats

1. QUEUE INP: jobname = control file specification, log file specification  
     To make an entry in the Batch input queue, INP:.
2. QUEUE output queue name: jobname = list of input specifications  
     To make an entry in an output spooling queue.
3. QUEUE listing file specifications/LIST = list of queue names  
     To obtain a listing of the entries in a queue.
4. The following six commands can be substituted for the various formats of the QUEUE command:
  - a. CPUNCH jobname = list of input specifications  
     equivalent to QUEUE CDP: jobname = list
  - b. PLOT jobname = list of input specifications  
     equivalent to QUEUE PLT: jobname = list
  - c. PRINT jobname = list of input specifications  
     equivalent to QUEUE LPT: jobname = list
  - d. PUNCH jobname = list of input specifications<sup>1</sup>  
     equivalent to QUEUE PTP: jobname = list
  - e. SUBMIT jobname = control file name, log file name  
     equivalent to QUEUE INP: jobname = control file, log file
  - f. TPUNCH jobname = list of input specifications  
     equivalent to QUEUE PTP: jobname = list

Queue names are taken from the following list:

|            |                                              |
|------------|----------------------------------------------|
| INP: (I:)  | Batch input queue                            |
| LPT: (L:)  | line printer output queue, default condition |
| CDP: (C:)  | card punch output queue                      |
| PTP: (PT:) | paper-tape punch output queue                |
| PLT: (PL:) | plotter output queue                         |

<sup>1</sup>The PUNCH command can be redefined by the installation to be equivalent to QUEUE CDP: jobname = list.

**QUEUE command (Cont)**

Command Formats (cont)

Control file specification is the file specification, plus switches and keyword parameters, for the control file being submitted to the Batch input queue. This file can be on any file structure that the user has access to; the default is DSK:. The filename is required, but the extension can be omitted; the default is .CTL. The asterisk construction is legal for the filename or extension.

Log file specification is the file specification for the file that is to be used to record actions taken during the execution of the control file. This file can be on any file structure in which the user can write. The default is the same file structure in which the control file resides. If the filename is missing, the log file is given the same name as the control file. If the extension is omitted, it is .LOG.

Jobname is the name of the job being entered into the queue. The default jobname is the name of the first file in the request not the first file given. These names are different when the first file given does not yet exist.

Input specifications are the file specifications for the disk files to be processed, and the various switches and keyword parameters that aid in constructing the queue entry. The files can be on any file structure that the queue processor has access to; the default is DSK:. The files can be in any directory, provided that the user has read-access to them; the default is the user's directory. The filename is optional; the default is \* for the first filename. The default for subsequent filenames is the last filename used. Note that the asterisk construction is legal only in the input specifications. The extension can be omitted because each queue has a default extension for the files to be processed. These default extensions are:

- .CTL - Batch input queue
- .LPT - line printer queue
- .CDP - card punch queue
- .PTP - paper-tape punch queue
- .PLT - plotter queue

The listing file specification is the description of the listing file. The default for the listing file destination is TTY unless a name is specified. If no queue names are specified, all queues for all the jobs of all users are listed.

Switches - Three categories of switches are provided. The first category contains the switches that define the operation; the second contains the switches that can appear only once because they affect the entire request; the third contains the switches specific to each file. In general, switches that precede the filename become the default for all succeeding files. This is true also for a device name, an extension, or a directory name that precedes a filename.

Queue-Operation Switches - Only one of this type of switch can be placed in a command string, because these switches define the type of queue request. This switch may appear anywhere in the command string.

**QUEUE command (Cont)**

Command Formats (cont)

General Queue Switches - Each of these switches can appear only once in a command string. They affect the entire request, generally in terms of scheduling. These switches can appear anywhere in the command string.

File-Control Switches - These switches affect the individual files in a request and must be adjacent to the filename in the command string. In order to change the defaults for the rest of the files, however, these switches must appear before a filename.

In the table of switches below, the following conventions have been used:

- ALL - Switches that can appear for both the Batch input queue and the output queues.
- INPUT - Switches that can appear only for the Batch input queue.
- LIST - Switches that can appear only for the listing file specification.
- OUTPUT - Switches that can appear only for the output queues.

| <u>Switch</u> | <u>Meaning</u>                                                                                                                                                                                                                                                                               | <u>Category</u> | <u>Queues</u> |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|---------------|
| /AFTER:t      | Process the request after the specified time. t is either in the form hhmm (time of day) or +hhmm (time later than the current time). The resulting AFTER time must be less than the DEADLINE time. If the switch, or the value of the switch, is omitted, no AFTER constraints are assumed. | General         | ALL           |
| /BEFORE:t     | Queue only the files with creation dates before time t where t = dd-mmm-yy hhmm.                                                                                                                                                                                                             | General         | OUTPUT        |
| /BEGIN        | Start the output on the nth page, card, or foot. The default is to begin output on the first unit.                                                                                                                                                                                           | File Control    | OUTPUT        |
| /CARDS:n      | Use n (decimal) as the maximum number of cards that can be punched by the job. If the switch is omitted, no cards are punched. If the switch is given with no value, 2000 cards is assumed as the maximum.                                                                                   | General         | INPUT         |

(continued on next page)

**QUEUE command (Cont)**

Command Formats (cont)

| <u>Switch</u>          | <u>Meaning</u>                                                                                                                                                                                                                                                                                      | <u>Category</u> | <u>Queues</u> |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|---------------|
| /CHARGE:a <sup>1</sup> | Charge the run to the specified account.                                                                                                                                                                                                                                                            | General         | ALL           |
| /COPIES:n              | Repeat the output the specified number of times (n must be less than 64). The default is one copy. If more than 63 copies are desired, two requests must be made.                                                                                                                                   | File Control    | OUTPUT        |
| /CORE:n                | Use n (in decimal K) as the maximum amount of core memory that the job can use. If the switch is omitted, the maximum of 25K is assumed; if the value of the switch is omitted, a maximum of 40K is assumed.                                                                                        | General         | INPUT         |
| /CREATE                | Make a new entry in the specified queue. This switch is the default for the queue-operation switches.                                                                                                                                                                                               | Queue Operation | ALL           |
| /DEADLINE:t            | Process the request before the specified time. t is either in the form hhmm (time of day) or +hhmm (time later than the current time). The resulting DEADLINE time must be greater than the /AFTER time. If the switch, or the value of the switch is omitted, no DEADLINE constraints are assumed. | General         | ALL           |
| /DEFER <sup>1</sup>    | Make a new entry in the specified queue, but the request is deferred until LOGOUT.                                                                                                                                                                                                                  | Queue Operation | ALL           |

(continued on next page)

<sup>1</sup>Not yet implemented.

**QUEUE command (Cont)**

Command Formats (cont)

| <u>Switch</u>     | <u>Meaning</u>                                                                                                                                                                                                                                                                                                                                                                        | <u>Category</u> | <u>Queues</u> |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|---------------|
| /DEPEND:n         | Specifies the initial value of the dependency count in decimal. When used with /MODIFY, this switch changes the dependency count of another job. If n is a signed number (+ or -), that number is added to or subtracted from the dependent job's count. If n is not a signed number, the dependent job's count is changed to n. If this switch is omitted, no dependency is assumed. | General         | INPUT         |
| /DISPOSE:DELETE   | Delete the file after spooling.                                                                                                                                                                                                                                                                                                                                                       | File Control    | ALL           |
| /DISPOSE:PRESERVE | Save the file after spooling. This is the default for files with extensions of .LST, .TMP, and if protection is 0xx, .CDP, .LPT, .PLT, .PTP.                                                                                                                                                                                                                                          | File Control    | ALL           |
| /DISPOSE:RENAME   | Rename the file from the specified directory immediately, remove it from the logged-out quota, and delete it after spooling. This is the default for files with extensions of .LST, .TMP, and if protection is 0xx, .CDP, .LPT, .PLT, .PTP.                                                                                                                                           | File Control    | ALL           |
| /F                | List the entries in the queue, but do not update the queues. Therefore, the list may not be an up-to-date listing of the queues but the listing will be faster than with /LIST.                                                                                                                                                                                                       | Queue Operation | LIST          |
| /FEET:n           | Use n (in decimal) as the maximum number of feet of paper tape that the job can punch. If the switch is omitted, no paper tape is punched. If the value is omitted, the default is 10*B+20 feet, where B is the number of blocks in the request.                                                                                                                                      | General         | INPUT         |

(continued on next page)

QUEUE command (Cont)

Command Formats (cont)

| <u>Switch</u>  | <u>Meaning</u>                                                                                                                                                                                                                                         | <u>Category</u> | <u>Queues</u> |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|---------------|
| /FILE:ASCII    | Specify that the input file format is to be interpreted as ASCII text. This is assumed for all files with extensions other than .DAT.                                                                                                                  | File Control    | OUTPUT        |
| /FILE:COBOL    | Specify that the input file format is to be interpreted as COBOL SIXBIT text.                                                                                                                                                                          | File Control    | OUTPUT        |
| /FILE:ELEVEN   | Specify that the input file format is to be interpreted as binary format.                                                                                                                                                                              | File Control    | OUTPUT        |
| /FILE:FORTRAN  | Specify that the input file format is to be interpreted as FORTRAN ASCII text (obeys FORTRAN carriage control characters). This is assumed for files with an extension of .DAT.                                                                        | File Control    | OUTPUT        |
| /FORMS:a       | Place the output on the named forms. The argument to the switch must be six alphabetic characters. Normal forms (14 x 11) are used if this switch is omitted. Narrow forms are 8-1/2 x 11.                                                             | General         | OUTPUT        |
| /HEADER:0 or 1 | Output block headers at beginning of the file if 1 (default); do not output headers if 0.                                                                                                                                                              | File Control    | OUTPUT        |
| /HELP          | Print a message giving the general format of the command string and explains the dialogue that is entered if the user needs additional help.                                                                                                           | -               | -             |
| /KILL          | Remove the specified entry from the specified queue. This switch requires an output specification; it does not default to LPT:*. The /KILL switch can be used for deleting a previously submitted request as long as the request has not been started. | Queue Operation | ALL           |

(continued on next page)

**QUEUE command (Cont)**

Command Formats (cont)

| <u>Switch</u> | <u>Meaning</u>                                                                                                                                                                                                                                                                                       | <u>Category</u> | <u>Queues</u> |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|---------------|
| /LIMIT:n      | Limit the output to the specified number of pages, cards, feet, or minutes.                                                                                                                                                                                                                          | General         | OUTPUT        |
| /LIST         | List the specified entries in the queue; the default entries are those for queues for all the jobs of all users.                                                                                                                                                                                     | Queue Operation | LIST          |
| /LOG          | Define the file that the spoolers will use to record their output. The default is jobname.LOG.                                                                                                                                                                                                       | File Control    | OUTPUT (LPT)  |
| /MODIFY       | Alter the specified parameters in the specified jobs; this switch requires that the user have access rights to the job. It also requires a queue name; it does not default to the LPT. This switch can be used to modify a previously submitted request as long as the request has not been started. | Queue Operation | ALL           |
| /NEW          | Accept request even if file does not yet exist. This is the default for the log file of Batch input queue.                                                                                                                                                                                           | File Control    | ALL           |
| /NOTE:a       | Output the specified text (a) in the output.                                                                                                                                                                                                                                                         | File Control    | OUTPUT        |
| /NULL         | Accept request even if there is nothing in the request. No error message is given.                                                                                                                                                                                                                   | General         | OUTPUT        |
| /OKBINARY     | Print files whose extensions include binary information. Normally files with extensions .SAV, .SHR, .LOW, .REL, and .HGH will not be in print queues.                                                                                                                                                | File Control    | OUTPUT (LPT)  |
| /OKNONE       | Do not produce message if no files match the wildcard construction.                                                                                                                                                                                                                                  | File Control    | OUTPUT        |

(continued on next page)

**QUEUE command (Cont)**

Command Formats (cont)

| <u>Switch</u> | <u>Meaning</u>                                                                                                                                                                                                                                                                                                                                                                                                 | <u>Category</u> | <u>Queues</u> |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|---------------|
| /OUTPUT:n     | Cause job to terminate with a /Z:n to KJOB (n is from 0 to 4).<br><br>N=0 Suppress all normal queuing performed at LOGOUT time.<br><br>N=1 Queue only the log file.<br><br>N=2 Queue only the log file and spooled output (e.g., *.LPT).<br><br>N=3 Queue the log file, spooled output, and *.LST files.<br><br>N=4 Queue the log file, spooled output, *.LST files, and any requests deferred to LOGOUT time. | General         | INPUT         |
| /PAGE:n       | Use n (decimal) as the maximum number of pages of output that the job can print. If the switch is omitted, the maximum is 200 pages; if only the value is omitted, a maximum of 2000 pages can be printed.                                                                                                                                                                                                     | General         | INPUT         |
| /PAPER:x      | Identical to /PUNCH:x, /PRINT:x, /TAPE:x, or /PLOT:x.                                                                                                                                                                                                                                                                                                                                                          | File Control    | OUTPUT        |
| /PHYSICAL     | Suppress logical device names for the specified device.                                                                                                                                                                                                                                                                                                                                                        | File Control    | ALL           |
| /PLOT:ASCII   | Plot the file in ASCII mode. If the /PLOT switch is omitted, the file is plotted in the data mode specified in the file.                                                                                                                                                                                                                                                                                       | File Control    | OUTPUT (PLT)  |
| /PLOT:BINARY  | Plot the file in binary mode. If the /PLOT switch is omitted, the file is plotted in the data mode specified in the file.                                                                                                                                                                                                                                                                                      | File Control    | OUTPUT (PLT)  |

(continued on next page)

**QUEUE command (Cont)**

Command Formats (cont)

| <u>Switch</u>   | <u>Meaning</u>                                                                                                                                                                                  | <u>Category</u> | <u>Queues</u> |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|---------------|
| /PLOT:IMAGE     | Plot the file in image mode. If the /PLOT switch is omitted, the file is plotted in the data mode specified in the file.                                                                        | File Control    | OUTPUT (PLT)  |
| /PRINT:ARROW    | Convert all control characters to up-arrow format except 011-015 and 020-024. This is the default.                                                                                              | File Control    | OUTPUT (LPT)  |
| /PRINT:ASCII    | Send the file to the line printer with no changes.                                                                                                                                              | File Control    | OUTPUT (LPT)  |
| /PRINT:OCTAL    | Print the file in octal.                                                                                                                                                                        | File Control    | OUTPUT (LPT)  |
| /PRINT:SUPPRESS | Suppress all character-control characters except for ASCII code characters LF and CR; this switch implies the use of the /PRINT:ARROW. Equivalent to operator command to spooler (SUPPRESS).    | File Control    | OUTPUT (LPT)  |
| /PRIORITY:n     | Give the specified external priority (n = 0 to 62) to the request. A larger number is greater priority. The default is 10 if no switch is given, and 20 if a switch is given without the value. | General         | ALL           |
| /PROTECT:nnn    | Specify a protection nnn (in octal) for this job or queue entry. If the switch, or the value of the switch, is omitted, the standard protection is assumed.                                     | General         | ALL           |
| /PUNCH:026      | Punch files in 026 Hollerith code. If the /PUNCH switch is not given, the files are punched according to the data mode of the file.                                                             | File Control    | OUTPUT (CDP)  |
| /PUNCH:ASCII    | Punch files in ASCII card code. If the /PUNCH switch is not given, the files are punched according to the data mode of the file.                                                                | File Control    | OUTPUT (CDP)  |

(continued on next page)

**QUEUE command (Cont)**

Command Formats (cont)

| <u>Switch</u>   | <u>Meaning</u>                                                                                                                                                                                                                                                                                        | <u>Category</u> | <u>Queues</u> |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|---------------|
| /PUNCH:BINARY   | Punch files in binary card format. If the /PUNCH switch is not given, the files are punched according to the data mode of the file.                                                                                                                                                                   | File Control    | OUTPUT (CDP)  |
| /PUNCH:D029     | Punch files in the old DEC 029 card code. If the /PUNCH switch is not given, the files are punched according to the data mode of the file.                                                                                                                                                            | File Control    | OUTPUT (CDP)  |
| /PUNCH:IMAGE    | Punch files in image mode. If the /PUNCH switch is not given, the files are punched according to the data mode of the file.                                                                                                                                                                           | File Control    | OUTPUT (CDP)  |
| /REMOVE         | Remove the file from the queue. This switch is valid only with the /MODIFY switch and can be used to remove a previously submitted file as long as the Batch System has not started processing the job.                                                                                               | File Control    | OUTPUT        |
| /REPORT:code    | Print the specified report within a COBOL report file. Code can be up to 12 characters in length.                                                                                                                                                                                                     | File Control    | OUTPUT (LPT)  |
| /RESTART:0 or 1 | A value of 0 (default) means the job cannot be requeued or restarted by the operator after a system crash. A message is sent to the job's log file. A value of 1 means the job will be requeued or restarted. The job should not be restartable if there are changes to the permanent file directory. | General         | INPUT         |
| /SEQ:n          | Specify a sequence number to aid in identifying a request to be modified or deleted.                                                                                                                                                                                                                  | General         | ALL           |
| /SINCE:t        | Queue only the files with creation dates after the specified time t where t is in the form dd-mmm-yy hhmm.                                                                                                                                                                                            | General         | OUTPUT        |

(continued on next page)

**QUEUE command (Cont)**

Command Formats (cont)

| <u>Switch</u>   | <u>Meaning</u>                                                                                                                                                            | <u>Category</u> | <u>Queues</u> |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|---------------|
| /SPACING:DOUBLE | Double-space the output lines.                                                                                                                                            | File Control    | OUTPUT (LPT)  |
| /SPACING:SINGLE | Single-space the printed lines (default).                                                                                                                                 | File Control    | OUTPUT (LPT)  |
| /SPACING:TRIPLE | Triple-space the printed lines.                                                                                                                                           | File Control    | OUTPUT (LPT)  |
| /START:n        | Start on n line of the file. If the switch, or the value of the switch, is omitted, the Batch System starts with the first line.                                          | File Control    | ALL           |
| /STRS           | Search for the file on all structures in the search list and takes each occurrence. The default is to take just the first occurrence of the file.                         | File Control    | OUTPUT        |
| /TAPE:ASCII     | Punch the tape in ASCII code. If the /TAPE switch is not given, the files are punched according to the data mode of the file.                                             | File Control    | OUTPUT (PTP)  |
| /TAPE:BINARY    | Punch the tape in binary mode. If the /TAPE switch is not given, the files are punched according to the data mode of the file.                                            | File Control    | OUTPUT (PTP)  |
| /TAPE:IBINARY   | Punch the tape in image-binary mode. If the /TAPE switch is not given, the files are punched according to the data mode of the file.                                      | File Control    | OUTPUT (PTP)  |
| /TAPE:IMAGE     | Punch the tape in image mode. If the /TAPE switch is not specified, the files are punched according to the data mode of the file.                                         | File Control    | OUTPUT (PTP)  |
| /TIME:hhmmss    | Specify the central processor time limit for the job. If no switch is specified, the limit is 5 minutes; if the switch is specified without a value, the limit is 1 hour. | General         | INPUT         |

(continued on next page)

**QUEUE command (Cont)**

Command Formats (cont)

| <u>Switch</u>        | <u>Meaning</u>                                                                                                                                                                                                                     | <u>Category</u> | <u>Queues</u> |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|---------------|
| /TPLOT:n             | Use n (decimal minutes) as the maximum amount of plotting time allowed for the job. If the switch is omitted, no plotter time is allowed; if the value is omitted but the switch is given, the maximum plotter time is 10 minutes. | General         | INPUT         |
| /UNIQUE: 0 or 1      | Run any number of Batch jobs under this project-programmer number at the same time, if 0. Runs only one Batch job at any one time, if 1 (default).                                                                                 | General         | INPUT         |
| /UNPRESERVED         | Output file only if not preserved.                                                                                                                                                                                                 | General         | OUTPUT        |
| /ZDEFER <sup>1</sup> | Create a new entry in a queue and defer it until LOGOUT; however, the deferred file is zeroed first so that all previous /DEFER requests from the current job are deleted.                                                         | Queue Operation | ALL           |

Characteristics

The QUEUE command (and its associated variations):

- Leaves the terminal in monitor mode.
- Runs the QUEUE program, thereby destroying the user's core image.
- Does not require LOGIN when only queue listings are desired.
- Depends on FTQCOM which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

<sup>1</sup>Not yet implemented.

**QUEUE command (Cont)**

Examples

- `._QUEUE FILEA,FILEB )` Enter files FILEA.LPT and FILEB.LPT in the line-printer queue under the jobname of FILEA.
- `._QUEUE INP:=TEST )` Enter file TEST.CTL in the Batch input queue under jobname TEST and log file with name TEST.LOG.
- `._QUEUE IN:PAYR=MAN )` Enter file MAN.CTL in the Batch input queue under jobname PAYR and log file with name MAN.LOG.
- `._QUEUE DSK:A.X=/LIST )` Place a queue listing of all jobs into file A.X in the user's disk area.
- `._QUEUE INP:FREED=FILEA/CREATE /PRIORITY:4/TIME:1:5 )` Place file FILEA.CTL in the Batch input queue with the jobname FREED. An external priority of 4 and CPU time limit of one minute and five seconds are set for the job. The log file is named FILEA.LOG.
- `._QUEUE INP:TEST=/KILL )` Remove the entry corresponding to TEST.CTL from the Batch input queue.
- `._QUEUE INP:JOBNAM=/MODIFY/TIME:200 )` Alter the time parameter of the entry corresponding to JOBNAM.CTL in the Batch input queue.
- `._QUEUE INP:=JOB.CTL/PAGES:500/TPLOT:20 )` Establish a limit of 500 pages and 20 minutes of plotting on the output generated by this job.
- `._QUEUE PLT:=JOB.PLT/LIMIT:20 )` Queue a file to PLTSPL with a limit of 20 minutes of plotting time.

**QUOLST program**

Function

The QUOLST program informs the user of both the amount of disk space he has used and the amount he has left on each file structure in his search list. This program also returns the amount of free space that the system has left for all users of the structure. Free system space on structures not in the user's search list is not output. This information can be obtained by typing SYSTAT /F.

The output given for each file structure consists of 1) the structure name, 2) the number of blocks used, and 3) the numbers of blocks left in the logged-in quota, in the logged-out quota, and on the structure.

Command Format

R QUOLST

Characteristics

The R QUOLST command:

Leaves the terminal in monitor mode.

Runs the QUOLST program, thereby destroying the user's core image.

Examples

.R QUOLST )

| USER: | USED | LEFT:(IN) | (OUT) | (SYS) |
|-------|------|-----------|-------|-------|
| STR   |      |           |       |       |
| DSKA: | 10   | 1000      | 100   | 4703  |
| DSKB: | 491  | 9509      | 4509  | 4240  |
| DSKC: | 0    | 10000     | 5000  | 396   |

.R QUOLST )

| USER: | USED | LEFT:(IN) | (OUT) | (SYS) |
|-------|------|-----------|-------|-------|
| STR   |      |           |       |       |
| DSKA: | 1022 | -22       | -922  | 4215  |
| DSKB: | 1735 | 78265     | 8265  | 36    |
| DSKC: | 0    | 2000      | 1000  | 6378  |

The user is over quota on DSKA: and must delete files before he can logout.

## R command

### Function

The R command loads a core image from the system device and starts it at the location specified within the file (.JBSA). It is equivalent to RUN SYS: file.ext core and is the usual way to run a system program that does not have a direct monitor command to run it.

This command clears all of user core. However, programs should not count on this action and should explicitly clear those areas of core that are expected to contain zeroes (i.e., programs should be self-initializing). This action allows programs to be restarted by a tC, START sequence without having to do another R command.

On magnetic tape, if the low or high segment is missing, a null record is output before the EOF for the missing segment so that two EOFs cannot occur consecutively. Therefore, a saved null segment does not appear as a logical EOT (2 EOFs in a row).

### Command Format

R file.ext core

Arguments are the same as in the RUN command except that SYS: is used as the default device. (In nondisk monitors, the default is the generic name that matches the system device.) Refer to the RUN command for a discussion of the core argument.

The extension applies to the low file, not the high file. An extension of .SHR, then .HGH, is assumed for the high file. If the user types an extension of .SHR or .HGH, the extension is treated as a null extension since .SHR and .HGH are confusing as low file extensions.

### Characteristics

The R command:

- Places the terminal in user mode.
- Runs a system program.

### Associated Messages

Refer to Chapter 4.

### Examples

```

.R PIP )
*
.R PIP 5 )
*

```

## REASSIGN command

### Function

The REASSIGN command allows one job to pass a device to a second job without having the device go through the monitor device pool (restricted or unrestricted). Both restricted and unrestricted devices can be reassigned. This command, applied to DECtapes, clears the copy of the directory currently in core, forcing the next directory reference to read a new copy from the tape, but does not clear the logical name assignment. If a device is INITed, a RELEASE UUO is performed unless the user issuing the command is reassigning the device to himself.

### Command Format

REASSIGN dev job

dev = the physical or logical name of the device to be reassigned. This argument is required.

job = the number of the job to which the device is to be reassigned. If no job is specified, the device is reassigned to the job issuing the command. This is useful when the user wants to force the next directory reference to come from the tape instead of core.

A logical name which is also a physical name can be reassigned only if the job issuing the command and the job to which the device is to be reassigned have the same project-programmer number, or the user issuing the command has operator privileges (logged-in under [1,2] or logged-in at OPR). However, a logical name cannot be duplicated; i.e., two devices cannot have the same logical name.

### Characteristics

The REASSIGN command:

Leaves the terminal in monitor mode.

Requires core.

Does not operate when the device is currently transmitting data.

### Restrictions

The job's controlling terminal cannot be reassigned.

**REASSIGN command (Cont)**

Associated Messages

Refer to Chapter 4.

Examples

.REASSIGN LPT:17 )  
:  
.REASSIGN CDP:4 )  
:

Reassign the line printer to job 17.

Reassign the card punch to job 4.

**REATTA program**

Function

The REATTA program allows a user to transfer his job from one terminal to another. Unlike the ATTACH command, REATTA does not require a password or that the terminal be of the same type that LOGIN recognizes in order to run the job. For example, usually a [1,2] job can run only on a local terminal. However, the REATTA program can be used to attach a [1,2] job from a local terminal to a remote terminal.

Before reattaching his job, the user should verify that the terminal to which he is attaching is turned on and working properly. Otherwise, it might be difficult to retrieve the job.

Command Format

.R REATTA )

REATTA responds by asking for the new terminal name.

TYPE NEW TTY NAME:

The user answers with either the new terminal name (e.g., CTY, TTY2) or number (e.g., 2). REATTA then responds with

FROM JOB n

:

on the old terminal, and

NOW ATTACHED TO JOB n

:

on the new terminal.

Characteristics

The R REATTA command:

Leaves the terminal in monitor mode.

Runs the REATTA program, thereby destroying the user's core image.

Restrictions

The R REATTA command is not available to Batch users.

**REATA program (Cont)**Associated Messages

Refer to Chapter 4.

Examples

.R REATA)

TYPE NEW TTY NAME: TTY27)

FROM JOB 7

;appears on old terminal

⋮

NOW ATTACHED TO JOB 7

;appears on TTY 27

⋮

**REENTER command**

Function

The REENTER command is similar to the DDT command. It copies the saved program counter value from .JBPC into .JBOPC and starts the program at an alternate entry point specified in .JBREN (must be set by the user or his program). If the job was executing a UUO when it was interrupted (i.e., in exec mode but not in TTY input wait or SLEEP mode), the monitor continues the job until the UUO is completed and then traps to the REENTER address in .JBREN. If the job is in TTY input wait or SLEEP mode, the trap to the REENTER address occurs immediately and .JBOPC contains the address of the UUO. If the job is in user mode, the trap also occurs immediately. Therefore, it is always possible to continue the interrupted program after trapping by executing a JRSTF@.JBOPC.

Command Format

REENTER

Characteristics

The REENTER command:

- Places the terminal in user mode.
- Requires core.
- Requires the user to have a job number.

Associated Messages

Refer to Chapter 4.

Example

.REE )

**RENAME command <sup>1</sup>**Function

The RENAME command changes the name of one or more files on disk or DECtape.

Command Format

RENAME arg

arg = a pair of file specifications separated by an equal sign, or a string of such pairs separated by commas:

RENAME new1 = old1, new2 = old2, ...

Device or file structure names can be specified only with the new filename and remain in effect until changed or until the end of command string is reached. In addition, a protection may be specified with the new filename and remains in effect only for that filename. This command accepts the full wildcard construction.

Characteristics

The RENAME command:

- Leaves the terminal in monitor mode.
- Runs the PIP program, thereby destroying the user's core image.
- Depends on FTCCLX which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

---

<sup>1</sup>This command runs the COMPIL program, which interprets the command before running PIP.

**RENAME command (Cont)**

Example

```
.RENAME T11.MAC=T1.MAC )  
FILES RENAMED:  
T1.MAC
```

```
.RENAME *.BAK=*.MAC )  
FILES RENAMED:  
T11.MAC  
T2.MAC  
T3.MAC
```

```
.RENAME TEST.MAC<057>=TEST.MAC )  
FILES RENAMED:  
TEST.MAC
```

⋮

**RESOURCES command**Function

The RESOURCES command prints the names of all available devices (except TTY's and PTY's), all file structures, and all physical units not in file structures (unless they are down or non-existent).

Command Format

RESOURCES

Characteristics

The RESOURCES command:

Leaves the terminal in monitor mode.  
Does not require LOGIN.

Example

```
.RES)  
DSKA,DSKB,DSKC,DPB0,DPB1,CDR0,2,PTR0,LPT0,1,2,3,DTA0,3,4,5,6,7,MTA0,1,2,  
PTP0,CDP0,PLT0,DIS0
```

**RESTORE program**

Function

The RESTORE program enables the user to place back onto disk that which was saved on the backup medium (magnetic tape, disk, or DECTape) with the BACKUP program. This includes restoring the entire disk or a subset of the disk. The data to be returned to the disk is read from the BACKUP SET file. This file contains the data that was saved with one BACKUP command. On a restore, either the BACKUP SET file can be scanned for the desired files or the index file can be searched to determine where the requested data is stored within the BACKUP SET file. The index file contains the directories of all areas written on the backup medium along with the relative block number in the BACKUP SET file where each file begins. When the entire backup medium is being restored, the RESTORE program starts at the beginning of the index file and continues until it reaches the last file in the index.

During a restore, a command recovery file is created that contains information concerning the portion of the user's command that has been executed and the portion that is remaining. This file resides on the disk and is updated as portions of the user's request are completed. The command recovery file is valuable if the system fails because only part of the restore need be redone.

As files are restored to disk, UFDs are created for each file structure on which the user has files. These newly created UFDs are then entered into the MFD.

Command Format

R RESTORE

The following commands may be typed by the user after the RESTORE program outputs a slash. These commands are stored in core and are not processed until the START command is given. The full wildcard construction may be used to replace the filename or the extension (refer to Paragraph 1.4.2.4).

| <u>Command</u> | <u>Explanation</u>                                                                                                                                                                                             |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BACKSPACE FILE | Backspaces the magnetic tape to a user file header and positions the tape before the header.                                                                                                                   |
| BACKSPACE SET  | Backspaces the magnetic tape to a BACKUP header and positions the tape either before the header or to the beginning of the tape if there is no BACKUP header (i.e., there is only one BACKUP set on the tape). |
| BACKSPACE UFD  | Backspaces the magnetic tape to a UFD header and positions the tape immediately before the header.                                                                                                             |

(continued on next page)

**RESTORE program (Cont)**

Command Format (cont)

| <u>Command</u>       | <u>Explanation</u>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DELETE dev:file.ext  | Deletes the named file from the designated device. This device must be one on which a BACKUP has been done.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| DENSITY MTAn:x       | Sets the magnetic tape density as specified by x.<br><div style="margin-left: 100px;">                     x = 2      200 bpi<br/>                     x = 5      556 bpi<br/>                     x = 8      800 bpi                 </div>                                                                                                                                                                                                                                                                                                                                                                                                    |
| DUMP ON dev:file.ext | The default is the system standard defined at MONGEN time. Dumps the contents of the BACKUP set file beginning at the present position and ending at the next file control word. All types of errors are ignored. The device on which a dump is to be written may not be a listing device.                                                                                                                                                                                                                                                                                                                                                      |
| ERROR DUMP/switch    | Returns to the last file control word and dumps the file if a transmission error of the type specified has occurred.<br>/switch = any or all of the following<br><div style="margin-left: 100px;">                     /CHECKSUM<br/>                     /PARITY<br/>                     /READ<br/>                     /WRITE                 </div>                                                                                                                                                                                                                                                                                         |
| ERROR HALT/switch    | Halts program execution if the type of error specified by /switch occurs during restoring. An asterisk is typed to the user so that he may type further instructions. The user may want to backspace the tape and dump it. The command recovery file is destroyed unless simply a START command is given. In this case, the current file is skipped and the next command in the recovery file is executed.<br>/switch = any or all of the following:<br><div style="margin-left: 100px;">                     /CHECKSUM<br/>                     /EXCEPT<br/>                     /PARITY<br/>                     /READ                 </div> |
| INDEX dev:file.ext   | Reads the index file with the designated filenames from the device specified.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

(continued on next page)

**RESTORE program (Cont)**

Command Format (cont)

| <u>Command</u>                                | <u>Explanation</u>                                                                                                                                                                                                                                                                                  |
|-----------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LOG dev:file.ext /switch                      | Writes a file on the specified device so that operations of the RESTORE program can be recorded. The default is DSK:RESTOR.LOG.<br><br>/switch = /ERROR<br><br>Logs only the errors. If this switch is omitted, all operations are recorded.                                                        |
| PARITY dev: ODD or EVEN                       | Specifies the parity on magnetic tape as odd or even. The default is odd.                                                                                                                                                                                                                           |
| RESTORE dev1: [p,p] file.ext<br>← dev2/switch | Writes the specified files from dev2 to the designated area on dev1. For example, if all of the disk is to be restored from the entire BACKUP SET file, the command is<br><br>RESTORE DSK: ← MTA1:<br><br>/switch = /EXCEPT file descriptor<br><br>Indicates the files that should not be restored. |
| REWIND dev:                                   | On magnetic tape, closes BACKUP set and rewinds tape. On disk, closes BACKUP set.                                                                                                                                                                                                                   |
| SET ACCESS dd-mmm-yy                          | Sets the access date to be used when restoring files. The files will be restored only if accessed after this date.                                                                                                                                                                                  |
| SET CREATION dd-mmm-yy                        | Sets the creation date to be used when restoring files. The files will be restored only if created after this date.                                                                                                                                                                                 |
| SKIP dev: FILE                                | Advances to next file trailer, EOF1, and positions after it.                                                                                                                                                                                                                                        |
| SKIP dev: SET                                 | Advances to next BACKUP set trailer and positions after it, or skips to end of the tape and positions the tape between the tape marks.                                                                                                                                                              |
| SKIP dev: UFD                                 | Advances to next UFD header, HDR1, and positions before the header.                                                                                                                                                                                                                                 |
| START                                         | Begins execution of a series of commands entered previously. Commands are not processed until a START command. If there are no commands to be processed when the START is executed, the command recovery file is searched for an executable command.                                                |
| UNLOAD dev:                                   | Performs a rewind and unload to the magnetic tape.                                                                                                                                                                                                                                                  |

**RESTORE program (Cont)**Command Format (cont)

The RESTORE program may be restarted by the user at any time. The user can cancel current requests and specify new ones with a `↑C ↑C START` sequence. The command recovery file is deleted with a `↑C START` sequence unless the next command given to RESTORE is a START. If this is the case, the command recovery file is searched and the RESTORE program proceeds according to the information in the file. A `↑C CONT` sequence does not delete the command recovery file; this sequence of commands completes the current requests.

Upon completion of all requests, the RESTORE program closes the log file and types an asterisk on the user's terminal indicating its readiness for more requests. Rewinds to the magnetic tape due to it being filled are actually rewind and unload operations to ensure that the magnetic tape is not overwritten. When the RESTORE program reaches completion, the magnetic tape last written on remains in position unless a REWIND command is given.

Characteristics

The R RESTORE command:

Runs the RESTORE program, thereby destroying the user's core image.

Associated Messages

Refer to Chapter 4.

**REWIND command <sup>1</sup>**

Function

The REWIND command rewinds a magnetic tape or a DECtape. This command is equivalent to the PIP command string:

dev: (MW)-

Command Format

REWIND dev:

dev: = a magnetic tape (MTAn) or a DECtape (DTAn).

Characteristics

The REWIND command:

- Leaves the terminal in monitor mode.
- Runs the PIP program, thereby destroying the user's core image.
- Depends on FTCCLX which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

Examples

.\_REW DTA4: )

.\_REWIND MTA1: )

---

<sup>1</sup>This command runs the COMPIL program, which interprets the command before running the PIP program.

## RUN command

### Function

The RUN command loads a core image from a retrievable storage device and starts at the location specified within the file (.JBSA).

If the program has two segments, both the low and high segments are set up. If the high file has extension .SHR (as opposed to .HGH), the high segment will be shared. Therefore, if the user has RUN (or GET) the same program, I/O will not usually be required for the high segment. A two-segment program may have a low file extension (.LOW).

The RUN command clears all of user core. However, programs should not count on this action and should explicitly clear those areas of core that are expected to contain zeroes (i.e., the programs should be self-initializing). This action allows programs to be restarted by a  $\mathcal{C}$ , START sequence without having to do another RUN command.

On magnetic tape, if the low or high segment is missing, a null record is output before the EOF for the missing segment so that two EOFs cannot occur consecutively. Therefore, a saved null segment does not appear as a logical EOT (2 EOFs in a row).

### Command Format

RUN dev:file.ext [proj,prog] core

dev: = the logical or physical name of the device containing the core image. The default device name is DSK:.(In nondisk monitors, the default is the generic name that matches the system device.)

file.ext = the name of the file containing the core image; .ext applies to the low file, not the high file. An extension of .SHR, then .HGH, is assumed for the high file. If the user types an extension of .SHR or .HGH, the extension is treated as a null extension since .SHR and .HGH are confusing as low file extensions. The default filename is the job's current name as set by the last R, RUN, GET, SAVE, or SSAVE command, the last SETNAM UUO, or the last command which ran a program.

[proj,prog] = the project-programmer number; required only if core image file is located in a disk area other than the user's.

core = the amount of core to be assigned to the sum of the low and high segments if different from minimum core needed to load the program or from the core argument of the SAVE command which saved the file.

If core < the minimum low segment size, then an error message occurs.

If core  $\geq$  the minimum low segment size and < the sum of the high segment and the minimum low segment size, then the core assignment is the low segment size.

If core  $\geq$  the sum of the minimum low segment and the high segment size, then the core assignment is the size of both the low and high segments to be used.

**RUN command (Cont)**

Command Format (cont)

Core arguments can be specified in units of 1024 words or 512 words (a page) by following the number with K or P, respectively. For example, 2P represents 2 pages or 1024 words. If K or P is not specified, K (1024 words) is assumed.

Note that on KA10 based systems (DECsystem-1040, 1050, 1055), the minimum unit of allocation is 1024 words. Therefore, all arguments are rounded up to the nearest multiple of 1024 words (e.g., 3P is treated as 2K on a KA10 based system).

Since previous core is returned, MTA must have the core argument because there is no directory telling how much core is for the low segment. Refer to Appendix D.

Characteristics

The RUN command:

Places the terminal in user mode.

Restrictions

On systems with a large amount of core memory, the user should not specify a core argument that forces the high segment to start higher than 400000 (i.e., a core argument of greater than 128K) unless the program's high segment is location independent. If this is done, the ILLEGAL UUO error message is likely to occur.

Associated Messages

Refer to Chapter 4.

Examples

```
._RUN TEST )  
._RUN HISTST [10,63] )  
._RUN DTA3:TEST1 )
```

## SAVE command

### Function

The SAVE command writes out a core image of the user's core area on the specified device. It saves any user program (two-segment sharable, one-segment nonsharable, or two-segment nonsharable) as one or two files. Later, when the program is loaded by a GET, R, or RUN command, it will be nonsharable. If DDT was loaded with the program, the entire core area is written; if not, the area starting from zero up through the program break (as specified by .JBFF) is written. Refer to DECsystem-10 Monitor Calls for a description of the job data area locations referenced by this command.

The SAVE command should be used instead of the SSAVE command when debugging a two-segment program. Refer to Appendix D for additional information on the SAVE command.

On magnetic tape, if the low or high segment is missing, a null record is output before the EOF for the missing segment so that two EOFs cannot occur consecutively. Therefore, a saved null segment does not appear as a logical EOT (2 EOFs in a row).

### Command Format

SAVE dev:file.ext [proj,prog] core

dev = the device on which the core image file is to be written. The default device name is DSK:. In nondisk monitors, the default is the generic name that matches the system device. The colon following the device name is required if a device is specified.

file.ext = the name to be assigned to the core image file. The default filename is the job's current name as set by the last R, RUN, GET, SAVE, or SSAVE command, the last command which ran a program (e.g., DIRECT), or the last SETNAM UUO.

ext applies to the low file, not the high file. An extension of .SHR, then .HGH, is assumed for the high file. If the user types an extension of .SHR or .HGH, the extension is treated as a null extension since .SHR and .HGH are confusing as low file extensions. If ext is omitted and the program has only one segment, the ext is assumed to be .SAV. If ext is omitted and the program has two segments, the high segment will have extension .HGH, and the low segment will have extension .LOW.

[proj,prog] = the name of the disk area on which the core image file is to be written.

core = the amount of core in which the program is to be run. This value is stored in JOBDAT as the job's core area (.JBCOR) and is used by subsequent RUN and GET commands. This argument is optional.

Core arguments can be specified in units of 1024 words or 512 words (a page) by following the number with K or P respectively. For example, 2P represents 2 pages or 1024 words. If K or P is not specified, K (1024 words) is assumed.

**SAVE command (Cont)**

Command Format (cont)

Note that on KA10 based systems (DECsystem-1040, 1050, 1055), the minimum unit of allocation is 1024 words. Therefore, all arguments are rounded up to the nearest multiple of 1024 words (e.g., 3P is treated as 2K on a KA10 based system).

If core is omitted, only the number of blocks required by the core image area (as explained in the RUN command description) is assumed.

Characteristics

The SAVE command:

- Leaves the terminal in monitor mode.
- Requires core.
- Does not operate when a device is currently transmitting data.

Associated Messages

Refer to Chapter 4.

Example

```
.SAVE )  
JOB SAVED
```

```
.SAVE DTA3:TEST)  
JOB SAVED
```

**SCHED command**Function

The SCHED command types out the schedule bits as set by the last privileged SET SCHED command. The schedule bits are as follows:

|     |                                                                         |
|-----|-------------------------------------------------------------------------|
| 0   | regular timesharing.                                                    |
| 1   | no further logins allowed except from CTY.                              |
| 2   | no further logins from remote terminals, and no answering of data sets. |
| 4   | batch jobs only.                                                        |
| 100 | device mounts can be done without operator intervention.                |
| 200 | unspooling allowed.                                                     |
| 400 | no operator coverage.                                                   |

Command Format

SCHED

Characteristics

The SCHED command:

Leaves the terminal in monitor mode.  
Does not require LOGIN.  
Depends on output from the SET SCHED command which is normally absent in the DECsystem-1040.

Example

.SCHED  
000400

Regular timesharing, but no operator coverage.

**SEND command**

Function

The SEND command provides a mechanism for one-way interconsole communication. (This command replaces the TALK command.) A line of information is transmitted from one terminal to another, with the identification of the terminal sending the information. With remote communications capabilities, SEND is able to differentiate between stations.

When the SEND command is sent from the central station operator's terminal (OPR) or from a terminal logged in as [1,2], it allows a broadcast of a line of information to all non-slaved terminals (including remote terminals) in the system. This allows important information to be dispersed, such as system shutdown or hardware problems. SEND ALL messages do not go to slaved terminals unless the SET TTY NO GAG bit is set to permit reception when the terminal is busy.

A busy test is made on single-destination messages before the message is sent unless the sender or the receiver of the message is OPR or a job logged-in as [1,2]. The receiver of the message is considered busy if his terminal is not at monitor command level. If the receiver is busy, the sender receives the message BUSY and the information is not sent, unless the receiving terminal has the TTY NO GAG bit set (refer to the SET TTY command). If the receiving terminal is turned off, the information appears to have been sent, since the hardware cannot detect this condition on hard-wired terminals.

Command Format

SEND dev: text

or

SEND JOB n text

dev = any physical terminal name (CTY included) or OPRnn. If OPRnn is specified, the message is sent to the operator at station nn. If OPR (nn is null) is specified, the message is sent to the operator at the user's logical station. If the terminal sending the message is the operator's terminal, the argument may be ALL to provide the broadcast operation.

n = the job number to which the message is to be sent.

The message printed on the receiving terminal appears as follows:

;;TTY n: - text

where

n is the TTY sending the message, and text is the message. A bell sounds on the receiving terminal when the message is sent.

**SEND command (Cont)**Characteristics

The SEND command:

Leaves the terminal in monitor mode.

Does not require LOGIN.

Depends on FTTALK which is normally absent in the DECsystem-1040.

Restrictions

The SEND command is not available to the Batch user.

Associated Messages

Refer to Chapter 4.

Examples

```
SEND OPR: PLEASE WRITE-ENABLE DTA3 )  
.
```

**SET BLOCKSIZE command**

Function

The SET BLOCKSIZE command sets a default blocksize for the specified magnetic tape.

Command Format

SET BLOCKSIZE dev: nnn

dev: = MTAn: where n is the number of the magnetic tape drive for which the blocksize is to be set, or a logical name associated with a physical magnetic tape. The user must have the magnetic tape assigned to him. This argument is required.

nnn = a decimal number up to a maximum of 4095 designating the block size for this magnetic tape. No additional checking is done for the legality of the specified number besides the check for the maximum 4095. This argument is required.

Characteristics

The SET BLOCKSIZE command:

Leaves the terminal in monitor mode.

Depends on both FTSET and FTMTSET which are normally absent in the DECsystem-1040.

Examples

```
.SET BLOCKSIZE MTA2:3956)
:
.ASSIGN MTA4:NAME: )
MTA4 ASSIGNED
.SET BLOCKSIZE NAME:2000)
:
```

**SET CDR command**Function

The SET CDR command sets the filename for the next card-reader spooling intercept (refer to DECsystem-10 Monitor Calls). This command is generally not needed, even when the card reader is being simulated on the disk via the spooling mechanism. It is included in case the user wishes to reset or change the spooling. In addition, the Batch Controller uses this command to read spooled input card decks.

Command Format

SET CDR filename

filename = one- to three-character filename to be used on next card-reader INIT.

Characteristics

The SET CDR command:

Leaves the terminal in monitor mode.

Depends on FTSET and FTSP which are normally absent in the DECsystem-1040.

Examples

```
.SET CDR A )  
.  
.SET CDR MAS )  
.
```

## SET CPU command

### Function

The SET CPU command allows a privileged user to change the CPUs on which his job can run. It is used in a multiprocessing system to specify whether the programs run under the job can be processed on the primary CPU, the secondary CPU, or either CPU. The job remains with the specified CPU until (1) another SET CPU command with a different specification is given, (2) a KJOB command is issued, or (3) the user's program overrides the SET CPU command by issuing the SETUJO with a different specification. If the SETUJO overrides the command, the specification given in the UJO remains in effect until a RESET or EXIT UJO or another SETUJO with a different specification is executed. When an EXIT or RESET UJO is executed, the job reverts back to the specification given in the last SET CPU command. When the user logs in, the CPU specification is usually set to ALL. The schedulers for each CPU compete for jobs with the ALL specification so that the load is dynamically balanced between CPUs. Therefore, this command is generally not needed but is provided in case the user wishes to change the CPU specification.

### Command Formats

1. SET CPU CP<sub>xn</sub>

adds the specified CPU to the job's CPU specification.

2. SET CPU NO CP<sub>xn</sub>

removes the specified CPU from the job's CPU specification.

3. SET CPU ALL

adds all of the CPUs to the job's CPU specification.

4. SET CPU ONLY CP<sub>xn</sub>

changes the CPU specification so that it includes only the specified CPU.

x = either U designating a logical name or A or I designating physical names for a KA10 processor (DECsystem-1055) or a KI10 processor (DECsystem -1077), respectively.

n = a decimal number from 0 to the number of processors in the system.

**SET CPU command (Cont)**Characteristics

The SET CPU command:

Leaves the terminal in monitor mode.

Depends on FTSET and FTMS which are normally absent in the DECsystem-1040, 1050, and 1070.

Restrictions

The privileges required for using this command are determined by bit 5 (JP.CCC) of the privilege word, .GTPRV.

Associated Messages

Refer to Chapter 4.

Examples

```
.SET CPU ONLY CPU1 )  
.  
.SET CPU CPA0 )  
.
```

**SET DENSITY command**

Function

The SET DENSITY command sets a default density for the specified magnetic tape.

Command Format

SET DENSITY dev: nnn

dev: = MTAn: where n is the number of the magnetic tape drive for which the density is to be set, or a logical name associated with a physical magnetic tape. The user must have the device assigned to him. This argument is required.

nnn = 200 bpi  
556 bpi  
800 bpi

This argument is required.

Characteristics

The SET DENSITY command:

Leaves the terminal in monitor mode.  
Depends on both FTSET and FTMTSET which are normally absent in the DECsystem-1040.

Examples

```
.SET DENSITY MTAS: 556)  
:
```

**SET DSKPRI command**Function

The SET DSKPRI command allows a privileged user to set the priority for his job's disk operations (data transfers and head positionings). The standard priority is 0, and the range of permissible values is -3 to +3. This means that a priority lower than the standard can be specified, as well as one higher than the standard. The priority specified applies to all disk I/O channels currently open or subsequently opened whose priority has not been explicitly set with a DISK. UUO (refer to DECSYSTEM-10 Monitor Calls). The priority specified in the SET DSKPRI command remains in effect until (1) another SET DSKPRI command is given with a different priority, (2) a KJOB command is issued, or (3) the user's program overrides the SET DSKPRI command by issuing a DISK. UUO with a different priority.

Command Format

SET DSKPRI n

n = a decimal number from -3 to +3 indicating the priority to be associated with the job's disk operations. When n = 0, the priority is the normal timesharing priority.

Characteristics

The SET DSKPRI command:

Leaves the terminal in monitor mode.  
Depends on both FTSET and FTDPRI which are normally absent in the DECSYSTEM-1040.

Restrictions

The privileges required for using this command are determined by bits 1 and 2 of the privilege word, .GTPRV. These two bits specify an octal number from 0-3. The user is always allowed a 0 priority.

Examples

```
._SET DSKPRI 2,
.
```

**SET HPQ command**

Function

The SET HPQ command allows a privileged user to place his job in a high-priority scheduler run queue. With this command, the user obtains a faster response and CPU time than in the normal timesharing queues. The job remains in the specified high-priority queue until (1) another SET HPQ command to a different high-priority queue is given, (2) a KJOB command is issued, or (3) the user's program overrides the SET HPQ command by issuing an HPQ UUC with a different value. If an HPQ UUC overrides the command, the level specified in the UUC remains in effect until a RESET or EXIT UUC or another HPQ UUC with a different value is executed. When an EXIT or RESET UUC is executed, the job is returned to the high-priority queue specified in the SET HPQ command.

