PDP-10 REFERENCE HANDBOOK

Prepared by
The Software Writing Group
Programming Department
Digital Equipment Corporation

Additional copies of this handbook may be ordered from the Program Library, DEC, Maynard, Mass. 01754. Order code: AIW. $5.00 each. Discounts available on five or more copies.

PDP-10 HANDBOOK SERIES

The material in this handbook, including but not limited to instruction times and operating speeds, is for information purposes and is subject to change without notice.

FOREWORD

One of the significant measures of the quality of a computing system like the PDP-10 is the utility and availability of systems documentation. Good manuals are the vital communications link between DEC and the people who use our systems.

This collection has been organized for the convenience of PDP-10 programmers, systems analysts, engineers and others who work at the machine language level.

I'm pleased to acknowledge here the work of the many DEC system designers, engineers, and system programmers who continue to advance the state of the time-sharing art in both hardware and software. Also, to our PDP-10 users, who during the past two years, have helped immeasurably to improve and refine the system, and to the DEC software writers and technical artists who prepared this volume, our special thanks.

Kenneth H. Olsen

President, Digital Equipment Corporation

PREFACE

This volume is a comprehensive library of information for experienced programmers, systems analysts, and engineers who are interested in writing and operating assembly-language programs in the PDP-10 time-sharing environment.

The first three chapters deal with program preparation. Chapters 4 and 5 are about loading, editing, testing, debugging, and running source language programs. Chapter 6 contains a miscellaneous collection of utility programs that have proven most useful to system designers and experienced programmers.

As we expect to improve this book in future revisions, all readers are earnestly requested to send corrections and comments to:

Manager, Software Writing Group
Programming Department
Digital Equipment Corporation
Maynard, Mass. 01754

A companion volume for beginning programmers and others who prefer to write programs in one of the popular compiler or conversational languages is scheduled for publication in 1970.

CONTENTS

SYSTEM OVERVIEW

The PDP-10 is the successful culminatioo of many years of computer design research — a process which has enabled Digital Equipment Corporatioo to provide better computers at the lowest possible prices.

Starting with the PDP-1 in 1959, DIGITAL has pioneered the development of real-time systems for scieoce and iodustry. Since then, each new system has increased in versatility, yet has coosistently decreased the cost of computation. The PDP-1 was the first powerful real-time computer for under $150,000. The PDP-g showed that an effective computer could sell for less than $20,000, and newer models io the PDP-g family have lowered the cost to less than $10,000.

In developing its time-sharing capability, DIGITAL has built a history of success very similar to the company's record in real-time applicatioos. DIGITAL's customers have been buildiog time-shariog systems around PDP computers since 1960. And, in 1963, DIGITAL developed its own time-sharing computer, PDP-6 — the first to be delivered with manufacturer-supplied hardware and software.

The PDP-10 reflects DIGITAL experience io both real-time and time-shariog. The system performs time-sharing and real-time operations equally well and simultaneously and provides concurrent batch processing.

In conversational time-sharing, up to 63 users at local and remote locations can simultaneously develop programs on remote consoles and receive answers to mathematical or engineering problems io seconds. PDP-10 time-sharing mooitors provide instantaneous respoose for the users so that they can perform on-line compositioo, editing, and debugging of programs in FORTRAN IV, MACRO-10, COBOL, BASIC, and AID, with the use of EDITOR, TECO and DDT. The monitors can handle any mixture of these languages and programs concurreotly. Aod most of the software is re-entrant so that multiple users can share the same compiler or utility program for increased efficiency.

For programs that doo't require immediate processiog, users may initiate batch processing — a task which proceeds coocurrently with time-sharing and real-time tasks. In batch processing, the PDP-10 can handle any stream of programs, such as a mixture of FORTRAN, MACRO-10 and COBOL. Normally, batch processiog operates without operator attention. However, the PDP-10 allows the operator to stop or start the batch system, rearrange the queue or call for a print-out to analyze program errors. The operator can also select and assign the desired input, output, and temporary storage devices to be used in batch processing.

Wheo real-time operations such as data acquisition and cootrol are the primary purpose of the PDP-10, system software provides response in microseconds, processing ioformation at speeds that meet the most demanding requirements. High priority real-time tasks are interfaced directly to the priority interrupt system and control their own input/output operations for unlimited flexibility. Less critical real-time jobs are mooitor cootrolled with a real-time clock assuring that each task does not exceed its allotted time and destroy the respoose of other programs. For even greater efficiency, time not used by the real-time programs can he used for conversational time-sharing and/or batch processing.

Structure of the PDP-10

Every PDP-10 uses one of three levels of mooitors to allocate resources and perform input/output functions. The single-user monitor is used for dedicated systems which operate one program at a time. The multi-programming monitor controls the execution of multiple programs in core, switching hetweeo them at microsecond speeds. The swapping mooitor effectively increases the available core by swapping programs between high speed disk or drum storage and core memory. Thus more users can be served hy a giveo amount of core. All language processors (FORTRAN, MACRO-10, COBOL, BASIC, and AID) operate identically under the multi-programming and swapping monitors.

To make efficieot use of memory, language processors and important utility programs are re-eotrant, that is, the pure code for each program can be shared by multiple users. Re-entrancy is possible since any program

may be separated into a pure segmeot that never requires modification, and a segment which contains code or data which is relevant only to a particular user. For example, a re-entrant system can service three FORTRAN users with one g K compiler (pure code) and three 2 K user areas, a total ðf 14 K, whereas a non-re-eotrant system would require 30 K for the same programs. Since more users can occupy a giveo amouot of core space, system response improves and swapping time is reduced. Dual protection and relocation registers protect the active user and allow the program segments to reside in two oon-contiguous sections of core memory.

The PDP-10 has a 36-bit word length allowing it to store 25 to 30 percent more information than a-32-bit system. The system stores five 7-bit USACII cbaracters whereas the smaller word length computer stores only four characters. It also provides more accuracy in single precision floating arithmetic than computers with 32-bit word size.

The PDP-10 has a large instruction repertoire to simplify assembly coding and reduce tbe size of higher level programs. Its 366 instructions divide logically ioto families and are easily learned. The list also includes an extensive set of floating point and byte manipulation instructions. Due to instruction set efficiency, fewer instructions are required to perform a given function. Assembly language programs are therefore shorter than with other computers and the instruction set simplifies monitor systems, language processors, and utility programs. For example, compiled programs are 30 to 50 per cent shorter, require less memory, and execute faster than those of comparable computers.

Sixteen high speed integrated circuit registers also help improve program execution. Depending on program requirements, these registers can serve as accumulators, normal memory location, and/or index registers. Intermediate results of computations are stored in the registers rather than in core memory; thus, oo instructions are needed to store and retrieve the data and data is available in oanoseconds. Fifteeo of the registers can be used as fast memory locations so that program segments with sixteen or fewer instructions can be executed repetitively at very high rates.

The PDP-10 memory bus structure gives the central processor and high speed data channels simultaneous access to separate memory modules. Only when tbe processor and a data channel access the same module does the processor lose a memory cycle. Modules contain up to four ports, allowing access to a total of four processors and/or cbannels. Such parallel operation improves processor utilization, yielding manyfold improvements over systems which provide only a single path to memory. The bus system allows each data channel to transmit full 36-bit words at speeds of up to ooe million words (5 million 7-bit characters) per second.

Memory can be modularly expanded to 262,144 words of core, all of wbich (including the 16 accumulators and 15 index registers) can be directly addressed. Total memory capacity can be comprised of combinations of modules in 8,192-, 16,384-, 32,768-, 65,356-, and 131,072-word blocks. Memory banks are asynchronous allowing interleaving and making it possible to iotermix memories of different cycle times.

To provide immediate service to real-time requests, the PDP-10 has a multi-level priority interrupt system. The system is a hardware feature, but is programmable for increased flexibility. Devices may be assigned to any level under program control and the eotire interrupt system or any level may be selectively turned on or off.

PDP-10 Configurations

The modularity of PDP-10 hardware and software makes it ecooomical to configure a wide variety of systems and easy to expand the systems in the field. (See PDP-10 hardware list in Appeodix A.) An individual can buy a single user system and; at some later date, expand to a small time-sharing system. Or the small time-shariog user can expand his system to serve 63 users simultaneously. Within any basic configuration, the nser has a wide cboice of memory sizes and speeds, ioput/ootput equipment, and storage facilities. For example, the input/output system alone can accommodate up to 128 discrete devices and device controllers, permitting almost limitless expansion of oo-line storage and other input/outpnt equipment.

The single-user system in Figure 1 can be simple or as elaborate as the user requires. As shown, it consists of an arithmetic processor, one or more core memory modules, a DECtape control and DECtape units[1], a console teletype, and a paper tape reader and punch. By adding more core memory and data line scanoer, the system can easily be cooverted to multi-programming.

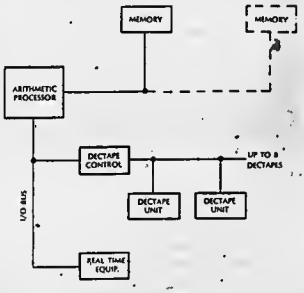


FIGURE 1. SINGLE-USER SYSTEM

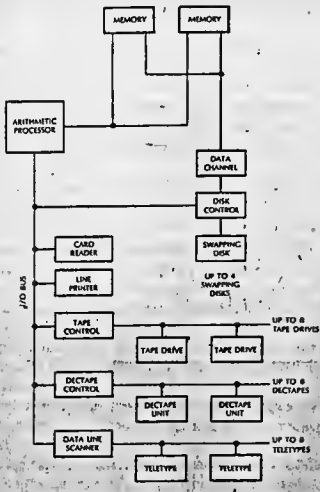


FIGURE 2. 8-USER SWAPPING SYSTEM

[1]A special random access magnetic tape designed by Digital Equipment Corporation.