Command Format

SET HPQ n

n = a decimal number from 0 to 15 indicating the high-priority queue to be entered. When n = 0, the queue is the normal timesharing run queue. Queue numbers from 1 to 15 are high-priority queues. The number of high-priority queues is an installation parameter and may be less than 15.

Characteristics

The SET HPQ command:

- Leaves the terminal in monitor mode.
- Depends on both FTSET and FTHPQ which are normally absent in the DECsystem-1040.

Restrictions

The privileges required for using this command are determined by bits 6 through 9 of the privilege word, .GTPRV. These four bits specify an octal number from 0-17, which is the highest priority queue attainable by the user.

Examples

```
.SET HPQ 4 )  
:  
:
```

**SET SPOOL command**Function

The SET SPOOL command adds devices to or deletes devices from the current list of devices being spooled for this job. Spooling is the mechanism by which I/O to or from slow-speed devices is simulated on disk. Devices capable of being spooled are: the line printer, the card punch, the card reader, the paper tape punch, and the plotter.

Command Formats

1. SET SPOOL dev1, dev2, ...devn  
adds the specified devices to the job's spool list.
2. SET SPOOL ALL  
places all spooling devices into the spool list.
3. SET SPOOL NONE  
clears the entire spool list.
4. SET SPOOL NO dev1, dev2, ...devn  
removes the specified devices from the job's spool list.  
  
dev1, dev2, ... devn = names of one or more devices to be added to or deleted from the current spool list.

Characteristics

The SET SPOOL command:

Leaves the terminal in monitor mode.

Depends on both FTSET and FTSP which are normally absent in the DECsystem-1040.

Restrictions

To unspool devices, the job must have (1) the privilege bit set in .GTPRV, (2) bit 28 (200 octal) set in the STATES word by the operator SET SCHED command, or (3) the user must be logged-in under [1,2].



**SETSRC program**

Function

The SETSRC program is used to manipulate the job's search list or the system's search list. A search list is defined to be the order of file structures that are to be searched whenever generic device DSK: is explicitly or implicitly specified by the user. This search list is originally defined by the system manager to include the file structures which the user can access. With the SETSRC program, the user can alter the search list defined for him by adding or deleting file structures.

The search list is in the form

fs1/s/s, fs2/s/s, ..., FENCE, ....., fs9/s/s

where fs is the name of the file structure and /s is a switch modifying the file structure. The file structures on the left of the FENCE comprise the active search list and represent the generic device DSK for this job. The files to the right of the FENCE comprise the passive search list and represent file structures that were once in the active search list. File structures are kept in the passive search list in order that quotas can be checked on a DISMOUNT or KJOB command. The FENCE represents the boundary between the active and passive search list.

Note that the MOUNT and DISMOUNT commands can also change the job's search list by adding or deleting a file structure. Since the SETSRC program does not create a UFD if one does not exist, the MOUNT command should be used to create a UFD. The name of the new file structure is placed at the end of the search list.

Refer to the SETSRC specification in the DECsystem-10 Software Notebooks for a complete description of the SETSRC program.

Command Format

R SETSRC

The user can then respond with any of the following commands:

Command

Explanation

A

Add one or more file structures to the existing search list. The file structures (with any switches) are appended to the beginning or the end of the active search list according to the following specifications:

1. If no asterisk appears in the specification (e.g., fs1, fs2) or if an asterisk appears before the file structure names (e.g., \*, fs1, fs2), the file structures are added to the end of the search list.

(continued on next page)

**SETSRC program (Cont)**

Command Format (cont)

| <u>Command</u> | <u>Explanation</u>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A (cont)       | 2. If the asterisk follows the file structure names (e.g., fs1, fs2, *), the file structures are added to the beginning of the search list.<br><br>3. If the asterisk appears in the middle of the file structures (e.g., fs1, *, fs2), the file structures before the asterisk are added to the beginning of the search list and the file structures after the asterisk are added to the end.<br><br>If the specified file structure is currently in the search list, it is removed and then added in the desired position. Therefore, this command can be used to reorder the search list. |
| C              | Create a new search list for this job. Any file structures in the current search list which are not in the new list are moved to the passive search list.                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| CP             | Create a new default directory path.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| CS             | Create a new system search list (i.e., the file structure search list for device SYS:). The user must be logged in under [1,2] to use this command.                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| H              | Obtain information about the available commands.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| M              | Modify the current search list and DSK specification by altering the switch settings for individual file structures. This command does not add or remove file structures from the search list.                                                                                                                                                                                                                                                                                                                                                                                               |
| R              | Remove file structures from the search list. They are placed on the right side of the FENCE (passive search list) so that on subsequent LOGOUTs or DISMOUNTs quota limits can be checked.                                                                                                                                                                                                                                                                                                                                                                                                    |
| T              | Type the search list of the job.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| TP             | Type the default directory path.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| TS             | Type the system search list.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

The following switches can be used in the SETSRC command string. Switches that modify file structures must appear immediately after the file structure that they modify. Other switches can appear anywhere in the command string. The switches can be abbreviated as long as the abbreviation is unique. The minimum number of characters is underlined below.

Switches that modify file structures

/CREATE Allow new files to be created on the file structure.

(continued on next page)

**SETSRC program (Cont)****Command Format (cont)**

**/NOCREATE** Do not allow new files to be created on the file structure when DSK is specified, but allow files to be superseded. Files can be created on the file structure if the user specifies the file structure name explicitly.

**/NOWRITE** Do not allow writing on the file structure for this job (i.e., the file structure is read only).

**/WRITE** Allow writing on the file structure.

If no switches are specified, **/CREATE** and **/WRITE** are assumed. For compatibility with previous versions of SETSRC, **/N** is equivalent to **/NOCREATE** and **/R** equivalent to **/NOWRITE**.

Switches that modify the directory path (used only with the CP command)

These switches can be typed in directly as commands by omitting the CP command and the slash (i.e., **/SCAN** is equivalent to **CP/SCAN**).

**/NOSCAN** Cancel the scan switch for the directory path.

**/SCAN** Set the scan switch for the directory path.

Switches that modify the DSK or SYS specification (used only with the C and M commands)

These switches can be typed in directly as commands by omitting the C or M command and the slash (i.e., **NOSYS** is equivalent to **M/NOSYS**).

**/LIB: [ proj, prog]** Set the job's library directory to the UFD [ proj, prog] and add it to the user's DSK specification. This means that if a file is not found in the user's directories in his search list, the library directory will then be searched for the file.

**/NOLIB** Remove the library directory from the user's DSK specification.

**/NOSYS** Remove the SYS specification from the user's DSK specification.

**/NONEW** Remove the [ 1,5] directory from the user's SYS specification.

**/SYS** Add the SYS specification to the user's DSK specification. This means that if a file cannot be found in the user's directories in his search list or in his library directory (if **/LIB: [ proj, prog]** has been specified), the system directory [ 1,4] will then be searched for the file.

**/NEW** Add the directory [ 1,5] to the user's SYS specification. This means that when the system directory is searched, the directory [ 1,5] will be searched before the directory [ 1,4].

**Characteristics****The R SETSRC command**

Places the terminal in user mode.

Runs the SETSRC program, thereby destroying the user's core image.

**SETSRC program (Cont)**

Restrictions

The user must be logged in under [1,2] to create a new system search list. The directory path commands (CP and TP) are meaningful only with the 5.04 and later monitors and only if FTSSD is on.

Examples

.R SETSRC )

\*I )

DSKB: , FENCE

The user's search list is defined as DSKB.

\*A DSKA: )

Add DSKA to the end of the search list.

\*I

DSKB: , DSKA: , FENCE

The user's search list is defined as DSKB, DSKA.

\*A DSKC: , \* )

Add DSKC to the beginning of the search list.

\*I )

DSKC: , DSKB: , DSKA: , FENCE

Remove DSKA from the search list.

\*R DSKA: )

\*I )

DSKC: , DSKB: , FENCE , DSKA:

The user's search list is defined as DSKC, DSKB.

\*M DSKB: /NOWRITE )

Do not allow writing on DSKB.

\*M /LIB: [27,500] )

Set the user's library directory to [27,500] and add it to the user's DSK specification.

\*SYS )

Add SYS: to the user's DSK specification.

\*I )

/LIB: [27,500] /SYS DSKC: , DSKB: /NOWRITE , FENCE , DSKA:

The user's DSK and SYS specifications are first followed by the user's search list.

\*TS )

DSKA: , DSKB: , DSKC: , FENCE

The system search list is defined as DSKA, DSKB, DSKC.

**SET TIME command**

Function

The SET TIME command sets a central processor time limit for a job. When the time limit is reached, the job is stopped and a message is typed. A timesharing job may be continued by typing CONT, but no time limit is in effect unless it is reset. A Batch job cannot be continued.

Command Format

SET TIME n

n = number of seconds of central processor time to which the job is limited. An argument of 0 cancels the time remaining.

Characteristics

The SET TIME command:

- Leaves the terminal in monitor mode.
- Depends on both FTSET and FTTLIM which are normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

Examples

```

.MAKE LOOP.F4)
*I10    CONTINUE
        GOTO 10
        END

$$
*EX$$

.TYPE LOOP.F4)
I0     CONTINUE
      GOTO 10
      END
    
```

Create a program with an indefinite loop.

Type the program.

(continued on next page)

**SET TIME command (Cont)**

Examples (cont)

```

LOAD LOOP)
FORTRAN: LOOP.F4
LOADING

```

Compile and load the program.

LOOP 2K CORE

EXIT

.SET TIME 5)

Set the time limit to 5 seconds.

.TIME)

Clear the incremental run time, so that the SET TIME command can be checked.

3.50

5.57

KILO-CORE-SEC=32

.START)

Start the loop.

? TIME LIMIT EXCEEDED

As expected, the time limit was exceeded.

.TIME)

3.00

10.57

KILO-CORE-SEC=67

:

**SET TTY or TTY command**

Function

The SET TTY command (or TTY command) declares properties of the terminal line on which the command is typed to the scanner service. With hardwired TTYS, the system manager can set the default conditions, so that this command is usually not needed. However, the user is likely to use this command on data sets, where the terminal cannot be predicted.

Command Formats

1. SET TTY NO word  
 equivalent to TTY NO word

2. SET TTY word  
 equivalent to TTY word

NO = the argument that determines whether a bit is to be set or cleared. This argument is optional.

word = the various words representing bits that may be modified by this command. The words are as follows:

|                    |                                                                                                                                                                                                                                                               |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SET TTY ALTMODE    | Converts the ALTmode codes of 175 and 176 to the ASCII standard escape character 033.                                                                                                                                                                         |
| SET TTY NO ALTMODE | Restores the individual identity of the codes 175 and 176.                                                                                                                                                                                                    |
| SET TTY BLANKS     | Restores multiple carriage return-line feeds and form feeds.                                                                                                                                                                                                  |
| SET TTY NO BLANKS  | Suppresses blank lines (consecutive carriage return-line feeds after the first) and outputs form feeds and vertical tabs as 2 carriage return-line feeds. This is used for the display terminal in order to prevent the output from moving up off the screen. |
| SET TTY CRLF       | Restores the carriage return.                                                                                                                                                                                                                                 |
| SET TTY NO CRLF    | The carriage return normally output at the end of a line exceeding the carriage width is suppressed.                                                                                                                                                          |
| SET TTY ECHO       | Restores the normal echoing of each character typed in.                                                                                                                                                                                                       |

(continued on next page)

**SET TTY command (Cont)**

Command Formats (cont)

- SET TTY NO ECHO            The terminal line has local copy and the computer should not echo characters typed in.
- SET TTY FILL n            The filler class n is assigned to this terminal. The filler character is always DEL (RUBOUT, 377 octal). No fillers are supplied for image mode output.
- SET TTY NO FILL           Equivalent to TTY FILL 0. Fillers for output and echoing are determined from the following:

| Character Name | Octal | Number of Fillers for Filler Class |        |        |          |
|----------------|-------|------------------------------------|--------|--------|----------|
|                |       | 0                                  | 1      | 2      | 3        |
| BS             | 010   | 0                                  | 2      | 6      | 6        |
| HT             | 011   | 0                                  | 1 or 2 | 1 or 2 | 1 or 2†  |
| LF             | 012   | 0                                  | 1      | 6      | 6        |
| VT             | 013   | 0                                  | 2      | 6      | 6        |
| FF             | 014   | 0                                  | 12     | 21     | 21       |
| CR             | 015   | 0                                  | 1 or 2 | 2 or 4 | 2 or 4†† |
| XON            | 021   | 0                                  | 1      | 1      | 1        |
| TAPE           | 022   | 0                                  | 1      | 1      | 1        |
| XOFF           | 023   | 0                                  | 1      | 1      | 1        |
| NTAP           | 024   | 0                                  | 1      | 1      | 1        |

†1 if 0-3 spaces to tab stop; 2 if 4-7 spaces to tab stop.  
 ††1 or 2 if CR is typed; 2 or 4 if CR is supplied because the line is too long.

- SET TTY FORM              This terminal has hardware FORM (PAGE) and VT (vertical tab) characters.
- SET TTY NO FORM         The monitor sends eight line feeds for a FORM and four line feeds for a VT.
- SET TTY GAG              The SEND command cannot be received at this terminal unless the terminal is at command level (initial state).
- SET TTY NO GAG         The SEND command can be received at this terminal even though it is not at command level.
- SET TTY LC                The translation of lower-case characters input to upper case is suppressed.

(continued on next page)

**SET TTY command (Cont)**

Command Formats (cont)

|                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SET TTY NO LC   | The monitor translates lower-case characters to upper case as they are received. In either case, the echo sent back by the monitor matches the case of the characters after translation. By looking at the printout, the user can determine what translation was performed by the monitor.                                                                                                                                                                         |
| SET TTY PAGE    | The user has the ability to temporarily suspend system timeout without losing it. The XOFF key (tS) suspends the timeout, and the XON key (tQ) restores it. The XOFF and XON keys are not echoed and are not sent to the user's program. This command is useful for display terminals where the user may want to read a page of text before it disappears from the screen. Note that this preempts the use of tS and tQ for reading paper tape (see SET TTY TAPE). |
| SET TTY NO PAGE | The timeout control ability of the XOFF and XON keys is disabled. The current interpretation of these keys depends on the last SET TTY TAPE command.                                                                                                                                                                                                                                                                                                               |
| SET TTY SLAVE   | The terminal becomes slaved, i.e., no commands may be typed on the terminal, and the terminal may be ASSIGNED by another user. The user can slave only his own terminal and must contact the operator in order to unslave it.                                                                                                                                                                                                                                      |
| SET TTY TAB     | This terminal has hardware TAB stops every eight columns.                                                                                                                                                                                                                                                                                                                                                                                                          |
| SET TTY NO TAB  | The monitor simulates TAB output from programs by sending the necessary number of SPACE characters.                                                                                                                                                                                                                                                                                                                                                                |
| SET TTY TAPE    | The XON key (tQ) causes the terminal to read paper tape. The XOFF key (tS) causes the terminal to stop reading paper tape.                                                                                                                                                                                                                                                                                                                                         |
| SET TTY NO TAPE | The XON key (tQ) and the XOFF key (tS) have no special paper tape function. They may have a PAGE function.                                                                                                                                                                                                                                                                                                                                                         |
| SET TTY WIDTH n | The carriage width (the point at which a free carriage return is inserted) is set to n. The range of n is 17 (two TAB stops) to 200 decimal.                                                                                                                                                                                                                                                                                                                       |

**SET TTY command (Cont)**

Characteristics

The SET TTY command:

Leaves the terminal in monitor mode.

Does not require LOGIN.

Depends on FTSET which is normally absent in the DECsystem-1040. However, the TTY command format can always be used.

**SET WATCH command**Function

The SET WATCH command sets the system to print incremental job statistics automatically. This command provides the user with a tool for measuring the performance of his programs.

Command Formats

1. SET WATCH  $arg_1, arg_2, \dots, arg_5$   
prints the specified WATCH statistics.
2. SET WATCH ALL  
prints all the WATCH statistics.
3. SET WATCH NONE  
eliminates the printing of all WATCH statistics.
4. SET WATCH NO  $arg_1, arg_2, \dots, arg_5$   
eliminates the printing of the specified WATCH statistics.

The following arguments enable printing whenever a monitor command switches the console from monitor to user mode.

arg = DAY prints the time of day, as [HH:MM.SS]

arg = VERSION prints the version of the program in standard format (refer to the VERSION command).

The following arguments enable printing whenever the console is returned to monitor mode via the ↑C, EXIT, HALT, ERROR IN JOB n, or DEVICE xxx OPR zz ACTION REQUESTED messages.

arg = READ prints the incremental number of disk blocks read modulo 4096.

arg = RUN prints the incremental run time.

arg = WAIT prints the wait time (time elapsed since the user started or continued the program).

arg = WRITE prints the incremental number of disk blocks written modulo 4096.

**SET WATCH command (Cont)**

Command Formats (cont)

Any combination of the arguments may be specified in any order. Statistics are not printed for commands that do not run programs, such as ASSIGN or PJOB. When a user logs in, his job is set to WATCH the statistics of which he has notified the system manager. The information on what statistics to WATCH is kept in ACCT.SYS.

The order of the error message is the same as the order of output. Therefore, a user who forgets either the argument or the significance of the statistics can find these out by examining the message. A single space is always typed between each statistic, whether the statistic appears or not; therefore, it is possible to tell which statistics are being typed.

NOTE

Enabling WATCH output interacts with the incremental data typed by the TIME and DSK commands.

Characteristics

The SET WATCH command:

Leaves the terminal in monitor mode.

Depends on FTSET and FTWATCH which are normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

**SET WATCH command (Cont)**Examples

1. .SET WATCH P)  
?ARGS ARE: DAY,RUN,WAIT,READ,WRITE,VERSION,ALL,NONE  
  
.SET WATCH)  
?ARGS ARE: DAY,RUN,WAIT,READ,WRITE,VERSION,ALL,NONE  
  
.SET WATCH DAY RUN WAIT READ WRITE  
  
.R PIP)  
[22:38:19]  
\*+C  
[0.10 2.95 457 243]
2. .SET WATCH VERSION DAY)  
  
.R TECO)  
[9:44:30]  
  
[S:TECO 22(64) + ]  
\*+C

**SKIP command <sup>1</sup>**

Function

The SKIP command spaces a magnetic tape forward a specified number of files or records or to the logical end of tape. This command, depending on its arguments, is equivalent to the following PIP command strings:

MTAn: (M #nA) ←  
MTAn: (M #nD) ←  
MTAn: (M #nT) ←

Command Formats

1. SKIP MTAn: x FILES  
    advances forward x files.
2. SKIP MTAn: x RECORDS  
    advances forward x records.
3. SKIP MTAn: EOT  
    advances forward to the logical end of tape.

The words FILES, RECORDS, and EOT can be abbreviated to F, R, and E, respectively.

Characteristics

The SKIP command:

Leaves the terminal in monitor mode.  
Runs the PIP program, thereby destroying the user's core image.  
Depends on FTCCLX which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

---

<sup>1</sup>This command runs the COMPIL program, which interprets the command before running the PIP program.

**SKIP command (Cont)**

Examples

.\_SKIP MTA0: 4 FILES )

.\_SKIP MTA1: EOT )

.\_SKIP MTA2: 20 RECORDS )

**SSAVE command**

Function

The SSAVE command is the same as the SAVE command except that the high segment, if present, will be sharable when it is loaded with the GET command. To indicate this sharability, the high segment is written with extension .SHR instead of .HGH. A subsequent GET will cause the high segment to be sharable. Because an error message is not given if the program does not have a high segment, a user can use this command to save system programs without having to know which are sharable.

On magnetic tape, if the low or high segment is missing, a null record is output before the EOF for the missing segment so that two EOFs cannot occur consecutively. Therefore, a saved null segment does not appear as a logical EOT (2 EOFs in a row).

The SAVE command rather than the SSAVE command, should be used when debugging the program. This is because a GET command after a SSAVE command does not reinitialize the original high segment from the file after the user modifies it with the D command or the DDT program. Refer to Appendix D for more information on the SSAVE command.

Command Format

SSAVE dev:file.ext [proj,prog] core

Arguments and defaults are the same as in the SAVE command,

Characteristics

The SSAVE command:

- Leaves the terminal in monitor mode.
- Requires core.
- Does not operate when a device is currently transmitting data.

Associated Messages

Refer to Chapter 4.

Example

```
.SSAVE DSK:TEST )  
JOB SAVED
```

:

(continued on next page)

**SSAVE command (Cont)**

Example (Cont)

.LOAD FILE1 ↵  
MACRO: FILE1  
LOADING

Compile and load program.

LOADER 1K CORE  
EXIT

.SSAVE ↵  
JOB SAVED

Save a sharable copy. The filename is taken from the routine that contained the starting address.

.GET ↵  
JOB SETUP

Get a sharable copy.

**START command**

Function

The START command begins execution of a program either previously loaded with the GET command or interrupted (e.g., IC). The old program counter is copied from .JBPC to .JBOPC. An explicit start address is optional, and, if omitted, the address supplied in the file (.JBSA) is used. If an address argument is specified and the job was executing a UUO when interrupted (i.e., it was in exec mode but not in TTY input wait or SLEEP mode), the monitor sets a status bit (UTRP) and continues the job at the location at which it was interrupted before trapping to the specified START address. When the UUO processing is completed, the monitor clears the status bit, sets .JBOPC to the address following the UUO, and then traps to the START address. If the job is in TTY input wait or SLEEP mode, the trap to the program occurs immediately, and .JBOPC contains the address of the UUO. If the job is in user mode, the trap also occurs immediately.

Command Format

START adr

adr = the address at which execution is to begin if other than the location specified within the file (.JBSA). This argument is optional. If adr is not specified, the address comes from .JBSA. A starting address of 0 may be specified.

Characteristics

The START command:

- Places the terminal in user mode.
- Does not operate when a device is currently transmitting data.
- Requires core.
- Requires LOGIN if an address argument is specified.

Associated Messages

Refer to Chapter 4.

Example

.\_START )

## SUBMIT command

### Function

The SUBMIT command is used to place entries into the input queue for the Batch system. This command is equivalent to the following form of the QUEUE command:

QUEUE INP: jobname = control file, log file

### Command Format

SUBMIT jobname = control file, log file

jobname = name of the job being entered into the queue.

control file = name of the control file. This file contains all monitor-level and user-level commands for processing by the Batch Controller (BATCON).

log file = name of the log file. This file is used by the Batch Controller to record its processing of the job.

Only the two files mentioned above can be specified in a request to the Batch input queue. The name of the control file is required; the log file name is optional and, if omitted, is taken from the control file. If the jobname is omitted, it is the name of the first file in the request, not the name of the first file given. If an extension is omitted, the following are assumed:

.CTL for the control file  
.LOG for the log file.

Three categories of switches can be used in the command string:

1. Queue-operation - Only one of these switches can be placed in the command string because they define the type of queue request. The switch used can appear anywhere in the command string.
2. General - Each switch in this category can appear only once in the command string because they affect the entire request. The switch used can appear anywhere in the command string.
3. File control - Any number of these switches can appear in the command string because they are specific to individual files within the request. The switch used must be adjacent to the file to which it applies. If the switch precedes the filename, it becomes the default for subsequent files.

**SUBMIT command (Cont)**

Command Format (cont)

The following switches can be used with the SUBMIT command.

| <u>Switch</u>   | <u>Explanation</u>                                                                                                                                                                                                                                                                                                                                                                    | <u>Category</u> |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| /AFTER:tt       | Process the request after the specified time; tt is either in the form of hhmm (time of day) or +hhmm (time later than the current time). The resulting AFTER time must be less than the DEADLINE time. If the switch, or the value of the switch, is omitted, no AFTER constraints are assumed.                                                                                      | General         |
| /CARDS:n        | Use n (decimal) as the maximum number of cards that can be punched by the job. If the switch is omitted, no cards are punched. If the switch is given with no value, 2000 cards is assumed as the default.                                                                                                                                                                            | General         |
| /CORE:n         | Use n (decimal K) as the maximum amount of core memory that the job can use. If the switch is omitted, 25K is the maximum. If the switch is specified, but the value is omitted, the default maximum is 40K.                                                                                                                                                                          | General         |
| /CREATE         | Make a new entry into the Batch input queue. This switch is the default for the queue-operation switches.                                                                                                                                                                                                                                                                             | Queue Operation |
| /DEADLINE:tt    | Process the request before the specified time; tt is either in the form hhmm (time of day) or +hhmm (time later than the current time). The resulting DEADLINE time must be greater than the AFTER time. If the switch, or the value of the switch, is omitted, no DEADLINE constraints are assumed.                                                                                  | General         |
| /DEPEND:n       | Specify the initial value of the dependency count (in decimal). When used with /MODIFY, this switch changes the dependency count of another job. If n is a signed number (+ or -), that number is added to or subtracted from the dependent job's count. If n is not a signed number, the dependent job's count is changed to n. If this switch is omitted, no dependency is assumed. | General         |
| /DISPOSE:DELETE | Delete the file after processing.                                                                                                                                                                                                                                                                                                                                                     | File Control    |

(continued on next page)

**SUBMIT command (Cont)**

Command Format (cont)

| <u>Switch</u>     | <u>Explanation</u>                                                                                                                                                                                                                            | <u>Category</u> |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| /DISPOSE:PRESERVE | Save the file after processing. This is the default for all files except those with extensions .TMP, .LST, .CDP, .LPT, .PLT, and .PTP.                                                                                                        | File Control    |
| /DISPOSE:RENAME   | Rename the file from the specified directory immediately, remove it from the logged-out quota, and delete it after processing. This is the default for files with extensions .TMP, .LST, .CDP, .LPT, .PLT, and .PTP.                          | File Control    |
| /F                | List the entries in the input queue, but do not update the queues. Therefore, the list may not be an up-to-date listing, but the listing will be faster than with /LIST.                                                                      | Queue Operation |
| /FEET:n           | Use n (decimal) as the maximum number of feet of paper tape that the job can punch. If the switch is omitted, no paper tape is punched. If the value is omitted, the default is 10*B+20 feet, where B is the number of blocks in the request. | General         |
| /KILL             | Remove the specified entry from the Batch input queue. This switch can be used for deleting a previously-submitted request as long as the request has not been started by the Batch Controller.                                               | Queue Operation |
| /LIST             | List the entries in the input queue; the default is all entries for all jobs of all users.                                                                                                                                                    | Queue Operation |
| /MODIFY           | Alter the specified parameters in the job. This switch requires that the user have access rights to the job. It can be used for altering a previously submitted request as long as the request has not been started by the Batch Controller.  | Queue Operation |
| /NEW              | Accept the request although the file does not yet exist. This is the default for the log file. When placing this switch with the control file, the user can submit his job and then create the control file.                                  | File Control    |

(continued on next page)

**SUBMIT command (Cont)**

Command Format (cont)

| <u>Switch</u>   | <u>Explanation</u>                                                                                                                                                                                                                                                                                                                                                                                       | <u>Category</u> |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| /OUTPUT:n       | Cause job to terminate with a /Z:n to KJOB (n is from 0 to 4).<br><br>N=0 Suppress all normal queuing performed at LOGOUT time.<br>N=1 Queue only the log file.<br>N=2 Queue only the log file and spooled output (e.g., *.LPT).<br>N=3 Queue the log file, spooled output, and *.LST files.<br>N=4 Queue the log file, spooled output, *.LST files, and any requests deferred to LOGOUT time (default). | General         |
| /PAGE:n         | Use n (decimal) as the maximum number of pages of output that the job can print. If the entire switch is omitted, the maximum is 200 pages; if only the value is omitted, the maximum is 2000 pages.                                                                                                                                                                                                     | General         |
| /PHYSICAL       | Suppress logical device name assignments for the device specified.                                                                                                                                                                                                                                                                                                                                       | File Control    |
| /PRIORITY:n     | Assign the specified external priority (n=0 to 62) to the request. The larger the number, the greater priority the job has. The default is 10 if no switch is given and 20 if the switch is specified without a value.                                                                                                                                                                                   | General         |
| /PROTECT:nnn    | Assign the protection nnn (octal) to the job. If the switch, or the value of the switch, is omitted, the standard protection is assumed.                                                                                                                                                                                                                                                                 | General         |
| /RESTART:0 or 1 | A value of 0 means the job is not requeued or restarted by the Batch Controller after a system crash (default). A message is sent to the job's log file. A value of 1 means the job is restarted by the Batch Controller.                                                                                                                                                                                | General         |
| /SEQ:n          | Specify a sequence number to help in identifying a request to be modified or deleted.                                                                                                                                                                                                                                                                                                                    | General         |

(continued on next page)

**SUBMIT command (Cont)**

Command Format (cont)

| <u>Switch</u>  | <u>Explanation</u>                                                                                                                                                                                                     | <u>Category</u> |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| /START:n       | Begin on the nth line of the control file. If the switch, or the value of the switch, is omitted, the Batch Controller starts with the first line.                                                                     | File Control    |
| /START:xxx     | Start at the statement labelled xxx (up to 5 characters) of the control file.                                                                                                                                          | File Control    |
| /TIME:hhmmss   | Specify the central processor time limit for the job. If no switch is specified, the limit is 5 minutes; if the switch is given without a value, the limit is 1 hour.                                                  | General         |
| /TPLOT:n       | Use n (decimal minutes) as the maximum amount of plotting time allowed for the job. If the switch is omitted, no plotter time is allowed; if the value is omitted, but the switch is given, the maximum is 10 minutes. | General         |
| /UNIQUE:0 or 1 | Run any number of Batch jobs under this project-programmer number at the same time, if 0. Run only one Batch job at any one time, if 1 (default).                                                                      | General         |

Characteristics

The SUBMIT command:

- Leaves the terminal in monitor mode.
- Runs the QUEUE program.
- Depends on FTQCOM which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

**SUBMIT command (Cont)**

Examples

.\_SUBMIT USRJOB=CONTRL,LOGFIL,)

The defaults are as follows:

1. control file name is CONTRL.CTL
2. log file name is LOGFIL.LOG
3. no cards punched (/CARDS:0)
4. maximum core of 25K (/CORE:25)
5. no dependency (/DEPEND:0)
6. control and log files are saved after spooling (/DISPOSE:PRESERVE)
7. no paper tape punched (/FEET:0)
8. all line printer output is spooled with the maximum pages being 200 (/OUTPUT:4, /PAGE:200)
9. priority is 10 (/PRIORITY:10)
10. standard protection is assumed (/PROTECT:nnn (standard))
11. job is not restarted after a crash (/RESTART:0)
12. control file is begun on the first line (/START:1)
13. maximum CPU time is 5 minutes (/TIME:0:05)
14. no plotter time allowed (/TPLOT:0)
15. only one job at a time under a given project-programmer number is run (UNIQUE:1)

.\_SUBMIT USRJOB=/MODIFY/FEET:35/CORE,)

Modify the original request to include 35 feet as the maximum number of feet of paper tape that the job can punch and 40K of core as the maximum amount of core that the job can use. This command is valid only if the job has not been started yet by the Batch system.

.\_SUPMIT USRJOB=/KILL,)

Kill the job only if it has not been started by the Batch system.

**SYSTAT command**Function

The SYSTAT command runs a system program which prints status information about the system. This information allows a user to determine the load on the system before logging-in.

To write the output on the disk as a file with name SYSTAT.TXT, assign device DSK with logical name SYSTAT.

The SYSTAT command types the status of the system: system name, time of day, date, uptime, percent null time (idle plus lost time), number of jobs in use.

It types the status of each job logged-in: job number; project-programmer number (\*\*, \*\* = detached, [OPR] = the project-programmer number of the operator, [SELF] = user's project-programmer number); terminal line number (CTY = console terminal, DET = detached, Pn = PTY number); program name being run; program size; job and swapped state (refer to DECsystem-10 Monitor Calls); run time since logged-in.

It types the status of high segments being used: name (PRIV = nonsharable, OBS = superseded); device or file structure name from which the segment came; directory name (\*\*, \*\* if detached); size (SW = swapped out, SWF = swapped out and fragmented, F = in core and fragmented on disk, SPY = user is executing the SPY UJO); number of users in core or on the disk.

The command types swapping space used, virtual core used, swapping ratio, active swapping ratio, virtual core saved by sharing, average job size.

It types status of busy devices: device name, job number, how device is assigned (AS = ASSIGN command, INIT = INIT or OPEN UJO, AS+INIT = both ways).

It types system file structures: free blocks, mount count, single-access job.

It types remote stations: number of station, status of station.

It types dataset control: number of the TTY, status of TTY.

**SYSTAT command (Cont)**

Command Format

SYSTAT arg

arg = one or more single letters (in any order) used to type any subset of the SYSTAT output. This argument is optional. The following message, produced by typing SYSTAT /H, lists the various arguments to the SYSTAT command.

```
.SYSTAT/H)
SYSTAT V467(5)
SYSTAT INSTRUCTIONS:
TYPE "SYS<C.RET.>" TO LIST THE ENTIRE STATUS, OR
TYPE "SYS " FOLLOWED BY ONE OR MORE LETTERS AS FOLLOWS--
B  BUSY DEVICE STATUS
D  DORMANT SEGMENT STATUS
E  NON-DISK ERROR REPORT
F  FILE STRUCTURE STATUS
H  THIS MESSAGE
J  JOB STATUS
L  OUTPUT TO LPT
N  NON-JOB STATUS (ALL BUT J)
O  OTHER SYSTEM STATUS
P  DISK PERFORMANCE
R  REMOTE STATION STATUS
S  SHORT JOB STATUS
T  DATASET STATUS
X  READ DSK:CRASH.XPN
NNN PRINTS JUST JOB NNN (. DOES THIS JOB)
[P,PN] PRINTS JUST JOBS WITH THAT PROJ-PROG (P AND/OR PN MAY BE *)
#NNN PRINTS JUST JOBS FROM TERMINAL NNN
      (ALSO, C=CTY, PNN=PTYNN, TNN=TTYNN,.-THIS TTY)
```

Characteristics

The SYSTAT command:

- Leaves the terminal in monitor mode.
- Runs the SYSTAT program, thereby destroying the user's core image.
- Does not require LOGIN.
- Depends on FTCCLX which is normally absent in the DECsystem-1040.

**SYSTAT command (Cont)**

Examples

.SYSTAT)

STATUS OF B50400-05 #40, AT 16:15:17 ON 11-APR-72

OPTIME 7:00:59, 108% NULL TIME = 105% IDLE + 3% LOST  
 53 JOBS IN USE OUT OF 64. 50 LOGGED IN, 2 DETACHED

| JOB | WHO      | LINE# | WHAT   | SIZE(K) | STATE | RUN TIME |
|-----|----------|-------|--------|---------|-------|----------|
| 1   | [OPR]    | DET   | DAEMON | 5+SPY   | SL SW | 20       |
| 2   | 40,64    | 131   | TECO   | 2+3     | TC SW | 7        |
| 3   | [OPR]    | 2     | OPSER  | 1+3     | HB SW | 3:43     |
| 4   | [OPR]    | 4     | UMOUNT | 2+3     | TC SW | 3:38     |
| 5   | 2,111    | 3     | PIP    | 1+4     | TC SW | 26       |
| 6   | 274,1431 | 60    | COBDDT | 14+8    | RN    | 12:15    |
| 7   | [OPR]    | CTY   | PIP    | 1+4     | TC SW | 43       |
| 8   | 16,35    | 20    | FDSYS  | 22      | TI SW | 1:37     |
| 9   | 347,1246 | 65    | PIP    | 1+4     | TC SW | 5        |
| 10  | 271,1131 | 64    | QUEUE  | 2+4     | TC SW | 29       |
| 11  | [OPR]    | P0    | BATCON | 2+3     | RN    | 2:45     |
| 12  | 271,701  | 61    | SEEK   | 2+8     | TI SW | 5:08     |
| 13  | 337,1113 | 114   | KJOB   | 1+3     | TC SW | 13       |
| 14  | 40,64    | 150   | TECO   | 2+3     | TI SW | 4:37     |
| 15  | 122,216  | 124   | KJOB   | 1+4     | CB    | 2        |
| 16  | 16,107   | P2    | SYSTAT | 6+SPY   | SL SW | 25       |
| 17  | 142,1243 | 62    | TECO   | 2+3     | TI    | 1:00     |
| 18  | 146,500  | 112   | TECO   | 2+3     | TI    | 2:09     |
| 19  | 40,65    | 147   | EDITS  | 2+6     | TI SW | 1:24     |
| 20  | 240,353  | 155   | TECO   | 2+3     | TI SW | 1        |
| 21  | [OPR]    | P1    | LPTSPL | 2       | IO    | 13:47    |
| 22  | 345,1417 | 74    | EDITS  | 2+6     | TC SW | 43       |
| 23  | [OPR]    | P3    | LPTSPL | 2       | IO    | 18:39    |
| 24  | [OPR]    | P4    | PTPSPL | 2+3     | HB SW | 1:25     |
| 25  | [OPR]    | 1     | OPSER  | 1+3     | HB SW | 46       |
| 26  | [OPR]    | P6    | OPROMO | 2+5     | TI SW | 1:38     |
| 27  | [OPR]    | P5    | OPROMO | 2+5     | SL SW | 1:31     |
| 28  | 360,1436 | 137   | UMOUNT | 2+3     | CB    | 1:01     |
| 29  | 125,207  | 15    | EDITS  | 2+6     | TI SW | 1        |
| 30  | 110,1412 | 127   | TECO   | 2+3     | DI    | 7        |
| 31  | 402,570  | 12    | PIP    | 1+4     | CB SW | 1:26     |
| 32  | 405,505  | 25    | PIP    | 1+4     | TC SW | 1:13     |
| 33  | 40,633   | 146   | TECO   | 2+3     | TI    | 5:33     |
| 34  | 271,701  | 14    | SEEK   | 2+8     | TI SW | 4        |
| 35  | 10,20    | 142   | D      | 13+34   | TI    | 33:14    |
| 36  | 413,521  | 21    | TECO   | 2+3     | TI    | 20       |
| 37  | 402,570  | P7    | DCTDE  | 4       | TC SW | 12       |

(continued on next page)

**SYSTAT command (Cont)**

Examples (cont)

```

38 122,1202 136 EDITS 2+6 TI SW 1
39 60,60 153 TECO 2+3 TO 6:33
40 406,104 35 PLEASE 1 SL SW 3
41 402,574 36 SPACE 2 TI SW 10
42 271,701 30 RAFCO 6+8 TI 2:11
43 14,1145 P10 QMOUNT 2+3 SL SW 1
44 14,1145 11 DIRECT 1+3 TC SW 45
45 2,5 135 QUEUE 3+3 CB SW 0
46 11,176 144 COMPCT 5 TO 2:16
47 10,33 DET QMOUNT 2+3 TO SW 1
48 10,34 6 TECO 4+3 TI 3:02
49 2,5 13 SYSTAT 6+SPY RN 0
50 110,1367 157 TECO 2+3 TI SW 16
51 340,1107 26 AID 2+9 RN SW 1
52 2,5 7 SYSTAT 4 CB 0
53 122,1007 122 TECO 2+3 TI SW 24
PNN CORRESPONDS TO TTY172+NN

```

HIGH SEGMENTS:  
PROGRAM DEVICE OWNER HIGH(K) USERS

```

OPSER DSKB SYS 3 SW 2
DIRECT DSKB SYS 3 SW 1
QMANGR DSKB SYS 3 2
PIP DSKB SYS 4 SW 5
TECO DSKB SYS 3 12
EDITS DSKB SYS 6 SW 4
(PRV) JOB 35 34 1
QUEUE DSKB SYS 4 2
LIBOL DSKB SYS 8 4
QMOUNT DSKB SYS 5 SW 2
PTPSPL DSKB SYS 3 SW 1
QMOUNT DSKB SYS 3 4
KJOB DSKB SYS 3 SW 1
AID DSKB SYS 9 SW 1

```

```

SWAPPING SPACE USED = 156/635 = 25%
VIRT. CORE USED = 254/635 = 40%
SWAPPING RATIO = 254/144 = 1.8
ACTIVE SWAPPING RATIO = 75/144 = 0.5
VIRT. CORE SAVED BY SHARING = 115/(115+254) = 31%
AVERAGE JOB SIZE = 163/53 = 3.1 + 206/53 = 3.9 TOTAL = 369/53 = 7.0

```

(continued on next page)

**SYSTAT command (Cont)**

Examples (cont)

```

BUSY DEVICES:
DEVICE  JOB      WHY      LOGICAL
LPT0    23      AS+INIT
LPT1    21      AS+INIT
DTA1    14       AS
DTA2    19       AS
DTA3    33       AS
DTA4     7       AS
DTA6    27      AS+INIT
DTA7    26      AS+INIT
MTA1    37       AS
MTA2    35       AS
MTA3    31       AS
PTP0    24       INIT
72 DISK DDBS
    
```

```

SYSTEM FILE STRUCTURES:
NAME     FREE     MOUNT
DSKA     705      12
DSKB    17220    54
DSKC    35895     5
DIAG    2835     3
TOTAL FREE 56655
    
```

```

DATASET CONTROL
TTY#     STATUS
60       IN USE
61       IN USE
62       IN USE
64       IN USE
65       IN USE
72       IN USE
74       IN USE
    
```

• KJOB

(continued on next page)

SYSTAT command (Cont)

Examples (cont)

.SYSTAT P )

STATUS OF B50400-05 #40 AT 16:27:49 ON 11-APR-72

DISK PERFORMANCE STATISTICS:

UNIT OR F/S

| UNIT OR F/S   | BR        | BW          | DR         | DW            | XR            | XW         | MR                   | MW                   |
|---------------|-----------|-------------|------------|---------------|---------------|------------|----------------------|----------------------|
| DSKA          | 716       | FREE        |            |               |               |            |                      |                      |
| FHA1(1RD002): | 716       | FREE, 0     | SEEKS      |               |               |            |                      |                      |
|               | 6616      | 177         | 59861      | 8811          | 0             | 0          | 30062                | 27935                |
| MSB           | ERRORS: 1 | DAT:1       | RETRIES:1  | 2CONI:4000,15 | 1CONI:4000,40 | 15         | 2DATAI:21            | 1DATAI:140071        |
| DSKB          | 17215     | FREE        |            |               |               |            |                      |                      |
| DPA0(102446): | 4445      | FREE, 54617 | SEEKS      |               |               |            |                      |                      |
|               | 30498     | 39483       | 12648      | 1155          | 0             | 0          | 41622                | 10489                |
| MSB           | ERRORS: 1 | DAT:11      | RETRIES:1  | 2CONI:15      | 1CONI:5,40    | 15         | 2DATAI:54661,40000   | 1DATAI:54661,40000   |
| DPA1(117986): | 4345      | FREE, 42791 | SEEKS      |               |               |            |                      |                      |
|               | 37609     | 41724       | 17037      | 2790          | 0             | 0          | 20563                | 9505                 |
| MSB           | ERRORS: 1 | DEV:8       | 1DAT:240   | RETRIES:1     | 2CONI:15      | 1CONI:5,40 | 15                   | 2DATAI:102261,240000 |
| DPA2(102376): | 4170      | FREE, 33215 | SEEKS      |               |               |            |                      |                      |
|               | 35205     | 35413       | 14911      | 1236          | 0             | 0          | 15503                | 7687                 |
| MSB           | ERRORS: 1 | DEV:6       | RETRIES:3  | 2CONI:15      | 1CONI:400     | 15         | 2DATAI:456661,400000 | 1DATAI:456661,0      |
| DSKC          | 35925     | FREE        |            |               |               |            |                      |                      |
| DPA5(151669): | 35925     | FREE, 2863  | SEEKS      |               |               |            |                      |                      |
|               | 1705      | 337         | 250        | 1             | 0             | 0          | 4236                 | 1014                 |
| MSB           | ERRORS: 1 | DAT:2945    | FREE, 6808 | SEEKS         |               |            |                      |                      |
| DIAG          | 2945      | FREE, 6808  | SEEKS      |               |               |            |                      |                      |
| DPA6(DIAG01): | 2945      | FREE, 6808  | SEEKS      |               |               |            |                      |                      |
|               | 2133      | 2532        | 8423       | 533           | 0             | 0          | 8199                 | 2602                 |
| MSB           | ERRORS: 1 | DAT:2945    | FREE, 6808 | SEEKS         |               |            |                      |                      |

ACTIVE SWAPPING STATISTICS:

| UNIT | R      | W      | USED(K)       |
|------|--------|--------|---------------|
| FHA0 | 809992 | 366056 | 281/335 = 84% |
| FHA1 | 397720 | 351376 | 181/300 = 60% |

.KJOB

**SYSTAT command (Cont)**

Examples (Cont)

```

SYS (10,*1)
SYSTAT V467(5)
10      [SELF]  10      SYSTAT  5+SPY  RN      3
14      10,133  0       TECO    2+3   TI      47
    
```

```

SYS /0)
SYSTAT V467(5)

STATUS OF      50416A SYSTEM #2 AT 1:16:11 P.M. ON 24-FEB-72

UPTIME 45:47, 60% NULL TIME = 55% IDLE + 5% LOST
15 JOBS IN USE OUT OF 37.  15 LOGGED IN, 1 DETACHED

SWAPPING SPACE USED = 60/350 = 19%
VIRT. CORE USED = 85/350 = 24%
SWAPPING RATIO = 85/18 = 4.7
ACTIVE SWAPPING RATIO = 5/18 = 0.3
VIRT. CORE SAVED BY SHARING = 5/(5+85) = 6%
AVERAGE JOB SIZE = 49/15 = 3.3 + 41/15 = 2.7  TOTAL = 90/15 = 6.0
    
```

**TECO command 1**

Function

The TECO command runs TECO and opens an already existing file on disk for editing. Refer to the TECO manual in the DECsystem-10 Software Notebooks.

Command Format

TECO dev:file.ext [proj,prog]

dev: = the device or file structure name containing the existing file. If omitted, DSK: is assumed.

file.ext = the filename and filename extension of the existing file. If omitted, the arguments of the last EDIT-class command are used.

[proj,prog] = the directory name in which the file appears. If omitted, the user's directory is assumed.

Characteristics

The TECO command:

Places the terminal in user mode.  
Runs the TECO program, thereby destroying the user's core image.  
Depends on FTCCLX which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

Example

```
._TECO TEST1.MAC )  
_*C  
._TECO DSKB:FILNAM.CBL [100,27] )
```

---

<sup>1</sup>This command runs the COMPIL program, which interprets the commands before running TECO.

**TIME command**Function

The TIME command causes typeout of the total running time since the last TIME command, followed by the total running time used by the job since it was initialized (logged-in), followed by the integrated product of running time and core size (KILO-CORE-SEC=). Time is typed in the following format:

hh:mm:ss.hh

where

hh = hours

mm = minutes

ss.hh = seconds to nearest hundredth.

Interrupt level and job scheduling times are charged to the user who was running when the interrupt or rescheduling occurred.

**NOTE**

If automatic runtime is enabled using the SET WATCH command, the incremental runtime is usually 0.

Command Format

TIME job

job = the job number of the job whose timing is desired. If job is omitted, the job to which the terminal is attached is assumed. In this case, monitor types out the incremental running time (running time since last TIME command) as well as the total running time since the job was initialized.

Characteristics

The TIME command:

Leaves the terminal in monitor mode.  
Does not require LOGIN.

Associated Messages

Refer to Chapter 4.

**TIME command (Cont)**

Example

```

.TIME)
0.38
0.38
KILO-CORE-SEC=1

```

The command is given for the first time after LOGIN; therefore, the incremental time equals the total time since LOGIN.

```

.TI)
0.00
0.38
KILO-CORE-SEC=1

```

```

.DIR/F)
NEW505 BAK      DSKB:  [10,770]
NEW505 RNO
FOO    SFD
C      MAC      DSKC:
PETALS F4
PETALS SAV
CSHELL MAC
CIO    MAC
TRYCIO MAC
METER  RNO
SURFIT ALG

```

```

.TI)
0.40
0.78
KILO-CORE-SEC=6

```

The DIRECT command took .40 seconds of runtime and 5 kilo-core-seconds.

## TPUNCH command

### Function

The TPUNCH command is used to place entries into the paper-tape punch output queue. This command is equivalent to the following form of the QUEUE command:

QUEUE PTP: jobname = list of input specifications.

The TPUNCH command can be further abbreviated to

PUNCH jobname = list of input specifications.

However, individual installations may redefine PUNCH to mean output to the card-punch queue instead of the paper-tape punch queue.

### Command Format

TPUNCH jobname = list of input specifications

jobname = name of the job being entered into the queue. The default is the name of the first file in the request not the name of the first file given. These differ when the first file given does not yet exist.

input specifications = a single file specification or a string of file specifications, separated by commas, for the disk files being processed. A file specification is in the form dev:file.ext [proj,prog].

dev: = any file structure to which PTPSPL will have access; the default is DSK:.

file.ext = names of the files. The filename is optional. The default for the first filename is \*, the default for subsequent files is the last filename used. The extension can be omitted; the default is .PTP.

[proj,prog] = a directory to which the user has access; the user's directory is assumed if none is specified.

The wildcard construction can be used for the input specifications.

If no arguments appear in the command string (i.e., only the command name is given), all entries in the paper-tape punch queue for all jobs are listed.

Switches that aid in constructing the queue entry can also appear as part of the input specifications. These switches are divided into three categories:

1. Queue-operation - Only one of these switches can be placed in the command string because they define the type of queue request. The switch used can appear anywhere in the command string.

(continued on next page)

**TPUNCH command (Cont)**

Command Format (cont)

- 2. General - Each switch in this category can appear only once in the command string because they affect the entire request. The switch used can appear anywhere in the command string.
- 3. File control - Any number of these switches can appear in the command string because they are specific to individual files within the request. The switch used must be adjacent to the file to which it applies. If the switch precedes the file-name, it becomes the default for subsequent files.

The following switches can be used with the TPUNCH command.

| <u>Switch</u> | <u>Explanation</u>                                                                                                                                                                                                                                                                                   | <u>Category</u> |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| /AFTER:tt     | Process the request after the specified time; tt is either in the form of hhmm (time of day) or +hhmm (time later than the current time). The resulting AFTER time must be less than the DEADLINE time. If the switch, or the value of the switch, is omitted, no AFTER constraints are assumed.     | General         |
| /BEFORE:t     | Queue only the files with a creation date before time t where t is in the form dd-mmm-yy.                                                                                                                                                                                                            | General         |
| /BEGIN:n      | Start the output on the nth foot of tape. The default is to begin output on the first foot.                                                                                                                                                                                                          | File Control    |
| /COPIES:n     | Repeat the output the specified number of times. N must be less than 64. If more than 63 copies are needed, two separate requests must be made. If this switch is omitted, one copy is made.                                                                                                         | File Control    |
| /CREATE       | Make a new entry into the paper-tape output queue. This switch is the default for the queue-operation switches.                                                                                                                                                                                      | Queue Operation |
| /DEADLINE:tt  | Process the request before the specified time; tt is either in the form hhmm (time of day) or +hhmm (time later than the current time). The resulting DEADLINE time must be greater than the AFTER time. If the switch, or the value of the switch, is omitted, no DEADLINE constraints are assumed. | General         |

(continued on next page)

**TPUNCH command (Cont)**

Command Format (cont)

| <u>Switch</u>     | <u>Explanation</u>                                                                                                                                                                                                                     | <u>Category</u> |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| /DISPOSE:DELETE   | Delete the file after spooling.                                                                                                                                                                                                        | File Control    |
| /DISPOSE:PRESERVE | Save the file after spooling. This is the default of all files except files with extensions of .LST, .TMP, and if the protection is 0xx, .PTP.                                                                                         | File Control    |
| /DISPOSE:RENAME   | Rename the file from the specified directory immediately, remove it from the logged-out quota, and delete it after spooling. If omitted, this is the default for files with extensions .LST, .TMP, and if the protection is 0xx, .PTP. | File Control    |
| /F                | List the entries in the paper-tape punch queue, but do not update the queues. Therefore, the list may not be an up-to-date listing, but the listing will be faster than with /LIST.                                                    | Queue Operation |
| /FILE:ASCII       | Indicate that the file format is ASCII text. This is the default.                                                                                                                                                                      | File Control    |
| /FILE:ELEVEN      | Indicate that the file format is MACX11 binary format.                                                                                                                                                                                 | File Control    |
| /KILL             | Remove the specified entry from the paper-tape punch queue. This switch can be used for deleting a previously submitted request as long as the request has not been started by the paper-tape punch spooler.                           | Queue Operation |

(continued on next page)

**TPUNCH command (Cont)**

Command Format (cont)

| <u>Switch</u> | <u>Explanation</u>                                                                                                                                                                                                                  | <u>Category</u> |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| /LIMIT:n      | Limit the output to the specified number of feet. The default is 10*B+20 feet, where B is the number of blocks in the request.                                                                                                      | General         |
| /LIST         | List the entries in the paper-tape punch queue; if the switch, along with all other switches, is omitted, all entries for all jobs of all users are listed.                                                                         | Queue Operation |
| /MODIFY       | Alter the specified parameters in the job. This switch requires that the user have access rights to the job. It can be used for altering a previously submitted request as long as the request has not been started by the spooler. | Queue Operation |
| /NEW          | Accept the request even if the file does not yet exist.                                                                                                                                                                             | File Control    |
| /NOTE:a       | Punch the specified text (a) in the output.                                                                                                                                                                                         | File Control    |
| /NULL         | Accept the request even if there is nothing in the request. No error message is given.                                                                                                                                              | General         |
| /OKNONE       | Do not output message if no files match the wild-card construction. This is assumed at KJOB time.                                                                                                                                   | File Control    |
| /PHYSICAL     | Suppress logical device name assignments for the device specified.                                                                                                                                                                  | File Control    |
| /PRIORITY:n   | Assign the specified external priority (n = 0 to 62) to the request. The larger the number, the greater priority the job has. The default is 10 if no switch is given and 20 if the switch is specified without a value.            | General         |
| /PROTECT:nnn  | Assign the protection nnn (octal) to the job. If the switch, or the value of the switch, is omitted, the standard protection is assumed.                                                                                            | General         |
| /REMOVE       | Remove the file from the queue. This switch is valid only with the /MODIFY switch and can be used to remove a previously submitted file as long as the spooler has not started processing the request.                              | File Control    |

(continued on next page)

**TPUNCH command (Cont)**

Command Format (cont)

| <u>Switch</u> | <u>Explanation</u>                                                                                                                        | <u>Category</u> |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| /SEQ:n        | Specify a sequence number to help identify a request to be modified or deleted.                                                           | General         |
| /SINCE:t      | Queue only the files with creation dates after the specified time t where t is in the form dd-mmm-yy.                                     | General         |
| /START:n      | Begin on the nth line of the file. If the switch, or the value of the switch, is omitted, the spooler starts with the first line.         | File Control    |
| /STRS         | Search for the file on all file structures in the search list and take each occurrence. The default is to take just the first occurrence. | File Control    |
| /TAPE:ASCII   | Punch the tape in ASCII code. If the /TAPE switch is not specified, the file is punched according to the data mode of the file.           | File Control    |
| /TAPE:BINARY  | Punch the tape in binary mode. If the /TAPE switch is not specified, the file is punched according to the data mode of the file.          | File Control    |
| /TAPE:IBINARY | Punch the tape in image-binary mode. If the /TAPE switch is not specified, the file is punched according to the data mode of the file.    | File Control    |
| /TAPE:IMAGE   | Punch the tape in image mode. If the /TAPE switch is not specified, the file is punched according to the data mode of the file.           | File Control    |
| /UNPRESERVED  | Output the files only if they are not preserved (i.e., the first digit is 0). This switch avoids redundant printing.                      | General         |

Characteristics

The TPUNCH command:

- Leaves the terminal in monitor mode.
- Runs the QUEUE program, thereby destroying the user's core image.
- Depends on FTQCOM which is normally absent in the DECsystem-1040.

**TPUNCH command (Cont)**

Associated Messages

Refer to Chapter 4.

Examples

```
.TPUNCH TENDMP.RFL/TAPE: BINARY/COPIES:5 )
```

Punch 5 copies, in binary mode, of the file DSK:TENDMP.REL.

## TYPE command <sup>1</sup>

### Function

The TYPE command directs PIP to type the contents of the named source file(s) on the user's terminal.

To stop the typing, type !C twice.

### Command Format

TYPE list

list = a single file specification or a string of file specifications separated by commas. The filename is required. The extension is required if the filename has an extension.

In addition, the full wildcard construction can be used.

Switches can be passed to PIP by enclosing them in parentheses in the TYPE command string. When COMPIL interprets the command string, it passes the switches on to PIP.

### Characteristics

The TYPE command:

Leaves the terminal in monitor mode.

Runs the PIP program, thereby destroying the user's core area.

Depends on FTCLX which is normally absent in the DECsystem-1040.

### Associated Messages

Refer to Chapter 4.

### Examples

```
.TYPE FILEA,DTA0:FILEB.MAC )
. TYPE *.TMP,DTA4:C )
```

---

<sup>1</sup>This command runs the COMPIL program, which interprets the command before running PIP.

**UNLOAD command <sup>1</sup>**

Function

The UNLOAD command rewinds and unloads a magnetic tape or a DECtape. This command is equivalent to the following PIP command string:

dev: (MU) ←

Command Format

UNLOAD dev:

dev: = a magnetic tape (MTAn) or a DECtape (DTAn).

Characteristics

The UNLOAD command:

- Leaves the terminal in monitor mode.
- Runs the PIP program, thereby destroying the user's core image.
- Depends on FTCCLX which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

Examples

.\_UNLOAD DTA7: )

.\_UNL MTA3: )

---

<sup>1</sup>This command runs the COMPIL program, which interprets the command before running the PIP program.

## VERSION command

### Function

The VERSION command prints the version number of the program in the user's core area (i.e., the last program run implicitly or explicitly). The version number is obtained from .JBVER and .JBHVR in the job data area and is printed in standard format. The output from this command is in one of the following representations:

|                     |                                                                                    |
|---------------------|------------------------------------------------------------------------------------|
| low + high          | The low and high segments are different.                                           |
| low                 | There is only a low segment.                                                       |
| low +               | The low and high segments are the same.                                            |
| + <sup>1</sup>      | A GETSEG UUO has been done to a high segment which matches the low segment.        |
| + high <sup>1</sup> | A GETSEG UUO has been done to a high segment which does not match the low segment. |
| blank <sup>1</sup>  | The high segment has been released.                                                |

With the VERSION command, the low and high segments are represented in the format

name version

With the SET WATCH VERSION command, the low and high segments are represented in one of three formats:

|                |                                                                         |
|----------------|-------------------------------------------------------------------------|
| name version   | The program is not from SYS:                                            |
| :name version  | The output is the result of a SETNAM UUO (e.g., at the end of loading). |
| S:name version | The program is a system program (not logical device SYS:).              |

The name is a SIXBIT name and the version is in standard format. When printing the version number, the standard format is:

major version minor version (edit) - group who modified program last

<sup>1</sup>Output only from the SET WATCH VERSION command.

**VERSION command (Cont)**

Function (cont)

The major version is octal; the minor version is alphabetic; the edit is octal and enclosed in parentheses; and the group who last modified the program is octal and preceded by a hyphen (0 = DEC development, 1 = all other DEC personnel, and 2-7 = customer use). There are no spaces separating the items, and if an item is zero, it does not appear in print. The parentheses and hyphen also do not appear in print if the corresponding item is zero. The following are examples of version numbers output in standard format.

- 10B(335)-1      major version 10, minor version B, edit number 335, group that modified program last 1.
- 7(5)            major version 7, minor version 0, edit number 5, group that modified program last 0.
- 54A             major version 54, minor version A, edit number 0, group that modified program last 0.

Command Format

VERSION

Characteristics

The VERSION command:

- Leaves the terminal in monitor mode.
- Depends on FTVERS which is normally absent in the DECsystem-1040.

Examples

```

.R TECO)
*!C
.VERSION)
TECO 22(64) +
.TYPE SAMPL.TXT)
THIS IS A TEXT FILE
.VERSION)
PIP 31(35) +

```

## WHERE command

### Function

The WHERE command enables the user to determine the station at which a specific peripheral device is located. If the station of a particular terminal is requested, the number returned is the physical location of the terminal which may or may not be the location of the controlling job. This depends on whether the user changed his job's logical location with the LOCATE command.

### Command Format

WHERE devn

dev = any physical device name and n is the unit number.

### Characteristics

The WHERE command:

Leaves the terminal in monitor mode.

Does not require LOGIN.

Depends on FTREM which is normally absent in the DECsystem-1040.

### Associated Messages

Refer to Chapter 4.

### Examples

|                              |                                               |
|------------------------------|-----------------------------------------------|
| <pre> 1  WHERE CDR2;) </pre> | <pre> ;station of CDR2. </pre>                |
| <pre> 2  WH TTY;) </pre>     | <pre> ;station of job's terminal. </pre>      |
| <pre> 4  WHE DPR;) </pre>    | <pre> ;station of job issuing command. </pre> |
| <pre> 1  WHERE CTY;) </pre>  | <pre> ;central station. </pre>                |

**ZERO command <sup>1</sup>**

Function

The ZERO command clears the directory of the output device. This command is equivalent to the following PIP command string:

dev: /Z ←

Command Format

ZERO dev:

dev: = a DECTape (DTAn) or a disk (DSK). This argument is required.

A directory name can be specified with ZERO DSK: and if the user has access to the specified directory, the directory is zeroed. If no directory is specified, the user's directory is assumed.

Characteristics

The ZERO command:

- Leaves the terminal in monitor mode.
- Runs the PIP program, thereby destroying the user's core image.
- Depends on FTCCLX which is normally absent in the DECsystem-1040.

Associated Messages

Refer to Chapter 4.

Examples

```
._ZER DTA4: )  
._ZERO DSK: )  
._ZER DSK: [27,40] )
```

---

<sup>1</sup>This command runs the COMPIL program, which interprets the command before running the PIP program.



## CHAPTER 3

# BATCH SYSTEM COMMANDS

The Batch System, operating under the control of the DECsystem-10 Operating System, increases system throughput by processing jobs that do not require human interaction. Types of jobs best suited for a batch environment are: large and long-running jobs, jobs that require large amounts of data, frequently run production jobs, and jobs that require little or no interaction with the user. Up to 14 Batch jobs can be processed concurrently without adversely affecting the running of timesharing jobs. Batch jobs may be entered from

1. Local devices
2. Remote devices
3. Interactive terminals.

### 3.1 BATCH COMPONENTS

The Batch System consists of a group of programs; some are used for Batch operations only, others are available for various operations of the total computing system.

The individual Batch components are: the Stacker, CDRSTK; the Queue Manager, QMANGR; the Batch Controller, BATCON; and the output spoolers, LPTSPL (line printer), CDPSPL (card punch), PLTSPL (plotter), and PTPSPL (paper-tape punch).

#### 3.1.1 The Stacker

The Stacker, CDRSTK, is responsible for

1. reading a sequential input stream from an input device,
2. separating the input by placing it in files according to the control cards contained in the input stream,
3. creating the job's log file and entering a report of its processing, and
4. entering the job into the Batch input queue.

When input is from the card reader, CDRSTK accepts ASCII, binary, 026, and DEC-029 Hollerith code. (Refer to Appendix B for tables of card codes.) The input is read in image mode, and CDRSTK converts it to one of the mentioned codes. If input is from any other device, only ASCII code is accepted.

CDRSTK creates three types of files during its copying of the input data: the user's data files, the Batch control file, and the job's log file. The data files are created according to the control cards in the input and are placed into the user's disk area. Programs and data are copied into these files and are passed to the job, while it is running, by the Batch Controller. Refer to Paragraph 3.3 for the description of the control cards that cause CDRSTK to copy information into these files.

A user control file is created for each valid job and is subsequently processed by the Batch Controller. This file contains all monitor level and user level commands encountered in the input. CDRSTK also enters commands resulting from the processing of certain control cards and any information that does not follow specific control card format. The control file is placed in the user's disk area. Refer to Paragraph 3.4 for a description of the Batch Controller commands that can be entered into the control file.

The job's log file contains a report of the CDRSTK's processing, along with a record of any operator intervention during its operation. This file is in the user's disk area along with the other CDRSTK-created files and is deleted after it is printed by the line printer spooler.

### 3.1.2 The Queue Manager

The Queue Manager, QMANGR, is the program that schedules jobs and maintains system queues. When CDRSTK finishes processing a job, it makes an entry into the Batch input queue. The Queue Manager computes and dynamically revises priorities for the job and notifies the Batch Controller when the job is to be run. Jobs are scheduled for running according to the parameters pertaining to each job and to the priorities established by the system. While the job is running, its queue entry is flagged to show it is in use, but the entry is not deleted from the queue until the job terminates. When the job is logged off the system, an output queue entry is usually made and the entry in the input queue is deleted. The Queue Manager again schedules the job's output and deletes the job's output queue entry only when the output is completely finished.

### 3.1.3 The Batch Controller

The Batch Controller, BATCON, controls all jobs entered into the Batch System. It reads the control file created by CDRSTK or the user and initiates and controls the running of the job by passing data and system program commands directly to it.

Monitor commands are examined by the Batch Controller and passed to the monitor for action. The Controller determines the destination of commands by interpreting the character in column 1 in each line of the control file. If column 1 contains a space or a tab, the spaces are ignored until a nonspace character is encountered. If column 1 contains an alphabetic or numeric character, the line is either at monitor command level or at user command level. If column 1 contains a special character, the Batch Controller interprets the line as follows:

\$ (dollar sign) - The interpretation depends upon the character in column 2.

If column 2 contains an alphabetic character, the line is copied to the log file as a comment because it is a Stacker control line and has already been processed.

If column 2 contains a numeric or special character, the line is treated as data.

If column 2 contains a dollar sign (\$), the initial dollar sign is suppressed and the line is treated as data.

If column 2 contains a line feed, vertical tab, or form feed, a blank line is entered into the log file.

(period) - The interpretation depends upon the character in column 2.

If column 2 contains an alphabetic character, the line is treated as a monitor command and the period is suppressed.

If column 2 contains a nonalphabetic character, the line is treated as data with the period as part of the data.

\* (asterisk) - The line is treated as a user-level command or program data and the asterisk is suppressed. This is the standard input data method for most system programs.

= (equal sign) - The line is treated as a user level command or program data. The equal sign is suppressed and final spaces and the end of the line are suppressed (i.e., not passed to the program). This line normally indicates a DDT or TECO command because these commands terminate with special characters rather than the end of the line and would not function properly if the end of the line were passed.

; (semicolon) - The line is treated as a comment to the log file.

% (percent sign) - The line is treated as part of a command level statement label. The percent sign is normally reserved for DEC use. If % is encountered when the job has had no error, the control file is advanced, unless a %FIN is encountered. In this case, the %FIN is executed. Refer to the discussion of the .IF command in Paragraph 3.4.5.

The Batch Controller does not examine the contents of any lines in the control file other than those destined for the monitor. However, when it encounters an up-arrow (^), it converts the up-arrow as follows:

If the character following the up-arrow is a numeric character, the up-arrow and the digit are passed to the job.

If the character following the up-arrow is an alphabetic character, the up-arrow and the character are translated to a control character; e.g., ^A is translated to CTRL-A.

(continued on next page)

If the character following the up-arrow is another up-arrow, the first up-arrow is ignored and the second up-arrow is treated as an up-arrow; e.g., ↑↑A is treated as ↑A (up-arrow A) and ↑↑↑A is treated as ↑↑A (up-arrow up-arrow A).

If the job is requesting input and is at monitor level, the control file is read until a command or intermediate level line is found. If a job is requesting input at data level and the next line is a monitor command, the Batch Controller inserts a control-C.

A Batch user may not issue the following monitor commands when his job is operating in batch mode: ATTACH, DETACH, CCONT, CSTART, and SEND. If these commands are used, the line is suppressed and flagged at BATERR in the log file and the job is continued. All other monitor commands and system program commands may be used by a job operating in batch mode.

The Batch Controller makes entries to the log file to record its processing of the control file and the job.

#### 3.1.4 The Output Spoolers

The output spoolers receive job output that has been placed into the output queues by the Queue Manager. Usually a job's output is placed in a line printer queue to be printed at a later time by the LPTSPL spooling program at the same station from which the input was received. The output filenames are in the form QxxSnn.LPT, where xx is a random number, and nn is the station number of the printer where the job is currently located. However, the user can also specify other output devices either in his programs within his job or by means of the QUEUE monitor command in his job. The first method causes output to the card punch, paper-tape punch, or plotter to be automatically spooled by the system. The second specifies nonstandard output spooling to any of the spooling devices.

### 3.2 SUBMITTING JOBS

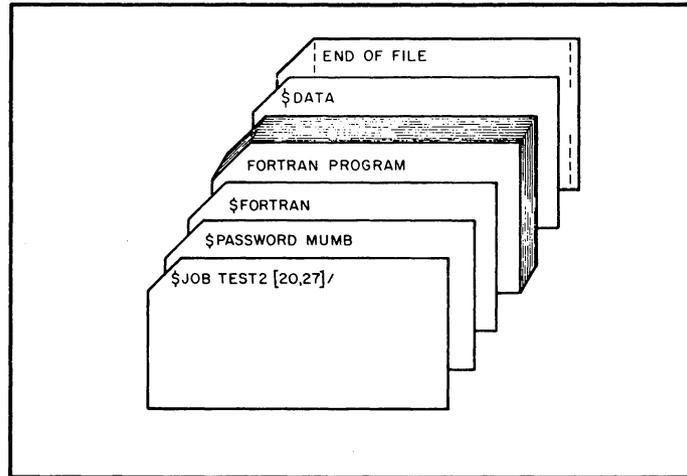
A job is a unit that consists of one step or a group of steps. It can contain (1) a single program and its related data, or several programs and their data, and (2) the monitor and user-level commands that are required to control the programs.

The Batch system allows the user to submit his job by one of the following three methods:

1. The user punches his job on cards, inserts control cards to CDRSTK, and leaves his cards at the designated place for the operator to run (refer to Paragraph 3.2.1).
2. The user creates his job as a file for input to CDRSTK (instead of having his job on cards) and then runs CDRSTK himself (refer to Paragraph 3.2.2).
3. The user bypasses CDRSTK by creating his own control file on disk for the Batch Controller and then enters his job into the Batch input queue from his terminal (refer to Paragraph 3.2.3).

### 3.2.1 Submitting a Job with Cards

With this method, a job is submitted via a deck of cards, bounded by the control cards that mark its beginning and end. Other control cards to CDRSTK are interspersed among the card deck to direct CDRSTK's processing. Figure 3-1 shows a job containing the appropriate control cards to CDRSTK. This job compiles, loads, executes, and lists a FORTRAN program.



10-0729

Figure 3-1 Typical Job on Cards

3.2.1.1 The \$JOB Card - This card notifies CDRSTK that a job is to be processed. CDRSTK creates a control file into which commands are placed for the Batch Controller and a log file on the disk. The first argument (TEST2) shown on this card is the user-assigned name for the job; the second argument ([20,27]) is the project-programmer number of the user. For a description of switches which can be used on this card, refer to Paragraph 3.3.9.

3.2.1.2 The \$PASSWORD Card - This card contains the PASSWORD associated with the project-programmer number specified on the \$JOB card. In Figure 3-1, the PASSWORD is MUMB, which was assigned to the user by the system manager. Refer to Paragraph 3.3.12 for more information on the \$PASSWORD card.

3.2.1.3 The \$FORTRAN Card - This card causes CDRSTK to insert a COMPILE monitor command (refer to Chapter 2) into the control file in order to cause the program to be compiled. Immediately following the \$FORTRAN card is the FORTRAN source program to be compiled. The source program is read into a disk file with the specified filename (or a default name if a filename is not given) and with an extension of .F4. Refer to Paragraph 3.3.8 for more information on the \$FORTRAN card.

3.2.1.4 The \$DATA Card - The card after the FORTRAN program is the \$DATA card. This card causes CDRSTK to insert an EXECUTE monitor command (refer to Chapter 2) into the control file in order to load and then execute the previously compiled program. Refer to Paragraph 3.3.3 for additional information on this card.

3.2.1.5 The End of File Card - The last card shown in the example is the end-of-file card. This card signals the end of the job. The card is recognized by CDRSTK as the end of the file because of the punches in rows 12, 11, 0, 1, 6, 7, 8, 9 in columns 1 and 80 of the card. Refer to Paragraph 3.3 for more information about this card.

3.2.1.6 Output - Once the program is punched on cards, the card deck is submitted to the operator, who in turn stacks the job in the card reader. The user receives his output in the form of line printer listings. Refer to Paragraph 3.5 for an explanation of the job output.

The CDRSTK control cards shown in Figure 3-1 are just a few of the control cards available to the user. For a complete description of all the CDRSTK control cards, refer to Paragraph 3.3.

### 3.2.2 Submitting a Job with a File

With this method, a job is submitted via a file contained on any input device that supports ASCII code. This file contains the program and data with card images of the control cards for CDRSTK. The following example shows the creation of a disk file containing a FORTRAN program and card images of CDRSTK control commands. Note that it corresponds to the card example in Paragraph 3.2.1.

```

.LOGIN 20,27)
JOB17 SSC4 TTY11
PASSWORD:
1020 15-MAK-72 WED
.MAKE JOB)
*I $JOB TEST2, ([20,27])
$FORTRAN)
C FORTRAN PROGRAM GOES HERE
$EOD)
$DATA)
$$
*EX$$
:

```

3.2.2.1 Image of the \$JOB Card - The first line of the file is an image of the \$JOB card. Note that the \$ character must be the first character of the line in order for CDRSTK to recognize it as a control command. This line causes a control file and a log file to be created on the disk when CDRSTK is run. The first argument (TEST2) is the user-assigned name for the job; the second ([20,27]) is the

project-programmer number of the user. The \$PASSWORD card image is not needed because the user is already logged-in when creating the input file. For additional information on the \$JOB card, refer to Paragraph 3.3.9.

3.2.2.2 Image of the \$FORTRAN Card - This line causes CDRSTK to insert a COMPILE monitor command (refer to Chapter 2) into the control file in order to compile the program. The source program follows immediately and is read into a disk file with the specified filename (or a default name if a filename is not given) and with an extension of .F4. Refer to Paragraph 3.3.8 for more information on the \$FORTRAN card.

3.2.2.3 Image of the \$EOD Card - This line indicates to CDRSTK the end of the FORTRAN program. Refer to Paragraph 3.3.6 for more information.

3.2.2.4 Image of the \$DATA Card - This line causes CDRSTK to insert an EXECUTE monitor command (refer to Chapter 2) into the control file in order to load and execute the program.

3.2.2.5 Running CDRSTK - Once the file is created and CDRSTK is run by the user, it processes the user-created file in the same manner as it processes input files of jobs entered directly by the operator. The user runs CDRSTK by typing

```
␣R CDRSTK ␣
```

CDRSTK responds with an asterisk, and then the user types in the following command

```
␣START dev:file.ext ␣
```

where dev: is the name of the device containing the input file for CDRSTK and file.ext is the name of the file. Using the above file, the command is

```
␣START DSK:JOB ␣
```

and CDRSTK responds with

```
!
```

When CDRSTK has completed its processing (i.e., when it has created the control and log files and has entered the job into the Batch input queue), it responds with

```
READY
```

```
*
```

indicating its readiness to accept another file. At this point, the user can enter another file or return to monitor mode with a `TC`.

The card images shown in the preceding example are only a few of the `CDRSTK` control card images available. Refer to Paragraph 3.3 for a complete description of all of the control cards.

### 3.2.3 Submitting a Job with a Control File to the Batch Controller

With this method, a job is submitted via the steps within a control file to the Batch Controller. The file must be a disk file and is created with a system editor. Since this file is processed directly by the Batch Controller, control card images are not used. The control file consists of monitor commands, user program commands, comments, and sequence control statements. Refer to Paragraph 3.4 for a description of control file commands. The following is an example of creating a control file. It assumes that a file named `DATA.F4` already exists on disk.

```

_MAKE JOB.CTL )
*I.EXECUTE /COMPILE DATA.F4 /LIST )
$$
EX$$

```

Once the control file is created, the user can enter the job into the Batch input queue one of three ways:-

1. `SUBMIT jobname = control file, log file`  
refer to the `SUBMIT` command in Chapter 2.
2. `QUEUE INP: jobname = control file, log file`  
refer to the `QUEUE` command in Chapter 2.
3. `R QUEUE`  
refer to the `QUEUE` specification in Notebook 7 of the DECsystem-10 Software Notebooks.

### 3.2.4 Interjob Dependency

Jobs are not necessarily run in the order that they are read into the Batch System. Priorities stipulated by the user on the `$JOB` card (refer to Paragraph 3.3.9) and additional parameters set by the Batch System are dynamically computed by the Queue Manager to determine in what order the jobs are run. However, it is often useful to submit several jobs that must be run in a specific order, for example, one job updates a master file and another job processes it. Therefore, the running of one job is dependent upon the running of the other. Although these jobs could be combined into one large job, it is sometimes necessary to keep them distinct; i.e., they might be submitted by different people at different

times. Because the jobs in the Batch System are run in order of priority, the user specifies an additional priority, an initial dependency count, on the \$JOB card of the dependent job. This dependency count becomes part of the queue entry. Any input queue entry that has a dependency count greater than zero cannot be scheduled. When the count becomes zero, the job is scheduled, based upon the time it was submitted and the time that the dependency count became zero. If the dependency count becomes negative, an advisory message is sent to the issuing job and to the dependent job. The dependency count can be altered by including the QUEUE command as part of any job upon which the dependent job is waiting. (Refer to the QUEUE monitor command.) The QUEUE command switch that allows the user to change the dependency count of another job is the /MODIFY/DEPEND:nn switch. If the user specifies a plus or minus sign before the count (nn), that number is subtracted from or added to the dependent job's count. If the user does not specify a sign, the dependent job's count is changed to the count specified in the /MODIFY/DEPEND: switch.

### 3.3 CDRSTK CONTROL CARDS

Control cards are interspersed among the input stream to aid CDRSTK in separating the input into the appropriate files, either the user's data files or the control file processed by the Batch Controller. The control cards contain a dollar sign (\$) in column 1 and an alphabetic character in column 2. These are the only cards read and interpreted by CDRSTK; the remainder of the input is separated and placed into the appropriate file. Note that if the user creates his own control file, he bypasses CDRSTK, and, therefore, does not use these control cards.

Only the first part of the command name or switch need be specified; as long as the name is unique, it is accepted. The first three characters of a command name are generally sufficient to ensure uniqueness. The standard comment and continuation conventions for the system can be used on the control card. A comment is preceded by a semicolon; characters after the semicolon to the end of the card are treated as comments. A card may be continued by placing a hyphen as the last non-TAB or non-space character before the end of the card. Comments beginning with a semicolon, TABs, and spaces can follow. All defaults for control card parameters are installation parameters.

The end-of-file is used to signal the end of the job. When input is from the card reader, an end-of-file card is used. Column 1 of this card has rows 4 and 5 blank and rows 6, 7, 8, and 9 punched. The recommended form of this card has columns 1 and 80 containing punches in rows 12, 11, 0, 1, 6, 7, 8, 9, with rows 2, 3, 4, and 5 blank, so that the card can be recognized in any orientation. When devices other than the card reader are used for input, the standard end-of-file for each device is treated by CDRSTK as the end of the job.

3.3.1

**\$ALGOL**

Function

This card causes CDRSTK to copy the named ALGOL program onto disk and to insert a COMPILE monitor command into the control file. The card is placed at the beginning of the source program. When the job is run, the specified program is compiled and temporary relocatable binary and listing files are created. The binary and listing files can be made permanent if the user renames them to change their protection. The source file can be preserved by means of the /PROTECT switch. The listing file is printed as part of the job's output.

Processor switches can be passed to the ALGOL compiler by including them in the command string. The position of these switches in the command determines their position in the COMPILE command generated by CDRSTK. For example

\$ALGOL /NOLIST (E,,N)

results in the following COMPILE command

.COMPILE /COMPILE DECKAA.ALG /NOLIST (E,,N)

Refer to Paragraph 1.5.7 for a description of the ALGOL processor switches.

Card Format

\$ALGOL dev:name.ext [proj,prog] (processor switches) /S<sub>1</sub>/S<sub>2</sub>.../S<sub>n</sub>

dev: = a file structure name. If omitted, DSK is assumed.

name.ext = the name of the file to be created on disk. If omitted, CDRSTK assigns a unique filename of the form DECKaa (where aa = AA through ZZ) with the extension .ALG. It is recommended, however, that the user select a distinct filename for each job that is in the Batch system simultaneously.

[proj,prog] = a directory name other than that specified on the \$JOB card. If omitted, the project-programmer number on the \$JOB card is used.

(processor switches) = the switches to be passed to the ALGOL compiler. They must be enclosed in parentheses and the slash cannot appear in connection with these switches.

/S<sub>1</sub>/S<sub>2</sub>.../S<sub>n</sub> = the switches that control the mode of input interpretation and the listing of the compiled program.

| <u>Switch</u> | <u>Meaning</u>                                                                                                                                         | <u>Default</u> |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| /ASCII        | The input is read in ASCII mode.                                                                                                                       | on             |
| /D029         | The card deck is read in the old DEC-029 format. This format is similar to ASCII and is available only in those installations that use DEC-029 format. | off            |
| /LIST         | A temporary listing file of the program is created.                                                                                                    | on             |

(continued on next page)

Card Format (cont)

| <u>Switch</u>           | <u>Meaning</u>                                                                                                                                                                                                                        | <u>Default</u>                                     |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------|
| /NOLIST                 | No listing file of the program is created.                                                                                                                                                                                            | off                                                |
| /PROTECT:nnn            | The protection to be set for the file (in octal).                                                                                                                                                                                     | The file is preserved only until KJOB for the job. |
| /SUPPRESS:<br>ON or OFF | When ON is specified, trailing blanks are suppressed. They are not suppressed when OFF is specified.                                                                                                                                  | on                                                 |
| /WIDTH:nn               | The maximum number of columns to be read. If the specified width is less than 80, only that number of columns is read. The remaining columns are treated as blank. Normally this switch is only used when the /SUPPRESS switch is on. | 80                                                 |
| /026                    | The card deck is read in 026 card code.                                                                                                                                                                                               | off                                                |

Restrictions

The /026 and /D029 switches apply only to card reader input. Input from other devices must be read in ASCII code; otherwise an error message is written in the log file and the job is terminated.

3.3.2

\$COBOL

Function

This card causes CDRSTK to copy the specified COBOL program onto disk. The card is placed at the beginning of the source program, and when CDRSTK reads the card, it inserts a COMPILE monitor command into the control file and copies the COBOL program into the file on the specified disk area. When the job is run, the program is compiled and a temporary relocatable binary file and a temporary listing file are created. The binary and listing files can be made permanent if the user renames them to change their protection. The source file can be preserved if the user specifies the /PROTECT switch. The listing file is printed as part of the job's output.

Processor switches can be passed to the COBOL compiler by including them in the command string. The position of these switches in the command determines their position in the COMPILE command generated by CDRSTK.

For example

\$COBOL (A,M,C) /PROTECT:057

results in the following COMPILE command

.COMPILE /COMPILE DECKAB.CBL (A,M,C) /LIST

Refer to Paragraph 1.5.7 for a description of the COBOL processor switches.

Card Format

\$COBOL dev:name.ext [proj,prog] (processor switches) /S<sub>1</sub>/S<sub>2</sub>.../S<sub>n</sub>

dev: = a file structure name. If omitted, DSK is assumed.

name.ext = the name of the file to be created on disk. If omitted, CDRSTK assigns a unique filename of the form DECKaa (where aa = AA through ZZ) with the extension .CBL. It is recommended, however, that the user select a distinct name for each job in the Batch system simultaneously.

[proj,prog] = a directory name other than that specified on the \$JOB card. If omitted, the project-programmer number on the \$JOB card is used.

(processor switches) = the switches to be passed to the COBOL compiler. They must be enclosed in parentheses and the slash cannot appear in connection with these switches.

/S<sub>1</sub>/S<sub>2</sub>.../S<sub>n</sub> = the switches that control the mode of input interpretation and the listing of the compiled program.

| <u>Switch</u>           | <u>Meaning</u>                                                                                                                                                                                                                        | <u>Default</u>                                     |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------|
| /ASCII                  | The input is read in ASCII mode.                                                                                                                                                                                                      | on                                                 |
| /D029                   | The card deck is read in the old DEC-029 format. This format is similar to ASCII format and is available only in those installations that use DEC-029 format.                                                                         | off                                                |
| /LIST                   | A temporary listing file of the program is created.                                                                                                                                                                                   | on                                                 |
| /PROTECT:nnn            | Specifies the protection to be set for the job (in octal).                                                                                                                                                                            | The file is preserved only until KJOB for the job. |
| /SEQUENCE               | The program is read in conventional COBOL format with sequence numbers in columns 1 through 6, and comments beginning in column 73. When this switch is specified, the default width is 72.                                           | on                                                 |
| /SUPPRESS:<br>ON or OFF | When ON is specified, trailing blanks are suppressed. They are not suppressed when OFF is specified.                                                                                                                                  | on                                                 |
| /WIDTH:nn               | The maximum number of columns to be read. If the specified width is less than 80, only that number of columns is read, the remaining columns are treated as blank. Normally this switch is used only when the /SUPPRESS switch is on. | 80                                                 |
| /026                    | The card deck is read in 026 card code.                                                                                                                                                                                               | off                                                |

Restrictions

The /026 and /D029 switches apply only to card reader input. Input from other devices must be read in ASCII code; otherwise, an error message is written in the log file and the job is terminated.

\$DATA

Function

This card causes CDRSTK to copy data into a file on the user's disk area and to insert an EXECUTE monitor command into the control file.

CDRSTK maintains a list of filenames of all source or relocatable programs that have been processed since the beginning of the job or the last \$DATA card read. Each time a program is copied by the CDRSTK, its name is placed in the list and given an extension of .REL. When the \$DATA card is read, CDRSTK places an EXECUTE command into the control file and copies the filenames of the programs into the EXECUTE command string. On the next \$language card, CDRSTK clears the list of filenames so that the next entries into the list reflect only those filenames copied since the last \$DATA command was read. When the job is run, the programs are loaded and executed. No compilation is performed because the programs are either in relocatable binary form or have been previously compiled because of the \$language card. If two \$DATA cards appear in a row, the same programs are reloaded and executed again.

Loader switches can be passed to the LOADER by placing them in the command string. When CDRSTK places these switches in the EXECUTE command, it converts them to the standard LOADER switch format (i.e., % switch). For instance,

\$DATA (S,F)

causes the following EXECUTE command to be generated

.EXECUTE ... %S %F

Card Format

\$DATA dev:name.ext [proj,prog] /S<sub>1</sub>/S<sub>2</sub>.../S<sub>n</sub>

dev: = a file structure name. If omitted, DSK is assumed.

name.ext = the filename of the file to be created. If omitted, CDRSTK creates a unique filename of the form Qaa (aa = AA through ZZ) with the extension .CDR.

It is recommended, however, that the user select a distinct name for each job that is in the Batch system simultaneously, so that he can distinguish the various output listings.

[proj,prog] = the directory name if different from the one specified on the \$JOB card. If omitted, the project-programmer number specified on the \$JOB card is used.

/S<sub>1</sub>/S<sub>2</sub>.../S<sub>n</sub> = switches that control the mode of reading and interpreting of the input media.

Card Format (cont)

| <u>Switch</u>           | <u>Meaning</u>                                                                                                                                                                                                                                                                                                                                                                     | <u>Default</u> |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| /ASCII                  | The input stream is read in ASCII mode.                                                                                                                                                                                                                                                                                                                                            | on             |
| /BINARY                 | The card deck is read in binary card form. This switch is ordinarily not necessary because the first column of each card is checked for punches in rows 7 and 9. If these rows are punched, the card is read in binary.                                                                                                                                                            | off            |
| /D029                   | The card deck is read in the old DEC-029 format. This format is similar to ASCII format and is available only for compatibility with old decks.                                                                                                                                                                                                                                    | off            |
| /IMAGE:n                | The card deck is read in image mode. The switch must be followed by a decimal number in the range 2 through 80. This causes ensuing cards to be read in image mode until either end-of-file is reached or a card is read that contains punches in all rows of column 1 and all rows in column n. The CDRSTK control commands are not recognized when cards are read in image mode. | off            |
| /PROTECT:nnn            | A protection of nnn (octal) is set for the file; if not specified, the file is preserved only until a KJOB command for the job.                                                                                                                                                                                                                                                    |                |
| /SUPPRESS:<br>ON or OFF | When ON is specified, trailing spaces are suppressed. They are not suppressed when OFF is specified.                                                                                                                                                                                                                                                                               | on             |
| /WIDTH:nn               | The maximum number of columns to be read. If the specified width is less than 80, only that number of columns is read. The remaining columns are treated as blanks. Normally, this switch is only used when the /SUPPRESS switch is on. For example, /WIDTH:72 causes the CDRSTK to disregard columns 73 through 80 and to suppress any trailing spaces up to column 72.           | 80             |
| /026                    | The card deck is read in 026 card code.                                                                                                                                                                                                                                                                                                                                            | off            |

The modes ASCII, 026, IMAGE, and D029 are mutually exclusive modes for interpreting Hollerith punches. When one of those modes is set, it remains in effect until changed (refer to the \$MODE card) or the end of file is reached.

The defaults for all modes are reset by the next \$MODE card or by individual switches in other control cards such as in the \$DECK card.

Requirements

If the data is contained within the programs instead of being a separate file, a \$DATA card or an EXECUTE command must be placed after the programs. The program will not be executed otherwise.

Restrictions

This card can be used only when the programs in the job have been entered with a \$language card or \$RELOCATABLE card, since CDRSTK maintains a list of the filenames of programs that are input with these commands. If the user wishes only to have the programs compiled, no \$DATA card or EXECUTE command should appear in the job.

3.3.4

\$DECK

Function

This card causes CDRSTK to copy all statements up to the next control card into a data file.

Card Format

\$DECK dev:name.ext [proj,prog] /S<sub>1</sub>/S<sub>2</sub>.../S<sub>n</sub>

dev: = a file structure name. The default is normally DSK.

name.ext = the user-assigned name and extension of the file to be created. If omitted, a unique filename in the form DECKaa (aa = AA through ZZ) is created by CDRSTK with the extension .CDR. It is recommended, however, that the user select a distinct name for each job that is in the Batch system simultaneously.

[proj,prog] = a disk area other than the one supplied on the \$JOB card. If omitted, the project-programmer number specified on the \$JOB card is used.

/S<sub>1</sub>/S<sub>2</sub>.../S<sub>n</sub> = switches that control the mode of reading and interpreting of the input media. The switches are identical to the switches described for the \$DATA card.

Restrictions

The /BINARY, /O26, /IMAGE, and /D029 switches apply only to card reader input. Input from other devices must be read in ASCII code; otherwise an error message is written in the log file and the job is terminated.

3.3.5

\$DUMP

Function

This card causes CDRSTK to insert a DUMP monitor command into the control file which invokes a dump, according to the arguments specified, when an error is detected by the Batch Controller.

Card Format

\$DUMP /command arg/command arg...

The commands and their arguments are the same as described for the DUMP program. (Refer to Chapter 2). Two of the commands useful to a Batch job are duplicated below.

| <u>Command</u> | <u>Argument</u>                         | <u>Meaning</u>                                  |
|----------------|-----------------------------------------|-------------------------------------------------|
| ALL            |                                         | Dumps the entire file.                          |
| DUMP           | dump descriptor,<br>dump descriptor,... | Dumps the specified bytes in the current modes. |

3.3.6

\$EOD

Function

This card terminates the input that is being copied into a data file by CDRSTK because of a preceding \$DECK card. All control cards with the exception of \$MODE perform this action, i.e., terminate the copying of input. If input is not being copied and this card is read, CDRSTK ignores it. \$EOD is only necessary when the user wishes to place a line of input which is not a CDRSTK control card after input that is being copied into a data file.

Card Format

\$EOD

3.3.7

\$ERROR  
\$NOERROR

Function

These cards are used to aid the Batch Controller in processing errors. They cause CDRSTK to insert an .IF statement into the control file; e.g., .IF (ERROR) or .IF (NOERROR). Refer to Paragraph 3.4.5 for an explanation of the .IF statement. These cards must appear at the point at which the error occurs.

Card Formats

\$ERROR statement

\$NOERROR statement

statement = an executable monitor or batch command preceded by a period. If the statement directs the Batch Controller to go to a statement label, the statement label line and any related lines must be included in the sequence of commands at the place the user wants it executed. For example,

```

$FORTRAN TEST1
.
.
.
$ERROR .GOTO A
$DATA TEST1DA
.
.
.
$ERROR .GOTO A
A: CONT
$FORTRAN TEST2
.
.
.

```

3.3.8

\$FORTRAN or \$F40

Function

This card causes CDRSTK to copy the named FORTRAN program onto disk and to insert a COMPILE monitor command into the control file. The card is placed at the beginning of the source program. When the job is run, the specified program is compiled and temporary relocatable binary and listing files are created. The binary and listing files can be made permanent if the user renames them to change their protection. The source file can be preserved by means of the /PROTECT switch. The listing file is printed as part of the job's output.

Processor switches can be passed to the FORTRAN compiler by including them in the command string. Their position in the command string determines their position in the COMPILE command generated by CDRSTK. For example,

```
$FORTRAN (A,S,D)/NOLIST
```

results in the following COMPILE command

```
.COMPILE /COMPILE DECKII.F4 (A,S,D)/NOLIST
```

Refer to Paragraph 1.5.7 for a description of the FORTRAN processor switches.

Card Format

\$FORTRAN dev:name.ext [proj,prog] (processor switches) /S<sub>1</sub>/S<sub>2</sub>.../S<sub>n</sub>

\$F40 dev:name.ext [proj,prog] (processor switches) /S<sub>1</sub>/S<sub>2</sub>.../S<sub>n</sub>

dev: = a file structure name. If omitted, DSK is assumed.

name.ext = the name of the file to be created on disk. If omitted, CDRSTK assigns a unique filename of the form DECKaa (where aa = AA through ZZ) with the extension .F4. It is recommended, however, that the user select a distinct name for each job in the Batch system simultaneously.

[proj,prog] = a directory name other than that specified on the \$JOB card. If omitted, the project-programmer number on the \$JOB card is used.

(processor switches) = the switches to be passed to the FORTRAN compiler. They must be enclosed in parentheses and the slash cannot appear in connection with these switches.

/S<sub>1</sub>/S<sub>2</sub>.../S<sub>n</sub> = the switches that control the mode of input interpretation and the listing of the compiled program.

| <u>Switch</u>           | <u>Meaning</u>                                                                                                                                                                                                                        | <u>Default</u>                                     |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------|
| /ASCII                  | The input is read in ASCII mode.                                                                                                                                                                                                      | on                                                 |
| /CREF                   | A cross-referenced listing file is created to be processed by the CREF program.                                                                                                                                                       | off                                                |
| /D029                   | The card deck is read in the old DEC-029 format. This format is similar to ASCII and is available only in those installations that use DEC-029 format.                                                                                | off                                                |
| /LIST                   | A temporary listing file of the program is created.                                                                                                                                                                                   | on                                                 |
| /M                      | The MACRO coding is included in the output listing.                                                                                                                                                                                   | off                                                |
| /NOLIST                 | No listing file of the program is created.                                                                                                                                                                                            | off                                                |
| /PROTECT:nnn            | The protection to be set for the file (in octal).                                                                                                                                                                                     | The file is preserved only until KJOB for the job. |
| /SUPPRESS:<br>ON or OFF | When ON is specified, trailing blanks are suppressed. They are not suppressed when OFF is specified.                                                                                                                                  |                                                    |
| /WIDTH:nn               | The maximum number of columns to be read. If the specified width is less than 80, only that number of columns is read. The remaining columns are treated as blank. Normally this switch is only used when the /SUPPRESS switch is on. | 72                                                 |
| /026                    | The card deck is read in 026 card code.                                                                                                                                                                                               | off                                                |

Restrictions

The /026 and /D029 switches apply only to card reader input. Input from other devices must be read in ASCII code; otherwise, an error message is written in the log file and the job is terminated.

\$JOB

Function

This card, in conjunction with the \$PASSWORD card (if required), causes CDRSTK to create a control file and a log file on disk into which commands are placed for the Batch Controller. The filename of the control file is the name of the job specified in the command string, and the extension is .CTL. CDRSTK also uses this name as the filename of the log file with an extension of .LOG. If the jobname is omitted from the command string, CDRSTK creates a unique name for the control file and log file. It is recommended, however, that the user select a distinct name for each job that is in the Batch system simultaneously, so that he can distinguish the various output listings. In general, the jobname used on input appears in the output queues. CDRSTK adds the control and log files to the directory of the specified project-programmer number.

The user may specify a wildcard designation (#) for the programmer number in the \$JOB card, for example,

\$JOB FLEX[4,#] or \$JOB FLEX [4]

This specification causes CDRSTK to look at ACCT.SYS (an administrative file) in order to determine if the wildcard option is allowed for this project. If it is, CDRSTK provides a unique programmer number within the project. If it is not allowed, CDRSTK returns an error message and continues with the next job.

Card Format

\$JOB name [proj,prog] /S<sub>1</sub>/S<sub>2</sub>.../S<sub>n</sub>

name = the user-assigned name for the job; if omitted, CDRSTK creates a unique name of the form JOBaaa (aaa = AAA through ZZZ) for the control and log files.

[proj,prog] = the project-programmer number of the user who submitted the job. This argument is required. A space or comma can separate this argument from the jobname.

/S<sub>1</sub>/S<sub>2</sub>.../S<sub>n</sub> = switches taken from the following group. These switches are optional.

| <u>Switch</u>         | <u>Meaning</u>                                                                                                               | <u>Default</u> |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------|----------------|
| /AFTER:dd-mmm-yy hhmm | The job cannot be run until after the specified date and time. The resulting AFTER time must be less than the DEADLINE time. | None           |
| /AFTER:++             | The job cannot be run until after the input time plus the number of minutes indicated by ++.                                 | None           |
| /CARDS:nk             | The maximum number of cards (up to 10K) that can be punched by the job (in decimal). K is optional.                          | 0              |

(continued on next page)

Card Format (cont)

| <u>Switch</u>            | <u>Meaning</u>                                                                                                                                                                         | <u>Default</u>                                  |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|
| /CHARGE:aa               | The job is charged to a user-specified account (aa = name of the account).                                                                                                             | None                                            |
| /CORE:nnk                | Maximum amount of core (in decimal) that can be used by the job up to the maximum allowed by the installation. K is optional.                                                          | 25K                                             |
| /DEADLINE:dd-mmm-yy hhmm | The job must be completed by the specified date and time. The resulting DEADLINE time must be greater than the AFTER time.                                                             | None                                            |
| /DEADLINE:++t            | The job must be started by the indicated number of minutes after it is input.                                                                                                          | None                                            |
| /DEPEND:nn               | Initial interjob dependency count (in decimal).                                                                                                                                        | 0                                               |
| /FEET:nn                 | The number of feet of paper tape that will be punched by the job.                                                                                                                      | 0                                               |
| /LOCATE:Snn              | Specifies the remote station of the job and where the output is to be sent.                                                                                                            | The station where the cards were input.         |
| /NAME:aa                 | The user's name in up to 12 characters.                                                                                                                                                | None                                            |
| /PAGES:nn                | The maximum number of pages in decimal to be printed by the job, including the log file and compilation listing.                                                                       | 100                                             |
| /PRIORITY:nn             | The external priority of the job; the highest priority that can be specified is 62 (decimal).                                                                                          | 0                                               |
| /PROTECT:nnn             | The protection (in octal) for the job, the control file, and the log file.                                                                                                             | Preserved only until KJOB is given for the job. |
| /RESTART:0 or 1          | If 0, the job cannot be restarted by the operator. The job can be restarted if 1 is specified. The job should not be restartable if there are changes to the permanent file directory. | 0                                               |
| /TIME:hh:mm:ss           | The limit placed on the amount of CPU time used by the job.                                                                                                                            | 5.0 (5 minutes)                                 |
| /TPLOT:mm                | The number of minutes of plotter time that the job will use.                                                                                                                           | 0                                               |
| /UNIQUE:0 or 1           | If 1, only one Batch job at a time is run using the specified directory. If 0, any number of Batch jobs can be run at the same time using the specified directory.                     | 1                                               |

Requirements

The \$JOB card must immediately follow the \$SEQUENCE card, or be the first card if the \$SEQUENCE card is not required.

\$MACRO

Function

This card causes the CDRSTK to copy the designated MACRO program onto disk and is placed at the beginning of the source program. When CDRSTK reads the card, it inserts a COMPILE monitor command into the control file and copies the MACRO program into the file on the specified disk area. When the job is run, the program is assembled and a temporary relocatable binary file and listing files are created. The binary and listing files can be made permanent if the user renames them to change their protection. The source file is preserved by means of the /PROTECT switch. The listing file is printed as part of the job's output.

Processor switches can be passed to the MACRO assembler by including them in the command string. Their position in the command string determines their position in the COMPILE command generated by CDRSTK. For example

```
$MACRO /PROTECT:055 (W,S,Q)
```

results in

```
.COMPILE /COMPILE DECKCB.MAC /PROTECT:055 (W,S,Q) /LIST
```

Card Format

```
$MACRO dev:name.ext [proj,prog] (processor switches) /S1/S2.../Sn
```

dev: = a file structure name. If omitted, DSK is assumed.

name.ext = the name of the file to be created on disk. If omitted, CDRSTK assigns a unique filename in the form DECKaa (aa = AA through ZZ) with the extension .MAC. However, it is recommended that the user select a distinct name for each job in the Batch system simultaneously.

[proj,prog] = a directory name other than that specified on the \$JOB card. If omitted, the project-programmer number on the \$JOB card is used.