The small swappiog system shown in Figure 2 can be expanded in eight user groups to the large time-sbaring system (Figure 3) which cao handle up to 63 users. The large system includes file storage and swapping storage units, additional memory, and more peripheral equipment. For very large systems, a file storage disk may replace or supplemeot the disk packs. A computer-based communication system may be substituted for the data lioe scanner, and synchronous data phone units can be used to connect the system to remote batch devices and other computers.

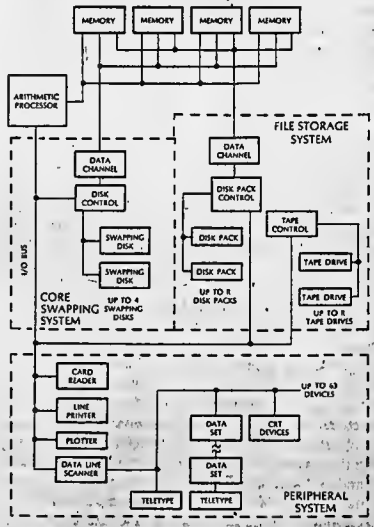


FIGURE 3. LARGE SWAPPING SYSTEM

The dual processor system in Figure 4 is one of many possible multi-processor configurations that the user can tailor bis mooitor to meet. Since the system shares both peripherals and core memory, both processors can access memory at the same time and can compote in parallel. Such an arrangement doubles the computing power of a single processor and more than doubles cost/effectiveoess, since the cost of the additiooal processor is only a small fractioo of overall system cost.

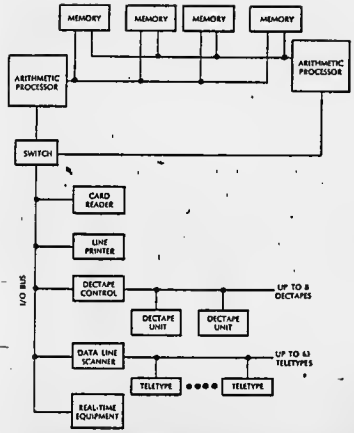


FIGURE 4. DUAL PROCESSOR SYSTEM

In other multi-processor systems, the processors may work independently or communicate through shared memory. One may serve as the input/output processor while the other performs most of the calculations. Or the processors can share all input/output and processing. Multi-processor systems can also combine a PDP-10 arithmetic processor with other DIGITAL computers.

In Summary

The PDP-10 exemplifies the versatility required for today's large computing tasks. With its 366-instruction repertoire, re-entrant software, multi-programming hardware, and flexible priority interrupt system, it provides power and excellent response for a multitude of applications. And with the system's wide range of hardware and software, the user can purchase to serve present needs, yet easily expand to meet future system requirements.

Every PDP-10 is backed by service — software support for a full range of customer assistance, service through a worldwide system of over 60 service centers, and formal training through a variety of available training courses.

At a cost of less than half that of comparable systems, the PDP-10 provides the best price/performance available today — another step toward Digital Equipment Corporation's goal of providing the most for every computing dollar.

SYSTEM DESCRIPTION

DEC PDP-10 software is divided into eight functional groupings with respect to common programming activities, as follows:

a. Source Program Preparation
b. Conversational Language Translators
c. Program Loading and Library Facilities
d. Debugging
e. Utilities
f. Calculators
g. Batch Processing
h. Monitoring

Groups a through g are programs called CUSPs (commonly used system programs) and are run under control of the Single-User, Multiprogramming non-disk, Multiprogramming disk or Swapping Monitor.

Source Program Preparation (EDITOR, LINED, TECO)

The DECtape Editor, LINED (Line Editor for Disk), and the Text Editor and Corrector (TECO) programs can be used to create (and later correct or modify) text files (e.g., Macro-10 and FORTRAN source language programs) for subsequent assembly or compilation. Editor creates and modifies files on DECtape; LINED creates and modifies files on disk; and TECO performs more complex editing functions on any standard I/O devices.

Language Translators (MACRO, F40)

The Macro-10 Assembler (MACRO) and the FORTRAN Compiler (F40) translate source programs written in the Macro-10 and FORTRAN IV languages, respectively, into binary machine language for subsequent loading and execution.

Program Loading and Library Facilities (LOADER, LIB40, JOBDAT, FUDGE2)

Loading is performed by the Linking Loader, which loads specified relocatable binary programs in core, links their references to each other, and searches the appropriate subroutine libraries (e.g., LIB40) for required subroutines. A job data area (JOBDAT) is created by the Loader for each program; this area is used to store the current status of the job during execution. Library files of binary programs can be updated by use of the File Update Generator (FUDGE2).

Debugging (DDT, CREF, GLOB)

After a program is compiled (or assembled), it can be loaded in conjunction with the Dynamic Debugging Technique (DDT) program and debugged. DDT allows the user to control program execution and to modify the program in any of several modes, including symbolic. For purposes of further program analysis (and for documentation), the user can use the Cross Reference Listing (CREF) program, which produces a cross-referenced listing of all symbols within his Macro program, and the Global Cross-Reference Listing (GLOB) program, which produces one to three listings of all global symbols encountered in one or more programs.

Utilities (PIP, CODE, SRCCOM, BINCOM)

A variety of utility programs are available for general-purpose data handling. Among these programs are: the Peripheral Interchange Program (PIP), which transfers data between any standard I/O devices; Code Translator (CODE), which performs translations between standard ASCII codes and code of other manufacturers; Source Compare (SRCCOM), which compares two versions of an ASCII file; and Binary Compare (BINCOM), which compares two versions of a binary file.

Conversational Languages (AID, BASIC)

Two problem-solving conversational languages for scientists, engineers, and students are included as part of the PDP-10 software: the Algebraic Interpretive Dialogue (AID), a program based on the RAND JOSS™ algebraic language; and Advanced BASIC®, a conversational language for scientific, business, and educational applications that includes among many other features a special set of matrix processing operations.

™ JOSS is the trademark and service mark of the RAND Corporation for its computer program and services using that program.

® Registered, Trustees of Dartmouth College.

Batch Processing (BATCH, STACK)

The Batch Processor (BATCH) monitors the sequential execution of a series of user jobs with a minimum of operator attention; operates as one of the "users" in a time-sharing environment and runs concurrently with the Batch-controlled jobs (as well as other jobs on the system); and permits constant communication by the operator. Job Stacker (STACK) prepares input stacks for BATCH and processes output stacks from BATCH.

Monitors

PDP-10 software includes two major categories of Monitors: the Single-User Monitor (10/30 configuration) and the Time-Sharing Monitors (10/40 and 10/50 configurations). The latter category includes the Multiprogramming non-disk Monitor (10/40), Multiprogramming disk Monitor (10/40) and the Swapping Monitor (10/50). The Swapping Monitor was used in the generation of all examples in this manual.

The 10/30 Monitor is a subset of the 10/40 and 10/50 Monitor. They are compatible at the source and relocatable binary levels. The 10/40 and 10/50 Monitors are compatible at the source, relocatable binary, and saved core image levels.

SYSTEM OPERATION

The following basic procedures and rules are necessary to communicate with the Monitor and load and execute DEC commonly used system programs (CUSPs), as well as user programs.

Step	Procedure
1.	To establish communication with the Monitor, place the Teletype in Monitor Command Mode by typing ↑C (i.e., hold down the CTRL key while striking C; Monitor will respond with a period (.). If you have a 10/30 or a 10/40 system, skip the LOGIN procedure in the next step.
2.	Type LOGIN followed by carriage return. The system will respond with JOB n NAME OF SYSTEM followed by a number sign. Then type your project-programmer number, followed by a carriage return. The system will then type PASSWORD: Then type your secret password followed by a carriage return. The system will keep it secret by not printing it on the paper. If the project-programmer number agrees with the password, you will be logged, and any messages of the day will be typed for you.
3.	Then, direct Monitor to load and start a program from the System Library (.R prog), start a program already loaded in core (.START), discontinue your job (.KJOB), or perform any of a variety of other operations. A complete list of Time-Sharing Monitor commands is given in Table 9-1.

All CUSPs (except EDITOR and LINED) supplied by DEC are device independent; therefore, the user must tell the CUSP, via a command string typein, which devices to use. Readiness to receive a command string is signalled by the CUSP via an asterisk (*) typeout after loading. For example, when the FORTRAN IV Compiler has been called and it has responded with an asterisk, the user types in a command string indicating:

a. The device containing the source program to be compiled
b. The device on which the binary output is to be placed and
c. The device on which the compilation listing is to be written

*binary-output-device, listing-device = source-device

Devices are specified by a 3-character device name (a fourth character, a digit, specifies the particular unit in the case of DECtapes, teletypes, and magnetic tapes), followed by a colon.

Device	Device Name
Card reader	CDR:
Card punch	CDP:
Line printer	LPT:
Paper tape reader	PTR:
Paper tape punch	PTP:
Teletype	TTY: or TTYn:
DECtape	DTAn:
Magnetic tape	MTAn:
Disk	DSK:

For file-oriented devices (DECtape and disk), a filename (maximum of six characters) is also required following the device name to specify either the specific file to be read or the filename to be assigned to the output. A filename can be further specialized by adding a 3-character extension name to it, preceded by a period (.). Extension names are generally used to classify a file into a particular category, and certain standard extensions are used and recognized throughout the system (e.g., .REL for relocatable binary files, .SAV for saved core image files, .MAC for Macro-10 source files, .F4 for FORTRAN source files, etc.). The following example shows a sample FORTRAN command string.

Example:

DTA1:BIN.REL,LPT: = DTA0:SOURCE

Compile the file designated as SOURCE on DECtape 0; write the binary output on DECtape 1, designating it BIN.REL; print the listing on the line printer.

TYPOGRAPHIC CONVENTIONS IN THIS MANUAL

All computer typeouts are underscored (single line) or enclosed in brackets (multiple lines).

All operator typeins are not underscored.

SYMBOLOGY USED IN CONSOLE EXAMPLES

↑C — Hold down the CTRL key while striking C. Normally echoes as ↑C.

↑x — Hold down the CTRL key while striking the "x" key, where "x" is any character. Normally echoes as ↑x.

Some special control symbols and their respective key designations for Models 33, 35, and 37 Teletypes are given below.

Key Designation	Symbol in This Manual	Models 33 and 35	Model 37
(TAPE)	↑R	Hold down CTRL key while striking R.	Same as 33/35
(TAPE) (not-TAPE)	↑T	Hold down CTRL key while striking T.	Same as 33/35
(BELL)	↑G	Hold down CTRL key while striking G.	Same as 33/35
(TAB) (horizontal tab)	→| or ↑I	Hold down CTRL key while striking I.	Strike TAB key
(FORM)	↑L	Hold down CTRL key while striking L.	Same as 33/35
(VT) (vertical tab)	↑K	Hold down CTRL key while striking K.	Same as 33/35
(XON)	↑Q	(Initialize paper tape reader input.) Hold down CTRL key while striking Q.	Same as 33/35
(XOFF)	↑S	(Terminate paper tape reader input.) Hold down CTRL key while striking S.	Same as 33/35
←	←	Hold down the SHIFT key while striking O.	Strike ← key
RETURN	)	Strike the RETURN key. Normally echoes back as a carriage return, line feed.	Same as 33/35
ALTMODE or PREFIX or ESC	$	Strike the ESC key (sometimes labeled ALTMODE or PREFIX)	Same as 33/35
[	[	Hold down the SHIFT key while striking K.	Strike [key

Key Designation	Symbol in This Manual	Models 33 and 35	Model 37
]	]	Hold down the SHIFT key while striking M.	Strike] key
↑	↑	When appearing alone (as in DDT), hold down the SHIFT key while striking N.	Strike ↑ key
<	<	Hold down the SHIFT key while striking ",".	Strike < key
>	>	Hold down the SHIFT key while striking ".".	Strike > key
LINE FEED	↓	Strike the LINE-FEED key.	Strike LINE SPACE key
RUBOUT	(RUBOUT)	Strike the RUBOUT key. Normally echoes back as a backslash (\), XXX, or a repeat of the character erased.	Strike DELETE key
\	\	Hold down the SHIFT key while striking L.	Strike the \ key
SPACE BAR	␣	Strike the space bar to space to indicated position. TAB can also be used in most instances.	Same as 33/35

NOTE

Due to recent changes in ASCII, some terminals may have the keytops "↑" (caret) and "←" (underline); these characters have the same codes as "^" and "_", respectively. Where possible, DEC will supply all teletypes with the arrow characters.

DEMONSTRATION PROGRAMS

The following demonstration programs illustrate the simplicity and flexibility of a PDP-10 software system:

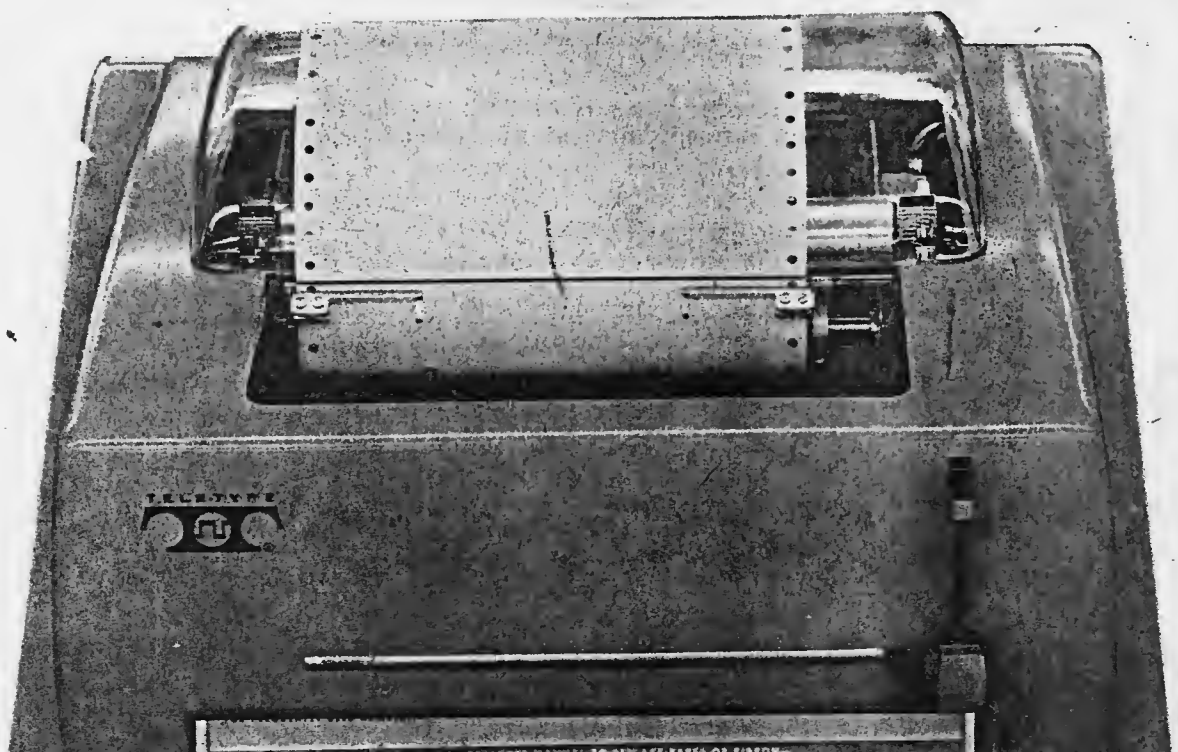
a. Demonstration #1 is a typical example of the procedure for creating a FORTRAN main program source file and a Macro-10 subprogram file. These two files are then translated, loaded, and executed together. A bug is encountered during execution, and the DDT (Dynamic Debugging Technique) program is used to correct the erroneous instruction. The programs are now executed successfully, their core image is saved for future execution, and the original source file is altered to reflect the correction made to the binary code.

b. Demonstration #2 is a more complex example. The sequence of operations is similar to that of Demonstration #1 (source program file creation, translation, loading, execution, debugging, saving the core image, and altering the source code to reflect changes made to the object code). In addition, such procedures as leaving the current job, logging in and beginning a second job, and then later returning to the original job are included. Figure I-1 contains the flow diagram for Demonstration #2.

Teletype KSR 35

have separate keys. Note also that both the keyboard arrangement and the labels differ somewhat. On both, the line feed (labeled "new line" on the 37) spaces the paper vertically at six lines to the inch, and must be combined with a return to start a new line. The local advance (feed) and return keys affect the printer directly and do not transmit codes. Appendix B lists the complete teletype code, ASCII characters, key combinations, and differences between the two models.

Indicators for the teletype are the TTY lights in the second row of the



panel at the top of bay 1. The numbered lights display the last character typed in from the keyboard (bit 8 is parity). The ACT lights indicate activity in the transmitter and receiver. The remaining lights display the PI assignment and flags (the Input and Output Done flags are labeled TTI FLAG and TTO FLAG).

Teletype KSR 37

Teletype manuals supplied with the equipment give complete, illustrated descriptions of the procedures for loading paper, changing the ribbon, and setting horizontal and vertical tabs. The first two procedures are fairly


TELETYPE

panel at the top of bay 4. The numbered ligh
typed in from the keyboard (bit 8 is parity)
activity in the transmitter and receiver. The re
assignment and flags (the Input and Output
FLAG and TTO FLAG).

Teletype manuals supplied with the equipme
descriptions of the procedures for loading pap
setting horizontal and vertical tabs. The firs

ngement and the
line" on the 37)

Teletype KSR 37

nts display the last character
). The ACT lights indicate
maining lights display the PI
Done flags are labeled TTI

ent give complete, illustrated
er, changing the ribbon, and
st two procedures are fairly

TELETYPE CODE

Even Parity Bit	7-Bit Octal Code	Character	Remarks
0	000	NUL	Null, tape feed. Repeats on Model 37. Control shift P on Model 35.
1	001	SOH	Start of heading; also SOM, start of message. Control A.
1	002	STX	Start of text; also EOA, end of address. Control B.
0	003	ETX	End of text; also EOM, end of message. Control C.
1	004	EOT	End of transmission (END); shuts off TWX machines. Control D.
0	005	ENQ	Enquiry (ENQRY); also WRU, "Who are you?" Triggers identification ("Here is . . . ") at remote station if so equipped. Control E.
0	006	ACK	Acknowledge; also RU, "Are you . . . ?" Control F.
1	007	BEL	Rings the bell. Control G.
1	010	BS	Backspace; also FEO, format effector. Backspaces some machines. Repeats on Model 37. Control H on Model 35.
0	011	HT	Horizontal tab. Control I on Model 35.
0	012	LF	Line feed or line space (NEW LINE); advances paper to next line. Repeats on Model 37. Duplicated by control J on Model 35.
1	013	VT	Vertical tab (VTAB). Control K on Model 35.
0	014	FF	Form feed to top of next page (PAGE). Control L.
1	015	CR	Carriage return to beginning of line. Control M on Model 35.
1	016	SO	Shift out; changes ribbon color to red. Control N.
0	017	SI	Shift in; changes ribbon color to black. Control O.
1	020	DLE	Data link escape. Control P (DC0).
0	021	DC1	Device control 1, turns transmitter (reader) on. Control Q (X ON).
0	022	DC2	Device control 2, turns punch or auxiliary on. Control R (TAPE, AUX ON).
1	023	DC3	Device control 3, turns transmitter (reader) off. Control S (X OFF).
0	024	DC4	Device control 4, turns punch or auxiliary off. Control T (~~TAPE~~, AUX OFF).
1	025	NAK	Negative acknowledge; also ERR, error. Control U.
1	026	SYN	Synchronous idle (SYNC). Control V.
0	027	ETB	End of transmission block; also LEM, logical end of medium. Control W.
0	030	CAN	Cancel (CANCL). Control X.
1	031	EM	End of medium. Control Y.
1	032	SUB	Substitute. Control Z.
0	033	ESC	Escape, prefix. This code is generated by control shift K on Model 35, but the Monitor translates it to 175.
1	034	FS	File separator. Control shift L on Model 35.
0	035	GS	Group separator. Control shift M on Model 35.