(processor switches) = the switches to be passed to the MACRO assembler. They must be enclosed in parentheses and the slash cannot appear in connection with these switches.

/S<sub>1</sub>/S<sub>2</sub>.../S<sub>n</sub> = the switches that control the mode of input interpretation and the listing of the assembled program.

| <u>Switch</u> | <u>Meaning</u>                                                                  | <u>Default</u> |
|---------------|---------------------------------------------------------------------------------|----------------|
| /ASCII        | The input is read in ASCII mode.                                                | on             |
| /CREF         | A cross-referenced listing file is created to be processed by the CREF program. | off            |

(continued on next page)

Card Format (cont)

| <u>Switch</u>           | <u>Meaning</u>                                                                                                                                                                                                                          | <u>Default</u>                                     |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------|
| /D029                   | The card deck is read in the old DEC-029 format. This format is similar to ASCII and available only in those installations that use DEC-029 format.                                                                                     | off                                                |
| /LIST                   | A temporary listing file of the program is created.                                                                                                                                                                                     | on                                                 |
| /NOLIST                 | No listing file of the program is created.                                                                                                                                                                                              | off                                                |
| /PROTECT:nnn            | The protection to be set for the file (in octal).                                                                                                                                                                                       | The file is preserved only until KJOB for the job. |
| /SUPPRESS:<br>ON or OFF | When ON is specified, trailing blanks are suppressed. When OFF is specified, they are not suppressed.                                                                                                                                   | on                                                 |
| /WIDTH:nn               | The maximum number of columns to be typed. If the specified width is less than 80, only that number of columns is read. The remaining columns are treated as blank. Normally, this switch is used only when the /SUPPRESS switch is on. | 80                                                 |
| /026                    | The card deck is read in 026 card code.                                                                                                                                                                                                 | off                                                |

Restrictions

The /026 and /D029 switches apply only to card reader input. Input from other devices must be read in ASCII code; otherwise, an error message is written in the log file and the job is terminated.

3.3.11

\$MODE

Function

This card causes CDRSTK to change the mode in which it is interpreting the input stream. The \$MODE card can be placed anywhere after the \$PASSWORD card in the command sequence and is terminated by another \$MODE card or the end-of-file (which terminates the job). This command does not terminate the copying of input preceded by a \$DECK card.

Card Format

\$MODE /S<sub>1</sub>/S<sub>2</sub>.../S<sub>n</sub>

/S<sub>1</sub>/S<sub>2</sub>.../S<sub>n</sub> = switches that control the mode of reading and interpreting of the input media. These switches are identical to the switches described for the \$DATA card.

Restrictions

The mode switches /026, /IMAGE, /D029, and /BINARY can be used only for card input. Input from other devices is always read as ASCII code. Thus, the only switches that can be used with the \$MODE card for devices other than the card reader are /SUPPRESS and /WIDTH.

3.3.12

\$PASSWORD

Function

This card contains the password associated with the project-programmer number specified in the \$JOB card. If the password does not match the password stored in the system for the specified project-programmer number, CDRSTK does not create any files and aborts the job. Use of this command is an installation option.

Card Format

\$PASSWORD password

password = 1 to 6 character password.

Requirements

If the \$PASSWORD card is required, it must immediately follow the \$JOB card.

3.3.13

\$RELOCATABLE

Function

This card causes CDRSTK to copy a relocatable binary program from cards to a file on the user's disk area. The cards are read in binary mode.

Card Format

\$RELOCATABLE dev:name.ext [proj,prog] /S<sub>1</sub>

dev: = a file structure name. If omitted, DSK is assumed.

name.ext = the name of the file into which the program is copied. If the filename is omitted, CDRSTK creates a unique name in the form DECKaa (aa = AA through ZZ). It is recommended that the user select a distinct name for each job in the Batch system simultaneously. If the extension is omitted, .REL is assumed.

(continued on next page)

Card Format (cont)

[proj, prog] = the disk directory if different from the one specified on the \$JOB card. If omitted, the project-programmer number on the \$JOB card is assumed.

/S<sub>1</sub> = /PROTECT:nnn (octal)

The protection for the file to be created. If not specified, the file is preserved only until a KJOB command for the job is executed.

Restrictions

Relocatable binary programs can only be read when the input is from cards.

The program following this command must be read in binary; the mode cannot be changed until a nonbinary file is copied. If an attempt is made to change the mode, an error message will be issued and the job will be aborted.

## 3.3.14

|            |
|------------|
| \$SEQUENCE |
|------------|

Function

This card specifies the job's unique sequence number. The use of this card depends on the requirements of the particular installation.

Card Format

\$SEQUENCE n

n = a decimal number

Requirements

If the installation requires this command, it must be the first card in the input stream.

## 3.4 BATCON CONTROL FILE COMMANDS

Ordinarily the Batch Controller reads the control file in a sequential manner. The commands described in this section can appear in the control file to interrupt the sequential processing of the control file in order to specify error recovery. If an error occurs in the job, the Batch Controller is notified of the error; the user has the option of including several methods of error recovery.

The user may include an .IF command in the control file. When the error occurs, the Batch Controller examines the next monitor level line in the control file for an .IF command to determine what action

to take on the error. It does not search past the next executable monitor line in the control file for the .IF command; therefore, if this command is used, it must be the next monitor command in the control file.

If the user does not wish to include an .IF command, he may include two types of error recovery routines in the control file, one type labeled %ERR (error processing for non-system programs) and the other labeled %CERR (error processing for compilers and system programs). A system program is one found on a device specified in the SYS search list in [1,4]. If SYS is assigned as a logical device name, the programs are considered user programs, not system programs. After an error occurs in the job and the next executable monitor line in the control file is not an .IF statement, the Batch Controller searches for the labeled error recovery control lines and processes the statements following these labels. These routines may be placed anywhere in the control file. Once the Batch Controller has processed the routine, it continues from that point in the control file; it does not read backwards over sections of the control file skipped in searching for the error routines. The following example shows the use of a %ERR error recovery routine.

```
.COMPILE SAMPLE /LIST )  
.MOUNT MTA:3 /VID:42936 )  
.EXECUTE )  
.DISMOUNT 3 )  
.R SORT )  
*MUMP.SRT+FOR04.DAT/R80/K1.10 )  
.QUEUE MUMP.SRT )  
%ERR:.CLOSE )  
.DUMP )  
.DISMOUNT 3 )  
%FIN:.DELETE FOR04.DAT )
```

Depending on the type of error found, the following operations are performed. If a compilation error occurs, only the compilation and the listing result. No tape is mounted. If an execution error results,

1. the program is compiled,
2. the tape is mounted,
3. the program begins execution,
4. the output is closed,
5. a quick dump of core is taken,
6. the tape is dismantled, and
7. the file FOR04.DAT is deleted.

If a SORT error occurs, the program compiles, the tape is mounted, the program is executed, and the file FOR04.DAT is deleted. Finally, if no errors result,

1. the program is compiled,
2. the tape is mounted,
3. the program is executed,
4. the tape is dismantled,
5. the sort is performed,
6. MUMP.SRT is printed, and
7. the file FOR04.DAT is deleted.

When the user is bypassing CDRSTK and creating his own control file, he may place a %FIN at the end of the control file. (CDRSTK, in creating the control file, automatically places a %FIN at the end.) This label is used for cleanup purposes, e.g., deleting the input files. In creating the control file, the user may place other %FIN's at various points in the file for periodic cleanup of his job. For example, this label is used in a special kind of error recovery. If the time allocated to the job runs to the maximum limit specified in the \$JOB command (refer to Paragraph 3.3.9) or by the Batch system, the user is given an additional 10% of his allocated time to cleanup his job before it is aborted. Because the user includes a %FIN, cleanup is performed and the results of the job's processing are not lost when the job is aborted. The user should be careful in using the %FIN in the control file because if the Batch Controller is searching for an error recovery routine and %FIN is placed before a %ERR or %CERR, the %FIN is executed and the Batch Controller assumes the error recovery routine has been satisfied and does not search any longer for %ERR or %CERR. Furthermore, a .GOTO label cannot bypass a %FIN label. Therefore, the best place to put a %FIN is as the last line in the control file.

If an error occurs in the job and the user either was not running a system program or has not included an .IF command or error recovery control lines, the Batch Controller initiates a standard quick dump of the user's core area and terminates the job (refer to the DUMP command in Chapter 2). The Batch Controller also initiates a dump if it is searching for a %ERR and reads a %FIN instead.

### 3.4.1

|         |
|---------|
| .BACKTO |
|---------|

#### Function

The .BACKTO command is used by Batch users to interrupt the sequential reading of the control file by the Batch Controller. Control is transferred in a backward direction. This command can be used with a .IF command to specify transfer of control to an error routine.

#### Command Format

.BACKTO label

label = label of a statement in the control file. This label is from one to six alphanumeric characters terminated with a colon and must not begin with a % character.

When the .BACKTO command is encountered, the Batch Controller searches for the labeled statement and transfers control to it. If the statement is not found, the job is terminated.

3.4.2

**.CHKPNT**

Function

The .CHKPNT command is used to aid in error recovery when a Batch job is terminated abnormally by a system failure. As many .CHKPNT commands as desired can be placed in the control file. When the job is restarted after the failure, the program begins at the location of the last .CHKPNT command instead of at the beginning of the program.

Command Format

.CHKPNT label

label = label of a statement in the control file. This label is from one to five alphanumeric characters. When the label appears with the statement in the control file, it must be followed by a double colon instead of the usual single colon (e.g., label :: statement).

3.4.3

**.ERROR**

Function

The .ERROR command causes the Batch Controller to recognize a message beginning with the specified character as an error in the job.

Command Format

.ERROR character

character = the beginning character of the line that is to be recognized as an error (e.g., %). If this argument is not specified, a ? at the beginning of a line is considered as an error.

3.4.4

**.GOTO**

Function

The .GOTO command is used by Batch users to interrupt the sequential reading of the control file by the Batch Controller. Control is transferred in a forward direction. This command may be used with a .IF command to specify transfer of control to an error routine.

Command Format

`.GOTO label`

label = label of a statement in the control file. The label appearing in the control file is from one to six alphanumeric characters terminated with a colon and must not begin with a % character.

When the `.GOTO` command is encountered, the Batch Controller searches for the labeled statement and transfers control to it. If the statement is not found before the end of the control file is reached, the job is terminated.

Examples

```
.EX TEST.MAC/L
.IF (ERROR) .GOTO A
.GOTO B
A:;.QUEUE LPT: =TEST.MAC

.GOTO B
B:;
```

## 3.4.5

|     |
|-----|
| .IF |
|-----|

Function

The `.IF` command is used by Batch users to aid the Batch Controller in processing errors. The Batch Controller recognizes the existence of an error when it encounters a line beginning with a question mark that is output from the job to the log file or a line that begins with the character specified in the `.ERROR` command. When the error occurs, this command must be the next monitor level command in the control file.

Command Format

`.IF (condition) statement`

(condition) = ERROR or NOERROR. The parentheses must be included.

statement = an executable monitor or batch command preceded by a period.

If the specified condition is true, the statement is executed. If the specified condition is not true, the Batch Controller processes the next line in the control file.

3.4.6

`.NOERROR`

Function

The `.NOERROR` command instructs the Batch Controller to ignore all errors (including messages beginning with a question mark) in the job. This is especially useful in TECO searches. However, the message

`?TIME LIMIT EXCEEDED`

always indicates that an error exists.

Command Format

`.NOERROR`

3.4.7

`.NOOPERATOR`

Function

The `.NOOPERATOR` command designates that no messages from the job are to be output to the controlling terminal.

Command Format

`.NOOPERATOR`

3.4.8

`.OPERATOR`

Function

The `.OPERATOR` command makes it possible for the job, or a program within the job, to communicate with the operator. Any message from the job, starting with the specified character (refer to Chapter 4), is typed on the controlling terminal. The job then waits for operator intervention and the operator's answer restarts the job.

(continued on next page)

Function (cont)

When the .OPERATOR command is in effect, the Batch Controller ignores an .IF statement unless the .NOOPERATOR command is given first, and proceeds to search for an error recovery routine labeled with either %ERR: or %CERR: (refer to Paragraph 3.4). This action is taken in order to minimize output to the operator in case of an unexpected transfer of control. However, when an error occurs, the Batch Controller preserves the error status across the .NOOPERATOR command and looks for the .IF statement as the next monitor-level command. In other words, an .IF statement following a .NOOPERATOR command will be executed. Refer to the following examples.

In the example below, the .IF statement will be ignored.

```
.OPERATOR %
.RUN TESPGR
.IF (ERROR) .GOTO TAG
```

However, in the following example, the .IF statement will be executed.

```
.OPERATOR %
.RUN TESPGR
.NOOPERATOR
.IF (ERROR) .GOTO TAG
```

Command Format

.OPERATOR character

character = the beginning character of the line that is to be sent to the operator (e.g., %). If this argument is not specified, \$ at the beginning of the line is assumed.

## 3.4.9

.REQUEUE

Function

The .REQUEUE command indicates to the Batch Controller that the job is to be requeued, instead of terminated, after an error. It is normally used with the .IF (ERROR) command (e.g., .IF (ERROR) .REQUEUE). The job is restarted after a default requeue time at the specified label in the control file.

Command Format

.REQUEUE label

3.4.10

**.REVIVE**

Function

The .REVIVE command causes all output from the job to be placed in the log file.

Command Format

.REVIVE

3.4.11

**.SILENCE**

Function

The .SILENCE command suppresses all output from the job except error messages to the log file. This means that the only lines appearing in the log file will be those that begin with a question mark.

Command Format

.SILENCE

### 3.5 JOB OUTPUT

The output from a user's job is normally in the form of printed listings containing the user's job output, compilation listings, any memory dumps requested by the user or initiated by the Batch Controller, and the log file indicating the processing performed by the programs in the Batch system. The results from the job and the log file are automatically placed in the queue for the line printer spooler, LPTSPL, unless the job was submitted with the /OUTPUT:0 switch. However, the user can output to any device in the system. When a user program specifies a slow-speed spooling device, the Batch system places the output into a queue for the appropriate spooler. If the user wishes specific files to be output to particular spooled devices outside of his programs, he can include the QUEUE monitor commands in his control file to specify the output device and any additional parameters that he wishes.

Compilation listings are produced from the \$language control cards unless the user specifies otherwise. These listings are automatically spooled to the line printer. The user can also include the COMPILE monitor command in his job with switches to produce listings.

The user can include any of the monitor DUMP commands or the CDRSTK card \$DUMP to request memory dumps during program testing. Under normal error conditions, the Batch Controller performs an automatic two-page dump for the user (refer to Paragraph 3.4).

### 3.5.1 The Log File

As part of its processing, CDRSTK creates a log file for each job so that the user can examine the processing performed by the CDRSTK and BATCON programs. The log file is the first part of the job's output. CDRSTK enters a record of its own processing, any errors detected, and any operator interventions. When the job is run, the Batch Controller places additional messages into the log file, including each line of the control file as it is passed to the job, any error conditions, and any operator actions. The LOGOUT program appends an accounting summary message to the log file when the job terminates. This message is similar to the message received when an interactive user logs off the system (refer to the KJOB command in Chapter 2). Note that the log file is appended to for jobs of the same name; thus it may be necessary to delete this file before running another job with the same name.

3.5.1.1 CDRSTK Messages - CDRSTK places six kinds of messages into the log file. The first line of each message is identified by the time that CDRSTK placed the message into the file and by an identifying word in columns 1 through 16. The identifier for each kind of message is taken from the following group:

- DATE -- gives the date, system name, CDRSTK version, and the input device.
- STACK -- identifies any CDRSTK nonerror message.
- STERR -- identifies any CDRSTK error message.
- CARD -- describes any card image not in an error message.
- STSUM -- identifies the summary message at the end of the CDRSTK's processing.
- STOPR -- describes any operator actions that occurred during the CDRSTK's processing.

The first entry in the log file always contains the identifier DATE and a message giving the date, the system name, the current version of CDRSTK, and the input device; for example,

```
10:20:06 DATE 13-MAY-71 5S03C System 40 CDRSTK version 7 device CDR1
```

The \$SEQUENCE and the \$JOB commands are the next two lines printed. The \$PASSWORD command is never printed for reasons of security. When the end-of-file is read, CDRSTK prints a summary message giving the number of cards read, the number of files and blocks written, and the number of each type of error that occurred. The summary is also placed in the system accounting file. An example of the job summary is given below.

11:25:38 STSUM End-of-File after 423 cards, 3 files (40 blocks) written  
 4 Hollerith errors (nonfatal)  
 2 Binary Sequence errors (fatal)  
 Job Aborted by CDRSTK

Between the beginning and ending messages, CDRSTK prints any operator actions as they occur, some nonerror messages, and reports of errors it has detected. The following are examples of nonerror messages from CDRSTK.

```
CARD    $JOB TESTA, [10,225]
STOPR   JOB STOPPED BY OPERATOR
STOPR   CONTINUED BY OPERATOR
```

3.5.1.2 CDRSTK Error Reporting - CDRSTK places messages in the log file that describe errors that have occurred during its processing. The following errors are detected, and their degree of severity is as specified:

Fatal Errors

- a. Error on the \$JOB card.
- b. Error on the \$PASSWORD card.
- c. Unrecognizable command on a CDRSTK control card.
- d. Error in a parameter on a CDRSTK control card.
- e. Binary sequence error - issued a maximum of five times per deck.
- f. Improper code (binary rather than Hollerith, or vice versa).

Nonfatal Errors

- a. Hollerith error (invalid punch).
- b. Missing end-of-file card.

Error messages are issued by CDRSTK to the log file either up to the first fatal error, or, for nonfatal errors, up to a maximum of 200 errors or errors on 10% of the total card count, whichever is greater. However, CDRSTK continues processing the job up to the end-of-file. The following are examples of error messages placed in the log file by CDRSTK.

```
JOB ABORTED BY OPERATOR
JOB ABORTED - HOLLERITH ERRORS
CARD #nnn    FATAL CARD
CARD #nnn    COL #nnn
CARD #nnn    CARD SEQUENCE ERROR
CARD #nnn    SWITCH ERROR
CARD #nnn    MODE ERROR
NON-BINARY CARD IN BINARY DECK
```

Each card-reading error results in a message which includes the first card column in error, the deck number and columns 1 through 30 of the \$DECK card, and the card number within the deck and within the job. The faulty card image appears on the next line with a backward slash (\) indicating the column in error.

```
11:15:05 STERR Hollerith error at col. 7 of card 241, card 73 in deck 2 ($FORTRAN MAIN)
3 \ORMAT ('FOO')
```

3.5.1.3 Batch Controller Messages - The Batch Controller messages are similar to those of the Stacker. The times followed by an identifying notation are placed in columns 1 through 16 of the first line of each message. The identifiers for the Batch Controller messages are described in the list below:

BVERS -- denotes the version of BATCON.  
 BDATE -- identifies the date BATCON processed the job.  
 BATCH -- identifies any Batch Controller nonerror message.  
 BAOPR -- describes any operator action.  
 BAERR -- denotes any Batch Controller error message:  
 MONTR -- identifies a line input or output at monitor level.  
 USER -- describes any line input or output at user level.  
 BASUM -- gives the Batch Controller summary message.

The first line in the log file printed by the Batch Controller is the version number. As each line in the control file is read, it is printed in the log file as well as being passed to the user program or to the monitor. Any time that the operator performs some action that affects the job, the Batch Controller records it in the log file. The BATCON program enters a message in the log file every time it generates a monitor command. For example, if a fatal error occurs in the job and the user has not included an .IF statement, a %ERR routine or a %CERR routine in the control file, the Batch Controller generates a DUMP command. It also generates a LOGIN and a KJOB monitor command for each job.

Any errors in the input that are detected by the Batch Controller are printed in the log file.

3.5.1.4 Batch Controller Error Reporting - The Batch Controller places the identifier BAERR on any line that it detects as being an error. The errors that are detected are listed below; the first three are fatal errors.

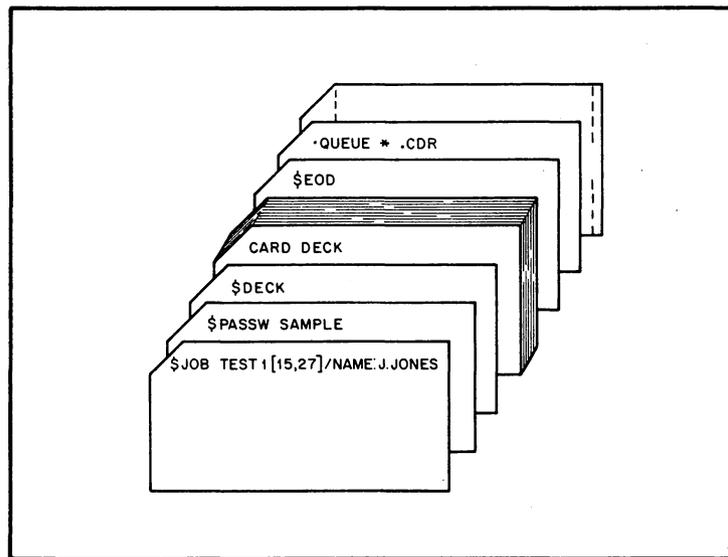
- a. Missing condition (ERROR or NOERROR) or missing statement in an .IF statement.
- b. Missing statement label in the .GOTO or BACKTO command.
- c. The labeled statement in a .GOTO command cannot be found after the .GOTO or before the .BACKTO command in the control file.
- d. Use of the ATTACH, DETACH, SEND, CCONT, and CSTART monitor commands.

Most user error conditions are not flagged by the Batch Controller, they are passed to the monitor where they are flagged as errors.

### 3.6 SAMPLE JOBS

The following sample job setups illustrate the versatility of the Batch System.

The first example, Figure 3-2, shows a setup to list a card deck with the QUEUE monitor command.

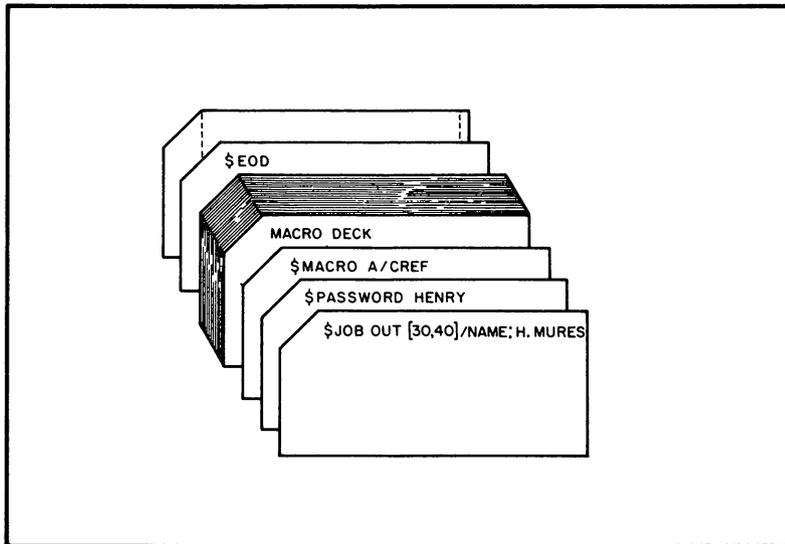


10-0730

Figure 3-2 Sample Job #1

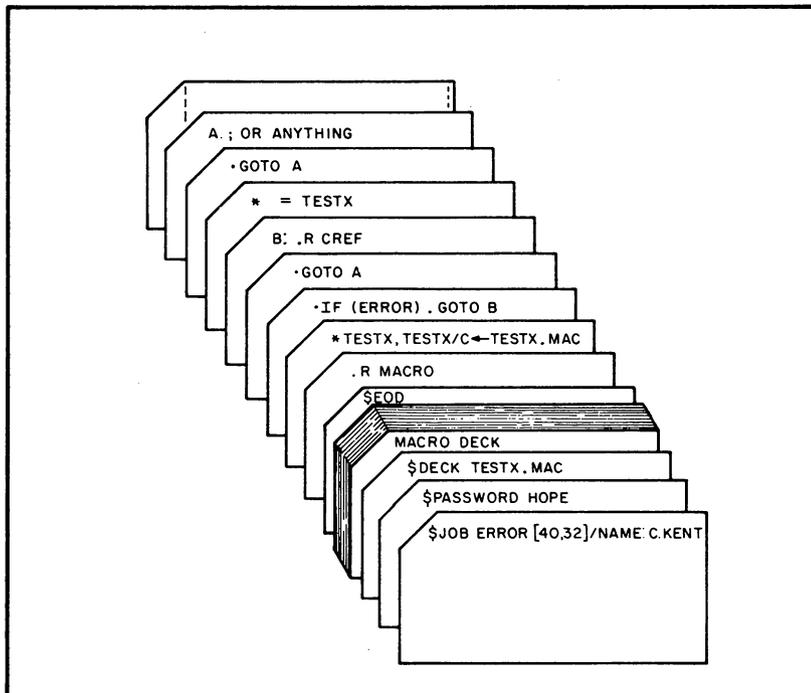
The second example, Figure 3-3, produces a CREF listing of a MACRO deck whether or not errors occur in the program.

The third example, Figure 3-4, illustrates the use of error processing commands.



10-0728

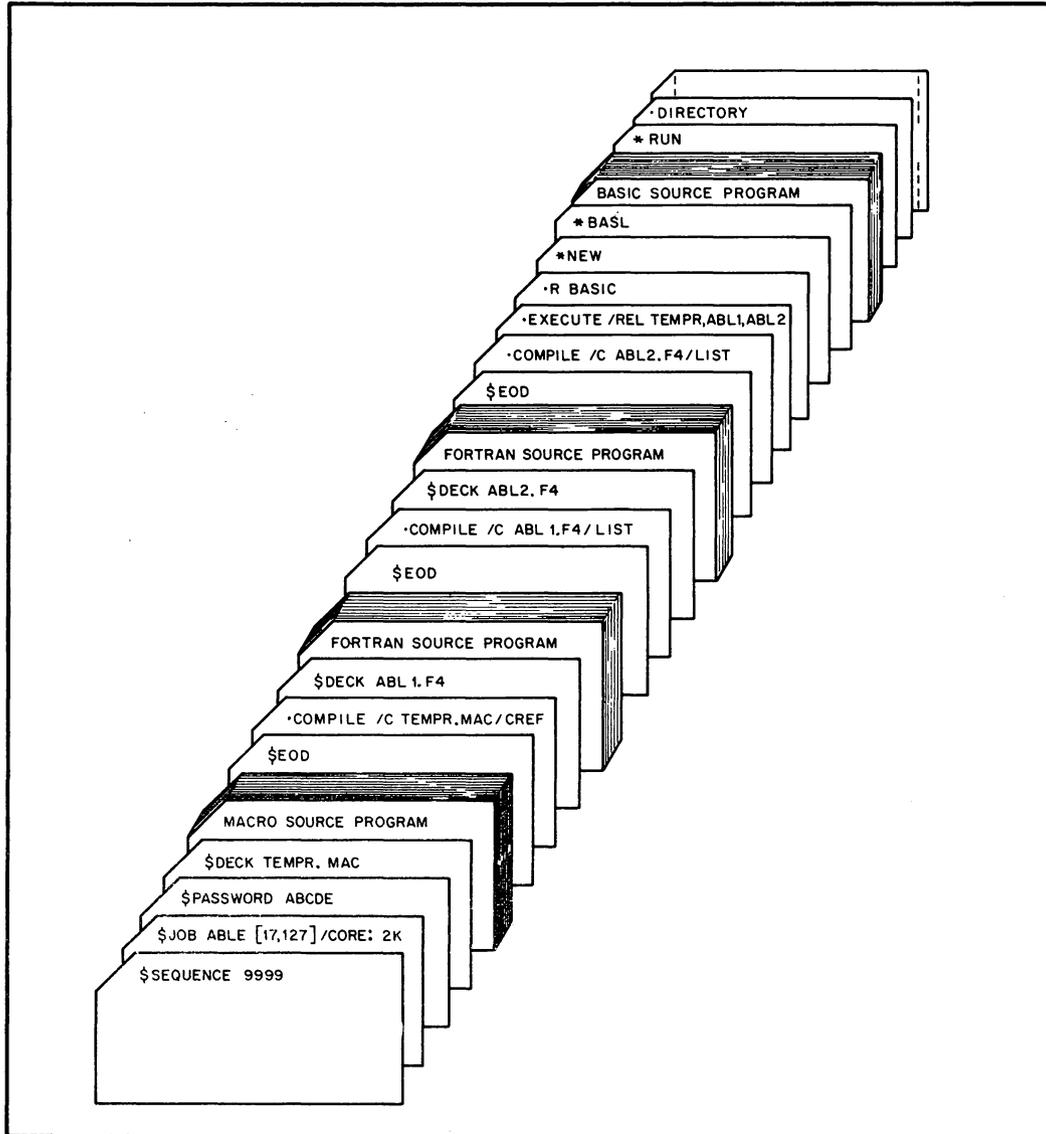
Figure 3-3 Sample Job #2



10-0727

Figure 3-4 Sample Job #3

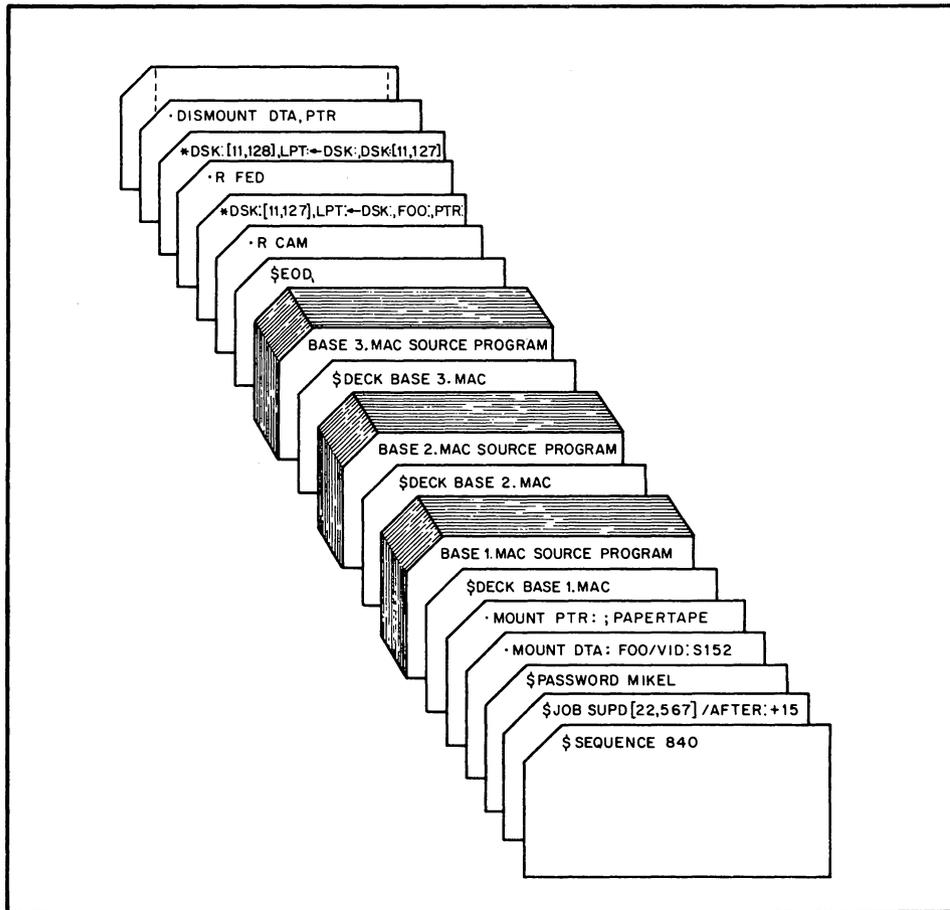
Figure 3-5 illustrates a MACRO assembly, two FORTRAN compilations, and execution of all three programs, and shows how monitor commands are entered along with the programs and the Stacker control cards.



10-0726

Figure 3-5 Sample Job #4

Figure 3-6 shows a simple SOUP update. Three base files are copied from cards to disk. The user files are on DECtape and the correction from DEC is on paper tape.



10-0725

Figure 3-6 Sample Job #5

## CHAPTER 4

# SYSTEM DIAGNOSTIC MESSAGES AND ERROR CODES

The following conventions are used in describing the system diagnostic messages:

|                     |                                            |
|---------------------|--------------------------------------------|
| dev                 | represents a legal device name.            |
| file structure name | represents a legal file structure name.    |
| file.ext            | represents a legal filename and extension. |
| adr                 | represents a user address.                 |
| n                   | represents a number.                       |
| abc                 | represents a disk unit or drive.           |
| x                   | represents an alphabetic character         |
| switch              | represents a switch.                       |

Most messages returned to the user fall in one of five categories. These categories are determined by the beginning character of the message.

? at the start of the message indicates a fatal error message.

% at the start of the message represents an advisory or warning message.

[ at the beginning of the message indicates a comment line.

\$ at the beginning of the message represents an operator/job communication line. A response is expected.

' (quote) at the beginning of the message represents a comment to the operator. No response is expected.

Programs and/or commands causing the error message are given in parentheses. (Note that the ONCE-only messages have been removed and placed in ONCE.RNO in the DECsystem-10 Software Notebooks.)

The descriptive text given with the message indicates what action the user should take when he receives the message. He can, if necessary, notify the operator of any problems that he is having by issuing the SEND, PLEASE, or R GRIPE command.

## 4.1 SYSTEM DIAGNOSTIC MESSAGES

The typein is typed back preceded and followed by ?

The monitor encountered an incorrect character (e.g., a letter in a numeric argument). The incorrect character appears immediately before the second ?.

For example:

```

.CORE ABC
?CORE A?

```

## ACCOUNTING SYSTEM FAILURE...

A program could not append an entry to the accounting file. Notify the operator. (LOGIN, LOGOUT).

## ?ADDRESS CHECK FOR DEVICE dev

(1) The monitor checked a user address on a UJO and found it to be too large ( $>C(.JBREL)$ ) or too small ( $\leq C(.JBPF1)$ ); in other words, the address lies outside the bounds of the user program  
 (2) The SAVED file is too large for the core assigned, or the file is not a core image file. (GET).

## \$ALL AREAS ON BACKUP

The BACKUP program has processed all of the project-programmer numbers specified and is now closing the associated files. (BACKUP).

## ?ALREADY ASSIGNED TO JOB n

The device is already assigned to another user's job (job n).

## ?AMBIGUOUS ABBREVIATION

A command or switch has been abbreviated to the point that it is not unique. (COMPIL).

## ?ARGS ARE: DAY, RUN, WAIT, READ, WRITE, VERSION, ALL, NONE

The user either did not type an argument or typed an illegal argument in the SET WATCH command string.

## dev: ASSIGNED

The device has been successfully assigned to the user's job.

?ASSIGNED TO JOB  $n_1, n_2, \dots$ 

If there is more than one device of the type specified, the numbers of the other jobs that have the same type of device are output, unless the user assigning the device has all the devices of the specified type. In this case, ?DEVICE ASSIGNED TO JOB is output.

## ?ATTACH TO USER JOB FAILED

DAEMON could not attach to the user's job. (DAEMON).

**\$BACKUP COMPLETED AT time**

The BACKUP program has successfully completed. (BACKUP).

**?BAD DENSITY**

The value given with the DENSITY command was not valid. (RESTORE).

**?BAD DIRECTORY FOR DEVICE DTAn**

The system cannot read or write the DECTape directory without getting some kind of error. This error often occurs when the user tries to write on a write-locked tape or use a DECTape that has never been written on.

**?BATCH ONLY**

The command issued can only be given by a batch job.

**BLOCK NOT FREE**

M specifies a unit or file structure logical block that is not free. (ALCFIL).

**n BLOCKS ALREADY ALLOCATED**

The file already exists. The new specification replaces, rather than updates, the old specification. (ALCFIL).

**?n1K BLOCKS OF CORE NEEDED**

The user's current core allocation is less than the contents of .JBFF.

**?BOMB OUT**

The location within INITIA that detected the error will be in AC 15 and the console lights. (INITIA).

**?BOOTSTRAP LOADER IS NOT IN COPY; TRY /L**

An attempt was made to write the bootstrap loader onto a DECTape via the /T switch before the loader was loaded into a core buffer and preserved with the COPY core image. (COPY program).

**?BOOTSTRAP LOADER WILL NOT FIT IN 3 BLOCKS**

The user's bootstrap loader is too big to fit into blocks 0, 1, and 2 of the output DECTape. (COPY program).

**?BUFFER CAPACITY EXCEEDED AND NO CORE AVAILABLE**

The buffer is not large enough to handle the number of lines required for looking ahead for matches, and additional core is not available. (FILCOM).

**?BUSY**

The terminal addressed is not communicating with the monitor (i.e., it is accepting a command or returning output from a command). The operator's terminal is never busy. (SEND, JCONT).

**?CANNOT DO I/O AS REQUESTED**

Input (or output) cannot be performed on one of the devices specified for input (output). For example, input may have been requested for a device that can only do output. (FUDGE2).

**?CANNOT DO OUTPUT TO DEVICE dev**

Output was attempted to a device that can only do input, or to a device assigned a logical name. (QUEUE).

**?CANNOT PROCESS EXTERNAL SYMBOLS**

External symbols were encountered while loading the bootstrap loader with the /L switch. (COPY program).

**?CANNOT PROCESS HIGH SEG'S**

While loading the bootstrap loader with the /L switch, high segment code was encountered. (COPY program).

**?CANNOT REATTACH FROM A BATCH SUBJOB**

Batch jobs are not allowed to reattach their jobs. (REATA).

**\$\$%CANT ACCESS COMMAND FILE - CONTINUING**

The command recovery file is not being created. This file contains information as to how much of the user's command has been processed and how much is remaining. Without this file, the user must start at the beginning if the system crashes. (BACKUP, RESTORE).

**?CANT ACCESS DEVICE dev**

The device specified cannot be INITed. The device is either in use or has an error, such as, being off-line. The user should request another device, or check this device for errors. (BACKUP, RESTORE).

**\$\$%CANT ACCESS INDEX DEVICE - CONTINUING dev**

The device specified for the index file cannot be INITed and an index file is not being created. The user can start over if he wants to create an index file. (BACKUP, RESTORE).

**?CANT ACCESS SYSTEM FILES**

ACCT.SYS could not be read. Only the operator may LOGIN until ACCT.SYS is ready. Consult the operator. (LOGIN).

**?CANT ADD TO YOUR FILE STRUCTURE SEARCH LIST n**

n is the error code from STRUO when trying to add a file structure to search list. (LOGIN).

**?CANT ATT TO JOB**

The project-programmer number specified is not that of the owner of the desired job, the project-programmer number was not given when it was required, or the PASSWORD given was incorrect. (ATTACH).

?dev CANT BE REASSIGNED

(1) The job's controlling terminal cannot be reassigned, or (2) the logical name would be duplicated, or (3) the logical name is a physical device name in the system and the job reassigning the device is either logged-in under a different project-programmer number or does not have operator privileges. (REASSIGN).

?CANT CONTINUE

The job was terminated due to (1) all ERROR IN JOB messages (except for HALT), (2) the EXIT UJO, (3) the CLOSE command, or (4) the REA command when the device was INITed, and the user attempted to continue his program at the point at which I/O was terminated. The job cannot be continued.

CANT CREATE NEW FILE STRUCTURE SEARCH LIST

The monitor cannot create a new file structure search list.

?CANT DECIPHER COMMAND

The command typed is not recognized by the BACKUP program. (BACKUP, RESTORE).

?CANT DECIPHER THAT

There is a syntax error in the command string. (MOUNT, DISMOUNT, FILE).

?CANT DET DEV

The user is not logged-in under [1,2].

?CANT ENTER OUTPUT FILE n file descriptor

The ENTER to write the output file failed; n is the disk error code. (DUMP).

?CANT EXPAND TABLE xxxx

The DUMP program ran out of core in attempting to expand the indicated table. (DUMP).

?CANT FIND INPUT FILE n file descriptor

DUMP cannot locate the file specified as the input file; n is the disk error code. (DUMP).

?CANT FIND FILE file.ext

The specified file could not be found.

?CANT GET SWAPPING PARAMETERS

DAEMON tried to obtain the job's swapping parameters and failed. (DAEMON).

?CANT GET SWAPPING POINTER FOR JOB

DAEMON tried to obtain the pointer to the user's job on the swapping space and could not because the GETTAB UJO failed. (DAEMON).

**?CANT GET USERS PPN**

DAEMON tried to obtain the user's project-programmer number and could not because a GETTAB UJO failed. (DAEMON).

**?CANT OPEN file structure name**

The file structure is mounted but cannot be opened. No UFD is created, though one may already exist. (LOGIN).

**?CANT OPEN CHANNEL FOR DEVICE dev**

The OPEN on the channel for the named device failed. (BACKUP, RESTORE).

**?CANT OPEN DEVICE dev**

The specified device does not exist or it is assigned to another user. (DAEMON).

**?CANT OPEN INDEX FILE**

The OPEN failed for the index file. (BACKUP, RESTORE).

**?CANT OPEN SWAP UNIT abc**

DAEMON attempted to use the indicated swapping unit and failed. (DAEMON).

**?CANT RELEASE UFD INTERLOCK FOR dev [p,p]**

The UFD interlock cannot be released for the named device. (BACKUP).

**?CANT RENAME-FILE PRESERVED**

An attempt was made via the /DISPOSE:RENAME switch to delete a preserved file (i.e., a file whose owner's field is greater than 0). (QUEUE).

**?CANT SET OUR SEARCH LIST**

DAEMON tried to set its search list and failed in its attempt. (DAEMON).

**?CANT SET SEARCH LIST = USER'S**

DAEMON attempted to set its file structure search list to be the same as the user's search list. (DAEMON).

**?COMMAND ERROR**

General catch-all error response for most commands. The syntax of the command is in error, and the command cannot be deciphered.

In FILCOM, one of the following errors occurred in the last command string typed.

1. There is no separator (+ or =) between the output and input specifications.
2. The input specification is completely null.
3. The two input files are not separated by a comma.
4. A file descriptor consists of characters other than alphanumeric characters.

(continued on next page)

5. FILCOM does not recognize the specified switch.
6. The project-programmer number is not in standard format, i.e., [proj,prog].
7. The value of the specified switch is not octal.
8. The first input file is followed by a comma but the second input file is null.

?COMMAND SYNTAX ERROR  
TYPE /H FOR HELP

An illegal command string was entered. (GLOB).

?COMMA REQUIRED IN DIRECTORY

A project-programmer number has been specified without the separating comma.  
(DUMP, QUEUE, BACKUP, RESTORE).

CONT BY OPR

The job has been continued by the operator. This message appears on the console of the job being continued. (JCONT).

?CONTROL AND LOG FILES MUST BE DISTINCT

The control file cannot be the same file as the log file. (QUEUE).

?2K CORE NEEDED AND NOT AVAILABLE

FILCOM needs 2K of core to initialize I/O devices and this core is not available from the monitor. (FILCOM).

%CPU<sub>n</sub> OPR1 ACTION REQUESTED

The Job's CPU specification includes a CPU which is not running or is not scheduling jobs. The monitor remembers the specification and uses the CPU as soon as it is started. If at least one CPU is running, the message is printed only once, since the job can run on another CPU.

?DAEMON FILE MUST BE WRITTEN ON A DISK

The device specified was a nondisk device. (DAEMON).

?DAEMON NOT RUNNING

The DAEMON program has not been initialized. It must be started by the operator to allow the DUMP and DCORE commands to operate. (DUMP, DCORE).

?DETACH UO FAILED

DAEMON could not detach itself from the TTY. Note that DAEMON does not detach itself if it is loaded with DDT. (DAEMON).

?DATA ERROR ON DEVICE PTR

A read error has occurred on the paper-tape reader. (COPY program).

**?DESTINATION DEVICE ERROR**

An I/O error occurred on the output device. (GLOB).

**?DEVICE CANT BE REASSIGNED**

(1) The job's controlling terminal cannot be reassigned, (2) the logical name would be duplicated, or (3) the logical name is a physical device name and the job reassigning the device is either logged in under a different project-programmer number or is not the operator.

**?DEVICE ERROR ON OUTPUT DEVICE**

A write error has occurred on the output file. (FUDGE2).

**?DEVICE INIT FAILURE**

The specified device has been assigned to another job or does not exist. (COPY program).

**?DEVICE MTA<sub>n</sub> NEEDS A WRITE RING, INSERT ONE AND TYPE <CR>**

This message is returned by the BACKUP and RESTORE programs.

**?DEVICE MUST BE A DECTAPE**

The only device that can be specified in the COPY command string is the DECTape. (COPY program).

**?DEVICE dev NOT A DIRECTORY DEVICE**

This message is returned by the BACKUP program.

**?DEVICE NOT ASSIGNABLE**

A non-privileged user cannot assign the requested device because it belongs to the restricted pool of devices. The user should try to assign the device with the MOUNT command. (ASSIGN).

**?DEVICE NOT AVAILABLE**

Specified device cannot be initialized because another user is using it or because it does not exist.

**?DEVICE WILDCARD ILLEGAL**

The wildcard construction cannot be used in the device specification. (DUMP, QUEUE, BACKUP, RESTORE).

**?DIALOG MODE NOT SUPPORTED**

The capability of interactive dialogue with the user has not been implemented. (QUEUE).

**?DIRECTORY FULL ON OUTPUT DEVICE**

There is no room in the file directory on the output device to add the updated file (nondisk devices only). (FUDGE2).

device name DISMOUNTED

The DISMOUNT command has completed.

?device name DISMOUNT INCOMPLETE

The DISMOUNT command was unsuccessful. In most cases, the reasons for failure have already been listed by nonerror messages.

DONT KNOW CTY LINE NUMBER

The DCORE command cannot be typed on CTY. (DAEMON).

?DOUBLE DEVICE ILLEGAL

Two device names appeared in a row without an intervening filename, or two colons appeared in a row, e.g., LPT:PTP: or DSKA ::FILEX. (DUMP, QUEUE).

?DOUBLE DIRECTORY ILLEGAL

Two directory names cannot appear without an intervening filename. (DUMP, QUEUE).

?DOUBLE EXTENSION ILLEGAL

Two extensions cannot appear without an intervening filename or comma. (DUMP, QUEUE).

?DOUBLE FILENAME ILLEGAL

Two filenames appeared in a row, or two periods appeared in a row; e.g., Q TEST1 TEST2 or TEXTX..MAC. (DUMP, QUEUE).

DPA<sub>n</sub> NO DRIVE AVAILABLE ON THIS CONTROLLER

The drives on the specified controller are all in use. (MOUNT).

?DSK CANT BE REASSIGNED

An attempt was made to reassign the prototype disk device data block (DDB).

?DSKCHR FAILURE <sub>n</sub> ON UNIT abc

The DSKCHR UJO gave an unexpected error return; <sub>n</sub> is the disk error code. Notify the operator. (DAEMON, KJOB).

%END OF TAPE id ON dev

PLEASE MOUNT TAPE id+1 AND TYPE <CR> TO CONTINUE:

This message is sent to the operator. (RESTORE).

%END OF TAPE id ON dev

PLEASE MOUNT NEXT TAPE AND TYPE <CR> TO CONTINUE

This message is sent to the operator. (BACKUP).

%END OF TAPE n ON MTAn AT time

The end of the tape has been reached. (RESTORE).

\$END OF TAPE n ON  $\left\{ \begin{array}{l} \text{DPA} \\ \text{MTA} \\ \text{DSK} \end{array} \right\} \times \text{AT time}$

This message appears in the log file. (BACKUP).

?ENTER ERROR n  
?DIRECTORY FULL

No additional files can be added to the directory of the output device; n is the disk error code. (GLOB).

?ENTER FAILURE

The DECTape directory is full (i.e., there is no room for the file to be written on the DECTape).

?ENTER FAILURE n

The output filename is null; n is the error code for an illegal filename (nondisk devices only). (FUDGE2).

?ENTER FAILURE FOR INDEX FILE

The ENTER failed for the index file. (BACKUP).

?ENTER FAILURE IN QUEUE MANAGER

QUEUE was unable to enter the files into the output queue. (QUEUE).

?ENTER FAILURE n ON  $\left\{ \begin{array}{l} \text{CCL} \\ \text{DAEMON} \end{array} \right\} \text{FILE}$

The ENTER to write the file failed; n is the disk error code.

?ENTRY BLOCK TOO LARGE PROGRAM name

The entry block of the named program is too large for the FUDGE2 entry table, which allows for 100 entry names. FUDGE2 can be reassembled with a larger table. (FUDGE2).

?ERROR CLOSING OUTPUT, STATUS = n

An I/O error occurred while closing the file on disk; n is the disk error code. (DUMP).

?ERROR IN JOB n

A fatal error occurred in the job or in the monitor while servicing the job. This typeout usually precedes a one-line description of the error.

?EXCEED LOG-OUT m QUOTA BY n BLOCKS

The total number of blocks for all the user's files exceeds the maximum permitted value (m) by the indicated amount n. The user may use PIP or the DELETE command to remove files. Until the user is under the limit, he cannot dismount the file structure. (DISMOUNT).

?EXECUTION DELETED

A program is prevented from being executed because of errors detected during assembly, compilation, or loading. Loading is performed, but the loader exits to the monitor without starting execution. (LOADER).

?EXPECTED FORMAT IS "NNNK" = 16K to 256K

The core-bank specified while processing the /T switch is not within the acceptable range or does not terminate with the letter K; e.g., 32 is not acceptable; 32K is. (COPY program).

%FAILURE ON { ENTER } FOR ERROR FILE--CONTINUING  
                  { OPEN }

The error file could not be generated. The BACKUP program is continuing without one. (BACKUP).

?FAILURE ON { INIT } FOR LOG FILE  
                  { OPEN }

The log file could not be generated. (BACKUP, RESTORE).

%FAILURE OUTPUTTING ERROR FILE--CONTINUING

The error file could not be output. The BACKUP program is continuing its processing. (BACKUP, RESTORE).

%FAILURE { READING } UFD FOR dev [proj,prog]  
                  { CREATING }

The UFD for the named device could not be read (BACKUP) or created (RESTORE).

%FAILURE TO INTERLOCK UFD FOR dev [proj,prog]

The UFD interlock for the named device failed. (BACKUP).

file structure name FILE ERRORS EXIST

One of the files in a file structure has an error status, as flagged in the UFD of that file structure. (LOGIN).

?FILENAME ALREADY IN USE

The specified file already exists. (COMPILE).

?FILENAME REQUIRED FOR INPUT QUEUE

A file cannot be entered into the Batch input queue without a filename. (QUEUE).

**?FILE n NOT IN SAV FORMAT**

The user indicated via the /X switch that the file is to be expanded but the specified file is not in compressed file format. N is either 1 or 2 indicating the first file or the second file. (FILCOM).

**?FILE n READ ERROR**

An error has occurred on either the first or second input device. (FILCOM).

**?FILE SWITCHES ILLEGAL IN OUTPUT FILE**

File switches cannot appear on the left of the equal sign, i.e., in the output specification. (QUEUE).

**? (3) FILE WAS BEING MODIFIED-file.ext**

Another user is modifying the file. (COMPIL).

**? (0) FILE WAS NOT FOUND-file.ext**

The named file could not be located. (COMPIL).

**?FORMAT OR READ ERROR IN AUXACC.SYS**

LOGIN unexpectedly found an end-of-file or an error in AUXACC.SYS. Notify the operator. (LOGIN).

**file.ext FOUND BAD BY FAILSAFE READING MTA**

The file in the file structure has an error status as flagged in the UFD of the file structure. (LOGIN).

**FROM JOB n**

An informative message telling the user the job number to which the console was attached or from which the console is detaching. (ATTACH, DETACH).

**?FUDGE2 SYNTAX ERROR**

An illegal command string was entered; for example, the left arrow was omitted or a program name was specified for the output file. (FUDGE2).

**?GIVING BACK TOO MUCH CORE**

An internal problem in the DUMP program. Notify your system programmer or software specialist. (DUMP).

**?HALT AT USER adr**

The user's program executed a HALT instruction at adr. Typing CONTINUE resumes execution at the effective address of the HALT instruction.

file.ext HARDWARE DATA READ ERROR DETECTED

The file has a hardware data read error flagged in the UFD of the file structure. (LOGIN).

file.ext HARDWARE DATA WRITE ERROR DETECTED

The file has a hardware data write error flagged in the UFD of the file structure. (LOGIN).

?HUNG DEVICE dev

If a device does not respond within a certain period after it is referenced, the system decides that the device is not functioning and outputs this message.

?ILLEGAL BACKUP DEVICE

The BACKUP operations can be done only on disk, magnetic tape, and DECtape. (BACKUP).

?ILLEGAL BLOCK TYPE

While loading the bootstrap loader with the /L switch, an unrecognizable block type was encountered by COPY. (COPY program).

?ILLEGAL COMMAND SYNTAX CHARACTER x

The character x is used incorrectly in the command string. (QUEUE, BACKUP).

?ILLEGAL DATA MODE FOR DEVICE dev AT USER adr

The data mode specified for a device in the user's program is illegal, such as dump mode for the terminal.

?drive ILLEGAL DRIVE NAME

The drive specified by the user is in conflict with the unit or controller type required by the units of the file structure. (MOUNT).

?ILLEGAL IN BATCH JOB

The ATTACH, DETACH, SEND, CCONT, and CSTART monitor commands cannot be used by a batch job.

?ILLEGAL JOB NUMBER

The job number is too large or is not defined in this configuration.

?ILLEGAL QUEUE DEVICE

The queue name specified cannot be used with the given switch. (QUEUE).

?ILLEGAL QUEUE NAME xxx

The queue is not one of the system queues, or the queue is a logical name. (QUEUE).

**?ILLEGAL TO CREATE REQUEST FOR SOMEONE ELSE**

Only the operator logged in under 1,2 can create queueing request for other users. (QUEUE).

**?ILLEGAL UUO AT USER adr**

An illegal UUO was executed at user location adr.

**?ILL INST. AT USER adr**

An illegal operation code was encountered in the user's program.

**?ILL MEM REF AT USER adr**

An illegal memory reference was made by the user's program. If this message occurred on a memory write, the error is at adr-1 since the program counter has been advanced. If it occurred on a memory read, then the illegal instruction is probably in location adr. The user should use the E command to first examine location adr-1 and then location adr in order to determine the illegal instruction. The index registers may also have to be examined.

**?INDEX FILE CANNOT GO TO A LISTING DEVICE**

This message is returned by the BACKUP and RESTORE programs.

**?INPUT AND OUTPUT DECTAPES MAY NOT BE THE SAME DEVICE**

The COPY program performs its operations on an input DECTape and an output DECTape. These DECTapes cannot be the same. (COPY program).

**?INPUT (or OUTPUT) BLOCK TOO LARGE**

A DECTape block number greater than 1101<sub>8</sub> was encountered. (COPY program).

**?INPUT (or OUTPUT) CHECKSUM OR PARITY ERROR**

A read (or write) error has been detected. (COPY program).

**?INPUT DEVICE dev CANNOT DO OUTPUT AT USER adr**

Output was attempted on a device that can only do input (e.g., the card reader).

**?INPUT (or OUTPUT) DEVICE ERROR**

The DECTape control unit has detected the loss of data or a missed block. (COPY program).

**?INPUT DEVICE NOT A DISK**

The input specifications in a QUEUE command must be disk files. (QUEUE).

**?INPUT ERROR**

An I/O error occurred while reading a temporary command file from the disk. File should be rewritten. (COMPILE).

?INPUT ERROR - file.ext FILE NOT FOUND

The specified file could not be found on the input device. (FILCOM).

%INPUT ERROR DSKn  $\left\{ \begin{array}{l} \text{file.UFD} \\ \text{file.MFD} \\ \text{file.SFD} \end{array} \right\}$  [proj,prog]

The BACKUP program cannot access the entries in the named directory. These entries will not be saved on the BACKUP medium. The BACKUP program continues by advancing to the next directory. (BACKUP).

?INPUT ERROR, STATUS = n

An I/O error occurred while reading the file from disk; n is the disk error code. A new INPUT command causes a new LOOKUP to be done. (DUMP, DAEMON).

?INPUT FAILED FOR FILE DSKn file.ext [proj,prog]

The INPUT failed for the specified file. (BACKUP, RESTORE).

?INPUT (or OUTPUT) PREMATURE END OF FILE

When copying a DECTape, COPY encountered the end of file before it expected it. This may happen when copying a PDP-9 DECTape to a PDP-10 DECTape. (COPY program).

?INSUFFICIENT CORE FOR QUEUE

There is not enough core in system at the time of the KJOB command to make an output queue entry. (QUEUE).

?INVALID ARGUMENT

The argument specified on a BACKSPACE or PARITY command is unknown. (BACKUP, RESTORE).

?INVALID ENTRY - TRY AGAIN

#

An illegal project-programmer number or password was entered and did not match identification in system. The user is to retype his project-programmer number and password. (LOGIN).

?I/O TO UNASSIGNED CHANNEL AT USER adr

An attempt was made to do an OUTPUT, INPUT, OUT, or IN to a device that the user's program has not initialized.

?x IS AN ILLEGAL  $\left\{ \begin{array}{l} \text{CHARACTER} \\ \text{SWITCH} \end{array} \right\}$

An illegal character or switch was encountered in the command string. (FUDGE2).

?symbol IS A MULTIPLY DEFINED LOCAL

The named symbol is in more than one symbol table with different values. (DUMP).

**?symbol IS AN UNDEFINED SYMBOL**

The named symbol is not in DUMP's symbol table. (DUMP).

**?symbol IS AN UNDEFINED SYMBOL TABLE NAME**

The named symbol table has not been loaded with an XTRACT command. (DUMP).

**?JOB CAPACITY EXCEEDED**

This message is received by a user who attempts to login after the maximum number of jobs that the system has been set to handle has been initiated. The user should login in at a later time. (LOGIN).

**?JOB NOT WAITING**

The job specified is not waiting to be continued. (JCONT).

**JOB SAVED**

The output is completed.

**JOBn USER [p,p] LOGGED OFF TTY n AT hhmm dd-mm-yy**  
**DELETED <ALL> n FILES**  
**SAVED <ALL> n FILES m TOTAL BLOCKS USED**  
**ANOTHER JOB STILL LOGGED IN UNDER [p,p]**  
**RUNTIME n MIN m SEC**

This information is typed as user logs off successfully. Note that m is total blocks allocated as opposed to blocks written. Therefore, it is always greater than or equal to the number of blocks written. Files are allocated in units of blocks called clusters. The system administrator selects the cluster size for each file structure, usually one block per cluster for FH file structures, and 5 or 10 blocks per cluster for DP file structures. (KJOB).

**?LANGUAGE PROCESSOR CONFLICT**

The use of the + construction has resulted in a mixture of source languages. (COMPIL).

**?LEVEL D ONLY**

The command issued is available only in 5-series monitors.

**?LINKAGE ERROR - RUN UO**

An I/O error occurred while reading a program from the device SYS:. (COMPIL).

**%LISTING DEVICE OUTPUT ERROR, STATUS =**

The device specified for the output has an error. A new OUT command selecting a new file can be given or an OUT and APPEND command sequence to try again. (DUMP).

**?LISTING ENTER FAILURE n**

The ENTER to write the output file failed; n is the disk error code. (QUEUE).

?LISTING OPEN FAILURE ON DEVICE dev

The OPEN failed on device dev. (QUEUE).

?LOCKED-OUT BY OPERATOR

The operator is preventing any new accesses to the file structure in order that it may be removed. (MOUNT).

file structure name LOGGED OUT QUOTA n EXCEEDED BY m BLOCKS

The user's allocation on the file structure named is greater than his logged out quota. The user must go through the CONFIRM dialogue and delete files until he is under the quota allowed to log off. (KJOB, LOGOUT).

%LOGICAL NAME WAS IN USE, DEVICE dev ASSIGNED

The user previously assigned this logical name to another device. The logical name is cleared from the first device and assigned to the second.

?LOGIN PLEASE

A command that requires the user to be logged in has been typed to the monitor; it cannot be accepted until the user performs a LOGIN.

?LOGIN PLEASE TO USE SWITCH CREATE

The user must be logged in to make a new entry into a system queue. (QUEUE).

?LOOKUP ERROR n

?file.ext FILE NOT FOUND

The named file cannot be found in the directory on the specified device. (GLOB)

%LOOKUP ERROR DSKn file [proj,prog]

The BACKUP or RESTORE program cannot access the indicated file and continues by skipping to the next file. (BACKUP, RESTORE).

?LOOKUP FAILED, "BSLDR.REL"

While processing the /L switch, COPY could not find the bootstrap loader named BSLDR.REL. (COPY program).

?LOOKUP FAILURE

The LOOKUP to read the disk file failed. This message is followed by a line explaining the reason for failure. (FUDGE2).

?file structure name LOOKUP FAILURE n

The LOOKUP to read the file failed; n is the disk error code.

?LOOKUP FAILURE FOR INPUT FILE n file

DUMP cannot read the input file. (DUMP).

?LOOKUP FAILURE n ON DAEMON FILE

The LOOKUP to read the DAEMON file failed; n is the disk error code. (DAEMON).

?MAX = n

A value was specified for an argument that is greater than the maximum value (n) allowed. (DUMP).

?MAY NOT LOGIN AS MFD PPN

No one can login as [1, 1] because this number is the project-programmer number of the MFD. (LOGIN).

?MAY NOT LOGIN { LOCAL  
REMOTE  
DATA SET  
BATCH JOB SUBJOB  
REMOTE CTY OR OPR }

ACCT.SYS entry does not permit the project-programmer number to login at the terminal that is being used. (LOGIN).

?MAY NOT LOGOUT WITH FILE STRUCTURES FOR LOGICAL NAMES

A file structure in the job's search list is assigned a logical name, and only physical device names are recognized. The user should deassign the logical names. (KJOB, LOGOUT).

?MEM PAR ERROR AT USER PC adr

The processor detected a memory parity error in the low or high segment while the job was executing. The adr is the address of the PC stored by the hardware rather than the user address of the parity error. The operator also receives an error message giving the range of absolute addresses in case memory reconfiguration is necessary. DAEMON is awakened in order to record the pertinent information about the error for field service personnel.

The user must start a new copy of his program by typing the appropriate monitor command R, RUN, or GET. He should not start the program over by typing START, since the error is likely to reoccur or the program operate with incorrect data.

?MFD { LOOKUP  
READ } FAILURE

The MFD cannot be accessed. (BACKUP).

?MORE THAN ONE { OUTPUT  
INPUT } DEVICE ILLEGAL

Files for the BACKUP operations can be taken from or written to only one device at a time. (BACKUP, RESTORE).

?MORE THAN ONE OUTPUT FILE ILLEGAL

Only one output queue-name may be specified in the QUEUE command string. (QUEUE).

device MOUNTED

The device is mounted and ready for use. The MOUNT command has completed. If a file structure was mounted, a list of the unit ID's and the drives on which they are mounted is output. (MOUNT).

?device MOUNT INCOMPLETE

The MOUNT command has not completed successfully. In most cases, the reasons for failure have already been listed by nonerror messages. In a Batch job, MOUNT INCOMPLETE not preceded by a message may indicate that the user is attempting to mount a spooled device without executing a SET SPOOL command to unspool the device. The user must have unspool privileges in his accounting file entry in order to unspool and mount spooled devices.

?MUST BE IN OWNER'S PROJECT FOR SINGLE ACCESS

The user may not request single-access (/SINGLE switch) unless he has the same project number as the owner of the file structure. This requirement is enforced since a user with single access may execute super-USETI/USETO UUOs. (MOUNT).

name MUST NOT BE A LOGICAL NAME

The structure named contains the operator request queue (3,3.UFD) and must not be the logical name for some other structure. (MOUNT).

?file structure name MUST NOT BE WRITE-PROTECTED

The named structure is being used to queue requests to the operator and therefore may not be write-protected, SETSRC may be used to change the protection. (MOUNT, DISMOUNT, FILE).

NAME:

The ACCT.SYS entry for this project-programmer number requires the user to type a name which matches the one in ACCT.SYS in order to login. (LOGIN).

?NEED 5.03 OR LATER FOR REATTACH COMMAND

The REATTA program depends on UUOs available in the 5.03 release of the monitor. The user attempted to run the program using an older monitor. (REATTA).

?NESTING TOO DEEP

The @ construction exceeds a depth of nine and may be due to a loop of @ command files. (COMPIL).

?NO CORE ASSIGNED

No core was allocated when the GET command was given and no core argument was specified in the GET.

NO DIFFERENCES ENCOUNTERED

No differences were found between the two input files. (FILCOM).

?(1) NO DIRECTORY FOR PROJECT-PROGRAMMER NUMBER - file.ext

A UFD does not exist for the requested project-programmer number. (COMPILE).

?NO END BLOCK ENCOUNTERED

The last block of the bootstrap loader program must be an end block (refer to the MACRO manual). (COPY program).

?NO ENTRY IN AUXACC.SYS  
NO SEARCH LIST OR UFDS CREATED

If the user has no entry in AUXACC.SYS, LOGIN does not create UFDS or a search list. User is logged-in and has UFDS if they existed previously. He may write only on file structures that have UFDS or read all file structures. He may also create a file structure search list with SETSRC. The user can create UFDS on those file structures for which he has an entry in QUOTA.SYS by using the MOUNT command. (LOGIN).

NO ENTRY IN QUOTA.SYS

The user may utilize the file structure, but no UFD is created if he does already have one. (MOUNT).

%NO INFO ON "name"

The user specified a feature that has no available documentation. (HELP).

?NO INPUT DEVICE SPECIFIED  
SPECIFY INPUT DEVICE NOW:

An input device name was not specified prior to the START command. (RESTORE).

?NO MODIFIER ALLOWED IN SWITCH switch

The switch specified cannot have an argument. (QUEUE).

NONE PENDING

None of the user's requests to the operator are pending.

?NON-EXISTENT DRIVE DPAn

The user has specified a drive that does not exist in the system. (MOUNT).

%NON-EXISTENT FILE input specification

The file specified for input could not be found. This message is not output if the /NEW switch is specified for the file. (QUEUE).

?NON-EX MEM AT USER adr

Usually due to an error in the monitor.

?NO OPR. JOB FOR THIS REQUEST

An operator request has been issued, but there is no OMOUNT running and enabled to service the request. The request is still queued unless the /PAUSE switch was given.

?NO OUTPUT DEVICE SPECIFIED  
SPECIFY OUTPUT DEVICE NOW:

An output device name was not specified prior to the START command. (BACKUP).

?NO PRIVILEGES TO SET CPU

The user does not have the privilege bits set by LOGIN from ACCT.SYS to change the CPU specification. The user should request that these privilege bits be set by the system manager.

?NO PRIVS TO UNSPOOL

The user does not have privileges to unspool devices, and the operator has not set bit 28 in the STATES word.

?NO REMOTE USERS. TRY AGAIN LATER

The operator has used the SET SCHEDULE command to prevent LOGINs from remote terminals. The message of the day is still typed. (LOGIN).

NO ROOM IN QUEUE, TRY AGAIN LATER

There is no room in the queue for the user's request to be sent to the operator. (MOUNT).

?(14) NO ROOM OR QUOTA EXCEEDED - file.ext

There is no room on the file structure or the user's quota on the file structure has been exceeded.

%NO RUNNING CPUS IN SPECIFICATION

If none of the CPUs in the job's CPU specification are running, the user receives this message every minute until the CPU is started or he types a new SET CPU command.

?NO START ADR

Starting address or reenter address is zero, because the user failed to specify the starting address in the END statement of the source program or in the START command. However, an implicit starting address of 0 may be specified.

?NO SUCH DEVICE

The device name does not exist or was not assigned to this job.

?NO SUCH JOB

An attempt was made to attach to a job that has not been initialized.

?NO SUCH STR

A nonexistent file structure was specified. (KJOB).

**?NO SUCH TTY**

The terminal number is not part of the system configuration.

**?NO SUCH UNIT**

The unit does not exist or all units of this type are in use.

**?NOT A JOB**

The job number is not assigned to any currently running job. (ATTACH, DSK, JCONT).  
There is no job logged in at this terminal. (CONTINUE).

**?NOT A SAVE FILE**

The file is not a core image file.

**?NOT A SPOOLING DEVICE**

The device specified is not one of the spooling devices (LPT, CDP, CDR, PTP, PLT).

**?NOT A STR - TRY AGAIN**

The file structure specified is not recognized by the monitor.

**?NOT A TTY**

The device name given is not a terminal. (REATTN).

**?drive NOT AVAILABLE**

The drive indicated by the user is not currently available. (MOUNT).

**?command NOT CODED**

A command that is not in this version of DUMP was specified in the command string. (DUMP).

**?NOT ENOUGH ARGUMENTS**

An insufficient number of files of one type has been specified. (FUDGE2).

**?NOT ENOUGH CORE**

The system cannot supply enough core to use as buffers or to read in a system program.  
(COMPILE).

**NOT ENOUGH DRIVES AVAILABLE**

There are currently not enough drives of the right type to mount the file structure. (MOUNT).

**NOT ENOUGH TABLE SPACE FOR SWAPPING UNITS**

There are more swapping units than DAEMON allowed for. DAEMON should be reassembled.  
(DAEMON).

?dev file.ext program NOT FOUND

The file or the program was not found on the device or in the file specified. If a program name is printed, this message may indicate that the program names in the command string appear in a sequence different from their sequence within the file. Therefore, the program may actually exist but was missed because of the incorrect sequence in the command string. (FUDGE2).

?file.SAV NOT FOUND

The program file requested cannot be found on the system device or the specified device.

drive NOT READY

The indicated drive is either off-line or physically write-locked when write-enabled was requested. The operator will be notified. (MOUNT).

?NOT YET SUPPORTED COMMAND CODE switch

A switch has been specified that is not implemented. (QUEUE).

NO UFD CREATED

The user may access the file structure, but he cannot write in his disk area since he has no UFD. (MOUNT).

?NULL DEVICE ILLEGAL

A colon has been found without a preceding device name. (QUEUE, BACKUP, RESTORE).

?NXM adr

While computing the value of an expression, a non-existent location was specified when referencing the input file. (DUMP).

?nk OF CORE NEEDED or ?nP OF CORE NEEDED

There is insufficient free core to load the files; n is the size being requested for the segment that failed (either high or low segment, not the sum of the high and low segments). This message occurs when the virtual core for the system has been exceeded or the core for this job has been exceeded. The user should type CORE, to determine what core has been exceeded, and whether the high or low segment was too big. K denotes 1024 words which is the unit of core allocation on a KA10-based system, and P denotes 512 words (one page) which is the unit of allocation on a KI10-based system.

?OFFSET = 1000 TO 777600 (OCTAL)

The offset specified by the user is not within the acceptable range. (COPY program).

?ONLY BATCH USERS MAY LOGIN. TRY AGAIN LATER

The operator has used the SET SCHEDULE command to prevent LOGINS, except for BATCH jobs. The message of the day is still typed. (LOGIN).

**?OPEN FAILED FOR DEVICE dev**

The OPEN for the named device failed. (BACKUP, RESTORE).

**?OPEN FAILURE ON DATA DEVICE dev**

The OPEN on the specified device failed. (DUMP).

**OPERATOR BUSY, HANG ON PLEASE.**

The user must wait for the operator to become available.

**OPERATOR NOTIFIED**

- (1) The operator is available and the user may continue typing his message. (PLEASE).
- (2) A request is queued to the operator to perform a specified action. (MOUNT, DISMOUNT).

**OPERATOR REQUESTED TO MOUNT UNITS**

A request is queued to the operator to mount and ready the packs on the proper drives. (MOUNT).

**OPERATOR REQUESTED TO READY DRIVES**

One or more drives (as specified by previous messages) are not ready. A request is queued to the operator. (MOUNT).

**OPERATOR REQUESTED TO REMOVE PACKS**

A request to physically remove the packs has been queued to the operator. (DISMOUNT).

**OTHER USERS - CANNOT SINGLE ACCESS**

Other users are currently using the file structure that has been specified with the single-access switch (/SINGLE). The switch is ignored. (MOUNT).

**OTHER USERS - CANT REMOVE**

A DISMOUNT command requesting physical removal (/REMOV switch) of a pack has been issued and there are other users of the pack. The switch is ignored. (DISMOUNT).

**OTHER USERS SAME PPN**

A program has determined that other jobs are currently logged-in under the same project-programmer number. (LOGIN, KJOB).

**?OUT OF BOUNDS**

The specified adr is not in the user's core area, or the high segment is write-protected and the user does not have privileges to the file that initialized the high segment. (D, E).

**?OUTPUT DEVICE dev CANNOT DO INPUT AT USER adr**

An attempt was made to input from an output device (e.g., the line printer).

?OUTPUT DEVICE ERROR

An error has occurred on the output device. (FILCOM).

?OUTPUT ERROR

An I/O error occurred while writing a temporary command file on disk. (COMPIL).

?OUTPUT ERROR, STATUS = n

An I/O error occurred while writing the file on disk; n is the disk error code. (DAEMON).

?OUTPUT INITIALIZATION ERROR

The output device cannot be initialized for one of the following reasons:

1. The device does not exist or is assigned to another job.
2. The device is not an output device.
3. The file cannot be placed on the output device. (FILCOM).

PASSWORD:

The user must type a PASSWORD which matches that in the ACCT.SYS entry for this project-programmer number. Echoing is suppressed to preserve PASSWORD security. If the user is at a half-duplex (local copy) terminal, this message is replaced by a sequence of random over-typed characters, over which the user types his PASSWORD. (LOGIN).

PAUSE... (IC TO QUIT, CR TO CONT)

The /PAUSE switch has been specified, and an operator action is about to be requested. IC aborts the command before the request is queued to the operator. Carriage return-line feed allows the command to continue, and the request is queued to the operator. (DISMOUNT).

?PC OUT OF BOUNDS AT USER adr

An illegal transfer has been made by the user program to user location adr.

?PLEASE KJOB OR DETACH

Attempt was made to LOGIN a job when the user already has a job initialized at that terminal. (LOGIN).

?PLEASE LOGIN AS [OPR]

The operator is the only person that can initialize DAEMON by typing R DAEMON.

?PLEASE TYPE IC FIRST

A command which would start a job has been issued after a CSTART or CCONT.

?PPN HAS EXPIRED

The current date is greater than the expiration date of the project-programmer number. The user may not login until expiration date is changed by the system manager. (LOGIN).

**?PROGRAM ERROR WHILE RESETTING MASTER DEVICE**

FUDGE2 cannot find the master device or cannot find the program on the master device. (FUDGE2).

**?PROJECT 1 MAY NOT BE PTY**

Project 1 is never allowed to login over a pseudo-TTY. (LOGIN).

**?PROTECTION FAILURE DSK file.ext [proj,prog]**

The user does not have access to the specified disk areas for either a read or a write. (BACKUP, RESTORE).

**?(2) PROTECTION FAILURE - file.ext**

There was a protection failure or the directory on DECtape had no room for the file. (COMPIL).

**?PTR INIT FAILURE**

The logical device PTR is not available or could not otherwise be initialized. (COPY program).

**QUOTA.SYS LOOKUP FAILURE**

The LOOKUP to read QUOTA.SYS failed. (MOUNT).

**QUOTA.SYS NOT ON STRUCTURE**

QUOTA.SYS is not part of this structure. The user may still use the file structure, but no UFD will be created. (MOUNT).

**QUOTA.SYS READ ERROR**

An I/O error occurred while reading QUOTA.SYS. (MOUNT).

**QUOTA.SYS WRONG FORMAT VERSION**

Wrong version of QUOTA.SYS is on the file structure being mounted. Consult the operator. (MOUNT).

**%READ ERROR DSKn file [proj,prog]**

The BACKUP or RESTORE program cannot input the designated file. (BACKUP, RESTORE).

**?file structure name RENAME FAILURE n**

The RENAME to change the protection of the file failed; n is the disk error code. (KJOB, LOGOUT).

**?(4) RENAME FILENAME ALREADY EXISTS - file.ext**

The new filename on a RENAME command already exists. (COMPIL).

REQUEST STORED  
n COMMANDS IN QUEUE

The request typed by the user has been placed in a queue to be performed when possible. n is the number of requests in the queue for all users. (FILE, MOUNT, DISMOUNT).

?REQUIRES DEVICE NAME

The device name or file structure name is required with the MOUNT and DISMOUNT commands.

\$RESTOR COMPLETED AT time

The RESTORE program has successfully completed. (RESTORE).

?RIGHT BRACKET REQUIRED IN DIRECTORY

The project-programmer number must be enclosed in square brackets. (QUEUE).

%SEARCH LIST DOES NOT ALLOW CREATES

There are no file structures available to the user on which he can write. Run MOUNT or SETSRC to modify the search list as necessary. (LOGIN).

%SEARCH LIST ERROR [proj,prog]

The BACKUP or RESTORE program is unable to obtain the search list for the named project-programmer number. The program advances to the next project-programmer number. (BACKUP, RESTORE).

%SEARCH LIST IS EMPTY

There are no file structures in the DSK: search list that are available to the user. He can run the SETSRC program to modify his search list. (LOGIN).

?SINGLE-ACCESS BY JOB n

The file structure is already single access by the indicated user. (MOUNT).

file.ext SOFTWARE CHECKSUM OR REDUNDANCY ERROR

The file has no error as flagged in the UFD of the file structure. (LOGIN).

?SOME OTHER TIME

The user is not scheduled to LOGIN at this time. He should try again when he is allowed to login. (LOGIN).

?SORRY, CANT OPEN DSK, PLEASE CALL THE OPERATOR

This message is returned from the GRIPE program.

?SORRY, CANT WRITE IN COMPLAINT AREA, PLEASE CALL THE OPERATOR

This message is returned from the GRIPE program.

?SORRY, COMPLAINT BASKET IS FULL, PLEASE CALL THE OPERATOR

This message is returned from the GRIPE program.

?SORRY, NO UFD FOR COMPLAINT BASKET, PLEASE CALL THE OPERATOR.

This message is returned from the GRIPE program.

START OF { BACKUP }  
          { RESTORE } VERSION n AT time YEAR nn DAY dd

The BACKUP or RESTORE program is beginning its processing. (BACKUP, RESTORE).

?STATION NOT IN CONTACT

The requested station is not in contact with the central station. (LOCATE).

?STATION NUMBER INVALID

The requested station number is not recognized by the system. (LOCATE).

STRUCTURE ALREADY MOUNTED

The requested file structure already exists and does not need to be physically mounted. (MOUNT).

?STRUCTURE NOT IN STRLST.SYS

The file structure name does not exist in the system administrator's file SYS:STRLST.SYS and, therefore, is not defined for the system. The operator or administrator may be requested to define the file structure by adding it to STRLST.SYS with the REACT program. (MOUNT).

?STRUUO FAILURE

The STRUUO UUO gave an error return. Notify the operator. (KJOB, LOGOUT).

%SUPERSEDING EXISTING FILE

A warning message indicating that a file already exists with the specified name. This file is being superseded. (TECO).

%SWAP READ ERROR UNIT abc STATUS = n

An I/O error occurred while reading the swapping space. The data is written into the DAEMON file as read. (DCORE).

?SWITCH ERROR

An illegal switch specification was given. (COPY program).

?switch SWITCH ILLEGAL

The switch specified cannot be used with the given queue name. (QUEUE, BACKUP, RESTORE).

?SWITCH VALUE TOO LARGE x

The value given to the switch exceeds the maximum value. (QUEUE).

?SYNTAX ERROR

There is a syntax error in the command string. Check for incorrect parentheses or two operators in a row.

?SYSSTR FAILURE

The SYSSTR UUO gave an error return. Notify the operator. (KJOB, LOGOUT).

?SYSTEM ERROR - xxxxxx

System errors designate operator or system errors and are not a direct fault of the user. They are typed for possible diagnostic used.

?SYSTEM NOT AVAILABLE

The operator has used the SET SCHED command to prevent LOGINs from timesharing terminals. The message of the day is still typed. (LOGIN).

?TABLE OVERFLOW - CORE UUO FAILED TRYING TO EXPAND TO xxx

The GLOB program requested additional core from the monitor, but none was available. (GLOB).

?THIS MONITOR WAS BUILT FOR A xxx AND WILL NOT RUN PROPERLY ON A yyy

The monitor is not running on the machine for which it was built. xxx and yyy are PDP-6, KA10, or KI10.

?TIME LIMIT EXCEEDED

The time limit allocated for the job has been reached. The job is stopped and the terminal is returned to monitor mode.

TIMESHARING WILL CEASE IN m HOURS n MINUTES

The KSYS command (OPSER) or SET KSYS UUO has been issued in order to stop timesharing on the system at the indicated time.

?TOO FEW ARGUMENTS

A command has been typed, but necessary arguments are missing.

?TOO MANY FILENAMES OR PROGRAM NAMES

More than 40 program names or filenames were specified in the command string. The user should separate the job into several segments. (FUDGE2).

?TOO MANY FILE STRUCTURES

The number of file structures exceeds the capacity of the monitor data base. The current limit is 14<sub>10</sub>. (ONCE ONLY).

**?TOO MANY NAMES or ?TOO MANY SWITCHES**

Command string complexity exceeds table space in the COMPIL program. (COMPIL).

**?TRANSMISSION ERROR**

During a SAVE, GET, or RUN command, the system received parity errors from the device, or was unable to read the user's file in some other way. This can be as simple as trying to write on a write-locked tape.

**?TRANSMISSION ERROR ON INPUT DEVICE dev**

A transmission error has occurred while reading data from the specified device. (FUDGE2).

**?TRIED TO OVERWRITE DATA WORD**

After writing the core image file, DAEMON backs up to overwrite a word not known previously (e.g., the length of the category). In overwriting the word, DAEMON encountered a deviation from the standard pattern used in originally writing the word. (DAEMON).

**?TRY LARGER ARG**

The specified argument is too small for the program. This message is followed by the standard output. (CORE).

**?TTY<sub>n</sub> ALREADY ATTACHED**

Job number is erroneous and is attached to another console, or another user is attached to the job.

**?TTY IN USE**

The terminal requested is already controlling a job or is otherwise in use. (REATTA).

**TYPE CORE BANK OR OFFSET FOR DTBOOT**

On a /T switch, COPY asks for a core bank or offset for the bootstrap loader. The core bank is 16K to 256K and the offset is 1000 to 777600 octal. (COPY program).

**TYPE H FOR HELP**

An unintelligible response or command has been typed. Either the filename or the CONFIRM: message is repeated, depending upon what was typed. (KJOB).

**?UFD ENTER FAILURE n**

Failure in trying to create UFD; n is the disk error code. Notify the operator. (LOGIN).

**%UFD ERROR DSK<sub>n</sub> [proj, prog]**

The BACKUP or RESTORE program cannot access the UFD (LOOKUP failure). It advances to the next UFD. (BACKUP, RESTORE).

?file structure name UFD INTERLOCK BUSY

Could not get UFD interlock when trying to set up a UFD. The UFD is not currently set up. Notify the operator. (LOGIN).

?UFD LOOKUP FAILURE n

A failure occurred in setting up a UFD; n is the disk error code. Notify the operator. (LOGIN).

?UFD OUTPUT FAILURE n

The output failed when trying to create the UFD (4-series); n is the software channel status. (LOGIN).

?file structure name UFD READ ERROR, STATUS = n

A read error occurred while reading the user's UFD on the file structure. Status n tells which error occurred. Notify the operator. (KJOB, LOGOUT).

?UFD RENAME FAILURE n

A failure occurred in setting up a UFD; n is the disk error code. Notify the operator. (LOGIN).

?UNDEFINED SWITCH switch

The specified switch is either undefined or not unique. (MOUNT, DISMOUNT).

?UNEQUAL NUMBER OF MASTER AND TRANSACTION PROGRAMS

On a replace request, the number of master programs (or files) does not equal the number of transaction programs (or files). (FUDGE2).

UNIT abc ALREADY MOUNTED ON DRIVE DPAn

The file structure is already mounted but is on different drives than the user specified. (MOUNT).

?UNKNOWN COMMAND

The monitor passed a command to COMPIL which COMPIL does not recognize. (COMPIL).

?UNKNOWN DEFAULT FOR SWITCH switch

The default condition is not known for the specified switch. (DUMP, QUEUE).

?UNKNOWN OR INVALID COMMAND - TYPE GO TO CONTINUE

This message is typed by the BACKUP and RESTORE programs.

?UNKNOWN SWITCH switch

The switch named has been mistyped. (DUMP, QUEUE).

**?UNKNOWN SWITCH VALUE n**

The argument specified with the switch has been mistyped. (DUMP, QUEUE).

**?UNRECOGNIZABLE SWITCH**

An ambiguous or undefined word followed a slash. (COMPIL).

**?UUO AT USER adr**

This message accompanies many error messages and indicates the location of the UUO that was the last instruction the user program executed before the error occurred.

**n VERIFICATION ERRORS**

On a word by word comparison requested via the /V switch, n discrepancies have been detected between the input DECTape and output DECTape. (COPY program).

**WAITING...**

A request has been queued to the operator and the command is waiting for the operator to complete the request. If the user does not want to wait for completion of the operator's action, he can type control-C without aborting the command. The operator action will still be completed. Later a DISMOUNT/CHECK or MOUNT/CHECK can be given to check for completion. (MOUNT, DISMOUNT).

**WAIT PLS**

The system's primary accounting file FACT.SYS was busy. It is retried for ten seconds before FACT.X01 is tried. This message can appear if many users are logging in simultaneously. (LOGIN, KJOB, LOGOUT).

**%WARNING - INPUT REQUEST USES ONLY TWO ENTRIES**

Only two files can be specified in the input queue request, the control file and the log file. (QUEUE).

**!WARNING NO INDEX ON OUTPUT FILE-CONTINUING**

The user has changed the structure of the index library file when deleting, appending, or inserting, thereby invalidating the index. The index has been removed from the new file. Re-indexing is required. (FUDGE2).

**?dev WASNT ASSIGNED**

The device is not currently assigned to the user's job and cannot be deassigned or reassigned by the job.

**?WASNT DET**

The specified device is not detached.

?WILDCARD ILLEGAL IN INPUT QUEUE FILE { NAME  
DIRECTORY  
EXTENSION }

The wildcard construction cannot be used when specifying the Batch input queue. (QUEUE).

?WILDCARD ILLEGAL IN OUTPUT { NAME  
DIRECTORY  
EXTENSION }

The wildcard construction cannot be used in the output queue specification. (QUEUE).

?WRITE LOCK ERROR

An attempt was made to write on a write-locked DECTape. (COPY program).

?WRONG FORMAT FOR SYMBOL

A symbol was given in the format program :symbol and a symbol name did not follow the colon; in other words, the colon must be followed by a symbol. (DUMP).

?WRONG FORMAT VERSION NUMBER IN SYSTEM FILES

Wrong version of ACCT.SYS or AUXACC.SYS is on the system. Consult the operator so that he can run REACT to change the accounting files. (LOGIN).

YOU ARE LOGGED IN AS n,m

When a user logs in with a unique programmer number (project, #), this message informs him of the project-programmer number that LOGIN assigned. (LOGIN).

?YOU DONT HAVE PRIVILEGES TO WRITE { DAEMON  
CCL } FILE

The user attempted to write in a file to which he did not have access. (DAEMON).

?1+1nK CORE  
VIR. CORE LEFT = 0

The swapping space or the core allocated to timesharing is all in use (i.e., there is no available virtual core). The user should wait a few minutes, and then attempt to login again. If this message still appears, it should be reported to the operator.

m+n/p CORE  
VIR. CORE LEFT = v

Key: m = number of blocks in low segment.  
n = number of blocks in high segment.  
p = maximum core per job. (Maximum physical user core unless limited by operator, or there are jobs locked in core (refer to DECsystem-10 Monitor Calls)).  
v = number of K blocks unassigned in core and on the swapping device.

Note that nK represents 1024-word blocks which is the unit of core allocation on a KA10-based system, and nP represents 512-word blocks which is the unit of allocation on a KI10-based system.

4.2 ERROR CODES

The following error codes are returned in AC on RUN and GETSEG UUOs, in location E + 1 on 4-word argument blocks of LOOKUP, ENTER, and RENAME UUOs, and in the right half of location E + 3 on extended LOOKUP, ENTER, and RENAME UUOs. The codes are defined in the S.MAC monitor file.

Table 4-1  
Error Codes

| Symbol | Code | Explanation                                                                                                                                                                                                                                                                                                              |
|--------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ERFNF% | 0    | File not found, illegal filename (0,*), or filenames do not match (UPDATE).                                                                                                                                                                                                                                              |
| ERIPP% | 1    | UFD does not exist on specified file structures. (Incorrect project-programmer number.)                                                                                                                                                                                                                                  |
| ERPRT% | 2    | Protection failure or directory full on DTA.                                                                                                                                                                                                                                                                             |
| ERFBM% | 3    | File being modified (ENTER).                                                                                                                                                                                                                                                                                             |
| ERAEF% | 4    | Already existing filename (RENAME) or different filename (ENTER after LOOKUP).                                                                                                                                                                                                                                           |
| ERISU% | 5    | Illegal sequence of UUOs (RENAME with neither LOOKUP nor ENTER, LOOKUP after ENTER).                                                                                                                                                                                                                                     |
| ERTRN% | 6    | <ul style="list-style-type: none"> <li>a. Transmission, device, or data error (RUN, GETSEG only).</li> <li>b. Hardware-detected device or data error detected while reading the UFD RIB or UFD data block.</li> <li>c. Software-detected data inconsistency error detected while reading the UFD or file RIB.</li> </ul> |
| ERNSF% | 7    | Not a saved file (RUN, GETSEG only).                                                                                                                                                                                                                                                                                     |
| ERNEC% | 10   | Not enough core (RUN, GETSEG only).                                                                                                                                                                                                                                                                                      |
| ERDNA% | 11   | Device not available (RUN, GETSEG only).                                                                                                                                                                                                                                                                                 |
| ERNSD% | 12   | No such device (RUN, GETSEG only).                                                                                                                                                                                                                                                                                       |
| ERILU% | 13   | Illegal UO (GETSEG only). No two-register relocation capability.                                                                                                                                                                                                                                                         |
| ERNRM% | 14   | No room on this file structure or quota exceeded (over-drawn quota not considered).                                                                                                                                                                                                                                      |
| ERWLK% | 15   | Write-lock error. Cannot write on file structure.                                                                                                                                                                                                                                                                        |
| ERNET% | 16   | Not enough table space in free core of monitor.                                                                                                                                                                                                                                                                          |
| ERPOA% | 17   | Partial allocation only.                                                                                                                                                                                                                                                                                                 |
| ERBNF% | 20   | Block not free on allocated position.                                                                                                                                                                                                                                                                                    |

(continued on next page)

Table 4-1 (Cont)  
Error Codes

| Symbol | Code | Explanation                                                                                                                                                                                                     |
|--------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ERNSD% | 21   | Cannot supersede an existing directory (ENTER).                                                                                                                                                                 |
| ERDNE% | 22   | Cannot delete a non-empty directory (RENAME).                                                                                                                                                                   |
| ERSNF% | 23   | Sub-directory not found (some SFD in the specified path was not found).                                                                                                                                         |
| ERSLE% | 24   | Search list empty (LOOKUP or ENTER was performed on generic device DSK and the search list is empty).                                                                                                           |
| ERLVL% | 25   | Cannot create a SFD nested deeper than the maximum allowed level of nesting.                                                                                                                                    |
| ERNCE% | 26   | No file structure in the job's search list has both the no-create bit and the write-lock bit equal to zero and has the UFD or SFD specified by the default or explicit path (ENTER on generic device DSK only). |
| ERSNS% | 27   | A GETSEG from a locked low segment is not for a high segment that is a dormant, active, or idle segment.                                                                                                        |



## APPENDIX A STANDARD FILENAME EXTENSIONS

Table A-1  
Filename Extensions

| Filename Extension | Meaning                                                                                            | Type of File  |
|--------------------|----------------------------------------------------------------------------------------------------|---------------|
| AID                | Source file in AID language                                                                        | Source        |
| ALG                | Source file in ALGOL language                                                                      | Source        |
| ALP                | Printer forms alignment                                                                            | ASCII         |
| ATO                | OPSER automatic command file                                                                       | ASCII         |
| B10                | Source file in BLISS-10                                                                            | Source        |
| B11                | Source file in BLISS-11                                                                            | Source        |
| BAC                | Output from the BASIC Compiler                                                                     | Object        |
| BAK                | Backup file from TECO or LINED                                                                     | Source        |
| BAS                | Source file in BASIC language                                                                      | Source        |
| BCM                | Listing file created by FILCOM (binary compare)                                                    | ASCII         |
| BIN                | Binary file for PDP-8 (DC68A)                                                                      | Object        |
| BKP                | Index file created by the BACKUP program                                                           | ASCII         |
| BLB                | Blurb file                                                                                         | ASCII         |
| BLI                | Source file in BLISS language                                                                      | Source        |
| BUG                | Saved to show a program error                                                                      | Object        |
| CAL                | CAL data and program files                                                                         | Object        |
| CBL                | Source file in COBOL language                                                                      | Source        |
| CCL                | Alternate convention for command file (@ command file construction for programs other than COMPIL) | ASCII         |
| CCO                | Listing of modifications to non resident software                                                  | ASCII         |
| CDP                | Spoiled output for card punch                                                                      | ASCII, Binary |
| CKP                | Checkpoint core image file created by COBOL operating system                                       | Binary        |

(continued on next page)

Table A-1 (Cont)  
Filename Extensions

| Filename Extension | Meaning                                                                     | Type of File  |
|--------------------|-----------------------------------------------------------------------------|---------------|
| CHN                | CHAIN file                                                                  | Object        |
| CMD                | Command file for indirect commands (@ construction for COMPIL)              | ASCII         |
| CMP                | Complaint file by GRIPE                                                     | ASCII         |
| COR                | Correction file for SOUP                                                    | ASCII         |
| CRF                | CREF (cross-reference) input file                                           | ASCII         |
| CTL                | MP batch control file                                                       | ASCII         |
| DAE                | Default output for DAEMON-taken core dumps                                  | Binary        |
| DAT                | Data (FORTRAN) file                                                         | ASCII, Binary |
| DDT                | Input file to FILDDT                                                        | ASCII         |
| DIR                | Directory from FILE command or DIRECT program                               | ASCII         |
| DMP                | COBOL compiler dump file                                                    | ASCII         |
| DOC                | Listing of modifications to the most recent version of the software         | ASCII         |
| DSE                | Directory sorted by extension                                               | ASCII         |
| DSF                | Directory sorted by filename                                                | ASCII         |
| ERR                | Error message file                                                          | ASCII         |
| F4                 | Source file in FORTRAN language                                             | Source        |
| FAL                | Source file in FAIL language                                                | Source        |
| FLO                | English language flowchart                                                  | ASCII         |
| FRM                | Blank form for handwritten records                                          | ASCII         |
| FUD                | FUDGE2 listing output                                                       | ASCII         |
| HGH                | Nonsharable high segment of a two-segment program (created by SAVE command) | Object        |
| HLP                | Help files containing switch explanations, etc.                             | ASCII         |
| IDA                | COBOL ISAM data file                                                        | ASCII, Binary |
| IDX                | Index file of a COBOL ISAM file                                             | ASCII, SIXBIT |
| INI                | Initialization file                                                         | ASCII, Binary |
| LAP                | Output from the LISP compiler                                               | ASCII         |
| LIB                | COBOL source library                                                        | ASCII         |
| LOG                | MP batch log file                                                           | ASCII         |

(continued on next page)

Table A-1 (Cont)  
Filename Extensions

| Filename Extension | Meaning                                                        | Type of File  |
|--------------------|----------------------------------------------------------------|---------------|
| LOW                | Low segment of a two-segment program (created by SAVE command) | Object        |
| LPT                | Spooled output for line printer                                | ASCII         |
| LSD                | Default output for DUMP program                                | ASCII         |
| LSP                | Source file in LISP language                                   | Source        |
| LSQ                | Queue listing created by QUEUE program                         | ASCII         |
| LST                | Listing data created by assemblers and compilers               | ASCII         |
| MAC                | Source file in MACRO language                                  | Source        |
| MAN                | Manual (documentation) file                                    | ASCII         |
| MAP                | Loader map file                                                | ASCII         |
| MEM                | Memorandum file                                                | ASCII         |
| MIM                | Snapshot of MIMIC simulator                                    | Binary        |
| MSB                | Music compiler binary output                                   | Object        |
| MUS                | Music compiler input                                           | Source        |
| OLD                | Backup source program                                          | Source        |
| OPR                | Installation and assembly instructions                         | ASCII         |
| OVR                | COBOL overlay file                                             | Object        |
| PAL                | Source file in PAL 10 (PDP-8 assembler)                        | Source        |
| P11                | Source program in MACX11 language                              | Source        |
| PL1                | Source file in PL1 language                                    | Source        |
| PLT                | Spooled output for plotter                                     | ASCII         |
| PTP                | Spooled output for paper-tape punch                            | ASCII, Binary |
| Qxx                | BAK files (all xx)                                             | ASCII         |
| QUD                | Queued data file                                               | ASCII, Binary |
| QUE                | Queue request file                                             | Binary        |
| QUF                | Master queue and request file                                  | Binary        |
| REL                | Relocatable binary file                                        | Object        |
| RIM                | RIM loader file                                                | Object        |
| RMT                | Read-in mode (RIM) format file (PIP)                           | Object        |
| RNC                | RUNOFF input for producing a .CCO file                         | ASCII         |
| RND                | RUNOFF input for producing a .DOC file                         | ASCII         |

Table A-1 (Cont)  
Filename Extensions

| Filename Extension | Meaning                                                                       | Type of File  |
|--------------------|-------------------------------------------------------------------------------|---------------|
| RNO                | Programming specifications in RUNOFF input                                    | ASCII         |
| RNP                | RUNOFF input for producing a .OPR file                                        | ASCII         |
| RSP                | Script response time log file                                                 | ASCII         |
| RST                | Index file created by the RESTORE program                                     | ASCII         |
| RTB                | Read-in mode (RIM10B) format file (PIP)                                       | Object        |
| SAV                | Low segment from a one-segment program (created by SAVE command)              | Object        |
| SCM                | Listing file created by FILCOM (source compare)                               | ASCII         |
| SCP                | SCRIPT control file                                                           | ASCII         |
| SEQ                | Sequential COBOL data file, input to ISAM program                             | ASCII, SIXBIT |
| SFD                | Subfile directory (restricted usage)                                          | Binary        |
| SHR                | Sharable high segment file of a two-segment program (created by SAVE command) | Object        |
| SNO                | Source file in SNOBOL language                                                | Source        |
| SNP                | Snapshot of disk by DSKLST                                                    | ASCII         |
| SRC                | Source files                                                                  | ASCII         |
| SVE                | .SAVed file from a single user monitor                                        | Object        |
| SYS                | Special system files                                                          | Binary        |
| TEC                | TECO macro                                                                    | ASCII         |
| TEM                | Temporary files                                                               | ASCII, Binary |
| TMP                | Temporary files                                                               | ASCII, Binary |
| TXT                | Text file                                                                     | ASCII         |
| UFD                | User file directory (restricted usage)                                        | Binary        |
| UPD                | Updates flagged in margin (FILCOM)                                            | ASCII         |
| WCH                | SCRIPT monitor (WATCH) file                                                   | ASCII         |
| XPN                | Expanded save file (FILEX)                                                    | Object        |

## APPENDIX B CARD CODES

Table B-1  
ASCII Card Codes

| ASCII Character | Octal Code | Card Punches | ASCII Character | Octal Code | Card Punches |
|-----------------|------------|--------------|-----------------|------------|--------------|
| NULL            | 00         | 12-0-9-8-1   | @               | 100        | 8-4          |
| CTRL-A          | 01         | 12-9-1       | A               | 101        | 12-1         |
| CTRL-B          | 02         | 12-9-2       | B               | 102        | 12-2         |
| CTRL-C          | 03         | 12-9-3       | C               | 103        | 12-3         |
| CTRL-D          | 04         | 9-7          | D               | 104        | 12-4         |
| CTRL-E          | 05         | 0-9-8-5      | E               | 105        | 12-5         |
| CTRL-F          | 06         | 0-9-8-6      | F               | 106        | 12-6         |
| CTRL-G          | 07         | 0-9-8-7      | G               | 107        | 12-7         |
| CTRL-H          | 10         | 11-9-6       | H               | 110        | 12-8         |
| TAB             | 11         | 12-9-5       | I               | 111        | 12-9         |
| LF              | 12         | 0-9-5        | J               | 112        | 11-1         |
| VT              | 13         | 12-9-8-3     | K               | 113        | 11-2         |
| FF              | 14         | 12-9-8-4     | L               | 114        | 11-3         |
| CR              | 15         | 12-9-8-5     | M               | 115        | 11-4         |
| CTRL-N          | 16         | 12-9-8-6     | N               | 116        | 11-5         |
| CTRL-O          | 17         | 12-9-8-7     | O               | 117        | 11-6         |
| CTRL-P          | 20         | 12-11-9-8-1  | P               | 120        | 11-7         |
| CTRL-Q          | 21         | 11-9-1       | Q               | 121        | 11-8         |
| CTRL-R          | 22         | 11-9-2       | R               | 122        | 11-9         |
| CTRL-S          | 23         | 11-9-3       | S               | 123        | 0-2          |
| CTRL-T          | 24         | 9-8-4        | T               | 124        | 0-3          |
| CTRL-U          | 25         | 9-8-5        | U               | 125        | 0-4          |
| CTRL-V          | 26         | 9-2          | V               | 126        | 0-5          |
| CTRL-W          | 27         | 0-9-6        | W               | 127        | 0-6          |
| CTRL-X          | 30         | 11-9-8       | X               | 130        | 0-7          |
| CTRL-Y          | 31         | 11-9-8-1     | Y               | 131        | 0-8          |
| CTRL-Z          | 32         | 9-8-7        | Z               | 132        | 0-9          |
| ESCAPE          | 33         | 0-9-7        | [               | 133        | 12-8-2       |
| CTRL-\          | 34         | 11-9-8-4     | \               | 134        | 0-8-2        |
| CTRL-]          | 35         | 11-9-8-5     | ]               | 135        | 11-8-2       |
| CTRL-†          | 36         | 11-9-8-6     | † ^             | 136        | 11-8-7       |
| CTRL-+          | 37         | 11-9-8-7     | + -             | 137        | 0-8-5        |
| SPACE           | 40         |              | \ `             | 140        | 8-1          |

NOTE: The ASCII character ESCAPE (octal 33) is also CTRL-[ on a terminal.