Even Parity Bit	7-Bit Octal Code	Character	Remarks
0	036	RS	Record separator. Control shift N on Model 35.
1	037	US	Unit separator. Control shift O on Model 35.
1	040	SP	Space.
0	041	!	
0	042	"	
1	043	#	
0	044	$	
1	045	%	
1	046	&	
0	047	'	Accent acute or apostrophe.
0	050	(	
1	051	)	
1	052	*	Repeats on Model 37.
0	053	+	
1	054	,	
0	055	-	Repeats on Model 37.
0	056	.	Repeats on Model 37.
1	057	/	
0	060	Ø	
1	061	1	
1	062	2	
0	063	3	
1	064	4	
0	065	5	
0	066	6	
1	067	7	
1	070	8	
0	071	9	
0	072	:	
1	073	;	
0	074	<	
1	075	=	Repeats on Model 37.
1	076	>	
0	077	?	
1	100	@	
0	101	A	
0	102	B	

Even Parity Bit	7-Bit Octal Code	Character	Remarks
1	103	C	
0	104	D	
1	105	E	
1	106	F	
0	107	G	
0	110	H	
1	111	I	
1	112	J	
0	113	K	
1	114	L	
0	115	M	
0	116	N	
1	117	O	
0	120	P	
1	121	Q	
1	122	R	
0	123	S	
1	124	T	
0	125	U	
0	126	V	
1	127	W	
1	130	X	Repeats on Model 37.
0	131	Y	
0	132	Z	
1	133	[	Shift K on Model 35.
0	134	\	Shift L on Model 35.
1	135	]	Shift M on Model 35.
1	136	↑	
0	137	←	Repeats on Model 37.
0	140	`	Accent grave.
1	141	a	
1	142	b	
0	143	c	
1	144	d	
0	145	e	
0	146	f	
1	147	g	

Even Parity Bit	7-Bit Octal Code	Character	Remarks
1	150	h	
0	151	i	
0	152	j	
1	153	k	
0	154	l	
1	155	m	
1	156	n	
0	157	o	
1	160	p	
0	161	q	
0	162	r	
1	163	s	
0	164	t	
1	165	u	
1	166	v	
0	167	w	
0	170	x	Repeats on Model 37.
1	171	y	
1	172	z	
0	173	{	▲
1	174	|	
0	175	}	This code generated by ALT MODE on Model 35. ▲
0	176	~	This code generated by ESC key (if present) on Model 35, but the Monitor translates it to 175.
1	177	DEL	Delete, rub out. Repeats on Model 37.

Keys That Generate No Codes

REPT	Model 35 only: causes any other key that is struck to repeat continuously until REPT is released.
PAPER ADVANCE	Model 37 local line feed.
LOCAL RETURN	Model 37 local carriage return.
LOC LF	Model 35 local line feed.
LOC CR	Model 35 local carriage return.
INTERRUPT, BREAK	Opens the line (machine sends a continuous string of null characters).
PROCEED, BRK RLS	Break release (not applicable).
HERE IS	Transmits predetermined 21-character message.

CARD CODES

▲ Character	PDP-10 ASCII	DEC 029	DEC 026	Character	PDP-10 ASCII	DEC 029	DEC 026
Space	040	*None*	*None*	@	100	8 4	8 4
!	041	11 8 2	12 8 7	A	101	12 1	12 1
"	042	8 7	0 8 5	B	102	12 2	12 2
#	043	8 3	0 8 6	C	103	12 3	12 3
$	044	11 8 3	11 8 3	D	104	12 4	12 4
%	045	0 8 4	0 8 7	E	105	12 5	12 5
&	046	12	11 8 7	F	106	12 6	12 6
'	047	8 5	8 6	G	107	12 7	12 7
(	050	12 8 5	0 8 4 ▲	H	110	12 8	12 8
)	051	11 8 5	12 8 4 ▲	I	111	12 9	12 9
*	052	11 8 4	11 8 4	J	112	11 1	11 1
+	053	12 8 6	12	K	113	11 2	11 2
,	054	0 8 3	0 8 3	L	114	11 3	11 3
-	055	11	11	M	115	11 4	11 4
.	056	12 8 3	12 8 3	N	116	11 5	11 5
/	057	0 1	0 1	O	117	11 6	11 6
0	060	0	0	P	120	11 7	11 7
1	061	1	1	Q	121	11 8	11 8
2	062	2	2	R	122	11 9	11 9
3	063	3	3	S	123	0 2	0 2
4	064	4	4	T	124	0 3	0 3
5	065	5	5	U	125	0 4	0 4
6	066	6	6	V	126	0 5	0 5
7	067	7	7	W	127	0 6	0 6
8	070	8	8	X	130	0 7	0 7
9	071	9	9	Y	131	0 8	0 8
:	072	8 2	11 8 2 *or* 11 0	Z	132	0 9	0 9
;	073	11 8 6	0 8 2	[	133	12 8 2	11 8 5
<	074	12 8 4	12 8 6	\	134	11 8 7	8 7
=	075	8 6	8 3	]	135	0 8 2	12 8 5
>	076	0 8 6	11 8 6	↑	136	12 8 7	8 5
?	077	0 8 7	12 8 2 *or* 12 0	←	137	0 8 5	8 2

Binary	7 9
Mode Switch	12 0 2 4 6 8
End of File	12 11 0 1

The octal codes given above are those generated by the Monitor from the column punches. The card reader interface actually supplies a direct binary equivalent of the column punch, as listed in the following two pages.

Column Punch	Character	Octal	Column Punch	Character	Octal
None	*Space*	0000	12 9	I	4001
0	0	1000	11 1	J	2400
1	1	0400	11 2	K	2200
2	2	0200	11 3	L	2100
3	3	0100	11 4	M	2040
4	4	0040	11 5	N	2020
5	5	0020	11 6	O	2010
6	6	0010	11 7	P	2004
7	7	0004	11 8	Q	2002
8	8	0002	11 9	R	2001
9	9	0001	0 1	/	1400
12 1	A	4400	0 2	S	1200
12 2	B	4200	0 3	T	1100
12 3	C	4100	0 4	U	1040
12 4	D	4040	0 5	V	1020
12 5	E	4020	0 6	W	1010
12 6	F	4010	0 7	X	1004
12 7	G	4004	0 8	Y	1002
12 8	H	4002	0 9	Z	1001

Column Punch	026 Data Processing	026 Fortran	029	DEC 026	DEC 029	Octal
12	&	+	&	+	&	4000
11	–	–	–	–	–	2000
12 0				?		5000
11 0				:		3000
8 2			:	←	:	0202
8 3	#	=	#	=	#	0102
8 4	@	'	@	@	@	0042
8 5		:	'	↑	'	0022
8 6		=	=	'	=	0012
8 7		"	"	\	"	0006
12 8 2		¢	¢	?	[	4202
12 8 3	.	.	.	.	.	4102
12 8 4	¤	)	<	)	<	4042
12 8 5		(	(	]	(	4022
12 8 6			+	<	+	4012

Column Punch	026 Data Processing	026 Fortran	029	DEC 026	DEC 029	Octal
12 8 7			|	!	↑	4006
11 8 2			!	:	!	2202
11 8 3	$	$	$	$	$	2102
11 8 4	*	*	*	*	*	2042
11 8 5			)	[	)	2022
11 8 6			;	>	;	2012
11 8 7			¬	&	\	2006
0 8 2			*See note*	;	]	1202
0 8 3	,	,	,	,	,	1102
0 8 4	%	(	%	(	%	1042
0 8 5			←	"	←	1022
0 8 6			>	#	>	1012
0 8 7			?	%	?	1006
12 11 0 1				*End of File*	*End of File*	7400
12 0 2 4 6 8				*Mode Switch*	*Mode Switch*	5252
7 9				*Binary*	*Binary*	*xx*05

NOTE: There is a single key for the 0 8 2 punch on the 029 but printing is suppressed.

The Monitor translates the octal code for the 12 0 punch in DEC 026 to 4202 (which corresponds to a 12 8 2 punch), and the code for 11 0 to 2202 (11 8 2).

APPENDIX C

MISCELLANY

CHAPTER 2
MONITOR COMMANDS

2.1 CONSOLE AND JOB CONTROL

The PDP-10 time-sharing system is a multiprogramming system. This means that control is transferred rapidly among a number of programs or processes in such a way that all the processes appear to be running simultaneously. Each process is called a job. In configuring and loading a time-sharing Monitor, the system administrator sets the maximum number of jobs which his system will handle simultaneously. This number may be up to 127 jobs if the system has enough core, disk storage, processor capacity, and time-sharing consoles to handle this load.

Jobs are initiated by users typing on a time-sharing console. A console is typically any of several models of Teletype machines but may also be a CRT (cathode ray tube) with a keyboard. The console may be directly connected to the computer or may be remotely connected via a private wire or the public telephone system.

There is not necessarily a one-to-one relationship between jobs and consoles. A console must initiate a job, but the DETACH and ATTACH commands (see Table 2.7) permit a job to "float" in a state where it is not associated with a particular console. Therefore a user may control several jobs from the same console. Each job is either in the ATTACHed or DETACHed mode depending on whether a console is currently associated with that job. At any point in time, each console is attached to at most one job. The console is often referred to as being in a "detached mode," but this results from a semantic confusion. It is really

meant that the job initiated from that console is in a detached mode. By typing an appropriate command, the job may be attached to the same console or to any other console in the system.