Table B-1 (Cont)  
ASCII Card Codes

| ASCII Character | Octal Code | Card Punches | ASCII Character | Octal Code | Card Punches |
|-----------------|------------|--------------|-----------------|------------|--------------|
| !               | 41         | 12-8-7       | a               | 141        | 12-0-1       |
| "               | 42         | 8-7          | b               | 142        | 12-0-2       |
| #               | 43         | 8-3          | c               | 143        | 12-0-3       |
| \$              | 44         | 11-8-3       | d               | 144        | 12-0-4       |
| %               | 45         | 0-8-4        | e               | 145        | 12-0-5       |
| &               | 46         | 12           | f               | 146        | 12-0-6       |
| '               | 47         | 8-5          | g               | 147        | 12-0-7       |
| (               | 50         | 12-8-5       | h               | 150        | 12-0-8       |
| )               | 51         | 11-8-5       | i               | 151        | 12-0-9       |
| *               | 52         | 11-8-4       | j               | 152        | 12-11-1      |
| +               | 53         | 12-8-6       | k               | 153        | 12-11-2      |
| ,               | 54         | 0-8-3        | l               | 154        | 12-11-3      |
| -               | 55         | 11           | m               | 155        | 12-11-4      |
| .               | 56         | 12-8-3       | n               | 156        | 12-11-5      |
| /               | 57         | 0-1          | o               | 157        | 12-11-6      |
| 0               | 60         | 0            | p               | 160        | 12-11-7      |
| 1               | 61         | 1            | q               | 161        | 12-11-8      |
| 2               | 62         | 2            | r               | 162        | 12-11-9      |
| 3               | 63         | 3            | s               | 163        | 11-0-2       |
| 4               | 64         | 4            | t               | 164        | 11-0-3       |
| 5               | 65         | 5            | u               | 165        | 11-0-4       |
| 6               | 66         | 6            | v               | 166        | 11-0-5       |
| 7               | 67         | 7            | w               | 167        | 11-0-6       |
| 8               | 70         | 8            | x               | 170        | 11-0-7       |
| 9               | 71         | 9            | y               | 171        | 11-0-8       |
| :               | 72         | 8-2          | z               | 172        | 11-0-9       |
| ;               | 73         | 11-8-6       | {               | 173        | 12-0         |
| <               | 74         | 12-8-4       |                 | 174        | 12-11        |
| =               | 75         | 8-6          | }               | 175        | 11-0         |
| >               | 76         | 0-8-6        | ~               | 176        | 11-0-1       |
| ?               | 77         | 0-8-7        | DEL             | 177        | 12-9-7       |

NOTE: The ASCII characters } and ~ (octal 175 and 176) are treated by the monitor as ALTmode which is often considered to be the same as ESCAPE.

Table B-2  
DEC-029 Card Codes

| Character | Octal Code | Card Punches | Character | Octal Code | Card Punches |
|-----------|------------|--------------|-----------|------------|--------------|
| SPACE     | 40         |              | @         | 100        | 8-4          |
| !         | 41         | 11-8-2       | A         | 101        | 12-1         |
| "         | 42         | 8-7          | B         | 102        | 12-2         |
| #         | 43         | 8-3          | C         | 103        | 12-3         |
| \$        | 44         | 11-8-3       | D         | 104        | 12-4         |
| %         | 45         | 0-8-4        | E         | 105        | 12-5         |
| &         | 46         | 12           | F         | 106        | 12-6         |
| '         | 47         | 8-5          | G         | 107        | 12-7         |
| (         | 50         | 12-8-5       | H         | 110        | 12-8         |
| )         | 51         | 11-8-5       | I         | 111        | 12-9         |
| *         | 52         | 11-8-4       | J         | 112        | 11-1         |
| +         | 53         | 12-8-6       | K         | 113        | 11-2         |
| ,         | 54         | 0-8-3        | L         | 114        | 11-3         |
| -         | 55         | 11           | M         | 115        | 11-4         |
| .         | 56         | 12-8-3       | N         | 116        | 11-5         |
| /         | 57         | 0-1          | O         | 117        | 11-6         |
| 0         | 60         | 0            | P         | 120        | 11-7         |
| 1         | 61         | 1            | Q         | 121        | 11-8         |
| 2         | 62         | 2            | R         | 122        | 11-9         |
| 3         | 63         | 3            | S         | 123        | 0-2          |
| 4         | 64         | 4            | T         | 124        | 0-3          |
| 5         | 65         | 5            | U         | 125        | 0-4          |
| 6         | 66         | 6            | V         | 126        | 0-5          |
| 7         | 67         | 7            | W         | 127        | 0-6          |
| 8         | 70         | 8            | X         | 130        | 0-7          |
| 9         | 71         | 9            | Y         | 131        | 0-8          |
| :         | 72         | 8-2          | Z         | 132        | 0-9          |
| ;         | 73         | 11-8-6       | [         | 133        | 12-8-2       |
| <         | 74         | 12-8-4       | \         | 134        | 11-8-7       |
| =         | 75         | 8-6          | ]         | 135        | 0-8-2        |
| >         | 76         | 0-8-6        | ↑ ^       | 136        | 12-8-7       |
| ?         | 77         | 0-8-7        | ← -       | 137        | 0-8-5        |

NOTE: Octal codes 0-37 and 140-177 are the same as in ASCII.

Table B-3  
DEC-026 Card Codes

| Character | Octal Code | Card Punches | Character | Octal Code | Card Punches |
|-----------|------------|--------------|-----------|------------|--------------|
| SPACE     | 40         |              | @         | 100        | 8-4          |
| !         | 41         | 12-8-7       | A         | 101        | 12-1         |
| "         | 42         | 0-8-5        | B         | 102        | 12-2         |
| #         | 43         | 0-8-6        | C         | 103        | 12-3         |
| \$        | 44         | 11-8-3       | D         | 104        | 12-4         |
| %         | 45         | 0-8-7        | E         | 105        | 12-5         |
| &         | 46         | 11-8-7       | F         | 106        | 12-6         |
| '         | 47         | 8-6          | G         | 107        | 12-7         |
| (         | 50         | 0-8-4        | H         | 110        | 12-8         |
| )         | 51         | 12-8-4       | I         | 111        | 12-9         |
| *         | 52         | 11-8-4       | J         | 112        | 11-1         |
| +         | 53         | 12           | K         | 113        | 11-2         |
| ,         | 54         | 0-8-3        | L         | 114        | 11-3         |
| -         | 55         | 11           | M         | 115        | 11-4         |
| .         | 56         | 12-8-3       | N         | 116        | 11-5         |
| /         | 57         | 0-1          | O         | 117        | 11-6         |
| 0         | 60         | 0            | P         | 120        | 11-7         |
| 1         | 61         | 1            | Q         | 121        | 11-8         |
| 2         | 62         | 2            | R         | 122        | 11-9         |
| 3         | 63         | 3            | S         | 123        | 0-2          |
| 4         | 64         | 4            | T         | 124        | 0-3          |
| 5         | 65         | 5            | U         | 125        | 0-4          |
| 6         | 66         | 6            | V         | 126        | 0-5          |
| 7         | 67         | 7            | W         | 127        | 0-6          |
| 8         | 70         | 8            | X         | 130        | 0-7          |
| 9         | 71         | 9            | Y         | 131        | 0-8          |
| :         | 72         | 11-8-2/11-0  | Z         | 132        | 0-9          |
| ;         | 73         | 0-8-2        | [         | 133        | 11-8-5       |
| <         | 74         | 12-8-6       | \         | 134        | 8-7          |
| =         | 75         | 8-3          | ]^        | 135        | 12-8-5       |
| >         | 76         | 11-8-6       | †         | 136        | 8-5          |
| ?         | 77         | 12-8-2/12-0  | ←         | 137        | 8-2          |

NOTE: Octal codes 0-37 and 140-177 are the same as in ASCII.

## APPENDIX C TEMPORARY FILES

The temporary files in Table C-1 are used by various programs in the DECsystem-10 computing system. These files are in the following form:

nnn xxx.TMP

where nnn is the user's job number in decimal, with leading zeroes to make three digits, and xxx specifies the use of the file.

Table C-1  
Temporary Files

| Name                                      | Meaning                                                                                                                                                                                                                                                                                          |
|-------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| nnn ALG.TMP                               | Read by ALGOL and contains one line for each program to be compiled. It may also contain the command NAME! which causes ALGOL to transfer control to the named program.                                                                                                                          |
| nnn AS1.TMP<br>nnn AS2.TMP<br>nnn AS3.TMP | Written, read, and deleted by COBOL and contains input to the COBOL assembler.                                                                                                                                                                                                                   |
| nnn BLI.TMP                               | Read by BLISS and contains one line for each program to be compiled.                                                                                                                                                                                                                             |
| nnn COB.TMP                               | Read by COBOL and contains one line for each program to be compiled. It may also contain the command NAME! which causes COBOL to transfer control to the named program.                                                                                                                          |
| nnn CPY.TMP                               | Written, read, and deleted by COBOL and contains copies of source files with library routines inserted.                                                                                                                                                                                          |
| nnn CRE.TMP                               | Read by CREF and contains commands for each file which has produced a CREF listing on the disk. COMPIL also reads this file each time a new CREF listing is generated to prevent multiple requests for the same file and to prevent discarding other requests that may not yet have been listed. |
| nnn DAE.TMP                               | Written by DAEMON to be read by DUMP.                                                                                                                                                                                                                                                            |
| nnn DMP.TMP                               | Read by DUMP as an input command file.                                                                                                                                                                                                                                                           |

Table C-1 (Cont)  
Temporary Files

| Name        | Meaning                                                                                                                                                                                                                                                                                                                                           |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| nnn EDS.TMP | Used by COMPIL to store the arguments of the most recent EDIT, CREATE, TECO, or MAKE command.                                                                                                                                                                                                                                                     |
| nnn EDT.TMP | Written by COMPIL and read by LINED or TECO. It contains a command for each EDIT, CREATE, TECO, or MAKE command. For the MAKE or CREATE commands, it contains the command<br><p style="text-align: center;">S file.ext [p,p] Ⓢ</p> For TECO or EDIT commands, it contains the command<br><p style="text-align: center;">S file.ext [p,p] &gt;</p> |
| nnn ERA.TMP | Written, read, and deleted by COBOL and is the error file.                                                                                                                                                                                                                                                                                        |
| nnn FOR.TMP | Read by FORTRAN and contains one line for each program to be compiled. It may also contain the command NAME! which causes FORTRAN to transfer control to the named program.                                                                                                                                                                       |
| nnn GEN.TMP | Written, read, and deleted by COBOL and contains the output of syntax processing.                                                                                                                                                                                                                                                                 |
| nnn KJO.TMP | Read by KJOB as an input command file.                                                                                                                                                                                                                                                                                                            |
| nnn LGO.TMP | Read by LOGOUT as an input command file.                                                                                                                                                                                                                                                                                                          |
| nnn LIN.TMP | Created by LINED and contains output file until the rename process.                                                                                                                                                                                                                                                                               |
| nnn LIT.TMP | Written, read, and deleted by COBOL and contains copy of the literal pool.                                                                                                                                                                                                                                                                        |
| nnn LOA.TMP | Read by LOADER and contains commands necessary for loading.                                                                                                                                                                                                                                                                                       |
| nnn MAC.TMP | Read by MACRO and contains one line for each program to be assembled. It may also contain the command NAME! which causes MACRO to transfer control to the named program.                                                                                                                                                                          |
| nnn P11.TMP | Read by MACX11 (the PDP-11 assembler for the PDP-10) and contains one line for each program to be assembled.                                                                                                                                                                                                                                      |
| nnn PLS.TMP | Read by PLEASE as an input command file.                                                                                                                                                                                                                                                                                                          |
| nnn PIP.TMP | Read by PIP and contains commands to implement the COMPIL-class commands that run PIP.                                                                                                                                                                                                                                                            |
| nnn QUE.TMP | Read by QUEUE as an input command file.                                                                                                                                                                                                                                                                                                           |
| nnn RNO.TMP | Read by RUNOFF and contains commands for each file which has produced a RUNOFF listing on the disk.                                                                                                                                                                                                                                               |
| nnn S01.TMP | Written, read, and deleted by COBOL and contains the intermediate sorted results of the data.                                                                                                                                                                                                                                                     |
| nnn SVC.TMP | Used by COMPIL to store the arguments of the most recent COMPILE, LOAD, EXECUTE, or DEBUG command.                                                                                                                                                                                                                                                |
| nnn SNO.TMP | Read by SNOBOL and contains one line for each program to be compiled.                                                                                                                                                                                                                                                                             |

(continued on next page)

Table C-1 (Cont)  
Temporary Files

| Name        | Meaning                                                            |
|-------------|--------------------------------------------------------------------|
| nnn TEC.TMP | Created by TECO and contains output file until the rename process. |
| nnn TMP.TMP | Created by LINED during the rename process.                        |
| nnn XFO.TMP | Created by FILEX as a result of the Q switch on the output side.   |
| nnn XFR.TMP | Created by FILEX as a result of the Q switch on the input side.    |

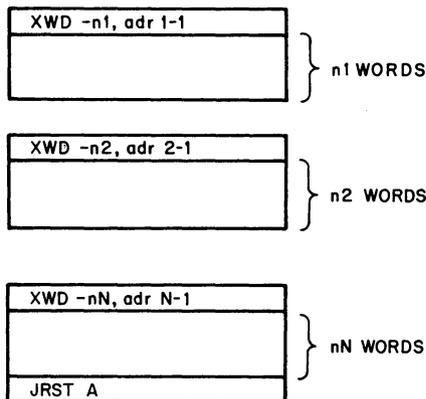


## APPENDIX D SAVE AND SSAVE COMMANDS

Before writing SAVed or LOW files in response to SAVE and SSAVE commands (refer to the individual command descriptions in Chapter 2), the monitor compresses the user's core image by eliminating consecutive blocks of zeroes. This technique is known as zero-compression and is used to save space on file media. Low segment files are zero-compressed on devices DTA, MTA, and DSK, but high segment files are not because the high segment can be shared at the time of the command.

SAVed files are ordinary binary files and can be copied using the /B switch in PIP. Files with the LOW or SAV extension may be read in dump mode, but must be reexpanded before being run. The monitor expands the file after input on a RUN, R, or GET command. The FILEX program may be used to expand the file for other purposes.

The data format of a zero-compressed SAVed file consists of a series of IOWDs and data block pairs and is terminated by a JRST A where A is the program starting address as specified by the contents of .JBSA. The format is as follows:



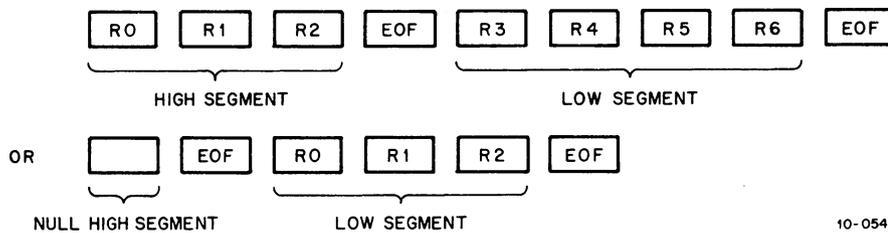
10-0544

Each IOWD describes the length of the following data block and the original location of the data in core. The LH of the IOWD can be positive in which case the number of words is taken as the number of words greater than 128K.

SAVED files are read into the user's core area starting at location .JBSAV and then are expanded to occupy the original relative locations. If the first word read is not an IOWD and is positive, an old-format, noncompressed saved file is assumed and no expansion is performed.

A SAVE command issued to a magnetic tape writes

- a. a high segment (possibly null)
- b. an EOF
- c. a low segment (possibly null)
- d. an EOF.



10-0540

The monitor does not determine the file size of a low segment on a GET from magnetic tape; therefore, a user must always specify a core argument or have enough core assigned to his job for the file.

To save file space, only the high segment up through the highest nonzero location (relative to high segment origin) loaded, as specified in the LH of .JBHRL, will be written by the SAVE command. If LH is zero (high segment created by CORE or REMAP UO) or DDT is present, the entire high segment will be written.

The LOADER indicates to the SAVE command how much data was loaded above the job data area in the low segment by setting the LH of .JBCOR to the highest location in the low segment that was not explicitly loaded with data (either zero or nonzero). Most programs are written so that only the high segment contains nonzero data. In this case, SAVE and SSAVE write only the high segments. This also saves file space and I/O time with the GET command.

A number of locations in the job data area need to be initialized on a GET, although there is no other data in the low segment. The SAVE command copies these locations into the first 10g locations of the

high segment, provided it is not sharable. The locations are referred to as the vestigial job data area (refer to DECsystem-10 Monitor Calls, Chapter 1). Therefore, the LOADER will load high segment programs starting at location 400010.

To prevent user confusion, SAVE and SSAVE delete a previous file with the extension .SHR or .HGH; therefore, SAVE deletes a file with the extension .SHR and SSAVE deletes a file with the extension .HGH. SAVE and SSAVE commands also delete files with the extension .LOW, if the high segment was the only segment written.

The regular access rights of the saved file indicate whether a user can perform a GET, R, or RUN command. These commands assume that the user wants to execute (but not modify) the high segment, independent of the access rights of the file used to initialize the segment. The monitor always enables the hardware user-mode write protect to prevent the user program from storing into the segment inadvertently.

To debug a reentrant system program, the user should make a private, nonsharable copy, rather than modify the shared version and possibly cause harm to other users. To make a private, nonsharable copy, the following commands are used:

|          |                                                                                                                                                                     |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GET prog |                                                                                                                                                                     |
| SAVE     | Writes a file in the user directory as nonsharable. The high segment in the user's addressing space remains sharable.                                               |
| GET      | Overlays the sharable program with the nonsharable one from the user's directory. Now the user can make patches while other users share the version in the library. |

If the user is debugging a sharable program in his UFD with the D command or the DDT program, it is recommended that the program be nonsharable instead of sharable. The reason for this is that the user may wish to modify the high segment during the debugging phase and later reinitialize the original unmodified high segment from the file with a GET command. However, since the high segment is sharable, the monitor will not do I/O into it, will not reinitialize it from the disk file, and the user will receive the modified high segment instead.

#### NOTE

DDT modifies the high segment when it inserts breakpoints.

The following examples are the incorrect and correct methods of debugging a sharable program. After the debugging phase is completed, the SSAVE command should be used to save the program.

Example 1: Incorrect Method

```

DEBUG prog )
EXECUTION
^C
.SSAVE ) ;SAVE should be used in debugging
.GET )
JOB SETUP )
.E 400010
400010/777777 777777 .D 0 0 )
.E )
400010/ 0 0
.GET )
JOB SETUP
.E 400010 ) ;not the original 777777 777777
400010/ 0 0
    
```

Example 2: Correct Method

```

DEBUG prog )
EXECUTION
^C
.SAVE )
.GET )
JOB SETUP
.E 400010 )
400010/777777 777777 .D 0 0 )
.E )
400010/ 0 0
.GET )
JOB SETUP
.E 400010 ) ;the original file
400010/777777 777777
    
```

Note that there are applications for a sharable data segment when the modified version of the sharable segment is wanted rather than the original segment as initialized from the file. The SSAVE command is then used.

A SAVE of a one-segment program and a SSAVE of a two-segment program of the same name can coexist in the same directory, and the monitor keeps the two versions separate. This allows for a common library, of reentrant and non-reentrant versions of the same system programs to service both the PDP-6 and the DECsystem-10. A sharable program may be superseded into the directory by the SSAVE command. The monitor clears the high segment in its table of sharable segments in use but does not remove the segment from the addressing space of users currently using it. Only the users doing a GET, R, or RUN command or a RUN or GETSEG UUC have the new sharable version.

When the SAVE or SSAVE command is used to save a sharable program with only a high file, the monitor does not modify the vestigial job data area. This prohibits unauthorized users from modifying the first 10 locations of a shared segment by executing a SAVE or SSAVE command. This restriction does

not exist if a low file is also written, because the GET command reads the low file after the high file, so that the real job data area locations are set from the low file. To change the version number of a sharable two-segment program with only a high file, the following commands are used.

```
GET prog
SAVE
GET
D nnn mmm 137
SSAVE
```

The SAVE command makes the program non-sharable so that the vestigial job data area can be modified by the SSAVE.



INDEX

A

↑ A (control-A) see Control commands (TECO)  
 A (append) command, 197 (INTRO TO TECO)  
 A (append) command, 257, 293, 301 (TECO)  
 A Switch, 399 (PIP)  
 Absolute Addresses, 17 (INTRO TO SOFTWARE)  
 /ACCESS, 520 (COMMANDS)  
 Accessing the system, 590 (COMMANDS)  
 Active search list, 662 (COMMANDS)  
 Adding comments, 104, 139 (BEGINNER'S BATCH)  
 Adding devices to spool list, 660 (COMMANDS)  
 Addition, 241 (TECO)  
 Advance command, 417 (PIP)  
 /AFTER switch (BEGINNER'S BATCH)  
   \$JOB card, 126  
   SUBMIT command, 141  
 /AFTER switch, 494, 605, 611, 620, 681, 697 (COMMANDS)  
 AID Interpreter, 20 (INTRO TO SOFTWARE)  
 ALCFIL program, 468 (COMMANDS)  
 ALCFIL program, 31 (INTRO TO SOFTWARE)  
 ALGOL (BEGINNER'S BATCH)  
   compiler switches, 111  
   definition, 91  
   deck, setting up, 105  
   job, examples, 169, 176  
   program, compiling and executing, 105  
 ALGOL, 18 (INTRO TO SOFTWARE)  
 \$ALGOL card, 105, 111 (BEGINNER'S BATCH)  
   examples, 112  
   switches, 111  
 \$ALGOL card, 718 (COMMANDS)  
 /ALGOL switch, 481, 513, 539, 585 (COMMANDS)  
 ALGOTS, 18, 31 (INTRO TO SOFTWARE)  
 /ALLOC, 520 (COMMANDS)  
 Allocating disk space, 468 (COMMANDS)

Allocating facilities, 464 (COMMANDS)

ASSIGN, 470  
 CLOSE, 479  
 CORE, 491  
 DEASSIGN, 512  
 DISMOUNT, 525  
 FINISH, 560  
 LOCATE, 589  
 MOUNT, 593  
 REASSIGN, 633  
 SET BLOCKSIZE, 653  
 SET CDR, 654  
 SET CPU, 655  
 SET DENSITY, 657  
 SET DSKPRI, 658  
 SET HPQ, 659  
 SET SPOOL, 660  
 SET TTY, 668  
 TTY, 668

Allocating I/O devices, 470, 593 (COMMANDS)

Allocating system resources, 77 (TIMESHARING)

ASSIGN, 77  
 CORE, 80  
 DEASSIGN, 79  
 DISMOUNT, 79  
 FINISH, 80  
 MOUNT, 78  
 REASSIGN, 79

Allocation (INTRO TO SOFTWARE)

File Storage, 11, 30

Allocator (INTRO TO SOFTWARE)

Shareable Resource, 10

Alphabetic case control (TECO)

in insert commands, 268  
 in search arguments, 282

Alphanumeric, definition, 92 (BEGINNER'S BATCH)

Alphanumeric argument, 196 (INTRO TO TECO)

Alphanumeric argument, 236, 240 (TECO)

Altmode, 235, 236, 239, 240, 251, 261, 266, 267, 288, 291, 295, 296, 319 (TECO)

ALTMODE key, 75 (TIMESHARING)

Altmode symbol ( Ⓢ ), 193, 194 (INTRO TO TECO)

## INDEX (Cont)

- & (ampersand), 242 (TECO)
  - Analyzing a core image file, 529 (COMMANDS)
  - AND, 242 (TECO)
  - Angle brackets, 382 (PIP)
  - < > (angle brackets), 288, 291 (TECO)
  - Angle bracket matching, V switch, 402 (PIP)
  - ' (apostrophe) command, 292, 317 (TECO)
  - Argument pair, 241 (TECO)
  - Arguments, 247 (TECO)
    - alphanumeric, 236, 240
    - numeric, 241, 251, 257
    - text, 240
  - Arguments, command, 195 (INTRO TO TECO)
  - Argument terminator, 251 (TECO)
  - Arithmetic/logical operators, 241, 242 (TECO)
  - ASCII code, definition, 92 (BEGINNER'S BATCH)
  - Assemble, definition, 92 (BEGINNER'S BATCH)
  - Assembler, definition, 92 (BEGINNER'S BATCH)
  - Assembler (INTRO TO SOFTWARE)
    - MACRO, 17
  - Assembling and executing a MACRO program, 107 (BEGINNER'S BATCH)
  - Assembly language, definition, 92 (BEGINNER'S BATCH)
  - Assembly listing, definition, 92 (BEGINNER'S BATCH)
  - ASSIGN command, 470 (COMMANDS)
  - ASSIGN command, 77 (TIMESHARING)
  - Assigning devices, 76 (TIMESHARING)
    - ASSIGN, 77
    - MOUNT, 78
  - Assigning input devices in programs (BEGINNER'S BATCH)
    - ALGOL
      - disk, 118
      - card reader, 119
    - COBOL
      - disk, 117
      - card reader, 118
    - FORTRAN
      - disk, 117
      - card reader, 119
  - Assigning names to DECTape, 395 (PIP)
  - Assignment, device, 10 (INTRO TO SOFTWARE)
  - \* (asterisk), 240, 242, 247, 248, 249, 262 (TECO)
  - \*i command, 271, 296 (TECO)
  - Asterisk construction, 452 (COMMANDS)
  - Asterisk construction, 66 (TIMESHARING)
  - Asterisk (\*) symbol usage, 375, 386, 406 (PIP)
  - Asterisk (\*) usage, 193 (INTRO TO TECO)
  - At (@) symbol usage, 375 (PIP)
  - @ (at sign modifier), 281 (TECO)
  - @I command, 266, 299 (TECO)
  - ATTACH command, 472 (COMMANDS)
  - ATTACH command, 81 (TIMESHARING)
  - Automatic timeout (TECO)
    - after searches, 282
    - flag, obtaining the value, 301
  - Auxiliary LINED commands, 362 (LINED)
  - Available devices, listing of, 640 (COMMANDS)
- B
- B, 258 (TECO)
  - B switch, 405 (PIP)
  - Back-arrow (SHIFT-O), 376 (PIP)
  - ← (back arrow) command, 280 (TECO)
  - ↑ (control-backslash) command, see control commands (TECO)
  - \ (backslash) command, 267, 302, 303 (TECO)
  - BACKSPACE command, 474 (COMMANDS)
  - Backspace file request, 416, 417 (PIP)
  - Backspace one file, 252 (TECO)
  - .BACKTO command, 123, 130, 146 (BEGINNER'S BATCH)
    - example, 146
  - .BACKTO command, 734 (COMMANDS)

## INDEX (Cont)

- Backup file, 193 (INTRO TO TECO)
- Backup file, 253, 255 (TECO)
- Back up one record, 252 (TECO)
- BACKUP program, 475 (COMMANDS)
- BACKUP program, 22 (INTRO TO SOFTWARE)
- BACKUP SET file, 475 (COMMANDS)
- BAK, 234, 249, 253 (TECO)
- BASIC (BEGINNER'S BATCH)
  - deck, setting up, 109
  - definition, 92
  - job, examples, 171, 178
  - program, running, 109
- BASIC, 18 (INTRO TO SOFTWARE)
- Batch (BEGINNER'S BATCH)
  - commands, 146
  - format, 138
  - control cards, 103
  - format, 103
  - output, 153
  - processing, definition, 92
  - queue, entering jobs, 139
- Batch (INTRO TO SOFTWARE)
  - Multiprogram, 12
- Batch command interpreter, 447 (COMMANDS)
- Batch controller, 710 (COMMANDS)
  - commands for, 732
- Batch Controller, 13 (INTRO TO SOFTWARE)
- Batch input queue, 680 (COMMANDS)
- Batch Operator intervention, 14 (INTRO TO SOFTWARE)
- Batch sample jobs, 743 (COMMANDS)
- Batch system commands, 709 (COMMANDS)
- BATCON, 710 (COMMANDS)
  - control file commands, 732
  - error reporting, 742
  - messages, 742
- BATCON, 12, 32 (INTRO TO SOFTWARE)
- /BEFORE, 494, 605, 611, 620, 697 (COMMANDS)
- /BEGIN, 494, 605, 611, 620, 697 (COMMANDS)
- Beginning in DDT, 510, 513 (COMMANDS)
- Bell, 236, 320 (TECO)
- Bell-space command, 321 (TECO)
- Binary mode switch (B), 401 (PIP)
- Bit (INTRO TO SOFTWARE)
  - Use, 28
- Blank page, 274, 275 (TECO)
- Blank tape, 252 (TECO)
- /BLISS, 481, 513, 539, 585 (COMMANDS)
- Block, 66 (TIMESHARING)
- Block Mode, 15 (INTRO TO SOFTWARE)
- Block Numbers (INTRO TO SOFTWARE)
  - Logical, 29
- Blocksize of magnetic tape, 653 (COMMANDS)
- Blocks of text, 309 (TECO)
- BOOTS, 32 (INTRO TO SOFTWARE)
- Boundary of the buffer, 259 (TECO)
- /B/P switch combination, 401 (PIP)
- Brackets, angle see Angle brackets (TECO)
- Brackets, square see Square brackets (TECO)
- Break character, 320 (TECO)
- Breakpoints, 22, 32 (INTRO TO SOFTWARE)
- Buffer, 32 (INTRO TO SOFTWARE)
  - Editing, 20
- Buffer, command string, 244 (TECO)
  - , editing, 239, 243, 244
  - pointer, 239, 256, 257, 258, 259, 260, 264, 267, 268, 278, 280, 296, 302
  - position, 241, 257
- Buffer, editing, 191 (INTRO TO TECO)
- Buffer boundary, 259 (TECO)
- Buffered Modes, 27, 28 (INTRO TO SOFTWARE)
- Buffer Pointer, 20, 32 (INTRO TO SOFTWARE)
- Buffer pointer, 198 (INTRO TO TECO)
- Buffers (INTRO TO SOFTWARE)
  - Ring of, 28
- Buffers, 231, 244, 257 (TECO)
- Byte descriptor, 534 (COMMANDS)

## INDEX (Cont)

## C

- ↑ C (control-C) command see Control commands (TECO)
- C (advance pointer by character) command, 198 (INTRO TO TECO)
- C command, 258, 293 (TECO)
- C switch, 399 (PIP)
- CAM, 21 (INTRO TO SOFTWARE)
- Capabilities (INTRO TO SOFTWARE)
  - macro, 18
- Card, definition, 92 (BEGINNER'S BATCH)
  - column, definition, 93
  - field, definition, 93
  - format, 103
  - output, specifying amount, 126, 142
  - row, definition, 93
- Card codes, 787 (COMMANDS)
- Card Punch, 49 (INTRO TO SOFTWARE)
- Card punch, J-switch, 418 (PIP)
- Card punch queue, 593 (COMMANDS)
- Card Readers, 48 (INTRO TO SOFTWARE)
- Card reader spooling intercept, 654 (COMMANDS)
- /CARDS switch (BEGINNER'S BATCH)
  - \$JOB card, 126
  - SUBMIT command, 142
- /CARDS switch, 620, 681 (COMMANDS)
- Cards to specify error recovery, 122, 129 (BEGINNER'S BATCH)
- Caret, 239 (TECO)
- Carriage return, 236, 239, 262, 267, 306, 320 (TECO)
- Carriage return symbol (↵), 193 (INTRO TO TECO)
- <CR> carriage return usage, 377 (PIP)
- Case control (TECO)
  - in insert commands, 268
  - in search arguments, 282
- Case flag, obtaining the value, 301 (TECO)
- Case flagging on timeout, 262 (TECO)
- Case match mode control in searches, 285 (TECO)
- Categories of messages, 747 (COMMANDS)
- Categories of TECO commands, 243 (TECO)
- Causing the current line to be retyped, 321 (TECO)
- CCONTINUE command, 500 (COMMANDS)
- CDRSTK, 709 (COMMANDS)
  - error reporting, 741
  - messages, 740
- CDRSTK, 12, 32 (INTRO TO SOFTWARE)
- Central processing unit, definition, 93 (BEGINNER'S BATCH)
- Central processor time limit, 666 (COMMANDS)
- Central site, definition, 93 (BEGINNER'S BATCH)
- Central site, 78 (TIMESHARING)
- %CERR, 733 (COMMANDS)
- CHAIN, 33 (INTRO TO SOFTWARE)
- CHAIN Files, 22 (INTRO TO SOFTWARE)
- Changing (TECO)
  - amount of error reporting, 324
  - maximum number of entries in the Q-register pushdown list, 251
- Changing CPU specification, 655 (COMMANDS)
- Changing filenames, 67 (TIMESHARING)
- Changing logical station, 589 (COMMANDS)
- Changing modes, 730 (COMMANDS)
- Changing switch in a queue entry, 141 (BEGINNER'S BATCH)
- Changing UFD or SFD protection code, 408 (PIP)
- Channels (INTRO TO SOFTWARE)
  - Software I/O, 28
- Character, definition, 93 (BEGINNER'S BATCH)
- Characters, control, 236 (TECO)
  - ,special, 235
- Character set, 234 (TECO)
- Character strings, 244, 296 (TECO)
- CHECKPOINT, 33 (INTRO TO SOFTWARE)
- /CHECKSUM, 521 (COMMANDS)
- .CHKPNT command, 735 (COMMANDS)

INDEX (Cont)

- Clearing directories, 470, 512, 633, 707 (COMMANDS)
- Clearing logical names, 512 (COMMANDS)
- CLOSE, 29 (INTRO TO SOFTWARE)
- Close, 252, 253, 274, 276, 307 (TECO)
- CLOSE command, 479 (COMMANDS)
- Close files, 192, 204 (INTRO TO TECO)
- Closing the current file, 361 (LINED)
- COBOL (BEGINNER'S BATCH)
  - compiler switches, 113
  - deck, setting up, 106
  - definition, 92
  - job, examples, 154, 174, 183
  - program
    - compiling and executing, 106
    - format, 114
- COBOL, 19 (INTRO TO SOFTWARE)
- \$COBOL card, 106, 113 (BEGINNER'S BATCH)
  - examples, 114
  - switches, 113
- \$COBOL card, 719 (COMMANDS)
- /COBOL switch, 481, 513, 539, 585 (COMMANDS)
- COBRG, 19 (INTRO TO SOFTWARE)
- CODE, 33 (INTRO TO SOFTWARE)
- Codes (COMMANDS)
  - card, 787
  - error, 780
- Codes (INTRO TO SOFTWARE)
  - Protection, 11, 30
- : (colon), 251 (TECO)
- : (colon modifier), 279, 281 (TECO)
- : (colon) search, 281, 294 (TECO)
- Colon (: ) usage, 376, 382 (PIP)
- Combinations of switches, 411 (PIP)
- Combine files, transfer without X-switch, 398 (PIP)
- Combine pages, 197 (INTRO TO TECO)
- Combining \* and ? wildcard symbols, 386 (PIP)
- Combining files, 486 (COMMANDS)
- Comma, 241 (TECO)
- Comma usage, 382 (PIP)
- Command, definition, 93 (BEGINNER'S BATCH)
- Command (COMMANDS)
  - arguments, 448
  - delay, 446
  - formats, 447
  - interpreters, 445
  - names, 447
- Command arguments, 195 (INTRO TO TECO)
- Command buffer, 278 (TECO)
- Command Control Language, 10 (INTRO TO SOFTWARE)
- Command Decoder, 25 (INTRO TO SOFTWARE)
- Command error, 195 (INTRO TO TECO)
- Command error, 240, 319 (TECO)
- Command errors, 421 (PIP)
- Command List, 28 (INTRO TO SOFTWARE)
- Command mode, 247 (TECO)
- Command recovery file, 475 (COMMANDS)
- Commands (COMMANDS)
  - Batch, 709
  - COMPILE-class, 453
  - system, 463
- Commands (INTRO TO TECO)
  - A (append page), 197
  - C (advance pointer by character), 198
  - D (delete character), 201
  - EF (close output file), 203
  - EG (close file, reexecute monitor command), 204
  - EX (close file), 204
  - HK (delete buffer), 201
  - HT (type entire buffer), 200
  - I (insert), 202
  - J (move pointer to beginning), 198
  - K (delete line), 199
  - L (move pointer by lines), 199
  - MAKE (make disk file), 192
  - N (search file), 206, 207
  - P (output buffer), 203
  - PW (output page), 203



## INDEX (Cont)

- Commands (cont) (INTRO TO TECO)
  - R (move pointer backwards by character), 198
  - S (search buffer), 205, 206
  - T (type), 200
  - TECO (initialize file for editing), 192
  - Y (yank next page), 197
  - ZJ (move pointer to end), 198
- Commands, 359 (LINED)
  - D, 360
  - E, 361
  - I, 360
  - P, 361
  - S, 362
- Commands (INTRO TO TECO)
  - erasing, 195
  - input, 197
- Commands (TECO)
  - edit class, 248
  - erasing, 235
  - immediate action, 235
  - monitor, 235, 248
  - that return a value, 243
- Commands, summary of, see Appendix C (TECO)
- Commands not available in Batch, 109, 138 (BEGINNER'S BATCH)
- Commands to specify error recovery, 148 (BEGINNER'S BATCH)
- Command string, (INTRO TO TECO)
  - execution, 194
  - syntax, 194
  - termination, 194
- Command string, 379 (PIP)
  - delimiters, 382
  - format, 379
- Command string buffer, 244 (TECO)
- Command strings, 239, 240, 244, 276, 277, 291, 296, 306, 319, 321 (TECO)
- Command string syntax, 239 (TECO)
- Comments, 104, 139 (BEGINNER'S BATCH)
- Comments, 294 (TECO)
- Communicating with operator, 466, 737 (COMMANDS)
  - DISMOUNT, 525
  - FILE, 553
  - GRIPE, 573
  - MOUNT, 593
  - PLEASE, 602
  - SEND, 651
- Communications (INTRO TO SOFTWARE)
  - Data, 50
  - Remote, 15
- COMP10, 21 (INTRO TO SOFTWARE)
- Comparing DECtapes, 488 (COMMANDS)
- Comparing files, 543 (COMMANDS)
- COMPIL, 33 (INTRO TO SOFTWARE)
- Compile, definition, 93 (BEGINNER'S BATCH)
- Compile-class command, 276 (TECO)
- COMPIL-class commands, 453 (COMMANDS)
  - switches, 457
- /COMPILE, 481, 513, 540, 586 (COMMANDS)
- COMPILE command, 480 (COMMANDS)
- COMPILE command, 276 (TECO)
- COMPILE command, 67 (TIMESHARING)
- Compiler, definition, 93 (BEGINNER'S BATCH)
- Compilers, 17 (INTRO TO SOFTWARE)
  - ALGOL, 18
  - BASIC, 18
  - COBOL, 19
  - FORTRAN, 19
- Compiling and executing a program (BEGINNER'S BATCH)
  - ALGOL, 105
  - COBOL, 106
  - FORTRAN, 107
- Compiling programs, 480 (COMMANDS)
  - ALGOL, 718
  - COBOL, 718
  - FORTRAN, 725
  - MACRO, 729

INDEX (Cont)

- Components (INTRO TO SOFTWARE)
  - DECsystem-10, 9
  - Multiprogram batch, 12
  - Operating System, 25
- Components, Batch, 709 (COMMANDS)
- Computer, definition, 93 (BEGINNER'S BATCH)
- Computer operator, definition, 93 (BEGINNER'S BATCH)
- Computing (INTRO TO SOFTWARE)
  - Multimode, 10
- Conditional branch, 281, 284 (TECO)
- Conditional command, 276 (TECO)
- Conditional execution, 286 (TECO)
- Conditional execution commands, 292, 294 (TECO)
- Conditional skip, 291 (TECO)
- CONFIGURATION category, 505 (COMMANDS)
- Connecting to a detached job, 472 (COMMANDS)
- Console data switches, 301, 302 (TECO)
- Contents of card decks, 103 (BEGINNER'S BATCH)
- Contents operators, 534 (COMMANDS)
- Continuation card, definition, 93 (BEGINNER'S BATCH)
- Continuation of information on a card, 104 (BEGINNER'S BATCH)
- Continuation of lines in control file, 139 (BEGINNER'S BATCH)
- CONTINUE, 277 (TECO)
- CONTINUE command, 485 (COMMANDS)
- CONTINUE command, 84 (TIMESHARING)
- Continuing a job, 578 (COMMANDS)
- Continuing a program, 84 (TIMESHARING)
- Control (PIP)
  - direct, 375
  - indirect, 375
- Control-C, 444, 446 (COMMANDS)
- Control cards, 103 (BEGINNER'S BATCH)
- Control characters, 193 (INTRO TO TECO)
  - Control characters, 235, 261, 286, (TECO)
  - Control characters, 73 (TIMESHARING)
    - Control-C, 73
    - Control-O, 75
    - Control-U, 74
  - Control commands, 717 (COMMANDS)
  - Control commands (TECO)
    - ↑ ↑, 270
    - ↑ \, 285
    - ↑ [, 235
    - ↑ A, 302, 303
    - ↑ C, 235, 275, 276, 290
    - ↑ E, 286, 301
    - ↑ F, 301, 303
    - ↑ G, 235, 306, 320
    - ↑ G ↑ G, 235, 320
    - ↑ G ↓, 321
    - ↑ H, 301, 303
    - ↑ L, 261
    - ↑ N, 286, 288, 301
    - ↑ O, 235, 260, 263
    - ↑ R, 271, 285, 286
    - ↑ S, 286, 288
    - ↑ T, 271, 285, 302, 303
    - ↑ V, 268, 283
    - ↑ V ↑ V, 269, 283
    - ↑ W, 268, 283
    - ↑ W ↑ W, 269, 283
    - ↑ U, 235, 306, 320, 321
    - ↑ X, 286, 288
    - ↑ Z, 275, 276
  - Control file, 99, 103, 137 (BEGINNER'S BATCH)
    - creating, 103, 137
    - definition, 93
    - examples, 137, 138, 160
    - format of lines, 138
    - putting commands in, 108
  - Control file, 680, 710, 716, 727 (COMMANDS)
    - commands, 732
  - Control key, 73 (TIMESHARING)
  - Control Language (INTRO TO SOFTWARE)
    - Command, 10
  - Controller (INTRO TO SOFTWARE)
    - Batch, 13



## INDEX (Cont)

- Controlling error reporting, 149 (BEGINNER'S BATCH)
- Controlling multiple jobs, 466 (COMMANDS)
  - ATTACH, 472
  - CCONT, 500
  - CSTART, 500
  - DETACH, 519
  - OPSER, 597
  - REATA, 635
- Controlling number of card columns read (BEGINNER'S BATCH)
  - \$ALGOL card, 112
  - \$COBOL card, 113
  - \$DATA card, 116
  - \$DECK card, 120
  - \$FORTRAN card, 124
  - \$MACRO card, 128
- Controlling object programs, 465 (COMMANDS)
  - CONTINUE, 485
  - DDT, 510
  - GET, 568
  - HALT, 574
  - JCONT, 578
  - R, 632
  - REENTER, 637
  - RUN, 646
  - START, 679
- Control-O, 447 (COMMANDS)
- Control-U, 447 (COMMANDS)
- Conventional COBOL format, 114 (BEGINNER'S BATCH)
- Conventional Format, 19 (INTRO TO SOFTWARE)
- Conventions, writing, 375 (PIP)
- Conventions and restrictions, 362 (LINED)
- Converting special characters to lower case, 270, 285 (TECO)
- /COPIES, 494, 605, 611, 621, 697 (COMMANDS)
- Copy (PIP)
  - all but specified files (DS switch), 397
  - files without combining, (X switch), 394
  - FORTRAN binary files, 401
- COPY command, 486 (COMMANDS)
- COPY program, 488 (COMMANDS)
- Copying, 395 (PIP)
- Copying data into disk files, 115 (BEGINNER'S BATCH)
- Copying DECtapes, 488 (COMMANDS)
- Copying into a data file, 723 (COMMANDS)
- Copying programs into disk files (BEGINNER'S BATCH)
  - ALGOL, 111
  - COBOL, 113
  - FORTRAN, 123
  - MACRO, 127
- Copying relocatable binary programs, 731 (COMMANDS)
- Copying trailing spaces into files (BEGINNER'S BATCH)
  - \$ALGOL card, 112
  - \$COBOL card, 114
  - \$DATA card, 116
  - \$DECK card, 120
  - \$FORTRAN card, 124
  - \$MACRO card, 128
- Core (BEGINNER'S BATCH)
  - definition, 94
  - specifying amount, 126, 142
- Core, 231, 238, 244, 247, 256, 257, 296, 309 (TECO)
- Core allocation, 491 (COMMANDS)
- CORE category, 505 (COMMANDS)
- Core check, 446 (COMMANDS)
- CORE command, 491 (COMMANDS)
- CORE command, 80 (TIMESHARING)
- Core expansion, 244, 256, 257 (TECO)
- Core image, 68 (TIMESHARING)
- Core-image files, 505, 529 (COMMANDS)
- Core Memories, 47 (INTRO TO SOFTWARE)
- /CORE switch (BEGINNER'S BATCH)
  - \$JOB card, 126
  - SUBMIT command, 142
- /CORE switch, 621, 681 (COMMANDS)

## INDEX (Cont)

- Core Utilization, 11 (INTRO TO SOFTWARE)
  - CORMAX, 33 (INTRO TO SOFTWARE)
  - CORMIN, 33 (INTRO TO SOFTWARE)
  - Correcting typing errors, 74 (TIMESHARING)
  - CPU, definition, 94 (BEGINNER'S BATCH)
  - CPUNCH command, 493 (COMMANDS)
  - CPU specification, 655 (COMMANDS)
  - CPU time, specifying amount, 127, 142 (BEGINNER'S BATCH)
  - CREATE, 359 (LINED)
  - Create, 248, 250 (TECO)
  - CREATE command, 498 (COMMANDS)
  - CREATE command, 63 (TIMESHARING)
  - /CREATE switch (SUBMIT command), 140 (BEGINNER'S BATCH)
  - /CREATE switch, 494, 605, 611, 621, 681, 697 (COMMANDS)
  - Creating, 30 (INTRO TO SOFTWARE)
  - Creating control file, 103, 137 (BEGINNER'S BATCH)
  - Creating entry in the Batch queue, 140 (BEGINNER'S BATCH)
  - Creating files, 498, 592 (COMMANDS)
    - control, 716, 727
    - library REL files, 482, 513, 540, 562, 586
  - Creating files, 62 (TIMESHARING)
    - CREATE, 63
    - MAKE, 64
  - Creating TECO macro, 271 (TECO)
  - CREF command, 499 (COMMANDS)
  - CREF command, 82 (TIMESHARING)
  - /CREF switch (BEGINNER'S BATCH)
    - \$COBOL card, 114
    - \$FORTRAN card, 124
    - \$MACRO card, 129
  - /CREF switch, 481, 513, 540, 586 (COMMANDS)
  - CREF Utility, 21, 34 (INTRO TO SOFTWARE)
  - Cross reference listing, definition, 93 (BEGINNER'S BATCH)
    - \$COBOL card, 113
    - \$FORTRAN card, 124
    - \$MACRO card, 129
  - Cross reference listing, 82 (TIMESHARING)
  - Cross-referenced listing files, 499 (COMMANDS)
  - CSTART command, 500 (COMMANDS)
  - CTRL key, 235 (TECO)
  - Current line, 239, 259, 261, 321 (TECO)
  - Cyclic Routines, 25 (INTRO TO SOFTWARE)
- D
- D command, 360 (LINED)
  - D (delete character) command, 201 (INTRO TO TECO)
  - D (delete) command, 264, 265, 293 (TECO)
  - D (deposit) command, 502 (COMMANDS)
  - D switch, 404, 409 (PIP)
  - DX switch, copy all but specified (PIP)
    - files, 397
  - DAEMON, 34 (INTRO TO SOFTWARE)
  - DAEMON program, 505, 529 (COMMANDS)
  - DAEMON-written file, 505 (COMMANDS)
  - Data, definition, 94 (BEGINNER'S BATCH)
  - Data Areas (INTRO TO SOFTWARE)
    - Sharable, 15
  - \$DATA card, 105, 106, 107, 115 (BEGINNER'S BATCH)
    - examples, 116
    - naming data files, 116
    - switches, 116
  - \$DATA card, 714, 715, 721 (COMMANDS)
  - Data Communications System, 50 (INTRO TO SOFTWARE)
  - Data file, 191 (INTRO TO TECO)
  - Data line in control file, format, 139 (BEGINNER'S BATCH)

Data Modes (INTRO TO SOFTWARE)  
     Buffered, 27, 28  
     Unbuffered, 27, 28

Data mode switches, 405 (PIP)

Data Transfers, 27 (INTRO TO SOFTWARE)

DATDMP, 34 (INTRO TO SOFTWARE)

DAYTIME command, 504 (COMMANDS)

DAYTIME command, 71 (TIMESHARING)

DCORE command, 505 (COMMANDS)

DDB category, 505 (COMMANDS)

DDT command, 510 (COMMANDS)

DDT program, 69 (TIMESHARING)

DDT Utility, 21, 34 (INTRO TO SOFTWARE)

/DEADLINE, 494, 605, 611, 621, 681, 697 (COMMANDS)

DEASSIGN command, 512 (COMMANDS)

DEASSIGN command, 79 (TIMESHARING)

Debug, definition, 94 (BEGINNER'S BATCH)

DEBUG, 276 (TECO)

DEBUG command, 513 (COMMANDS)

DEBUG command, 69 (TIMESHARING)

Debugging (COMMANDS)  
     reentrant programs, 797  
     sharable programs, 797

Debugging programs, 67 (TIMESHARING)  
     DEBUG, 69

Decimal number, 266 (TECO)

\$DECK card, 109, 119 (BEGINNER'S BATCH)  
     examples, 120  
     switches, 120

\$DECK card, 723 (COMMANDS)

Decoder (INTRO TO SOFTWARE)  
     Command, 25

DECsystem-10 Components, 9 (INTRO TO SOFTWARE)

DECsystem-10 Family, 9, 45 (INTRO TO SOFTWARE)

DECsystem-1040, 45 (INTRO TO SOFTWARE)

## INDEX (Cont)

DECsystem-1050, 45 (INTRO TO SOFTWARE)

DECsystem-1055, 45 (INTRO TO SOFTWARE)

DECsystem-1070, 45 (INTRO TO SOFTWARE)

DECsystem-1077, 46 (INTRO TO SOFTWARE)

DECtape control, remote, 553 (COMMANDS)

DECtape copy routine, 488 (COMMANDS)

DECtape tape names, 397 (PIP)

DECtape to paper tape copy, Y (PIP)  
     switch, 403

Defining limits for a job, 141 (BEGINNER'S BATCH)

DELETE command, 517 (COMMANDS)

DELETE command, 67 (TIMESHARING)

Delete disk, 410 (PIP)

Delete files (D switch), 409 (PIP)

Delete sequence number (N switch), 399 (PIP)

Delete trailing spaces, T switch, 402 (PIP)

Deleting (BEGINNER'S BATCH)  
     control file, 143  
     job from the queue, 140  
     log file, 143

Deleting a line, 360 (LINED)

Deleting characters, 74 (TIMESHARING)

Deleting DECtape and disk files, 517 (COMMANDS)

Deleting devices from spool list, 660 (COMMANDS)

Deleting files when over quota, 580 (COMMANDS)

Deleting multiple lines, 361 (LINED)

Deletion commands, 201 (INTRO TO TECO)

Delimiters, command string, 382 (PIP)

/DENSITY, 521 (COMMANDS)

Density, magnetic tape, 657 (COMMANDS)

Density and parity parameters, 415 (PIP)  
     switches for setting, 415

/DEPEND, 622, 681 (COMMANDS)

Dependency, 716 (COMMANDS)  
     setting the, 622, 681

INDEX (Cont)

- Dependency (INTRO TO SOFTWARE)
  - Job, 13
- Deposit command, 502 (COMMANDS)
- Depositing information, 502 (COMMANDS)
- Describing actions to be performed by Batch, 140 (BEGINNER'S BATCH)
- Designating particular groups of characters as a match in searches, 286 (TECO)
- DETACH command, 519 (COMMANDS)
- DETACH command, 81 (TIMESHARING)
- /DETAIL, 521 (COMMANDS)
- Determining the command that caused an error, 322 (TECO)
- Determining station of a device, 706 (COMMANDS)
- Device, 231, 248, 249, 250, 251, 253 (TECO)
- Device Assignment, 10 (INTRO TO SOFTWARE)
- Device name, 448 (COMMANDS)
- Device Name (INTRO TO SOFTWARE)
  - Logical, 28
- Device name, 383 (PIP)
- Device name, 233, 251 (TECO)
- Devices (BEGINNER'S BATCH)
  - mounting, definition, 95
  - peripheral, definition, 95
- Devices (INTRO TO SOFTWARE)
  - Input/Output, 48
  - Real-Time, 14
  - Sharing, 11
- Devices, 75 (TIMESHARING)
- Device station, 706 (COMMANDS)
- Digit numeric protection code values, 391 (PIP)
- Digit string, 302 (TECO)
- DIRECT, 34 (INTRO TO SOFTWARE)
- DIRECT command, 520 (COMMANDS)
- DIRECT command, 66, 82 (TIMESHARING)
- Direct control, 375 (PIP)
- Directory Files, 29, 34 (INTRO TO SOFTWARE)
- Directory identifier, 387, 389 (PIP)
- Directory listings, 520 (COMMANDS)
- Directory names, 451 (COMMANDS)
- Directory structured devices, 250 (TECO)
- Disconnecting a job, 81 (TIMESHARING)
- Disconnecting the terminal, 519 (COMMANDS)
- Disk, definition, 94 (BEGINNER'S BATCH)
- Disk area, 251 (TECO)
- Disk area, 62 (TIMESHARING)
- Disk deletion, 410 (PIP)
- Disk priority, 658 (COMMANDS)
- Disk Quotas, 30 (INTRO TO SOFTWARE)
- Disk System, 29, 48 (INTRO TO SOFTWARE)
- Disk usage, 527 (COMMANDS)
- DISMOUNT command, 525 (COMMANDS)
- DISMOUNT command, 79 (TIMESHARING)
- /DISPOSE switch, 143 (BEGINNER'S BATCH)
  - /DISPOSE:DELETE, 143
  - /DISPOSE: PRESERVE, 143
  - /DISPOSE: RENAME, 144
- /DISPOSE switch, 494, 605, 611, 622, 681, 698 (COMMANDS)
- Divide, 242 (TECO)
- Documentation (COMMANDS)
  - obtaining, 575
  - updating, 545
- \$ (dollar sign), 261, 286 (TECO)
- Double altmode, 194 (INTRO TO TECO)
- Drum System, 47 (INTRO TO SOFTWARE)
- DSK, 30, 34 (INTRO TO SOFTWARE)
- DSK command, 527 (COMMANDS)
- DSKLST, 35 (INTRO TO SOFTWARE)
- DSKRAT, 35 (INTRO TO SOFTWARE)
- Dump, 154 (BEGINNER'S BATCH)
  - definition, 94
  - example, 162
- DUMP, 35 (INTRO TO SOFTWARE)
- \$DUMP card, 723 (COMMANDS)
- DUMP command, 529 (COMMANDS)



Dump descriptor, 534 (COMMANDS)  
 Dumping files, 529, 530 (COMMANDS)  
 DUMP program, 530 (COMMANDS)

E

↑ E (control-E) see Control commands (TECO)  
 E command, 536 (COMMANDS)  
 E command, 361 (LINED)  
 E command, 293 (TECO)  
 EB command, 249, 253 (TECO)  
 E switch, 399 (PIP)  
 Echo, 236 (TECO)  
 EDDT, 35 (INTRO TO SOFTWARE)  
 EDIT, 359 (LINED)  
 Edit, 248, 250 (TECO)  
 EDIT command, 537 (COMMANDS)  
 EDIT command, 64 (TIMESHARING)  
 Edit-class command, 248 (TECO)  
 Editing, programmed, 243 (TECO)  
 Editing Buffer, 20 (INTRO TO SOFTWARE)  
 Editing buffer, 239, 243, 254, 256, 260, 265,  
 266, 267, 272, 278, 279, 296 (TECO)  
 Editing existing files, 248 (TECO)  
 Editing files, 537, 693 (COMMANDS)  
 Editing files, 192 (INTRO TO TECO)  
 Editing files, 64 (TIMESHARING)  
     EDIT, 64  
     TECO, 65  
 Editing line-sequence numbered files, 254 (TECO)  
 Editing process, 191 (INTRO TO TECO)  
 Editors (INTRO TO SOFTWARE)  
     LINED, 20  
     RUNOFF, 21  
     SOUP, 21  
     TECO, 20  
 EF (close output file) command, 203 (INTRO TO  
 TECO)  
 EF command, 274 (TECO)

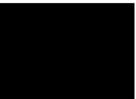
## INDEX (Cont)

EG (close file, reexecute monitor command)  
 command, 204 (INTRO TO TECO)

EG command, 275, 276, 308 (TECO)  
 EH command, 301, 323 (TECO)  
 EM command, 251, 252 (TECO)  
 End-of-file card, 105, 121 (BEGINNER'S BATCH)  
 End-of-file card, 714, 717 (COMMANDS)  
 End-of-file flag, obtaining value, 301 (TECO).  
 End-of-file record, 252 (TECO)  
 End-of-file writing, 538 (COMMANDS)  
 End of page indicator, 191 (INTRO TO TECO)  
 ENTER, 29 (INTRO TO SOFTWARE)  
 Entering items in system queues, 618 (COMMANDS)  
 Entering job, 137 (BEGINNER'S BATCH)  
     into batch's queue, 139  
 EO command, 301, 305 (TECO)  
 EO value, 304 (TECO)  
 \$EOD card, 108, 121 (BEGINNER'S BATCH)  
 \$EOD card, 715, 724 (COMMANDS)  
 EOF command, 538 (COMMANDS)  
 = (equal sign) command, 300, 303 (TECO)  
 Equals (=) symbol delimiter, 376, 379, 382 (PIP)  
 ER, 251, 254 (TECO)  
 Erasing (TECO)  
     entire command string, 320  
     single character, 320  
 Erasing commands, 195 (INTRO TO TECO)  
 Erasing commands, 235, 306, 319, 320 (TECO)  
 Erasing files, 67 (TIMESHARING)  
 %ERR, 733 (COMMANDS)  
 Error, command See Command Error (TECO)  
 \$ERROR card, 122 (BEGINNER'S BATCH)  
 \$ERROR card, 724 (COMMANDS)  
 Error codes, 780 (COMMANDS)  
 .ERROR command, 146 (BEGINNER'S BATCH)  
     example, 147

INDEX (Cont)

- .ERROR command, 735 (COMMANDS)
- Error handling, 363 (LINED)
- Error in command, 195 (INTRO TO TECO)
- Error message categories, 747 (COMMANDS)
- Error message flag, obtaining the value, 301, 324 (TECO)
- Error messages, 153 (BEGINNER'S BATCH)
- Error messages, (COMMANDS)
  - BATCON, 742
  - CDRSTK, 741
  - system, 747
- Error messages, 209 (INTRO TO TECO)
- Error messages, 363 (LINED)
- Error messages, 419 (PIP)
  - general, 422
  - TMPCOR (device TMP), 424
- Error messages, 321, 322, 325 (TECO)
- Error recovery, 131, 150 (BEGINNER'S BATCH)
  - examples, 132, 134, 150, 151
- Error recovery, 724, 732 (COMMANDS)
- Error Recovery, Batch, 13 (INTRO TO SOFTWARE)
- Error recovery, G-switch, 417 (PIP)
- Error reporting (COMMANDS)
  - BATCON, 742
  - CDRSTK, 741
- Errors (PIP)
  - file reference, 420
  - I/O, 418
  - Y-switch, 422
- Errors, 319 (TECO)
- ESC, 235, 236 (TECO)
- ES command, 282, 301 (TECO)
- ET command, 261, 301, 303 (TECO)
- ET flag, 301, 303 (TECO)
- EU command, 301 (TECO)
- EW command, 248, 251, 252, 274 (TECO)
- EX (close file) command, 204 (INTRO TO TECO)
- EX command, 275, 276, 307, 308 (TECO)
- Examine command, 536 (COMMANDS)
- Examining core locations, 536 (COMMANDS)
- Examining object programs, 466 (COMMANDS)
  - D, 502
  - DCORE, 505
  - DUMP, 529, 530
  - E, 536
- Examples (BEGINNER'S BATCH)
  - \$ALGOL card, 112
  - ALGOL job, 169, 176
  - .BACKTO command, 146
  - BASIC job, 110, 171, 178
  - \$COBOL card, 114
  - COBOL job, 154, 174, 183
  - Control file, 138, 160
  - \$DATA card, 116
  - \$DECK card, 120
  - dump, 162
  - .ERROR command, 147
  - error recovery, 132, 134, 150, 151
  - \$FORTRAN card, 124
  - FORTRAN job, 172
  - .GOTO command, 148
  - job, 138, 169, 176
  - loader map, 156
  - log file, 158, 161
  - \$MACRO card, 129
  - MOUNT command, 175, 183
  - mounting tapes, 175, 183
  - .NOERROR command, 149
  - output, 154, 160
  - submitting jobs, 144
  - use of \$EOD card, 108, 122
- Examples, 361 (LINED)
- Examples, (TIMESHARING)
  - assigning devices, 78
  - compiling and executing, 69
  - creating a file, 63
  - logging in, 61
  - KJOB CONFIRM dialogue, 72
  - SYSTAT, 85
- Exclamation point, 292 (TECO)
- Exclamation symbol (!), 382 (PIP)
- Execute, definition, 94 (BEGINNER'S BATCH)
- EXECUTE, 276 (TECO)



## INDEX (Cont)

EXECUTE command, 539 (COMMANDS)  
 EXECUTE command, 68 (TIMESHARING)  
 Executing programs, 539, 679, 721 (COMMANDS)  
 Executing programs, 67 (TIMESHARING)  
     EXECUTE, 68  
 Execution, 239, 244 (TECO)  
 Execution of command string, 194 (INTRO TO  
 TECO)  
 Exit, 275, 276, 277 (TECO)  
 Exit commands, 204 (INTRO TO TECO)  
 Exit command, 275, 276, 278 (TECO)  
 Exiting from PIP, 375 (PIP)  
 Expression, 534 (COMMANDS)  
 Extension, definition, 94 (BEGINNER'S BATCH)  
 Extensions, filename, 783 (COMMANDS)  
 Extension, filename, 192 (INTRO TO TECO)  
 EZ command, 253 (TECO)

## F

↑ F (control-F) see Control commands (TECO)  
 F command, 293 (TECO)  
 F switch, 407 (PIP)  
 /F, QUEUE, 612, 622, 682, 698 (COMMANDS)  
 \$F40 card, 725 (COMMANDS)  
 Facility allocation, 464 (COMMANDS)  
     ASSIGN, 470  
     CLOSE, 479  
     CORE, 491  
     DEASSIGN, 512  
     DISMOUNT, 525  
     FINISH, 560  
     LOCATE, 589  
     MOUNT, 593  
     REASSIGN, 633  
     SET BLOCKSIZE, 653  
     SET CDR, 654  
     SET CPU, 655  
     SET DENSITY, 657  
     SET DSKPRI, 658  
     SET HPQ, 659  
     SET SPOOL, 660

Facility allocation, (cont)  
     SET TTY, 668  
     TTY, 668  
 FAILSAFE, 35 (INTRO TO SOFTWARE)  
 Fatal error, character recognized as, 146  
 (BEGINNER'S BATCH)  
 Features enabled by EO values greater than 1,  
 305 (TECO)  
 Feature test switches, 463 (COMMANDS)  
 FED, 21 (INTRO TO SOFTWARE)  
 /FEET switch (BEGINNER'S BATCH)  
     \$JOB card, 126  
     SUBMIT command, 142  
 /FEET switch, 622, 682 (COMMANDS)  
 Fields, filename, 384 (PIP)  
 FILCOM program, 543 (COMMANDS)  
 FILDDT, 35 (INTRO TO SOFTWARE)  
 /FILE, 612, 623, 698 (COMMANDS)  
 File (INTRO TO SOFTWARE)  
     Directory, 29  
     Log, 13  
 File, definition, 94 (BEGINNER'S BATCH)  
 File, definition, 191 (INTRO TO TECO)  
 File access protection codes, 383, 390, 391  
 (PIP)  
 File Backup Utility, 22 (INTRO TO SOFTWARE)  
 FILE command, 553 (COMMANDS)  
 File-control switches, 143 (BEGINNER'S BATCH)  
 File control switches, 493 (COMMANDS)  
 File creation, 191 (INTRO TO TECO)  
 File devices, 191 (INTRO TO TECO)  
 File Directory, 29, 34 (INTRO TO SOFTWARE)  
 File directory switches, 406 (PIP)  
 File Handler, 25, 29 (INTRO TO SOFTWARE)  
 File manipulation, 465 (COMMANDS)  
     ALCFIL, 468  
     BACKSPACE, 474  
     BACKUP, 475  
     COPY, 486

## INDEX (Cont)

- File manipulation (cont)
  - CPUNCH, 493
  - DELETE, 517
  - DIRECT, 520
  - EOF, 538
  - FILE, 553
  - FILEX, 557
  - LIST, 584
  - PLOT, 604
  - PRESERVE, 609
  - PRINT, 610
  - PROTECT, 616
  - QUEUE, 618
  - RENAME, 638
  - RESTORE, 641
  - REWIND, 645
  - SKIP, 675
  - SUBMIT, 680
  - TPUNCH, 696
  - TYPE, 702
  - UNLOAD, 703
  - ZERO, 707
- Filename, definition, 94 (BEGINNER'S BATCH)
- Filename, 451 (COMMANDS)
- Filename, 192 (INTRO TO TECO)
- Filename, 233, 234, 248, 249, 250, 251 (TECO)
- Filename, 62 (TIMESHARING)
- Filename extension, definition, 94 (BEGINNER'S BATCH)
- Filename extension, 451, 783 (COMMANDS)
- Filename extension, 234, 249, 251, 253, (TECO)
- Filename extension, 62 (TIMESHARING)
- Filename fields, 383 (PIP)
- Filenames, 384 (PIP)
  - generation of, 396
- File Owner, 30 (INTRO TO SOFTWARE)
- File protection, 616 (COMMANDS)
- File protection codes, 409 (PIP)
  - changing of, 408
  - UFD and SFD, 392
- File reference errors, 420 (PIP)
- File request, backspace, 416 (PIP)
- Files, 11 (INTRO TO SOFTWARE)
  - Chain, 22
  - Directory, 29, 35
  - Named, 29, 39
  - Sharing, 11
- Files, 61 (TIMESHARING)
- Files, temporary, 454, 791 (COMMANDS)
- File selection, 249, 251 (TECO)
- File specification, 451 (COMMANDS)
- File specification, 29, 35 (INTRO TO SOFTWARE)
- File specification, 380, 381 (PIP)
  - delimiters, 381
- File Storage Allocation, 11, 30 (INTRO TO SOFTWARE)
- File structure names, 451 (COMMANDS)
- File structures, 29, 35 (INTRO TO SOFTWARE)
- File System, 11, 29 (INTRO TO SOFTWARE)
- File transfer, 395 (PIP)
  - nondirectory device to directory device, 395
- File transfer program, 557 (COMMANDS)
- FILEX program, 557 (COMMANDS)
- FILEX program, 22, 35 (INTRO TO SOFTWARE)
- Filler classes, 669 (COMMANDS)
- %FIN, 734 (COMMANDS)
- FINISH command, 560 (COMMANDS)
- FINISH command, 80 (TIMESHARING)
- Flow control commands, 291 (TECO)
- FN command, 280 (TECO)
- Format (BEGINNER'S BATCH)
  - \$ALGOL card, 111
  - .BACKTO command, 123, 133
  - Batch command, 138
  - Batch command card, 104
  - Card, 103
  - \$COBOL card, 111
  - control cards, 103
  - \$DATA card, 115

## Format (cont) (BEGINNER'S BATCH)

- data cards, 104
- data line, 139
- \$DECK card, 120
- end-of-file card, 121
- \$EOD card, 122
- \$ERROR card, 122
- .ERROR command, 146
- \$FORTRAN card, 123
- .GOTO command, 123, 147
- .IF command, 148
- .IF (ERROR) command, 122, 148
- .IF (NOERROR) command, 130, 148
- \$JOB card, 125
- lines in control file, 138
- \$MACRO card, 128
- monitor command
  - card, 104
  - line, 138
- \$NOERROR card, 130
- .NOERROR command, 149
- \$PASSWORD card, 131
- program cards, 104
- QUEUE INP: monitor command, 139
- \$SEQUENCE card, 131
- SUBMIT monitor command, 139
- system program command
  - card, 104
  - line, 139

Format, 237 (TECO)

Formats, command, 447 (COMMANDS)

Formatting command strings, 306 (TECO)

Form feed, 191 (INTRO TO TECO)

- removal, 197
- symbol, 194

Form feed characters, 234, 236, 237, 238, 256, 261, 272, 273, 274, 309, 310, 312 (TECO)

Form feed flag, 256, 273, 301, 302 (TECO)

Form feed processing, 309 (TECO)

/FORMS, 495, 606, 612, 623 (COMMANDS)

## FORTRAN (BEGINNER'S BATCH)

- compiler switches, 124
- deck, setting up, 106
- definition, 94
- job, examples, 172, 180
- program, compiling and executing, 107

## INDEX (Cont)

FORTRAN, 19 (INTRO TO SOFTWARE)

FORTRAN binary files, copying, 401 (PIP)

\$FORTRAN card, 106, 123 (BEGINNER'S BATCH)

- examples, 124
- switches, 123

\$FORTRAN card, 713, 715, 725 (COMMANDS)

FORTRAN carriage control character

- interpretation, 401 (PIP)

/FORTRAN switch, 481, 513, 540, 586 (COMMANDS)

FS command, 279 (TECO)

/FUDGE, 482, 513, 540, 586 (COMMANDS)

FUDGE command, 562 (COMMANDS)

FUDGE2 program, 563 (COMMANDS)

FUDGE2 program, 36 (INTRO TO SOFTWARE)

Functional groups of commands, 463 (COMMANDS)

Functions, optional, 393 (PIP)

Functions of control cards, 103 (BEGINNER'S BATCH)

## G

↑ G (control-G) see Control commands (TECO)

G command, 293, 296 (TECO)

G switch, error recovery, 417 (PIP)

Gaining access (COMMANDS)

- of a device, 470, 593
- to a file structure, 593
- to the system, 590

General error messages, 422 (PIP)

General QUEUE switches, 493 (COMMANDS)

General Switches, 141 (BEGINNER'S BATCH)

Generating cross-referenced listing, 481 (COMMANDS)

- global symbols, 569

Generating library REL files, 482, 513, 540, 562, 586 (COMMANDS)

INDEX (cont)

- Generic Name, 30, 36 (INTRO TO SOFTWARE)
  - /GENLSN (generate line-sequence numbers) (TECO)
    - switch, 254
  - GET command, 568 (COMMANDS)
  - GET command, 84 (TIMESHARING)
  - Getting date and time, 71 (TIMESHARING)
  - Getting information, 70, 85 (TIMESHARING)
    - DAYTIME, 71
    - PJOB, 71
    - RESOURCES, 85
    - SYSTAT, 85
    - TIME, 71
  - Getting job number, 71 (TIMESHARING)
  - Getting on the system, 59 (TIMESHARING)
  - Getting running time, 71 (TIMESHARING)
  - Global symbols, 569 (COMMANDS)
  - GLOB program, 569 (COMMANDS)
  - GLOB program, 36 (INTRO TO SOFTWARE)
  - Glossary, 31 (INTRO TO SOFTWARE)
  - .GOTO command, 123, 130, 147 (BEGINNER'S BATCH)
    - example, 148
  - .GOTO command, 735 (COMMANDS)
  - GRIPE program, 573 (COMMANDS)
  - GRIPE program, 36 (INTRO TO SOFTWARE)
  - Groups of commands, 463 (COMMANDS)
- H
- † H (control-H) see Control commands (TECO)
  - H, 258 (TECO)
  - H switch, 405 (PIP)
  - HALT command, 574 (COMMANDS)
  - HALT command, 84 (TIMESHARING)
  - Handler, (INTRO TO SOFTWARE)
    - File, 25, 29
    - Service Request, 9
    - UUO, 25, 28
  - Hardware, 9, 45 (INTRO TO SOFTWARE)
  - Hardware requirements, 375 (PIP)
  - /HEADER, 612, 623 (COMMANDS)
  - HELP command, 575 (COMMANDS)
  - High priority queues, 659 (COMMANDS)
  - High-Priority Run Queues, 15 (INTRO TO SOFTWARE)
  - HK (delete buffer) command, 201 (INTRO TO TECO)
  - Holding a job until a specified time, 126, 141 (BEGINNER'S BATCH)
  - How Batch reads (BEGINNER'S BATCH)
    - card decks, 108
    - control files, 146
  - How to use Batch, 99 (BEGINNER'S BATCH)
  - HP command, 310, 311 (TECO)
  - HT (type entire buffer) command, 200 (INTRO TO TECO)
- I
- I (insert) command, 202, 203 (INTRO TO TECO)
  - I command, 360 (LINED)
  - I command, 266, 267 (TECO)
  - \*i command, 271, 296 (TECO)
  - I switch, 405 (PIP)
  - Identification code, 60 (TIMESHARING)
  - Identifier (PIP)
    - DECTape, 397
    - directory, 387
  - Identifying the job, 127 (BEGINNER'S BATCH)
  - Identifying the user, 130 (BEGINNER'S BATCH)
  - Idle state, 247 (TECO)
  - .IF command (BEGINNER'S BATCH)
    - .IF (ERROR), 122, 148
    - .IF (NOERROR), 129, 148
  - Ignore card sequence numbers, (PIP)
    - (E switch,) 399



- INDEX (cont)
- Ignoring fatal error messages, 149  
(BEGINNER'S BATCH)
  - Illegal commands, 209 (INTRO TO TECO)
  - Immediate action commands, 235 (TECO)
  - Implementation, 365 (LINED)
  - INBUF, 28 (INTRO TO SOFTWARE)
  - Increasing the number of entries in the pushdown  
stack, 243 (TECO)
  - Increment, 295 (TECO)
  - INDEX file, 475 (COMMANDS)
  - Indirect commands, 454 (COMMANDS)
  - Indirect control, 375 (PIP)
  - Information about job, 467 (COMMANDS)
    - DSK, 527
    - PJOB, 601
    - QUOLST, 631
    - SETSRC, 662
    - SET TIME, 666
    - SET WATCH, 672
    - TIME, 694
  - Information about system, 467 (COMMANDS)
    - DAYTIME, 504
    - RESOURCES, 640
    - SCHED, 650
    - SYSTAT, 686
    - VERSION, 704
    - WHERE, 706
  - INIT, 28 (INTRO TO SOFTWARE)
  - INITIA, 36 (INTRO TO SOFTWARE)
  - INITIA command, 577 (COMMANDS)
  - Initialization, 247, 250 (TECO)
  - Initialization of TECO, 192 (INTRO TO TECO)
  - Initializing a file for processing, 362 (LINED)
  - Initializing a job, 463 (COMMANDS)
    - Batch, 712
    - INITIA, 577
    - LOGIN, 590
  - Input, 233, 234, 249, 251, 253, 255, 256,  
272, 273, 276, 278, 279 (TECO)
  - Input commands, 197 (INTRO TO TECO)
  - Input file, 233, 272, 276, 278, 279, 288 (TECO)
  - Input file devices, 191 (INTRO TO TECO)
  - Input/Output Devices, 48 (INTRO TO SOFTWARE)
  - Input/Output Routines, 25, 28 (INTRO TO  
SOFTWARE)
  - Input queue, Batch, 680 (COMMANDS)
  - Input Queue, 13 (INTRO TO SOFTWARE)
  - INPUT UO, 29 (INTRO TO SOFTWARE)
  - Insert, 233, 234, 248, 264, 265 (TECO)
  - Inserting (TECO)
    - control characters, 271
    - single control characters as text, 271
    - succeeding control characters as text, 271
  - Inserting a line, 360 (LINED)
  - Inserting multiple lines, 360 (LINED)
  - Insertion, 307 (TECO)
  - Insertion command, 264 (TECO)
  - Insert sequence numbers, S-switch, 400 (PIP)
  - Inside-text command, 271, 285 (TECO)
  - Integer divide, 242 (TECO)
  - Interjob dependency, 716 (COMMANDS)
  - Interpreters, 17 (INTRO TO SOFTWARE)
    - AID, 20
  - Interpreting control-characters as text in  
searches (TECO)
    - single characters, 285
    - succeeding characters, 285
  - Interpreting printed output, 153 (BEGINNER'S  
BATCH)
  - Interrupting sequential reading of control file,  
734, 735 (COMMANDS)
  - Interrupt System (INTRO TO SOFTWARE)
    - Priority, 14, 46
  - Intervention, (INTRO TO SOFTWARE)
    - Batch Operator, 14
  - I/O (PIP)
    - errors, 418
    - messages, 419

INDEX (cont)

I/O Channels (INTRO TO SOFTWARE)

Software, 28

I/O Service Routines, 10, 28 (INTRO TO SOFTWARE)

Iteration, 289, 296 (TECO)

J

J (move pointer to beginning) command, 198 (INTRO TO TECO)

J command, 258 (TECO)

J switch, card punch, 418 (PIP)

JCONTINUE command, 578 (COMMANDS)

Job, 99 (BEGINNER'S BATCH)

definition, 94

entering to Batch, 137

examples, 138, 169, 176

submitting, 139

examples, 144

Job, 443, 712 (COMMANDS)

\$JOB card, 105, 125 (BEGINNER'S BATCH)

switches, 126

\$JOB card, 713, 714, 727 (COMMANDS)

JOB category, 505 (COMMANDS)

Job contained in a file, 714 (COMMANDS)

Job Dependency, 13 (INTRO TO SOFTWARE)

Job in a control file, 716 (COMMANDS)

Job information, 467 (COMMANDS)

DSK, 527

PJOB, 601

QUOLST, 631

SETSRC, 662

SET TIME, 666

SET WATCH, 672

TIME, 694

Job initialization, 463 (COMMANDS)

Batch, 712

INITIA, 577

LOGIN, 590

Job Locking, 14 (INTRO TO SOFTWARE)

Jobname, definition, 91 (BEGINNER'S BATCH)

Jobname, 493, 604, 610, 619, 680, 696 (COMMANDS)

Job number, 60 (TIMESHARING)

Job on cards, 713 (COMMANDS)

Job output, Batch, 739 (COMMANDS)

Job search list, 662 (COMMANDS)

Job Search List, 30, 35 (INTRO TO SOFTWARE)

Jobstep, definition, 95 (BEGINNER'S BATCH)

Job termination, 466 (COMMANDS)

KJOB, 579

K

K, definition, 95 (BEGINNER'S BATCH)

K (delete line) command, 201 (INTRO TO TECO)

K command, 264, 265 (TECO)

KA10, 46 (INTRO TO SOFTWARE)

KI10, 46 (INTRO TO SOFTWARE)

Killing your job, 579 (COMMANDS)

/KILL switch, 140 (BEGINNER'S BATCH)

/KILL switch, 495, 606, 612, 623, 682, 698 (COMMANDS)

Kinds of printed output, 153 (BEGINNER'S BATCH)

KJOB, 23 (INTRO TO SOFTWARE)

KJOB command, 579 (COMMANDS)

KJOB command, 72 (TIMESHARING)

L

↑ L (control-L) see Control commands (TECO)

L (move pointer by lines) command, 199 (INTRO TO TECO)

L command, 259, 293 (TECO)

L switch, 406 (PIP)

Label, definition, 95 (BEGINNER'S BATCH)

Language (INTRO TO SOFTWARE)

Command Control, 10

Machine, 17

Language Programming (INTRO TO SOFTWARE)

Symbolic, 17

- INDEX (cont)
- Leaving the system, 72 (TIMESHARING)
  - Leaving the terminal in monitor mode, 500 (COMMANDS)
  - LIB40, 37 (INTRO TO SOFTWARE)
  - LIBOL, 19, 37 (INTRO TO SOFTWARE)
  - /LIBRARY, 514, 540, 586 (COMMANDS)
  - Library Maintenance Program, 19 (INTRO TO SOFTWARE)
  - /LIMIT, 495, 606, 612, 624, 699 (COMMANDS)
  - Line, 237, 259, 260, 268 (TECO)
  - Line continuation, 139 (BEGINNER'S BATCH)
  - LINED program, 498, 537 (COMMANDS)
  - LINED program, 20 (INTRO TO SOFTWARE)
  - LINED program, 62, 64 (TIMESHARING)
  - Line editor, 359 (LINED)
  - Line feed, 236, 237, 256, 262, 306, 320 (TECO)
  - Line feed symbol ( $\downarrow$ ), 194 (INTRO TO TECO)
  - Line numbers, 20 (INTRO TO SOFTWARE)
  - Line Printer, 49 (INTRO TO SOFTWARE)
  - Line printer listing, FORTRAN, (PIP) (P switch), 400
  - Line printer output, 81 (TIMESHARING)
    - CREF, 82
    - DIRECT, 82
    - PRINT, 81
  - Line printer output, specifying amount, 127, 142 (BEGINNER'S BATCH)
  - Line printer queue, 610 (COMMANDS)
  - Line sequence numbered files, 254 (TECO)
  - Line sequence numbers, 365 (LINED)
  - Line sequence numbers, 238, 254 (TECO)
  - Line terminators, 237 (TECO)
  - List (INTRO TO SOFTWARE)
    - Command, 28
    - Job Search, 30
  - /LIST, COMPIL, 482, 515, 540, 586 (COMMANDS)
  - /LIST, DIRECT, 521 (COMMANDS)
  - /LIST, QUEUE, 495, 606, 612, 624, 682, 699 (COMMANDS)
  - LIST command, 584 (COMMANDS)
  - Listing available devices, 640 (COMMANDS)
  - Listing cross-referenced files, 499 (COMMANDS)
  - Listing directories, 520 (COMMANDS)
  - Listing directories, 66, 82 (TIMESHARING)
  - Listing files, 66, 81 (TIMESHARING)
  - Listings, 154 (BEGINNER'S BATCH)
  - Listing source files, 584 (COMMANDS)
  - List limited source directory, (PIP)
    - F-switch, 407
  - List source device directory, (PIP)
    - L switch, 406
  - Literal type-out mode, 261 (TECO)
  - /LMAP, 515, 541, 587 (COMMANDS)
  - LOAD, 276 (TECO)
  - LOAD command, 585 (COMMANDS)
  - LOAD command, 68 (TIMESHARING)
  - Loader map, 154 (BEGINNER'S BATCH)
    - example, 156
  - LOADER program, 22, 37 (INTRO TO SOFTWARE)
  - LOADER program, 68 (TIMESHARING)
  - LOADER switches, 460 (COMMANDS)
  - Loading and starting core image file, 632, 646 (COMMANDS)
  - Loading and writing a Bootstrap loader, 488 (COMMANDS)
  - Loading core image file, 568 (COMMANDS)
    - and starting, 632, 646
  - Loading files, 585 (COMMANDS)
  - Loading PIP, 375 (PIP)
  - Loading programs, 67 (TIMESHARING)
    - LOAD, 68
  - Load point, 252 (TECO)
  - LOCATE command, 589 (COMMANDS)

## INDEX (cont)

- Locking Jobs, 14 (INTRO TO SOFTWARE)  
 /LOG, 612, 624 (COMMANDS)  
 Log file, 100, 153 (BEGINNER'S BATCH)  
     definition, 95  
     examples, 158, 161  
 Log file (COMMANDS)  
     BACKUP, 475  
     BATCH, 680, 710, 727, 740  
 Log File, 13 (INTRO TO SOFTWARE)  
 Logged-In Quota, 30 (INTRO TO SOFTWARE)  
 Logged-Out Quota, 30 (INTRO TO SOFTWARE)  
 Logging on, 590 (COMMANDS)  
 Logging off, 579 (COMMANDS)  
 Logical AND, 242 (TECO)  
 Logical Block Numbers, 29 (INTRO TO SOFTWARE)  
 Logical device name, 449 (COMMANDS)  
     example of, 450  
 Logical Device Name, 28, 38 (INTRO TO SOFTWARE)  
 Logical device name, 383 (PIP)  
 Logical device name, 76 (TIMESHARING)  
 Logical OR, 242 (TECO)  
 Logical station, 589 (COMMANDS)  
 Logical Station, 16 (INTRO TO SOFTWARE)  
 Login check, 446 (COMMANDS)  
 LOGIN command, 590 (COMMANDS)  
 LOGIN command, 60 (TIMESHARING)  
 LOGIN program, 23, 38 (INTRO TO SOFTWARE)  
 LOGIN program, 60 (TIMESHARING)  
 LOGOUT, 23 (INTRO TO SOFTWARE)  
 LOOKFL, 38 (INTRO TO SOFTWARE)  
 LOOKUP, 29 (INTRO TO SOFTWARE)  
 Loop, 278, 289 (TECO)  
 Lower case flagging, 262 (TECO)
- M
- M command, 296, 313 (TECO)  
 Machine Language, 17 (INTRO TO SOFTWARE)  
 MACRO (BEGINNER'S BATCH)  
     assembler switches, 128  
     deck, setting up, 107  
     program, assembling and executing, 107  
 MACRO, 17 (INTRO TO SOFTWARE)  
 Macro, 313, 314 (TECO)  
 Macro Capabilities, 18 (INTRO TO SOFTWARE)  
 \$MACRO card, 107, 127 (BEGINNER'S BATCH)  
     examples, 129  
     switches, 128  
 \$MACRO card, 729 (COMMANDS)  
 /MACRO switch, 482, 515, 541, 587 (COMMANDS)  
 /MACX11, 482, 515, 541, 587 (COMMANDS)  
 Magnetic tape, 251, 252, 255 (TECO)  
 Magnetic tape density, 657 (COMMANDS)  
 Magnetic tape switches, 415, 416 (PIP)  
 Magnetic Tape Systems, 48 (INTRO TO SOFTWARE)  
 Main uses of TECO, 247 (TECO)  
 MAKE command, 592 (COMMANDS)  
 MAKE (make disk file) command, 192 (INTRO TO TECO)  
 MAKE command, 247, 248, 250, 252, 307, 308 (TECO)  
 MAKE command, 64 (TIMESHARING)  
 Manager (INTRO TO SOFTWARE)  
     Queue, 13  
 Manipulating core images, 83 (TIMESHARING)  
     GET, 84  
     R, 83  
     RUN, 83  
     SAVE, 83

## INDEX (cont)

## Manipulating files, 465 (COMMANDS)

ALCFIL, 468  
 BACKSPACE, 474  
 BACKUP, 475  
 COPY, 486  
 CPUNCH, 493  
 DELETE, 517  
 DIRECT, 520  
 EOF, 538  
 FILE, 533  
 FILEX, 557  
 LIST, 584  
 PLOT, 604  
 PRESERVE, 609  
 PRINT, 610  
 PROTECT, 616  
 QUEUE, 618  
 RENAME, 638  
 RESTORE, 641  
 REWIND, 645  
 SKIP, 675  
 SUBMIT, 680  
 TPUNCH, 696  
 TYPE, 702  
 UNLOAD, 703  
 ZERO, 707

## Manipulating files, 65 (TIMESHARING)

DELETE, 67  
 DIRECT, 66  
 RENAME, 67  
 TYPE, 66

## Manipulating terminals, 80 (TIMESHARING)

ATTACH, 81  
 DETACH, 81  
 SEND, 80

## /MARKS, 521 (COMMANDS)

## Master file, 563 (COMMANDS)

## Master File Directory, 29, 38 (INTRO TO SOFTWARE)

## Maximum length of search strings, 278 (TECO)

## Memory (INTRO TO SOFTWARE)

Core, 47  
 Secondary, 11

## Merge, 310, 311 (TECO)

## Message of the day, 590 (COMMANDS)

## Messages (COMMANDS)

BATCON, 742  
 CDRSTK, 740  
 system, 747

## MFD, 29, 38 (INTRO TO SOFTWARE)

## - (minus), 242 (TECO)

## \$MODE card, 730 (COMMANDS)

## Modes (INTRO TO SOFTWARE)

Block, 15  
 Buffered Data, 28  
 Single, 15  
 Unbuffered Data, 28

## Modifiers, 281 (TECO)

## /MODIFY switch, 141 (BEGINNER'S BATCH)

## /MODIFY switch, 495, 606, 613, 624, 682, 699 (COMMANDS)

## Modifying amount of core assigned, 491 (COMMANDS)

## Modifying text, 192 (INTRO TO TECO)

## MONEY, 38 (INTRO TO SOFTWARE)

## MONGEN, 23, 38 (INTRO TO SOFTWARE)

## Monitor, definition, 95 (BEGINNER'S BATCH)

## Monitor, 59 (TIMESHARING)

## Monitor command (BEGINNER'S BATCH)

card format, 104  
 definition, 95  
 line format, 138

## Monitor command language, 443 (COMMANDS)

## Monitor commands, 235, 248 (TECO)

## Monitor mode, 444 (COMMANDS)

## Monitor Support Programs, 23 (INTRO TO SOFTWARE)

## MOUNT command, 100 (BEGINNER'S BATCH)

example, 175, 183

## MOUNT command, 593 (COMMANDS)

## MOUNT command, 78 (TIMESHARING)

## Mounting a device, definition, 95 (BEGINNER'S BATCH)

INDEX (cont)

- Mounting tapes, 100 (BEGINNER'S BATCH)
    - examples, 175, 183
  - Moving a file to Batch's disk area, 144 (BEGINNER'S BATCH)
  - /MULTI, 594 (COMMANDS)
  - Multimode Computing, 10 (INTRO TO SOFTWARE)
  - Multiple job control, 466 (COMMANDS)
    - ATTACH, 472
    - CCONT, 500
    - CSTART, 500
    - DETACH, 519
    - OPSER, 597
    - REATA, 635
  - Multiply, 242 (TECO)
  - Multiprocessing, 10, 38 (INTRO TO SOFTWARE)
  - Multiprogram Batch, 99 (BEGINNER'S BATCH)
  - Multiprogram Batch, 12 (INTRO TO SOFTWARE)
  - Multiprogramming, definition, 95 (BEGINNER'S BATCH)
  - Multiprogramming, 443 (COMMANDS)
  - Multiprogramming, 11, 28, 39 (INTRO TO SOFTWARE)
  - Multi-purpose commands, 307 (TECO)
- N
- † N (control-N) see Control commands (TECO)
  - N (search file) command, 206, 207 (INTRO TO TECO)
  - N command, 279, 281, 293, 310 (TECO)
  - N switch, delete sequence number, 399 (PIP)
  - nA command, 301 (TECO)
  - n\ command, 266 (TECO)
  - nK CORE, 244, 256 (TECO)
  - nI (\$) command, 266 (TECO)
  - Name (INTRO TO SOFTWARE)
    - Generic, 30, 36
    - Logical Device, 28, 38
  - Named Files, 29, 39 (INTRO TO SOFTWARE)
  - Naming control files, 140 (BEGINNER'S BATCH)
  - Naming data files on the \$DATA card, 116 (BEGINNER'S BATCH)
  - Naming files, 62 (TIMESHARING)
  - Naming files with octal constants, 385 (PIP)
  - Naming jobs, 125, 140 (BEGINNER'S BATCH)
  - Naming log files, 140 (BEGINNER'S BATCH)
  - Negation, 242 (TECO)
  - /NEW, 496, 606, 613, 624, 682, 699 (COMMANDS)
  - nnnTEC.TMP, 249, 253 (TECO)
  - No case flagging, 262 (TECO)
  - /NOCOMPILE, 482, 515, 541, 587 (COMMANDS)
  - \$NOERROR card, 129 (BEGINNER'S BATCH)
  - \$NOERROR card, 724 (COMMANDS)
  - .NOERROR command, 149 (BEGINNER'S BATCH)
    - example, 149
  - .NOERROR command, 737 (COMMANDS)
  - /NOLIST switch (BEGINNER'S BATCH)
    - \$ALGOL card, 111
    - \$FORTRAN card, 124
    - \$MACRO card, 129
  - /NOLIST switch, 482, 515, 541, 587 (COMMANDS)
  - Non-directory to Directory copy (PIP)
    - operation, 395
  - Non-Resident Software, 10, 17 (INTRO TO SOFTWARE)
    - .NOOPERATOR command, 737 (COMMANDS)
  - No prevailing case conversion, 270 (TECO)
  - /NOSEARCH, 515, 541, 587 (COMMANDS)
  - /NOTE, 607, 613, 624 (COMMANDS)
  - NOTICE.TXT, 590 (COMMANDS)
  - Null, 234 (TECO)
  - Null extension, 234, 251 (TECO)
  - Null page, 238 (TECO)

## INDEX (cont)

- /NULL switch, 496, 607, 613, 624, 699 (COMMANDS)
- Numbers (INTRO TO SOFTWARE)
  - Line, 20
  - Logical Block, 29
- # symbol, 385 (PIP)
- Numeric argument, 196 (INTRO TO TECO)
- Numeric argument, 241, 251, 257, 281 (TECO)
- Numeric operators, 242 (TECO)
  -
- † ○ (control-0) see Control commands (TECO)
- command, 292 (TECO)
- switch, insert sequence numbers and increment by one, 400 (PIP)
- Object program, definition, 95 (BEGINNER'S BATCH)
- Object program control, 465 (COMMANDS)
  - CONTINUE, 495
  - DDT, 510
  - GET, 568
  - HALT, 574
  - JCONT, 578
  - R, 632
  - REENTER, 637
  - RUN, 646
  - START, 679
- Object program examination, 466 (COMMANDS)
  - D, 502
  - DCORE, 505
  - DUMP, 529, 530
  - E, 536
- Object program preparation, 465 (COMMANDS)
  - COMPILE, 480
  - CREF, 499
  - DEBUG, 513
  - EXECUTE, 539
  - FUDGE, 562
  - FUDGE2, 563
  - LOAD, 585
  - SAVE, 648
  - SSAVE, 677
- Obtaining cross reference listing (BEGINNER'S BATCH)
  - \$COBOL card, 114
  - \$FORTRAN card, 124
  - \$MACRO card, 129
- Obtaining documentation, 575 (COMMANDS)
- Obtaining entries in queues, 618 (COMMANDS)
- Obtaining information, 70, 85 (TIMESHARING)
  - DAYTIME, 71
  - PJOB, 71
  - RESOURCES, 85
  - SYSTAT, 85
  - TIME, 71
- Obtaining job number, 601 (COMMANDS)
- Obtaining more information about errors, 323 (TECO)
- Obtaining printable dumps, 530 (COMMANDS)
- Obtaining the date and time, 504 (COMMANDS)
- Obtaining the value of (TECO)
  - automatic timeout flag, 301
  - case flag, 301
  - end-of-file flag, 301
  - error message flag, 301, 324
  - search mode flag, 286
  - timeout mode switch, 261
  - version number flag, 301
- Octal constants as filename, 385 (PIP)
- Octal numbers, 242 (TECO)
- /OKBINARY, 613, 624 (COMMANDS)
- /OKNONE, DIRECT, 521 (COMMANDS)
- /OKNONE, QUEUE, 496, 607, 613, 624, 699 (COMMANDS)
- OMOUNT program, 525, 554, 593 (COMMANDS)
- OMOUNT program, 39 (INTRO TO SOFTWARE)
- OPEN, 28 (INTRO TO SOFTWARE)
- Opening a file, 192 (INTRO TO TECO)
- Opening a new file, 248 (TECO)
- Operating procedure, 191 (INTRO TO TECO)

INDEX (cont)

- Operating system, 443 (COMMANDS)
    - commands, 463
  - Operating System, 9, 25 (INTRO TO SOFTWARE)
  - Operating the terminal, 73 (TIMESHARING)
  - Operator, computer, definition, 93 (BEGINNER'S BATCH)
  - .OPERATOR command, 737 (COMMANDS)
  - Operator communication, 466 (COMMANDS)
    - DISMOUNT, 525
    - FILE, 553
    - GRIPE, 573
    - MOUNT, 593
    - PLEASE, 602
    - SEND, 651
  - Operator communication, 78, 79 (TIMESHARING)
  - Operator Intervention (INTRO TO SOFTWARE)
    - Batch, 14
  - Operators (INTRO TO SOFTWARE)
    - Programmed, 25, 27, 40
  - Operators, 242 (TECO)
  - OPSER program, 597 (COMMANDS)
  - OPSER program, 23 (INTRO TO SOFTWARE)
  - Optional functions, 393 (PIP)
  - Optional PIP functions, 415 (PIP)
  - OR, 242 (TECO)
  - OUT, 29 (INTRO TO SOFTWARE)
  - OUTBUF, 28 (INTRO TO SOFTWARE)
  - /OUTPUT, 625, 683 (COMMANDS)
  - Output (BEGINNER'S BATCH)
    - card, 100
    - line printer, 100
    - paper tape, 100
    - plotter, 100
    - tape, 100
  - Output, 234, 250, 251, 252, 272, 274, 276, 277, 287, 302, 307, 310 (TECO)
  - Output, Batch, 739 (COMMANDS)
  - Output, specifying amount (BEGINNER'S BATCH)
    - card, 126, 142
    - line printer pages, 127, 142
    - paper tape, 126, 142
    - plotter time, 127, 143
  - Output commands, 203 (INTRO TO TECO)
  - Output commands, 272 (TECO)
  - Output error, 313 (TECO)
  - Output file, 233, 272, 273, 275, 278, 280 (TECO)
  - Output file devices, 191 (INTRO TO TECO)
  - Output from a job, 153 (BEGINNER'S BATCH)
  - Output Spoolers, 11, 13 (INTRO TO SOFTWARE)
  - OUTPUT UUO, 29 (INTRO TO SOFTWARE)
  - Owner (INTRO TO SOFTWARE)
    - File, 30
- P
- P (output buffer) command, 203 (INTRO TO TECO)
  - P command, 361 (LINED)
  - P command, 272, 274, 280 (TECO)
  - P switch, prepare FORTRAN output for Line Printer listing, 400 (PIP)
  - Page, 237, 238, 239, 256, 267, 272, 279, 287, 302, 309, 311 (TECO)
  - Pages, 491 (COMMANDS)
  - Pages, 191, 192 (INTRO TO TECO)
    - combining, 197
    - reading into buffer, 197
  - Pages, specifying number to print, 127 (BEGINNER'S BATCH)
  - /PAGE switch (SUBMIT command), 142 (BEGINNER'S BATCH)
  - /PAGE switch, 625, 683 (COMMANDS)
  - /PAGES switch (\$JOB card), 127 (BEGINNER'S BATCH)
  - /PAPER, 625 (COMMANDS)
  - Paper-tape output, specifying amount, 126, 142 (BEGINNER'S BATCH)
  - Paper tape punch queue, 696 (COMMANDS)
  - Parentheses, 242 (TECO)
  - Parentheses usage, 382, 415 (PIP)
  - /PARITY, 521 (COMMANDS)
  - Partial allocation, 468 (COMMANDS)

## INDEX (cont)

- Parts of error messages, 321 (TECO)
- Passing devices to jobs, 633 (COMMANDS)
- Passive search list, 662 (COMMANDS)
- Password, definition, 95 (BEGINNER'S BATCH)
- Password, 448 (COMMANDS)
- Password, 61 (TIMESHARING)
- \$PASSWORD card, 99, 130 (BEGINNER'S BATCH)
- \$PASSWORD card, 713, 731 (COMMANDS)
- /PAUSE, 525, 594 (COMMANDS)
- % (percent sign) command, 295, 314 (TECO)
- Period, 234, 286, 293 (TECO)
- . (period), 258, 293 (TECO)
- Period (.) usage, 192 (INTRO TO TECO)
- Period (.) usage, 377, 382 (PIP)
- Peripheral devices, definition, (BEGINNER'S BATCH)
- Peripheral devices, 383 (PIP)
- Peripheral devices, 75 (TIMESHARING)
- Permanent switch, 457 (COMMANDS)
- /PHYSICAL, DIRECT, 521 (COMMANDS)
- /PHYSICAL, QUEUE, 496, 607, 613, 625, 683, 699 (COMMANDS)
- Physical device name, 448 (COMMANDS)
  - example of, 450
- Physical device name, 383 (PIP)
- Physical device name, 75 (TIMESHARING)
- PIP command errors, 421 (PIP)
- PIP program, 22, 39 (INTRO TO SOFTWARE)
- PIP program, 192 (INTRO TO TECO)
- PJOB command, 601 (COMMANDS)
- PJOB command, 71 (TIMESHARING)
- PLEASE, 39 (INTRO TO SOFTWARE)
- PLEASE command, 602 (COMMANDS)
- /PLOT, 607, 625 (COMMANDS)
- PLOT command, 604 (COMMANDS)
- Plotter output queue, 604 (COMMANDS)
- Plotters, 49 (INTRO TO SOFTWARE)
- Plotter time, specifying amount, 127, 143 (BEGINNER'S BATCH)
- Plotting files, 604 (COMMANDS)
- + (plus), 241 (TECO)
- Pointer (INTRO TO SOFTWARE)
  - Buffer, 20
- Pointer, buffer see Buffer pointer (TECO)
- Pointer, buffer, 198 (INTRO TO TECO)
- Pointer position, 198 (INTRO TO TECO)
- Position, buffer, 241 (TECO)
- Preparing dumps, 505, 529 (COMMANDS)
- Preparing object programs, 465 (COMMANDS)
  - COMPILE, 480
  - CREF, 499
  - DEBUG, 419
  - EXECUTE, 539
  - FUDGE, 562
  - FUDGE2, 563
  - LOAD, 585
  - SAVE, 648
  - SSAVE, 677
- Preparing source files, 465 (COMMANDS)
  - CREATE, 498
  - EDIT, 537
  - MAKE, 592
  - TECO, 693
- Preserving (BEGINNER'S BATCH)
  - control file, 143
  - log file, 143
- Preserving files, 609 (COMMANDS)
- Primary Processor, 10 (INTRO TO SOFTWARE)
- /PRINT, 613, 626 (COMMANDS)
- PRINT command, 610 (COMMANDS)
- PRINT command, 81 (TIMESHARING)
- Printed, output, 154 (BEGINNER'S BATCH)
  - kinds, 153
- Printers (INTRO TO SOFTWARE)
  - Line, 49
- Printing a line, 361 (LINED)