2.1.1 Monitor Mode and User Mode

From the user's point of view, his console is in one of two states - monitor mode or user mode. In monitor mode, each line the user types in is sent to the Monitor Command Interpreter. The execution of certain commands (as noted in the tables below) places the console in user mode. Once the program is in user mode, the console becomes simply an input/output device for that user. In addition, user programs will use the console for two purposes. The user program will accept command strings from the console or will use the console as a direct input/output device. Example:

monitor mode	.R PIP	monitor command
user mode	*DSK:FOO←TTY:	user program command string
user mode	THIS IS FILE FOO↑Z	user program using console as an input device
monitor mode	.R MACRO	monitor command
user mode	*TTY:,←DSK:PROG1	user program command string
user mode	code	user program using console as an output device

The special character ↑C (produced by typing C with the CONTROL Key depressed) is used to stop a user program and return the console to monitor mode. There are certain commands which

cause the user program to start or continue running (as noted in the tables below) but which leave the console in monitor mode.

When the system is started, each console is in monitor mode ready for users to begin typing in commands. However, if the system becomes fully loaded (i.e., all the jobs that the system can accommodate have been initiated), then any unused consoles enter a special state where any command typed in will receive either the message "JOB CAPACITY EXCEEDED" or "X."

2.2 COMMAND INTERPRETER AND COMMAND FORMAT

Each command is a line of ASCII characters in upper and/or lower case. Spaces and non-printing characters preceding the command name are ignored. The Monitor Command Interpreter will not interpret or execute a line of comments preceded by a semicolon. Every command to the Monitor Command Interpreter must be terminated by pressing the RETURN key on the console. If the command is not understood, an error message is typed out by the Monitor and the mode is unchanged.

2.2.1 Command Names

Command names are strings from one to six letters. Characters after the sixth are ignored. Only enough characters to uniquely identify the command need be typed. In the tables which follow, the commonly used abbreviation of the command name is shown. Installations which choose to implement additional commands should take care to preserve the uniqueness of the first few letters of existing commands.

2.2.2 Arguments

Arguments follow the command name, separated from it by

a space or any printing character that is not a letter or a numeral. Argument formats are described under the associated commands.

If the Monitor Command Interpreter recognizes the command name, but a necessary argument is missing, the Monitor responds with

TOO FEW ARGUMENTS

Extra arguments are ignored.

2.2.3 Login Check (Disk Monitor Systems)

If a user who has not logged in (see Table 2.1) types a command requiring him to be logged in, the disk Monitor systems will respond with

LOGIN PLEASE

and the user's command will not be executed. Login is not required by a non-disk Monitor system.

2.2.4 Job Number Check (Non-disk Monitor Systems)

If the non-disk Monitor system recognizes a command name which requires a job number and no job number has been assigned, the Monitor assigns a job number, n, and responds with

JOB n

and a line identifying the Monitor version. The Monitor will then proceed to execute the command.

2.2.5 Core Storage Check

If the Monitor Command Interpreter recognizes a command name which requires core storage to have been allocated to the job and the job has no core, the Monitor responds with

NO CORE ASSIGNED

The user's command is not executed.

2.2.6 Delayed Command Execution

If the Monitor Command Interpreter recognizes a command that requires all devices to be inactive and the job has devices actively transmitting data to or from its core area, the execution of the command will be delayed until the devices are inactive. A command is also delayed if a job is swapped out to the disk and the command requires core residence. It will be executed when the job is returned to core.

2.2.7 Completion-of-Command Signal

Most commands are processed without delay. The completion of each command is signaled by the output of a carriage return, line feed. If the console is left in Monitor mode, a period follows the carriage return, line feed. If the console is left in user mode, any response other than the carriage return, line feed comes from the user's program. For example, all standard DEC CUSPS immediately send an asterisk (*) to the user's console to indicate their readiness to accept user-mode command strings.

2.3 SYSTEM ACCESS CONTROL COMMANDS

Access to the system is limited to authorized personnel. The system administrator provides each authorized user with a project number, a programmer number, and a password. The project and programmer numbers are octal numbers up to six digits each. The project-programmer numbers will identify not only the user but also his file storage area on the disk. The password is from one to five ASCII characters. To LOGIN successfully the project-

programmer numbers and the password typed in by the user must match the project-programmer numbers and password stored in the system accounting file (ACCT.SYS [1,1]).

Table 2-1

Monitor Command to Gain Access to the System

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
LOGIN	LOG	LOGIN initializes a Monitor routine to accept the user's LOGIN data. The following is the procedure used to gain access to the system. .LOGIN JOB n PDP-10 4S.50F Job number assigned to user, followed by Monitor name and version number. # System types out number sign to indicate user should type his project-programmer number. proj,prog User types in his project-programmer number PASSWORD: System requests user to type his password. User types password followed by carriage return. To maintain password security, the Monitor will not echo the password. 1135 8-AUG-69 TTY23 †C If user entries are correct, Monitor responds with time, date, TTY number, †C and a period, indicating readiness to accept a command.	u R D	LOGIN.PLEASE ? The user has typed a command that the Monitor cannot accept unless the user logs in. ?INVALID ENTRY - TRY AGAIN An illegal project-programmer number was entered or the password did not match. ?1+1/nK CORE VIR. CORE LEFT=0 System core and swapping space exceeded.

*Characteristics:

d = places job in detached mode
m = places job in Monitor mode
u = places job in user mode

L = LOGIN required (Disk Monitor)
A = no active device
C = core required

J = requires a job number.
R = runs a CUSP thereby replacing previous program in user's addressing space.
D = available only in Multiprogramming Disk and in swapping systems, not in Multiprogramming non-disk systems.

2.4 FACILITY ALLOCATION COMMANDS

The Monitor allocates peripheral devices and core memory to users upon request and protects these allocated facilities from interference by other users. The Monitor maintains a pool of available facilities from which a user can draw.

A user should never abandon a time-sharing console without returning his allocated facilities to the Monitor pool. Until a user returns his allocated facilities to the pool no other users may utilize them.

All devices controllable by the system are listed in Table 5-1. Associated with each device is a physical name, consisting of three letters and zero to three numerals to specify unit number. A logical device name may also be assigned by the user. This logical name of one to six alphanumeric characters of the user's choice is used synonymously with a physical device name in all references to the device. In writing a program, the user may use arbitrarily selected device names which he assigns to the most convenient physical devices at runtime. All references to devices in the Monitor pool are made by physical names or by assigned logical names.

When a device is assigned to a job, it is removed from the Monitor's pool of available facilities. Any attempt by another job to reference the device fails. The device is returned to the pool when the user deassigns it or kills his job.

Table 2-2

Monitor Commands to Allocate Facilities

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
ASSIGN[1] phys-dev log-dev	AS	To assign an I/O device to the user's job for the duration of the job or until a DEASSIGN command is given. phys-dev — Any device listed in Table 5-1[2]. This argument is required. log-dev — A logical name assigned by the user	m L J	dev: ASSIGNED The device has been successfully assigned to the job. NO SUCH DEVICE Device name does not exist. ALREADY ASSIGNED TO JOB n The device has already been assigned to another user's job. LOGICAL NAME ALREADY IN USE DEVICE dev: ASSIGNED The user has previously assigned this logical name to another device.
DEASSIGN[1] dev	DEA	Returns one or more devices currently assigned to the user's job to the Monitor's pool of available devices. dev — If this argument is not specified, all devices assigned to the user's job are deassigned. If this argument is specified, it can be either the logical or physical device name.	m L J	NO SUCH DEVICE Device name does not exist. DEVICE WASN'T ASSIGNED The device isn't currently assigned to this job.
REASSIGN dev job	REA	Allows one job to pass a device to a second job without going through the Monitor device pool. dev — The physical or logical name of the device to be reassigned. Cannot be a user console. job — The number of the job to which the device is to be reassigned.	m L J C A	DEVICE dev WASN'T ASSIGNED The device isn't currently assigned to this job. JOB NEVER WAS INITIATED The job number specified has not been initialized. NO SUCH DEVICE The device does not exist. DEVICE CAN'T BE REASSIGNED A user's console Teletype cannot be reassigned.

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
FINISH dev	F	Terminates any input or output currently in progress on the device. dev — The logical or physical name of the device on which I/O is to be terminated. If no name is specified, I/O is terminated on all devices assigned to the job.	m L J C A	NO SUCH DEVICE Either the device does not exist or it was not assigned to this job.
TALK dev	TA	To allow the user to type directly to another user's console. dev — Must be one of the following: CTY - Console Teletype TTYn - Where n can be in the range of 0 through 77. OPR - Operator's console (the Teletype designated as such when the Monitor was initialized).	m	BUSY The console addressed is either (1) not in the Monitor mode or (2) is not positioned at the left margin. (OPR is never busy.)
CORE n	COR	To modify the amount of core assigned to the user's job. n = o — The low and high segments disappear from the job's virtual addressing space. n > o — Total number of 1K blocks of core to be assigned to the job from this point on. If n is omitted, Monitor types out the same response as when an error occurs, but does not change core assignment.	m J A C	10/40 Systems: m/p 10/50 Reentrant Systems: m+n/p CORE VIR. CORE LEFT=v Key: m = number of 1K blocks in low segment n = number of 1K blocks in high segment p = maximum K per job swapping systems-max. physical user core non-swapping systems - free + dormant core v = number of K unassigned in core and swapping device

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
RESOURCES	RES	To print out all the available devices (except TTY's) and the number of free blocks on the disk.	m	

*Refer to footnote in Table 2-1

[1]The ASSIGN 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. The DEASSIGN command applied to DECtapes performs the same function. (See 5.7.7 for further details.)

[2]If DTA or MTA is used, the Monitor performs a search for an available drive and then types out DTAn (or MTAn) ASSIGNED.

Examples showing use of logical and physical names:

User types	.ASSIGN DTA,ABC	
Monitor responds	DEVICE DTA6 ASSIGNED	(successful)
User then types	.ASSIGN DTA,DEF	(find another unit)
Monitor responds	NO SUCH DEVICE	(all in use)
User then types	.ASSIGN PTP,ABC	(reserve paper tape punch)
Monitor responds	LOGICAL NAME ALREADY IN USE DEVICE PTP ASSIGNED	(paper tape punch is reserved, but ABC still refers to DTA6 only)
User then types	.ASSIGN DTA1,DEF	
Monitor responds	ALREADY ASSIGNED TO JOB 2	(another user has it)

User then types	.R PIP	(request for system program PIP)
User then types	*PTP:=ABC:FOO	(command string to PIP asking that file FOO be transferred from device ABC (which is now assigned as DTA6) to device PTP (which is assigned to user).