INDEX (cont)

- Printing files, 610 (COMMANDS)
- Printing incremental job statistics, 672
- Printing source files, 584 (COMMANDS)
- Printing system statistics, 686 (COMMANDS)
- Printing version numbers, 672, 704 (COMMANDS)
- Print summary of PIP functions, (PIP)
  - Q switch, 411
- /PRIORITY, 496, 607, 613, 626, 683, 699 (COMMANDS)
- Priority Interrupt System, 14, 46 (INTRO TO SOFTWARE)
- Processing errors, 724, 732 (COMMANDS)
- Processors, 46 (INTRO TO SOFTWARE)
  - Primary, 10
  - Secondary, 10
- Processors, 68 (TIMESHARING)
- Processor switches, 458 (COMMANDS)
- Producing cross-referenced listing, 481 (COMMANDS)
  - global symbols, 569
- Producing REL files, 480 (COMMANDS)
- Program (BEGINNER'S BATCH)
  - definition, 95
  - object, definition, 95
  - source definition, 95
- Program (INTRO TO SOFTWARE)
  - Reentrant, 11
  - Source Library Maintenance, 19
- Program execution, 539 (COMMANDS)
- Programmed editing, 243 (TECO)
- Programmed Operators, 25, 27, 40 (INTRO TO SOFTWARE)
- Programmer number, 251 (TECO)
- Programming, definition, 95 (BEGINNER'S BATCH)
- Programming (INTRO TO SOFTWARE)
  - Symbolic Language, 17
- Programs (COMMANDS)
  - system, 463
- Programs (INTRO TO SOFTWARE)
  - Monitor Support, 23
  - Spooling, 13
- Project number, 251 (TECO)
- Project-programmer area, 248, 249, 255 (TECO)
- Project-programmer number, 125 (BEGINNER'S BATCH)
  - definition, 95
- Project-programmer number, 448, 590 (COMMANDS)
- Project-programmer number, 251, 253 (TECO)
- Project-programmer number, 60 (TIMESHARING)
- [proj,prog], 125 (BEGINNER'S BATCH)
  - definition, 95
- [proj,prog], 251 (TECO)
- Proj, prog number pairs, 389 (PIP)
- Properties of terminals, 668 (COMMANDS)
- /PROTECT, DIRECT, 521 (COMMANDS)
- /PROTECT, QUEUE, 496, 607, 613, 626, 683, 699 (COMMANDS)
- PROTECT command, 616 (COMMANDS)
- Protecting old macros from new features in TECO, 305 (TECO)
- Protection codes, 616 (COMMANDS)
- Protection Codes, 11, 30 (INTRO TO SOFTWARE)
- Protection codes, 390, 391 (PIP)
  - changing of, 408
  - digit numeric values, 391
- Protection codes, 66 (TIMESHARING)
- Providing printable dumps, 530 (COMMANDS)
- Pseudo-TTYS, 23 (INTRO TO SOFTWARE)
- Punch Card, 49 (INTRO TO SOFTWARE)
- /PUNCH, 496, 626 (COMMANDS)
- PUNCH command, 696 (COMMANDS)
- Punching cards, 493 (COMMANDS)
- Pushdown stack, increasing the number of entries, 243 (TECO)

## INDEX (cont)

- Putting commands in the control file, 108  
(BEGINNER'S BATCH)
- Putting comments in TECO macros, 292 (TECO)
- PW (output page) command, 203 (INTRO TO  
TECO)
- PW command, 272, 273, 274, 310, 312 (TECO)
- PWY command, 273 (TECO)
- Q
- Q command, 295 (TECO)
- QMANGR, 710 (COMMANDS)
- QMANGR, 12, 40 (INTRO TO SOFTWARE)
- Q-register, 243, 244, 295, 296 (TECO)
- Q-register commands, 295 (TECO)
- Q-register pushdown list, 297 (TECO)
- Q-register pushdown stack, 243 (TECO)
- Q switch, print summary of PIP functions, 411  
(PIP)
- ? (question mark) command, 304, 322 (TECO)
- Question mark construction, 452 (COMMANDS)
- Question mark (?) symbol, 386, 406 (PIP)
- Queue, 40 (INTRO TO SOFTWARE)
- Input, 13
- QUEUE command, 618 (COMMANDS)
- Queue, definition, 96 (BEGINNER'S BATCH)
- entering a job into, 139
- QUEUE INP: monitor command, 139  
(BEGINNER'S BATCH)
- Queue manager, 710 (COMMANDS)
- Queue Manager, 13 (INTRO TO SOFTWARE)
- Queue names, 618 (COMMANDS)
- Queue operation switches, 140 (BEGINNER'S  
BATCH)
- Queue operation switches, 493 (COMMANDS)
- Queues, 26 (INTRO TO SOFTWARE)
- High-Priority Run, 15
- Queuing files, 581 (COMMANDS)
- QUOLST program, 631 (COMMANDS)
- QUOLST program, 41 (INTRO TO SOFTWARE)
- Quota (INTRO TO SOFTWARE)
- Logged-In, 30
- Logged-Out, 30
- Quotas (INTRO TO SOFTWARE)
- Disk, 30
- Quotas, typing, 631 (COMMANDS)
- QUOTA.SYS, 593 (COMMANDS)
- " (quotation mark) command, 282, 314 (TECO)
- R
- R (control-R) see Control commands (TECO)
- R command, 632 (COMMANDS)
- R (move pointer backwards by character)  
(INTRO TO TECO)
- command, 198
- R command, 258 (TECO)
- R command, 83 (TIMESHARING)
- R switch, Rename Source Files, 407 (PIP)
- REACT, 41 (INTRO TO SOFTWARE)
- Readers, Card, 48 (INTRO TO SOFTWARE)
- Reading a card deck, 108 (BEGINNER'S BATCH)
- Reading control file backward, 734 (COMMANDS)
- Reading control file forward, 735 (COMMANDS)
- Reallocating disk space, 468 (COMMANDS)
- Real-Time, 14 (INTRO TO SOFTWARE)
- Real-Time Devices, 14 (INTRO TO SOFTWARE)
- Real-Time Requirements, 14 (INTRO TO SOFTWARE)
- Rearranging, 309 (TECO)
- REASSIGN command, 633 (COMMANDS)
- REASSIGN command, 79 (TIMESHARING)
- Reassigning devices, 79 (TIMESHARING)
- Reattaching jobs, 472, 635 (COMMANDS)
- REATA program, 635 (COMMANDS)
- Receiving output, 100 (BEGINNER'S BATCH)

## INDEX (cont)

- Recording complaints, 573 (COMMANDS)
- Recovering from errors, 100, 131, 150 (BEGINNER'S BATCH)
- Recovery, Error, 13 (INTRO TO SOFTWARE)
- REENTER, 277, 290 (TECO)
- REENTER command, 637 (COMMANDS)
- Reentrant Program, 11 (INTRO TO SOFTWARE)
- Reentry to TECO, 205 (INTRO TO TECO)
- /REL, 516, 541, 587 (COMMANDS)
- RELEASE, 29 (INTRO TO SOFTWARE)
- Release, 251, 253 (TECO)
- Relocatable binary, 68 (TIMESHARING)
- \$RELOCATABLE card, 731 (COMMANDS)
- Remembered arguments, 454 (COMMANDS)
- Remembered arguments, 65 (TIMESHARING)
- Remote Communications, 15 (INTRO TO SOFTWARE)
- Remote DECtape control, 553 (COMMANDS)
- Remote Station, 15, 51 (INTRO TO SOFTWARE)
- Remote users, 78 (TIMESHARING)
- /REMOVE, DISMOUNT, 526 (COMMANDS)
- /REMOVE, QUEUE, 496, 607, 614, 627, 699 (COMMANDS)
- Removing file structure from search list, 525 (COMMANDS)
- Rename, 249, 253 (TECO)
- Rename backup file, 193 (INTRO TO TECO)
- RENAME command, 638 (COMMANDS)
- RENAME command, 67 (TIMESHARING)
- Rename (R) function, 408 (PIP)
- Renaming files, 638 (COMMANDS)
  - with any protection, 616
  - with standard protection, 609
- Replacing a line, 360 (LINED)
- /REPORT, 614, 627 (COMMANDS)
- Request Handler, Service, 9 (INTRO TO SOFTWARE)
- .QUEUE command, 738 (COMMANDS)
- Required control cards, 105 (BEGINNER'S BATCH)
- RERUN, 19 (INTRO TO SOFTWARE)
- Resident Operating System, 9, 25 (INTRO TO SOFTWARE)
- Resource Allocator, Sharable, 10 (INTRO TO SOFTWARE)
- RESOURCES command, 640 (COMMANDS)
- RESOURCES command, 85 (TIMESHARING)
- /RESTART, 627, 683 (COMMANDS)
- Restarting batch job after failure, 735 (COMMANDS)
- RESTORE program, 641 (COMMANDS)
- RESTORE program, 22 (INTRO TO SOFTWARE)
- Restoring files, 641 (COMMANDS)
- Restoring TECO to no prevailing case conversion, 270, 284 (TECO)
- Restricted devices, 464 (COMMANDS)
- Retrieval Information, 29 (INTRO TO SOFTWARE)
- Return a numeric value, 281, 301 (TECO)
- Returning devices, 512, 525, 560, 579 (COMMANDS)
- Returning devices (TIMESHARING)
  - DEASSIGN, 79
  - DISMOUNT, 79
  - FINISH, 80
  - KJOB, 72
- Returning to the monitor, 73 (TIMESHARING)
- RETURN key, 74 (TIMESHARING)
- Retyping the current line, 321 (TECO)
- .REVIVE command, 739 (COMMANDS)
- Rewind, 252, 255 (TECO)
- REWIND command, 645 (COMMANDS)
- Rewinding MTA, DTA, 645 (COMMANDS)
- Rewinding and unloading MTA, 703 (COMMANDS)
- Ring of Buffers, 28 (INTRO TO SOFTWARE)
- R LINED, 362 (LINED)
- Routines (INTRO TO SOFTWARE)
  - Cyclic, 25

## INDEX ( cont)

- Routines (cont)  
 I/O Service, 10  
 Input/Output, 25, 28  
 R TECO command, 247, 249, 250 (TECO)  
 Rubout, 235, 236, 268, 306, 307, 319 (TECO)  
 RUBOUT key, 446 (COMMANDS)  
 RUBOUT key, 195 (INTRO TO TECO)  
 RUBOUT key, 74 (TIMESHARING)  
 Rubout symbol (  $\text{\textcircled{R}}$  ), 194 (INTRO TO TECO)  
 /RUN, 521 (COMMANDS)  
 RUN command, 646 (COMMANDS)  
 RUN command, 83 (TIMESHARING)  
 Running CDRSTK, 715 (COMMANDS)  
 Running jobs, 99, 103, 137 (BEGINNER'S BATCH)  
 ALGOL, 105  
 BASIC, 109  
 COBOL, 106  
 FORTRAN, 106  
 MACRO, 107  
 Running programs, 463 (COMMANDS)  
 Running programs, 61, 83 (TIMESHARING)  
 Running time, 71 (TIMESHARING)  
 RUNOFF Editor, 21, 41 (INTRO TO SOFTWARE)  
 /RUNOFFSET, 521 (COMMANDS)  
 Run Queues (INTRO TO SOFTWARE)  
 High-Priority, 15
- S
- ↑ S (control-S) see Control commands (TECO)  
 S (search buffer) command, 205, 206 (INTRO TO TECO)  
 S command, 362 (LINED)  
 S command, 279, 281, 293 (TECO)  
 S Switch, insert sequence numbers, 400 (PIP)  
 Sample jobs, Batch, 743 (COMMANDS)  
 SAVE command, 648, 795 (COMMANDS)  
 SAVE command, 83 (TIMESHARING)  
 Saving core images, 648, 677 (COMMANDS)  
 Saving core images, 83 (TIMESHARING)  
 Saving files, 475, 579 (COMMANDS)  
 Saving previous command string, 296 (TECO)  
 SCHED command, 650 (COMMANDS)  
 Schedule bits, 650 (COMMANDS)  
 Scheduler, 25, 26 (INTRO TO SOFTWARE)  
 SCRIPT program, 42 (INTRO TO SOFTWARE)  
 Search, 278, 279, 280 (TECO)  
 Search command modifiers, 281 (TECO)  
 Search commands, 205 (INTRO TO TECO)  
 Searching (TECO)  
 and deleting strings, 279, 280  
 and replacing strings, 279, 280  
 in "either-case-mode", 285,  
 in "exact-case-mode", 285, 286  
 partly in "exact-case" and partly in  
 "either-case" mode, 285  
 Searching back in the control file, 123, 130,  
 146 (BEGINNER'S BATCH)  
 Searching forward in the control file, 123, 130,  
 147 (BEGINNER'S BATCH)  
 Search list, 662 (COMMANDS)  
 Search List, Job, 30 (INTRO TO SOFTWARE)  
 Search string, 278, 279, 281 (TECO)  
 Secondary Memory, 11 (INTRO TO SOFTWARE)  
 Secondary Processor, 10 (INTRO TO SOFTWARE)  
 Segments, 11, 42 (INTRO TO SOFTWARE)  
 ; (semicolon) command, 289, 291 (TECO)  
 SEND command, 651 (COMMANDS)  
 SEND command, 80 (TIMESHARING)  
 Sending messages, 466 (COMMANDS)  
 GRIPE, 573  
 PLEASE, 602  
 SEND, 651  
 Separator character, 286 (TECO)  
 \$SEQUENCE card, 105, 111 (BEGINNER'S BATCH)  
 \$SEQUENCE card, 732 (COMMANDS)

## INDEX (cont)

- Sequence number, delete (N switch), 399 (PIP)
- Sequence number, ignore card (E switch), 399 (PIP)
- Sequence number and increment by one, O switch, insert, 400 (PIP)
- Sequence numbers, S-switch, insert, 400 (PIP)
- /SEQUENCE switch (\$COBOL card), 114 (BEGINNER'S BATCH)
- /SEQUENCE switch, 497, 607, 614, 627, 683, 700 (COMMANDS)
- Service Request Handler, 9 (INTRO TO SOFTWARE)
- Service Routines (INTRO TO SOFTWARE)
  - I/O, 10
- SET BLOCKSIZE command, 653 (COMMANDS)
- SET CDR command, 654 (COMMANDS)
- SET CPU command, 655 (COMMANDS)
- Set data mode switches, B, H and I, 405 (PIP)
- SET DENSITY command, 657 (COMMANDS)
- SET DSKPRI command, 658 (COMMANDS)
- SET HPQ command, 659 (COMMANDS)
- SET SPOOL command, 660 (COMMANDS)
- SETSRC program, 662 (COMMANDS)
- SETSRC program, 42 (INTRO TO SOFTWARE)
- SET TIME command, 666 (COMMANDS)
- Setting (TECO)
  - case flagging mode, 262
  - EO value, 305
  - TECO to a prevailing case conversion mode, 269, 283
  - version number of TECO, 305
- Setting up a card deck, 104 (BEGINNER'S BATCH)
  - ALGOL, 105
  - BASIC, 99
  - COBOL, 106
  - FORTRAN, 106
  - MACRO, 107
- Setting up a job, 99 (BEGINNER'S BATCH)
- SET TTY command, 668 (COMMANDS)
- SET WATCH command, 672 (COMMANDS)
- SFD, 29, 42 (INTRO TO SOFTWARE)
- SFD (full directory path) identifiers, 388 (PIP)
- Sharable Data Areas, 15 (INTRO TO SOFTWARE)
- Sharable Resource Allocator, 10 (INTRO TO SOFTWARE)
- Sharing Devices, 11 (INTRO TO SOFTWARE)
- Sharing Files, 11 (INTRO TO SOFTWARE)
- .SILENCE command, 739 (COMMANDS)
- /SINCE, 497, 607, 614, 627, 700 (COMMANDS)
- /SINGLE, 594 (COMMANDS)
- Single Mode, 15 (INTRO TO SOFTWARE)
- Skip, 273, 274 (TECO)
- SKIP command, 675 (COMMANDS)
- Skip one file, 252 (TECO)
- Skip one record, 252 (TECO)
- Skip to end-of-tape, 252 (TECO)
- / (slash), 242 (TECO)
- / (Slash) command, 323 (TECO)
- Slice, Time, 27 (INTRO TO SOFTWARE)
- /SLOW, 521 (COMMANDS)
- /SNOBOL, 482, 516, 541, 587 (COMMANDS)
- Software (INTRO TO SOFTWARE)
  - Non-Resident, 10, 17
- Software, definition, 96 (BEGINNER'S BATCH)
- Software I/O Channels, 28 (INTRO TO SOFTWARE)
- /SORT, 522 (COMMANDS)
- SORT program, 19 (INTRO TO SOFTWARE)
- SOUP Editor, 21 (INTRO TO SOFTWARE)
- Source (BEGINNER'S BATCH)
  - deck, definition, 96
  - language, definition, 96
  - program, definition, 96
- Source file preparation, 465 (COMMANDS)
  - CREATE, 498
  - EDIT, 537
  - MAKE, 592
  - TECO, 693
- Source files (COMMANDS)
  - listing, 584
  - typing, 702

- Source Library Maintenance Program, 19  
(INTRO TO SOFTWARE)
- Space, 236, 241, 306 (TECO)
- Space symbol ( $\Delta$ ), 194 (INTRO TO TECO)
- /SPACING, 614, 628 (COMMANDS)
- Spacing magnetic tape (COMMANDS)
- backwards, 474
  - forward, 675
- Special characters, 446 (COMMANDS)
- Batch, 711
- Special characters, 235, 266, 292 (TECO)
- Special functions, 415 (PIP)
- Special "lower case" characters, 270, 285 (TECO)
- Special numeric values, 300 (TECO)
- Specification File, 29, 35 (INTRO TO SOFTWARE)
- Specifying amount (BEGINNER'S BATCH)
- cards to be punched, 126, 142
  - core, 126, 142
  - CPU time, 127, 142
  - pages to be printed, 127, 142
  - paper tape to be punched, 126, 142
  - plotter time, 127, 143
- Specifying blocksize, 653 (COMMANDS)
- Specifying character to be recognized as a fatal error, 146 (BEGINNER'S BATCH)
- Specifying conventional COBOL format, 114 (BEGINNER'S BATCH)
- Specifying disposal of a file, 143 (BEGINNER'S BATCH)
- Specifying error recovery, 131, 150 (BEGINNER'S BATCH)
- Specifying number for a job, 131 (BEGINNER'S BATCH)
- Specifying parameters for a file, 143 (BEGINNER'S BATCH)
- Splitting, 310 (TECO)
- Spoolers (INTRO TO SOFTWARE)
- Output, 11, 13
- Spooling, 618, 660, 712 (COMMANDS)
- card punch files, 493
  - input queue, 680
- Spooling (cont)
- line printer files, 584, 610
  - paper tape punch files, 696
  - plotter files, 604
- Spooling, 11, 42 (INTRO TO SOFTWARE)
- Spooling, 76, 81 (TIMESHARING)
- Spooling Programs, 13 (INTRO TO SOFTWARE)
- Square brackets, 382 (PIP)
- [ ] (square brackets), 297 (TECO)
- SSAVE command, 677, 795 (COMMANDS)
- STACKER, 709 (COMMANDS)
- STACKER, 12 (INTRO TO SOFTWARE)
- Stack mode, 445 (COMMANDS)
- Standard COBOL format, 114 (BEGINNER'S BATCH)
- Standard for DECsystem-10 line sequence numbers, 365 (LINED)
- Standard Format, 19 (INTRO TO SOFTWARE)
- Standard optional functions, 393 (PIP)
- Standard PIP switches, 393 (PIP)
- Standard processor, 457 (COMMANDS)
- Standard processor, 68 (TIMESHARING)
- Standard protection, 616 (COMMANDS)
- renaming files to, 609
- /START, 497, 608, 614, 628, 684, 700 (COMMANDS)
- START command, 679 (COMMANDS)
- START command, 84 (TIMESHARING)
- Starting core image file and loading, 632, 646 (COMMANDS)
- Starting the program (COMMANDS)
- at alternate entry point, 637
  - at beginning, 679
  - at DDT, 510, 513
- Starting the program, 84 (TIMESHARING)
- Station (INTRO TO SOFTWARE)
- Logical, 16
  - Remote, 15, 51
- Station of a device, 706 (COMMANDS)

## INDEX (cont)

- Steps to enter a job to Batch, 100  
(BEGINNER'S BATCH)
- Stop Teletype output, 200 (INTRO TO TECO)
- Stopping the job, 574 (COMMANDS)
- Storage Allocation File, 11, 30 (INTRO TO SOFTWARE)
- Strings, command, 235, 240, 244 (TECO)
- /STRS, 497, 608, 614, 628, 700 (COMMANDS)
- Structures, 29 (INTRO TO SOFTWARE)
- Subfile Directory, 29, 42 (INTRO TO SOFTWARE)
- Subfile Directory (SFD), 388 (PIP)
- Subjobs, 597 (COMMANDS)
- Subjobs, 23 (INTRO TO SOFTWARE)
- SUBMIT monitor command, 139 (BEGINNER'S BATCH)
  - switches, 140
- SUBMIT monitor command, 680 (COMMANDS)
- Submitting a job, 99, 137, 139 (BEGINNER'S BATCH)
  - examples, 144
- Submitting Batch jobs, 712 (COMMANDS)
- Subroutine, 244 (TECO)
- Subtraction, 242 (TECO)
- /SUMMARY, 522 (COMMANDS)
- Superseding, 30 (INTRO TO SOFTWARE)
- /SUPLSN (suppress line-sequence numbers)
  - switch, 254 (TECO)
- Support Programs (INTRO TO SOFTWARE)
  - Monitor, 23
- Suppressing listings (BEGINNER'S BATCH)
  - ALGOL, 112
  - FORTRAN, 124
  - MACRO, 129
- Suppressing terminal output, 75 (TIMESHARING)
- /SUPPRESS:OFF switch (BEGINNER'S BATCH)
  - \$ALGOL card, 112
  - \$COBOL card, 114
  - \$DATA card, 116
  - \$DECK card, 120
  - \$FORTRAN card, 124
  - \$MACRO card, 128
- Swapper, 25, 27 (INTRO TO SOFTWARE)
- Swapping, 11 (INTRO TO SOFTWARE)
- Switch combinations, 413 (PIP)
- Switches, 393 (PIP)
  - magnetic tape, 415, 416
  - for setting density and parity parameters, 415
- Switches for line-sequence numbers, 254 (TECO)
- Switches in SUBMIT command, 140 (BEGINNER'S BATCH)
- Switch summary, 412 (PIP)
- Symbolic Language Programming, 17 (INTRO TO SOFTWARE)
- Symbols, 236 (TECO)
- Symbols used in document, 193 (INTRO TO TECO)
- Syntax of command string, 194 (INTRO TO TECO)
- SYSDPY, 42 (INTRO TO SOFTWARE)
- SYSTAT command, 686 (COMMANDS)
- SYSTAT command, 85 (TIMESHARING)
- SYSTAT program, 43 (INTRO TO SOFTWARE)
- System (INTRO TO SOFTWARE)
  - Data Communications, 50
  - Disk, 48
  - Drum, 47
  - File, 11, 29
  - Magnetic Tape, 48
  - Multiprogramming, 11, 26
  - Operating, 9, 25
  - Priority Interrupt, 14, 46
- System commands, 463 (COMMANDS)
- System information, 467 (COMMANDS)
  - DAYTIME, 504
  - RESOURCES, 640
  - SCHED, 650
  - SYSTAT, 686
  - VERSION, 704
  - WHERE, 706
- System messages, 747 (COMMANDS)
- System program command (BEGINNER'S BATCH)
  - card format, 104
  - line format, 139
- System programs, 463 (COMMANDS)

## INDEX (cont)

- System queues, 618 (COMMANDS)
  - System status, 686 (COMMANDS)
  - System status, 85 (TIMESHARING)
- T
- ↑ T (control-T) see Control commands (TECO)
  - T (type) command, 200 (INTRO TO TECO)
  - T command, 260, 293 (TECO)
  - T switch, delete trailing spaces, 402 (PIP)
  - TAB, 236, 237 (TECO)
  - TAB codes, 399 (PIP)
  - Tab command, 266 (TECO)
  - Tab symbol ( → ), 194 (INTRO TO TECO)
  - Tab to space conversion, W-switch, 402 (PIP)
  - Tag, 292 (TECO)
  - /TAPE, 628, 700 (COMMANDS)
  - TECO command, 693 (COMMANDS)
  - TECO (initialize file for editing) command, 192 (INTRO TO TECO)
  - TECO command, 248, 249, 250, 253, 256, 308 (TECO)
  - TECO command, 65 (TIMESHARING)
  - TECO filnam.ext command, 249 (TECO)
  - TECO program, 592, 693 (COMMANDS)
  - TECO program, 20, 43 (INTRO TO SOFTWARE)
  - TECO program, 62, 65 (TIMESHARING)
  - Teletype, 231, 233 (TECO)
  - Teletype output, stopping, 200 (INTRO TO TECO)
  - Teletypes and Terminals, 49 (INTRO TO SOFTWARE)
  - Temporary case conversion, 268, 283 (TECO)
  - Temporary files, 454, 791 (COMMANDS)
  - Temporary file, 253 (TECO)
  - Temporary file, 65 (TIMESHARING)
  - Temporary switch, 457 (COMMANDS)
  - TENDMP program, 43 (INTRO TO SOFTWARE)
  - Terminal, 59, 73 (TIMESHARING)
  - Terminal I/O, 29 (INTRO TO SOFTWARE)
  - Terminal properties, 668 (COMMANDS)
  - Terminating commands, 74, 75 (TIMESHARING)
  - Terminating input into a data file, 724 (COMMANDS)
  - Terminating I/O, 479 (COMMANDS)
  - Terminating jobs, 560 (COMMANDS)
    - KJOB, 579
  - Termination of command string, 194 (INTRO TO TECO)
  - Terminator, 379 (PIP)
  - Terminator, argument, 251 (TECO)
  - Text argument, 240, 266, 278, 281, 292 (TECO)
  - Text block movement, 243 (TECO)
  - Text modification, 192 (INTRO TO TECO)
  - Text string, 534 (COMMANDS)
  - Text typeout, 200 (INTRO TO TECO)
  - Textual arguments, 196 (INTRO TO TECO)
  - /TIME, 628 (COMMANDS)
  - TIME command, 694 (COMMANDS)
  - TIME command, 71 (TIMESHARING)
  - Time of day, 301, 303 (TECO)
  - Timesharing, 10 (INTRO TO SOFTWARE)
  - Time slice, 27 (INTRO TO SOFTWARE)
  - /TITLES, 522 (COMMANDS)
  - TMPCOR (device TMP) error messages, 424 (PIP)
  - Total running time, 694 (COMMANDS)
  - /TPLOT switch (BEGINNER'S BATCH)
    - \$JOB card, 127
    - SUBMIT command, 143
  - /TPLOT switch, 629, 684 (COMMANDS)
  - TPUNCH command, 696 (COMMANDS)
  - Trace mode, 304 (TECO)
  - Trailing spaces, 399 (PIP)
  - Transaction file, 563 (COMMANDS)
  - Transfer function, 394 (PIP)
  - Transferring files, 486, 557 (COMMANDS)

## INDEX (cont)

- Transfers (INTRO TO SOFTWARE)
    - Data, 27
  - Transfer without X-switch (combine files), 398 (PIP)
  - Translating (TECO)
    - group of characters to lower case, 269, 283
    - group of characters to upper case, 269, 283
    - single characters to lower case, 269, 283
    - single characters to upper case, 269, 283
  - Translating programs, 67 (TIMESHARING)
    - COMPILE, 67
  - TTY command, 668 (COMMANDS)
  - TTY used as an I/O device, 234 (TECO)
  - Two-Pass Assembler, 17 (INTRO TO SOFTWARE)
  - Type-ahead technique, 446 (COMMANDS)
  - TYPE command, 702 (COMMANDS)
  - TYPE command, 66 (TIMESHARING)
  - Type-in, 306 (TECO)
  - Type-out, 260, 300 (TECO)
  - Type-out commands, 260, 300 (TECO)
  - Type-out mode, 262 (TECO)
  - Typing amount of core assigned, 491 (COMMANDS)
  - Typing disk usage, 527 (COMMANDS)
  - Typing errors, 319 (TECO)
  - Typing line over, 74 (TIMESHARING)
  - Typing numeric value (TECO)
    - in decimal, 300
    - in octal, 300
  - Typing quotas, 631 (COMMANDS)
  - Typing running time, 694 (COMMANDS)
  - Typing source files, 702 (COMMANDS)
  - Typing source files, 66 (TIMESHARING)
  - Typing system schedule, 650 (COMMANDS)
  - Typographical error correction, 195 (INTRO TO TECO)
- U
- ↑ U (control-U) see Control commands (TECO)
  - ↑↑ (control up-arrow), 270, 285, 302 (TECO)
  - U command, 293, 295 (TECO)
  - U switch, copy DECtape blocks 0, 1 and 2, 398 (PIP)
  - UFD, 29, 43 (INTRO TO SOFTWARE)
  - UFD and SFD File protection codes, 392 (PIP)
  - UFD-only identifiers, 388 (PIP)
  - UMOUNT program 525, 554, 593 (COMMANDS)
  - UMOUNT program, 43 (INTRO TO SOFTWARE)
  - Unbuffered Data Modes, 27, 28 (INTRO TO SOFTWARE)
  - Unconditional branch, 292 (TECO)
  - Underline, 236 (TECO)
  - Underscoring, 377 (PIP)
  - /UNIQUE, 629, 684 (COMMANDS)
  - /UNITS, 522 (COMMANDS)
  - Unload, 252 (TECO)
  - UNLOAD command, 703 (COMMANDS)
  - Unloading magnetic tape, 703 (COMMANDS)
  - /UNPRESERVED, 497, 608, 614, 629, 700 (COMMANDS)
  - Unrestricted devices, 464, 471 (COMMANDS)
  - Up-arrow, 236 (TECO)
  - Up-arrow, Batch, 711 (COMMANDS)
  - Up-arrow-O, 242 (TECO)
  - Up-arrow (↑) symbol usage, 377 (PIP)
  - Update, 253 (TECO)
  - Update mode, FILCOM, 545 (COMMANDS)
  - Updating, 30 (INTRO TO SOFTWARE)
  - Updating REL files, 563 (COMMANDS)
  - Upper case flagging, 262 (TECO)
  - Use Bit, 28 (INTRO TO SOFTWARE)
  - User directories, 65 (TIMESHARING)
  - User File Directory, 29, 43 (INTRO TO SOFTWARE)
  - User File Directory (UFD), 380 (PIP)
  - User mode, 444 (COMMANDS)
  - USER SET, 475 (COMMANDS)
  - Uses of TECO, 262 (TECO)

## Utilities (INTRO TO SOFTWARE)

CREF, 21  
 DDT, 21  
 File Backup, 22  
 FILEX, 22  
 LOADER, 22  
 PIP, 22

Utilization, Core, 11 (INTRO TO SOFTWARE)

UUO Handler, 25, 27 (INTRO TO SOFTWARE)

UUOS, 25, 27 (INTRO TO SOFTWARE)

## V

↑ V (control-V) see Control commands (TECO)

V command, 293 (TECO)

V switch, match angle brackets, 402 (PIP)

/Vx, KJOB, 581 (COMMANDS)

Values, obtaining (TECO)

automatic timeout flag, 301  
 case flag, 301  
 end-of-file flag, 301  
 error message flag, 301, 324  
 search mode flag, 286  
 timeout mode switch, 261  
 version number flag, 301

VERSION command, 704 (COMMANDS)

Version number flag, obtaining the value, 301 (TECO)

Vertical tab, 236, 237 (TECO)

Vestigial job data area, 797 (COMMANDS)

/VID, 594 (COMMANDS)

## W

↑ W (control-W) see Control commands (TECO)

W command, 293 (TECO)

W switch, convert tabs to spaces, 402 (PIP)

/WENABL, 594 (COMMANDS)

WHERE command, 706 (COMMANDS)

/WIDTH switch (BEGINNER'S BATCH)

\$ALGOL card, 112  
 \$COBOL card, 113  
 \$DATA card, 116  
 \$DECK card, 120  
 \$FORTRAN card, 124  
 \$MACRO card, 128

## INDEX (cont)

/WIDTH switch, 522 (COMMANDS)

Wildcard characters, 386 (PIP)

Wildcard construction, 452 (COMMANDS)

/WLOCK, 595 (COMMANDS)

/WORDS, 522 (COMMANDS)

Writing and loading a Bootstrap loader, 488 (COMMANDS)

Writing complaints, 573 (COMMANDS)

Writing conventions, 376 (PIP)

Writing core-image files, 505, 529 (COMMANDS)

Writing end of file, 538 (COMMANDS)

## X

↑ X (control-X) see Control commands (TECO)

X command, 295, 296 (TECO)

X switch, copy files without combining, 394, 404 (PIP)

## Y

Y (yank) command, 197 (INTRO TO TECO)

Y command, 249, 253, 256, 301 (TECO)

Y switch, DECtape to Paper Tape (PIP)

copy, 403  
 errors, 422

Yank command, 256 (TECO)

## Z

↑ Z (control-Z) see Control commands (TECO)

Z, 258, 267, 316 (TECO)

/Z, KJOB, 581 (COMMANDS)

Z switch, 411 (PIP)

ZJ (move pointer to end) command, 198 (INTRO TO TECO)

ZJ command, 259 (TECO)

ZERO command, 707 (COMMANDS)

Zero-compression, 795 (COMMANDS)

Zeroing DECtapes, 488 (COMMANDS)

**READER'S COMMENTS**

DECsystem-10  
USERS HANDBOOK  
DEC-10-NGZB-D

Digital Equipment Corporation maintains a continuous effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback: your critical evaluation of this document. Please give specific page and line references when appropriate.

**ERRORS NOTED IN THIS PUBLICATION:**

---

---

---

---

---

**SUGGESTIONS FOR IMPROVEMENT OF THIS PUBLICATION:**

---

---

---

---

---

---

---

---

DEC also strives to keep its customers informed about current DEC software and publications. Thus, the following periodically distributed publications are available upon request. Please check the publication(s) desired.

- PDP-10 User's Bookshelf, a bibliography of current programming documents.
- Program Library Price List, a list of available software documents and programs.

Name \_\_\_\_\_ Date \_\_\_\_\_

Organization \_\_\_\_\_

Please describe your position \_\_\_\_\_

Street \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip Code \_\_\_\_\_

-----  
**Fold Here** -----

-----  
**Do Not Tear - Fold Here and Staple** -----

**FIRST CLASS  
PERMIT NO. 33  
MAYNARD, MASS.**

**BUSINESS REPLY MAIL  
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES**

**Postage will be paid by:**

**Digital Equipment Corporation  
Software Information Services  
146 Main Street  
Maynard, Massachusetts 01754**



# DIGITAL EQUIPMENT CORPORATION **digital** WORLD-WIDE SALES AND SERVICE

## MAIN OFFICE AND PLANT

146 Main Street, Maynard, Massachusetts, U.S.A. 01754 • Telephone: From Metropolitan Boston: 646-8600 • Elsewhere: (617)-897-5111  
 TWX: 710-347-0212 Cable: DIGITAL MAYN Telex: 94-8457

### UNITED STATES

#### NORTHEAST

##### REGIONAL OFFICE

275 Wyman Street, Waltham, Massachusetts 02154  
 Telephone: (617)-890-0320/0330 TWX: 710-324-6919

##### WALTHAM

15 Lunda Street, Waltham, Massachusetts 02154  
 Telephone: (617)-891-1030 TWX: 710-324-6919

##### CAMBRIDGE/BOSTON

899 Main Street, Cambridge, Massachusetts 02139  
 Telephone: (617)-491-6130 TWX: 710-320-1167

##### ROCHESTER

130 Allens Creek Road, Rochester, New York 14618  
 Telephone: (716)-461-1700 TWX: 710-253-3078

##### CONNECTICUT

240 Pomeroy Ave., Meriden, Conn. 06450  
 Telephone: (203)-237-8441/7466 TWX: 710-461-0054

#### MID-ATLANTIC — SOUTHEAST

##### REGIONAL OFFICE:

U.S. Route 1, Princeton, New Jersey 08540  
 Telephone: (609)-452-2940 TWX: 510-685-2338

##### NEW YORK

95 Cedar Lane, Englewood, New Jersey 07631  
 Telephone: (201)-871-4984, (212)-584-6955, (212)-736-0447  
 TWX: 710-991-9721

##### NEW JERSEY

1239 Route 46, Parsippany, New Jersey 07054  
 Telephone: (201)-335-3300 TWX: 910-987-8319

##### PRINCETON

U.S. Route 1  
 Princeton, New Jersey 08540  
 Telephone: (609) 452-2940 TWX: 510-685-2338

##### LONG ISLAND

1 Huntington Quadrangle  
 Suite 1507 Huntington Station, New York 11746  
 Telephone: (516)-694-4131, (212)-985-8095

##### PHILADELPHIA

Station Square Three, Paoli, Pennsylvania 19301  
 Telephone: (215)-947-4900/4410 Telex: 510-686-8395

##### WASHINGTON

Executive Building  
 6811 Kenilworth Ave., Riverdale, Maryland 20840  
 Telephone: (301)-779-1800/752-8797 TWX: 710-826-9862

##### DURHAM/CHAPEL HILL

2704 Chapel Hill Boulevard  
 Durham, North Carolina 27707  
 Telephone: (919)-469-3347 TWX: 510-927-0912

##### ORLANDO

Suite 130, 7001 Lake Ellenor Drive, Orlando, Florida 32809  
 Telephone: (305)-851-4450 TWX: 810-850-0180

##### ATLANTA

2815 Clearview Place, Suite 100,  
 Atlanta, Georgia 30340  
 Telephone: (404)-451-3734/3735/3736 TWX: 810-757-4223

##### EUROPEAN HEADQUARTERS

Digital Equipment Corporation International Europe  
 81 Route de l'Aire  
 1211 Geneva 26, Switzerland  
 Telephone: 42 79 50 Telex: 22 683

##### FRANCE

Equipment Digital S.A.R.L.  
 PARIS  
 327 Rue de Charenton, 75 Paris 12 8<sup>me</sup>, France  
 Telephone: 344-78-07 Telex: 21339

##### GRENOBLE

10 rue Auguste Ravier, F-38 Grenoble, France  
 Telephone: (76) 87 87 32 Telex: 32 882 F (Code 212)

##### GERMANY

Digital Equipment GmbH  
 MUNICH  
 8 Muenchen 13, Wallensteinplatz 2  
 Telephone: 0811-35031 Telex: 524-226

##### COLOGNE

5 Koeln, Bismarckstrasse 7,  
 Telephone: 0221-522181 Telex: 888-2269  
 Telegram: Flip Chip Koeln

##### FRANKFURT

6078 Neu-Isenburg 2  
 Am Forsthaus Gravenbruch 5-7  
 Telephone: 06102-5528 Telex: 41-76-82

##### HANNOVER

3 Hannover, Podbielskiestrasse 102  
 Telephone: 0511-69-70-95 Telex: 922-952

##### AUSTRIA

Digital Equipment Corporation Ges.m.b.H  
 VIENNA  
 Meran/Hoferstrasse 136, 1150 Vienna 15, Austria  
 Telephone: 85 51 86

##### UNITED KINGDOM

Digital Equipment Co., Ltd.  
 U.K. HEADQUARTERS  
 Arkwright Road, Reading, Berks  
 Telephone: 0734-583955 Telex: 84327

##### READING

The Evening Post Building, Tessa Road  
 Reading, Berks.

##### BIRMINGHAM

29/31, Birmingham Road, Sutton Coldfield, Warwickshire  
 Telephone: (0044) 21-355 5501 Telex: 337 960

##### MANCHESTER

13 Upper Precinct, Walkden, Manchester M28 5AZ  
 Telephone: 061-790-8411 Telex: 668666

##### LONDON

Bilton House, Uxbridge Road, Ealing, London W.5.  
 Telephone: 01-579-2334 Telex: 22371

##### EDINBURGH

Shiel House, Craighill, Livingston,  
 West Lothian, Scotland  
 Telephone: 32705 / Telex: 727113

##### NETHERLANDS

THE HAGUE  
 Digital Equipment N.V.  
 Sir Winatou Churchillaan 370  
 Rijswijk/The Hague, Netherlands  
 Telephone: 070-995-160 Telex: 32533

##### BELGIUM

BRUSSELS  
 Digital Equipment N.V./S.A.  
 108 Rue D'Arlon  
 1040 Brussels, Belgium  
 Telephone: 02-139256 Telex: 25297

#### MID-ATLANTIC — SOUTHEAST (cont.)

##### KNOXVILLE

6311 Kingston Pike, Suite 21E  
 Knoxville, Tennessee 37919  
 Telephone: (615)-588-6571 TWX: 810-583-0123

##### CENTRAL

REGIONAL OFFICE:  
 1850 Frontage Road, Northbrook, Illinois 60062  
 Telephone: (312)-498-2500 TWX: 910-686-0655

##### PITTSBURGH

400 Penn Center Boulevard  
 Pittsburgh, Pennsylvania 15235  
 Telephone: (412)-243-9404 TWX: 710-797-3657

##### CHICAGO

1850 Frontage Road, Northbrook, Illinois 60062  
 Telephone: (312)-498-2500 TWX: 910-686-0655

##### ANN ARBOR

230 Huron View Boulevard, Ann Arbor, Michigan 48103  
 Telephone: (313)-761-1150 TWX: 810-223-8053

##### INDIANAPOLIS

21 Beachway Drive — Suite G  
 Indianapolis, Indiana 46224  
 Telephone: (317)-243-8341 TWX: 810-341-3436

##### MINNEAPOLIS

Suite 111, 8030 Cedar Avenue South,  
 Minneapolis, Minnesota 55420  
 Telephone: (612)-854-6562-3-4-5 TWX: 910-576-2818

##### CLEVELAND

Park Hill Bldg., 35104 Euclid Ave.  
 Willoughby, Ohio 44094  
 Telephone: (216)-946-8484 TWX: 810-427-2608

##### ST. LOUIS

Suite 110, 115 Progress Pky., Maryland Heights,  
 Missouri 63043  
 Telephone: (314)-878-4310 TWX: 910-764-0831

##### DAYTON

3101 Kettering Blvd., Dayton, Ohio 45439  
 Telephone: (513)-299-7377 TWX: 810-459-1676

##### MILWAUKEE

8531 W. Capitol Drive, Milwaukee, Wisconsin 53222  
 Telephone: (414)-463-9110 TWX: 910-262-1199

##### DALLAS

8555 North Stemmons Freeway  
 Dallas, Texas 75247  
 Telephone: (214)-638-4880 TWX: 910-861-4000

##### HOUSTON

3417 Millem Street, Suite A, Houston, Texas 77002  
 Telephone: (713)-524-2961 TWX: 910-881-1651

### INTERNATIONAL

#### SWEDEN

Digital Equipment Aktiebolag  
 STOCKHOLM  
 Vretensvagen 2, S-171 54 Solna, Sweden  
 Telephone: 86 13 90 Telex: 170 50  
 Cable: Digital Stockholm

#### NORWAY

Digital Equipment  
 OSLO  
 c/o Firma Service  
 Weldenmarthnesgate 84-B-86  
 Oslo 1, Norway  
 Telephone: 37 19 85, 37 02 30 Telex: 166 43

#### DENMARK

Digital Equipment Corporation  
 COPENHAGEN  
 Vesterbrogade 140, 1620 Copenhagen V

#### SWITZERLAND

Digital Equipment Corporation S.A.  
 GENEVA  
 81 Route de l'Aire  
 1211 Geneva 26, Switzerland  
 Telephone: 42 79 50 Telex: 22 683

#### ZURICH

Schweizerstrasse 21  
 CH-8005 Zurich, Switzerland  
 Telephone: 01/60 35 66 Telex: 56059

#### ITALY

Digital Equipment S.p.A.  
 MILAN  
 Corso Garibaldi 49, 20121 Milano, Italy  
 Telephone: 872 748 694 394 Telex: 33615

#### SPAIN

MADRID  
 Atajo Ingenieros S.A., Enrique Larreta 12, Madrid 16  
 Telephone: 215 35 43 / Telex: 27249

#### BARCELONA

Atajo Ingenieros S.A., Ganduxer 76, Barcelona 6  
 Telephone: 221 44 66  
 Digital Equipment Corporation Ltd.

#### AUSTRALIA

Digital Equipment Australia Pty. Ltd.  
 SYDNEY  
 P.O. Box 491, Crow's Nest  
 N.S.W. Australia 3065  
 Telephone: 439-2598 Telex: AA20740  
 Cable: Digital, Sydney

#### MELBOURNE

60 Park Street, South Melbourne, Victoria, 3205  
 Telephone: 696-142 Telex: AA40616

#### PERTH

643 Murray Street  
 West Perth, Western Australia 6005  
 Telephone: 214-993 Telex: AA92140

#### BRISBANE

139 Merivale Street, South Brisbane  
 Queensland, Australia 4101  
 Telephone: 444-047 Telex: AA40616

#### ADELAIDE

6 Montrose Avenue  
 Norwood, South Australia 5067  
 Telephone: 631-339 Telex: AA82825

#### CENTRAL (cont.)

##### NEW ORLEANS

310 Ridgelake Drive, Suite 108  
 Metairie, Louisiana 70002  
 Telephone: 504-837-0257

##### WEST

REGIONAL OFFICE  
 310 Soquel Way, Sunnyvale, California 94086  
 Telephone: (408)-735-8200

##### ANAHEIM

801 E. Ball Road, Anaheim, California 92805  
 Telephone: (714)-776-8932/8730 TWX: 910-591-1189

##### WEST LOS ANGELES

1510 Cotner Avenue, Los Angeles, California 90025  
 Telephone: (213)-479-9791/4918 TWX: 910-342-6999

##### SAN DIEGO

3444 Hancock Street  
 San Diego, California 92110  
 Telephone: (714)-298-0591, 0593 TWX: 910-335-1230

##### SAN FRANCISCO

1400 Terra Bella  
 Mountain View, California 94040  
 Telephone: (415)-964-6200 TWX: 910-373-1266

##### PALO ALTO

560 San Antonio Rd., Palo Alto, California 94306  
 Telephone: (415)-969-6200 TWX: 910-373-1266

##### OAKLAND

7850 Edgewater Drive  
 Oakland, California 94621  
 Telephone: (415)-635-5453/7830 TWX: 910-366-7238

##### ALBUQUERQUE

6303 Indian School Road, N.E.  
 Albuquerque, N.M. 87110  
 Telephone: (505)-296-5411/5428 TWX: 910-989-0614

##### DENVER

2305 South Colorado Blvd., Suite #5  
 Denver, Colorado 80222  
 Telephone: (303)-757-3332/758-1656/758-1659  
 TWX: 910-931-2690

##### SEATTLE

1521 130th N.E., Bellevue, Washington 98005  
 Telephone: (206)-454-4058/455-5404 TWX: 910-443-2306

##### SALT LAKE CITY

431 South 3rd East, Salt Lake City, Utah 94111  
 Telephone: (801)-328-9838 TWX: 910-925-5834

##### PHOENIX

4358 East Broadway Road  
 Phoenix, Arizona 85040  
 Telephone: (602)-268-3488 TWX: 910-950-4691

##### PORTLAND

Suite 168  
 5319 S.W. Canyon Court, Portland, Ore. 97221  
 Telephone: (503) 297-3761/3765

#### NEW ZEALAND

Digital Equipment Corporation Ltd.  
 AUCKLAND  
 Hilton House, 430 Queen Street, Box 2471 A,  
 Auckland, New Zealand  
 Telephone: 75-533

#### CANADA

Digital Equipment of Canada, Ltd.  
 CANADIAN HEADQUARTERS  
 150 Rosamond Street, Carleton Place, Ontario  
 Telephone: (613)-257-2615 TWX: 610-561-1651

#### OTTAWA

120 Holland Street, Ottawa 3, Ontario K1Y 0X7  
 Telephone: (613)-725-2193 TWX: 610-562-8907

#### TORONTO

230 Lakeshore Road East, Port Credit, Ontario  
 Telephone: (416)-274-1241 TWX: 610-492-4306

#### MONTREAL

9675 Cote de Liesse Road  
 Dorval, Quebec, Canada 760  
 Telephone: 514-636-9393 TWX: 610-422-4124

#### EDMONTON

5331 - 103 Street  
 Edmonton, Alberta, Canada  
 Telephone: (403)-434-9333 TWX: 610-831-2248

#### VANCOUVER

Digital Equipment of Canada, Ltd.  
 2210 West 12th Avenue  
 Vancouver 9, British Columbia, Canada  
 Telephone: (604)-736-5616 TWX: 610-929-2006

#### ARGENTINA

BUENOS AIRES  
 Cosain S.A.  
 Virrey del Pino 4071, Buenos Aires  
 Telephone: 52-3185 Telex: 012-2284

#### VENEZUELA

CARACAS  
 Cosain S.A. (Sales only)  
 Apartado 50939  
 Salana Grande No. 1, Caracas  
 Telephone: 72-9637 Cable: INSTRUVEN

#### CHILE

SANTIAGO  
 Cosain Chile Ltda. (sales only)  
 Casilla 14588, Correo 15, Santiago  
 Telephone: 396713 Cable: COACHIL

#### JAPAN

TOKYO  
 Rikei Trading Co., Ltd. (sales only)  
 Kozato-Kaikari Bldg.  
 No. 18-14, Nishishimbashi 1-chome  
 Minato-Ku, Tokyo, Japan  
 Telephone: 5915246 Telex: 781-4208

#### DIGITAL EQUIPMENT CORPORATION INTERNATIONAL

Kowa Building No. 17, Second Floor  
 2-7 Nishi-Azabu 1-Chome  
 Minato-Ku, Tokyo, Japan  
 Telephone: 404-58946 Telex: TK-6428

#### PHILIPPINES

Stanford Computer Corporation  
 P.O. Box 1608  
 416 Dasmarinas St., Manila  
 Telephone: 49-68-86 Telex: 742-0352

#### INDIA

H.S. Sonawala Mg Director (Sales Only)  
 HINDITRON SERVICES PUT LTD.  
 68/A Nepean Sea Road  
 Bombay, India

digital

decsystem10 handbook series