NOTE: The user does not type the period or the asterisk. The period is the Monitor response to the user and the asterisk is the CUSP response. The user must terminate every command to the Monitor Command Interpreter by pressing the RETURN key on the Teletype.

2.5 SOURCE FILE PREPARATION COMMANDS

The following commands call in the editing programs and cause these programs to open a specified text file for editing. Two of these commands call the TECO CUSP and two call the LINED CUSP (a disk-oriented version of EDITOR). For each editor, one command causes an existing file to be opened for changes and the other command causes a new file to be created. Each command requires a filename as its argument and may have an optional extension.

Filenames are from one to six letters or digits. All letters or digits after the sixth are ignored. A filename is terminated by any character other than a letter or digit. If a filename is terminated by a period, a filename extension is assumed to follow. A filename extension is from one to three letters or digits. It is generally used to indicate file format. The filename extension is terminated by any character other than a letter or digit.

The following are the standard meanings for file extensions:

.TMP	Temporary file
.MAC	Source file in MACRO language
.F4	Source file in FORTRAN IV language
.CBL	Source file in COBOL language (available in 1970)
.LST	Listing or CREF data
.REL	Relocatable binary file
.CMD	Command file, for @ construction
.SAV	Core dump, from SAVE command
blank	Unspecified ASCII text file

Each time one of these commands is executed the command with its arguments is "remembered" as a file on the disk. Because of this, the filename last edited may be recalled for the next edit without specifying the arguments again. For example, if the command

```
.CREATE PROG1.MAC
```

is executed, then the user may later type the command

```
.EDIT
```

instead of

```
.EDIT PROG1.MAC
```

assuming no other source file preparation command was used in the interim.

Table 2-3

Monitor Commands to Prepare Source Files

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
EDIT file.ext	ED	Runs LINED (Line Editor for Disk) and opens an already existing sequence-numbered file on disk for editing.	u L R D J	See Table 2-4
CREATE file.ext	CREA	Runs LINED and opens a new file on disk for creation.	u L R D J	See Table 2-4
TECO file.ext	TE	Runs TECO (Text Editor and Corrector) and opens an already existing non-sequence-numbered file on disk for editing.	u L R D J	See Table 2-4
MAKE file.ext	M	Runs TECO and opens a new file on disk for creation.	u L R D J	See Table 2-4

*Refer to footnote in Table 2-1

Table 2-4

Monitor Command Diagnostic Messages

(For File Manipulation Commands)

Message	Meaning
COMMAND ERROR	The COMPIL CUSP cannot decipher the command.
DEVICE NOT AVAILABLE	Specified device could not be initialized.
DISK NOT AVAILABLE	Device DSK: could not be initialized.

Table 2-4 (Cont)

Monitor Command Diagnostic Messages

(For File Manipulation Commands)

Message	Meaning
EXECUTION DELETED (typed by LOADER)	Errors detected during assembly, compilation, or loading prevent a program from being executed. Loading will be performed, but LOADER will EXIT to the Monitor without starting execution.
FILE IN USE OR PROTECTED	A temporary command file could not be entered in the user's UFD.
INPUT ERROR	I/O error occurred while reading a temporary command file from the disk.
LINKAGE ERROR	I/O error occurred while reading a CUSP from device SYS:.
NESTING TOO DEEP	The @ construction exceeds a depth of nine; may be due to a loop of @ command files.
NO SUCH FILE - file.ext	Specified file could not be found (may be a source file or a file required for operation of COMPIL CUSP).
NOT ENOUGH CORE	System cannot supply enough core for use as buffers or to read in a CUSP.
OUTPUT ERROR	I/O error occurred while writing a temporary command file on disk.
PROCESSOR CONFLICT	Use of + construction has resulted in a mixture of source languages.
TOO MANY NAMES or TOO MANY SWITCHES	Command string complexity exceeds table space in COMPIL CUSP.
UNRECOGNIZABLE SWITCH	An ambiguous or undefined word followed a slash (/).

2.6 FILE MANIPULATION COMMANDS

Each of the following commands performs complex functions which would require a number of commands on a less sophisticated system. The commands in Table 2-5 list the user's files and file directories and cause his source files to be compiled, loaded, and executed.

Table 2-5

Monitor Commands to Manipulate Files

Commands	Abbreviation	Explanation	Characteristics*	Monitor Messages
TYPE list	TY	Directs PIP (Peripheral Interchange Program) to type contents of named source file(s) on user's Teletype. list — A single file specification, or a string of file specifications separated by commas. A file specification is the same as that described for COMPILE, LOAD, EXECUTE, and DEBUG commands. In addition, the * construction can be used as follows: filename.* — All files with this filename and any extension *.ext — All files with this extension and any filename *.* — All files Examples: TYPE FILEA, DTA0:FILEB.MAC,*.TMP TYPE A,DTA4:B,C[15,107]	m L R D	See Table 2-4
LIST list	LI	Directs PIP to list contents of named source file(s) on the line printer (LPT). Examples: LIST TEST.* LIST *.MAC LIST DTA4:A,B,C	m L R D	See Table 2-4
DIRECT dev	DI	If dev: is omitted or DSK:, directory listing of user's disk files is typed on the user's Teletype. If DTAn: is specified, directory of that DECtape is typed. Two switches can be used with the DIRECT command: /F List short form of directory (i.e., omit dates) /L List on line printer (LPT) instead of Teletype. The : may be omitted in dev.	m L R D	See Table 2-4

Commands	Abbreviation	Explanation	Characteristics*	Monitor Messages
DELETE list	DEL	Deletes one or more files from disk or DECtape. If a device name is specified, it remains in effect until changed or end of command string is reached.	m L R D	See Table 2-4
RENAME arg	REN	Changes the name of one or more files on disk or DEC tape. The arg is a pair of file specifications separated by an = sign, or a string of such pairs separated by commas: RENAME new1 = old1,new2 = old2,... Device names can be specified only with the new filename and remain in effect until changed or end of command string is reached.	m L R D	See Table 2-4
CREF	CREF	Runs CREF and lists on the line printer any CREF listing files generated by previous COMPILE, LOAD, EXECUTE, and DEBUG commands using the /CREF switch. The file containing the names of these CREF-listing files is then deleted so that subsequent CREF commands will not list them again.	m L R D	See Table 2-4
COMPILE list	COM	Produces relocatable binary file(s) for the specified program(s). The use of the MACRO assembler and/or the FORTRAN IV compiler is determined as follows. Condition / Action If no .REL (binary) file — Translate source file If source-file [date, time] is later than binary-file [date, time] — Translate source file If other than above — Do not translate source file; use current .REL (binary) file. Source File Extension / Translator Used .MAC — MACRO assembler .F4 — FORTRAN IV compiler (F40) Other than above, or null — "Standard processor" is used (see 2.6.2). The list of files which may be a single file specification, or a string of file specifications separated by commas. A file specification consists of a filename (with or without an extension) and may include a device name (if the source file is not disk) or a project-programmer number (if the source file is not in the user's disk area). Examples: PROG1,PROG1.MAC,PROG1.F4,PROG1.XYZ,DTAO:PROG1 PROG1[10,16],PROGA,DTAO:PROGB PROGC.MAC (See 2.6.1, 2.6.2)	m L R D	See Table 2-4

Commands	Abbreviation	Explanation	Characteristics*	Monitor Messages
LOAD list	LDA	Performs the CDMPILE function for the specified program(s), then runs LOADER and loads the .REL files.	m L R D	See Table 2-4
EXECUTE list	EX	Performs the CDMPILE and LDAD functions for the specified program(s) and begins execution of the loaded program.	u L R D	See Table 2-4
DEBUG list	DEB	Performs the CDMPILE and LDAD functions and, in addition, prepares for debugging. DDT (the Dynamic Debugging Technique program) is loaded first, followed by the user's programs with local symbols. DDT is entered on completion of loading. Examples: COMPILE PROGA EXECUTE DTA1:TEST.MAC DEBUG/L FILEA,FILEB,FILEC/N,FILED	u L R D	See Table 2-4
		(Generate listings for FILEA,FILEB,and FILED; see 2.6.2) LOAD FILEA,FILEB,%60000FILEC (Pass origin switch to LOADER; see "Loader-Switches" 2.6.4)		

*Refer to footnote in Table 2-1

Each time a COMPILE, LOAD, EXECUTE, or DEBUG COMMAND is executed, the command with its arguments is "remembered" as a file on the disk. Because of this, the filename last used may be recalled for the next command without specifying the arguments again. (See last paragraph in Section 2.5)

2.6.1 Extended Command Forms

The commands shown in Table 2-5 are adequate for the compilation and execution of a single program or a small group of programs at one time. However, the assembly of large groups of programs, such as the FORTRAN library or the Time-Sharing

Monitor, is more easily accomplished by means of one or more of the extended command forms.

2.6.1.1 The @ File

When there are many program names and switches, they can be put into a file so that they 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 of the commands in Tables 2-3 and 2-5.

The "@ file" must appear at any point after the first word in the command. In this construction "file" must be a file-name, which may have an extension and project-programmer numbers. 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 command file specified 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 themselves may contain the "@ file" construction to a depth of nine levels. If this indirecting process should result in files pointing in a loop, the maximum depth will rapidly be exceeded and an error message will be produced.

The following rules are used in the handling of 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) The characters CARRIAGE RETURN,LINE-FEED, AND ALTMODE are ignored if the first non-blank character after a sequence of returns, line-feeds, and altmodes is a comma. Otherwise, they are treated either as commas by the COMPILE, LOAD, EXECUTE, and DEBUG commands or as command terminators by all the other commands appearing in Tables 2-3 and 2-5.

c) Blank lines are completely ignored since 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 through the line-feed is ignored.

e) If command files are sequenced, the sequence numbers are ignored.

2.6.1.2 The "+" Construction[1]

A single relocatable binary file may be produced from a collection of input source files by means of the "+" construction. For example, a user may wish to compile the parameter file, S.MAC, the switch file, FT50SB.MAC, and the file that is the body of the program, APRSER.MAC. This is specified by the following command:

```
COMPILE S + FT50SB + APRSER
```

The name of the last input file in the string is given to any output (.REL and/or .LST) files (e.g., APRSER. in the foregoing example). The source files in the "+" construction may each con-

[1]Used in COMPILE, LOAD, EXECUTE, and DEBUG commands only.

tain device and extension information and project-programmer numbers.

2.6.1.3 The "=" Construction[1]

Usually the filename of the binary file is the same as that of the source file, with the extension specifying the difference. This can be changed by use of the "=" construction, which allows a filename other than the source filename to be given to the output file. For example, if a binary file is desired with the name BINARY.REL from a source program with the name SOURCE.MAC, the following command is used.

```
COMPILE BINARY = SOURCE
```

This same technique may be used to specify an output name to a file produced by use of the "+" 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
```

2.6.1.4 The "<>" Construction[1]

The "<>" construction causes the programs within the angle brackets to be assembled with the same parameter file. If a + 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 S.MAC, the user may type

```
COMPILE S + LPTSER, S + PTPSER, S + PTRSER
```

But by using the angle brackets, the command becomes

```
COMPILE S + <LPTSER,PTPSER,PTRSER>
```

The user cannot type

```
COMPILE <LPTSER,PTPSER,PTRSER> + S
```

[1]Used in COMPILE, LOAD, EXECUTE, and DEBUG commands only.

2.6.2 Compile Switches[1]

The COMPILE, LOAD, EXECUTE, and DEBUG commands may be modified by a number of switches. Each switch is preceded by a slash (/SWITCH) and is terminated by any non-alphanumeric character, usually a space or a comma. An abbreviation may be used as long as it uniquely identifies a particular switch.

These switches may be either temporary or permanent. A temporary switch is appended to the end of the filename, without an intervening space, and applies only to that file.

Example: COMPILE A,B/MACRO,C (The MACRO assembler applies only to file B)

A permanent switch is set off from filenames by spaces, commas or any combination of the two. It applies to all the following files unless modified by a subsequent switch.

Example:

```
COMPILE A,/MACRO,B,C
COMPILE A /MACRO B,C
COMPILE A,/MACRO B,C
```

2.6.2.1 Compilation Listings

Listing files may be generated by the use of switches. The listings may be of the ordinary or the dross-reference type. The operation of the switch produces a disk file with the extension .LST.

The compile-switches "LIST" and "NOLIST" cause listing and non-listing of programs. These switches may be used as either temporary or permanent switches.

[1]Used in COMPILE, LOAD, EXECUTE, and DEBUG commands only.

```
COMPILE /LIST A,B,C
```

will generate listings of all three programs.

```
COMPILE A/LIST, B,C
```

will generate a listing only of program A.

```
COMPILE /LIST A, B/NOLIST, C
```

will generate listings of programs A and C.

The compile-switch "CREF" is just like "LIST", except that a cross-reference listing is generated, to be processed later by the program "CREF".

Unless the /LIST or /CREF is specified, no listing file is generated. The LIST command is used to obtain printer output of regular listing files and the CREF command to obtain printer output of CREF listing files.

Since the "LIST", "NOLIST", AND "CREF" switches are so commonly used, the switches "L", "N", and "C" are defined with the corresponding meanings, even though there are (for instance) other switches beginning with the letter "L". Thus the command

```
COMPILE /L A
```

produces a listing file "A.LST" (as well as, of course, "A.REL").

2.6.2.2 The "Standard Processor"

The "standard processor" is used to compile or assemble programs which do not have the extensions .MAC, .F4, or .REL. There are a number of switches for setting the "standard processor". If all source files are kept with the appropriate extensions, this subject can be disregarded.

If the command

```
COMPILE A
```

is executed and there is a file named "A." (that is, with a blank

extension), then "A." will be translated to "A.REL" by the "standard processor". Similarly, if the command

```
COMPILE FILE.NEW
```

is executed, the extension ".NEW", although meaningful to the user, does not specify a language, so the "standard processor" will be used. For these cases the user must be able to control the setting of the "standard processor".

The "standard processor" is FORTRAN IV at the beginning of each command string.

The "standard processor" may be changed by the following compile-switches:

MACRO	change standard to MACRO
M	same as MACRO
FORTRAN	change standard to FORTRAN IV
F	same as FORTRAN
REL	change standard to use RELocatable binary; i.e., use existing .REL files, even though a newer source file may be present. (Useful primarily in LOAD, EXECUTE, DEBUG commands).

These switches may be used as "temporary" or "permanent". For example, assume that programs A, B, and C exist on the disk, with blank extensions. Then

```
COMPILE A, B/M, C
```

will cause A and C to be translated by FORTRAN, B by MACRO.

```
COMPILE A, /M B, C
```

will cause A to be translated by FORTRAN, B and C by MACRO.

NOTE

Programs with .MAC and .F4 extensions are always translated by the extension implied, regardless of the "standard processor."

2.6.2.3 Forced Compilation

The compilation (or assembly) occurs if the source file is at least as recent as the relocatable binary file. If the binary is newer than the source, there is not normally any need to perform the translation.

There are cases, however, where such extra translation may be desirable, as for instance, when one desires a listing of the assembly. To force such an assembly, the switch "COMPILE" is provided, again in both temporary and permanent form. For example:

COMPILE /CREF / COMPILE A, B, C

will create cross-reference listing files A.LST, B.LST, and C.LST, even though current .REL files may exist. In fact, the binary files will also be recreated.

The corresponding switch "NOCOMPILE" is also provided, to turn off the forced-compile made. Note that this differs from the /REL switch which turns off even the normal compilation caused by a source file newer than the .REL file.

2.6.2.4 Library Searches

The LOADER normally performs a library search of the FORTRAN library. Sometimes it is necessary to search other files as libraries. To do this, the compile-switches "LIBRARY" and (its complement) "NOSEARCH" are provided.

These switches may be used as either "permanent" or "temporary".

For example, suppose a special library file named SPCLIB.REL were kept on device SYS at a particular installation. Then to compile and load a user program, library search the

special library, and then search the normal FORTRAN library, the following command could be used:

```
LOAD MAIN,SYS:SPCLIB/LIB
```

At this point, it should be noted that the program SPCLIB is not assembled simply because its source file is presumably not on device SYS. The COMPILE process will compile any program named in the command string, if its source is present and not older than the .REL file, unless prevented by the /REL switch.

2.6.2.5 Loader Maps

Loader maps are produced during the loading process by the compile-switch "MAP". When this switch is encountered, a loader map is requested from the Loader. The map will be written with filename MAP.MAP, in the user's disk area.

This compile-switch is the one exception to the "permanent compile-switch" rule, in that it causes only one map to be output, even though it may appear as a permanent switch.

2.6.3 Processor Switches [1]

Occasionally it is necessary to pass switches to the assembler or compiler. Recall that for each translation (assembly or compilation), a command string is sent to the translator containing three parts: the source files, a binary output file, and a listing file. If the user wishes to add switches to those files, he must do so as follows:

a) If the "+" construction is used, group the switches according to each related source filename.

b) Group the switches according to the three types of files (source, binary, and listing) for each source filename.

[1]Used in COMPILE, LOAD, EXECUTE, and DEBUG commands only.

c) For each source filename, separate the groups of switches by commas.

d) Enclose all the switches for each source filename 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 filename to which it refers.

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 PDP-6(S), and eleminate the MACRO coding from 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 comma must appear.

2.6.4 Loader Switches[1]

In unusually complex loading processes, it may be necessary to pass loader-switches to the LOADER to direct its

[1]Used in COMPILE, LOAD, EXECUTE, and DEBUG commands only.

operation. These are passed via the COMPILE, LOAD, EXECUTE, and DEBUG commands. These switches must be passed to the LOADER (not to the compiler or assembler). This is accomplished by the % character. The % has the same meaning as that of the / in the Loader's command string. Also, like the /, it takes 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, %60000C,D
```

The most commonly used switches are:

%S Load with symbols

%nO Set program origin to n

%F Cause early search of FORTRAN library

%P Prevent FORTRAN library search

2.6.5 Temporary Files

The COMPIL CUSP deciphers the commands found in Tables 2-3 and 2-5 and constructs new commands for the CUSPS that were referenced. These new commands are written as temporary files on the disk, as are all of the Monitor-level commands. COMPIL and the other CUSPS transfer control directly to one another without requiring additional typed-in commands from the user.

Temporary filenames have 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. In the filenames listed below, job number 1 will be assumed.

2.6.5.1 001SVC.TMP

This file contains the most recent COMPILE, LOAD, EXECUTE, or DEBUG command which included arguments. It is used to remember those arguments. See section 2.6.

2.6.5.2 001EDS.TMP

This file contains the most recent EDIT, CREATE, TECO, or MAKE command which included an argument. It is used to remember that argument. See section 2.5

2.6.5.3 001MAC.TMP

This file contains commands to MACRO. It is written by COMPIL, and ready by MACRO. It contains one line for each program to be assembled, and (if required) the command

NAME!

to cause MACRO to transfer control to the named CUSP ("name" may be F40, LOADER, etc.).

2.6.5.4 001FOR.TMP

This file corresponds exactly to the one described in the preceding section, except that it is read by the FORTRAN IV compiler, F40.

2.6.5.5 001PIP.TMP

This file is written by COMPIL and read by PIP. It contains ordinary PIP commands to implement the DIRECTORY, LIST, TYPE, RENAME, and DELETE commands.

2.6.5.6 001CRE.TMP

This file is written by COMPIL and read by CREF. It contains commands to CREF corresponding to each file which has produced a CREF listing on the disk.

COMPIL also reads this file, if it exists, each time a new CREF listing is generated, to prevent multiple requests

for the same file, and to prevent discarding other requests which may not yet have been listed.

2.6.5.7 001EDT.TMP

This file is written by COMPIL for each EDIT, CREATE, TECO, or MAKE command, and is read by either the LINED or TECO CUSP.

For the commands MAKE or CREATE, it contains the command

Sfile.ext (ALTMODE)

For the commands TECO or EDIT, it contains the command

Sfile.ext (RETURN) (LINEFEED)

2.7 RUN CONTROL COMMANDS

By using a run control command, 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 console. Files stored on disk and DECtape are addressable by name. Files on magnetic tape require the user to preposition the tape to the beginning of the file.

Taole 2-6

Monitor Commands to Call, Load, and Control Programs

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
RUN dev file.ext [proj,prog] core	RU	To load a core image from a retrievable storage device and start it at the location specified within the file (JOBSA). If the program has two segments, both the low and high segments will be set up. If the high file has extension .SHR (as opposed to .HGH), the high segment will be shared. A two-segment program may have a low file extension (.LOW). dev — The logical or physical name of the device containing the core image. file.ext — The name of the file containing the core image; if .ext is omitted, it is assumed to be SHR + LOW, HGH + LOW, or SAV. See SAVE, SSAVE. [proj.prog] — Project-programmer number; required only if core image file is located in a disk area other than the user's. core — Amount of core to be assigned if different from minimum core needed to load the the program or from the core argument of the SAVE command which saved the file. Since previous core is returned, MTA must have this argument because there is no directory to tell how much core for low segment.	u L J	dev: NOT AVAILABLE The device has been assigned to another job. NO SUCH OEVICE The device does not exist. nK OF CORE NEEOEO There is insufficient free core to load the file. NOT A DUMP FILE The file is not a core image file. TRANSMISSION ERROR A parity or device error occurred during loading.
R file.ext core	R	Same as RUN SYS: file.ext core. The R command is the usual way to run a CUSP that does not have a direct Monitor command to run it.	u L J R	Same as RUN
GET dev file.ext [proj,prog] core	G	Same as RUN command except that Monitor types out JOB SETUP and does not start execution.	m L J A	Same as RUN

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
START adr	ST	Begins execution of a program previously loaded with the GET command. adr — The address at which execution is to begin if other than the location specified within the file (JOBSA). If adr is not specified, the starting address comes from JOBSA.	u L J C A	NO CORE ASSIGNED No core was allocated to the user when the GET command was given and no core argument was specified in the GET. NO START ADR Starting address was 0 because ueer failed to specify a starting address in END statement of source program.
HALT (↑C)	↑C	Places the console in Monitor mode and transmits a HALT command to the Monitor Command Interpreter. Stops the job and stores the program counter in the job data area (JOBPC).	m	
CONT	CON	Starts the program at the saved program counter address stored in JOBPC by a HALT command (↑C) or a HALT instruction.	u L J C	CAN'T CONTINUE The job was halted due to a Monitor-detected error and can't be continued.
DDT	DD	Copies the saved program counter value from JOBPC into JOBOPC and starts the program at an alternate entry point specified in JOBDDT (beginning address of DDT as set by Linking Loader). DDT contains commands to allow the user to start or resume at any desired address	u L J C	NO START ADR DDT starting address was 0 (JOBDDT).
REENTER	REE	Similar to the DDT command. Copies saved program counter value from JOBPC into JOBOPC and starts program at an alternate entry point specified in JOBREN (must be set by the user or his program).	u L J C	NO START ADR REENTER starting address was 0 (JOBREN).
E adr	E	Examines a core location in the user's area (high or low segment). adr — If this argument is specified, the contents of the location are typed out in half-word octal mode. Adr is required the first time the E or D command is used. If adr is not specified, the contents of the location following the previously specified E adr or the location of the previous D adr are typed out.	m L J C	OUT OF BOUNDS The specified adr is not in the user's core area, or the user does not have read privileges to file which initialized the high segment.

Command	Abbreviation	Explanation	Characteristics*	
D lh rh adr	D	Deposits information in the user's core area (high or low segment). lh — The octal value to be deposited in the left half of the location. rh — The octal value to be deposited in the right half of the location. adr — The address of the location into which the information is to be deposited. 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.	m L J C	OUT OF BOUNDS The specified adr is not in the user's core area, or high segment is write protected and user does not have write privileges to file which initialized the high segment.
SAVE dev file.ext core	SA	Writes out a core image of the user's core area on the specified device. Saves any user program (reentrant, one segment non-reentrant, or two segment non-reentrant) as one or two files. Later when the program is loaded by a GET, R, or RUN command, it will be non-reentrant. 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 JOBFF) is written. dev — The device on which the core image file is to be written. file.ext — The name to be assigned to the core image file. 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. core — Amount of core in which the program is to be run. This value is stored in the job core area (JOBCOR) and is used by the RUN and GET commands. Specified as number of 1K blocks.	m L J C A	n 1K BLOCKS OF CORE NEEDED The user's current core allocation is less than the contents of JOBFF. DEVICE NOT AVAILABLE Device dev is assigned to another user. TRANSMISSION ERROR An error was detected while reading or writing the core image file. DIRECTORY FULL The directory of device dev is full; no more files can be added. JOB SAVED The output is completed.

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
		If core is omitted, only the number of blocks required by the core image area (as explained above) is assumed.		
SSAVE dev file.ext core	SSA	Same as SAVE except that the high segment 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 CUSP's without having to know which are sharable.	m L J A C	

*Refer to footnote in Table 2-1

2.7.1 Additional Information on SAVE and SSAVE

Low segment files will be zero compressed on all devices (DTA,MTA,DSK), but high segment files will not since the high segment may be shared at the time of the command. Saved files are ordinary binary files and can be copies using the /B switch in PIP.

In order to save file space, only the high segment up through the highest location (relative to high segment origin) loaded, as specified in the LH of JOBHRL, will be written by the SAVE command. If LH is zero (high segment created by CORE or REMAP UUO) or DDT is present, the entire high segment will be written.

It is possible for most programs to be written so that only the high segment contains non-zero data. This will also save file space and I/O time with the GET command. SAVE will write the high segment (.HGH) only. The LOADER will indicate to the SAVE command that no data was loaded above the Job Data area in the low segment by setting the LH of JOBCOR to the highest location loaded in the low segment with non-zero data.

There are a number of locations in the Job Data area which need to be initialized on a GET, even though there is no other data in the low segment. The SAVE command copies these locations into the first 10_8 locations of the high segment, provided it is not sharable. These 10 locations are referred to as the Vestigial Job Data area. 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. Both commands also delete a file 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 do a GET, R, or RUN command. These commands will 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 will always enable the hardware user-mode write protect to prevent the user program from storing into the segment inadvertently.

To debug a reentrant CUSP which is in the system directory, the user should make a private, non-sharable copy, rather than modifying the shared version and possibly causing

harm to other users. To make a private, non-sharable copy, the following commands are used.

a) GET SYS CUSP

b) SAVE dev CUSP — Writes a file in the user directory as non-sharable. The high segment in the user's addressing space remains sharable.

c) GET dev CUSP — Overlays the sharable program with the non-sharable one from the user's directory. Now the user can make patches while other users share the version in the system directory.

The Monitor will keep the shared and the non-shared versions separate from each other. A sharable program may be superseded into the directory by the SSAVE command. The Monitor will clear the high segment in its table of storable segments in use but will 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 UUO will 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 will not modify the Vestigial Job Data area unless the user has write privileges to the file which initialized the shared segment. This prohibits unauthorized users from modifying the first 10 locations of a shared segment. This restriction does not exist if a low file is also written, since the GET command reads the low file after the high file. The real Job Data area locations are set from the low file.

2.8 BACKGROUND JOB CONTROL COMMANDS

A job is a background, or detached, job if it is not under control of a user console. Any console can initiate any number of background jobs. I/O to the console while a job is running in a background mode causes the job to stop until a console is attached.

Table 2-7

Monitor Commands to Control Background Jobs

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
PJOB	PJ	Monitor responds by typing the job number to which the user's console is attached. 10/40 System - If the console is not attached to a job, Monitor assigns a job number and types the job number and a line identifying the Monitor version. 10/50 System - If the console is not attached to a job, Monitor responds with LOGIN PLEASE.	m L J	
CSTART CCONT	CS CC	Identical to the START and CONT commands, respectively, except that the console is left in the Monitor mode. To Use: 1. Begin the program with the console in user mode. 2. Type control information to the program, then type ↑C to halt job with console in Monitor mode. 3. Type CCONT to allow job to continue running and leave console in Monitor mode. 4. Further Monitor commands can now be entered from the console. Caution: These commands should not be used when the user program (which is continuing to run) is also requesting input from the console.	m L J C	Same as START and CONT.
DETACH	DET	Disconnects the console from the user's job without affecting the status of the job. The user console is now free to control another job, either by initiating a new job or attaching to a currently running background job.	d L	

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
ATTACH job	AT	Connects a console to a background job. job — The job number of the job to which the console is to be attached. [proj,prog] — The project-programmer number of the originator of the desired job. May be omitted if same as job to which console is currently attached. The operator (device OPR) may always attach to a job even though another console is attached, provided he specifies the proper [proj,prog].	m	If an error message occurs, the console remains attached to its current job. TTYn ALREADY ATTACHED Job number typed is erroneous and is attached to another console, or another user is attached to the job. NOT A JOB The job number is not assigned to any currently running job. CAN'T ATTACH TO JOB The project-programmer number entered is not that of the originator of the desired job.

*Refer to footnote in Table 2-1

2.9 JOB TERMINATION COMMANDS

When a user leaves the system, all facilities allocated to his jobs must be returned to the Monitor facility pool so that they are available to other users.

Table 2-8

Monitor Command to Terminate Jobs

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
KJOB	K	In Multiprogramming Systems: Stops all allocated 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: All of the above procedures. In addition, if user has any files, responds with: CONFIRM:	m A	

Command	Abbreviation	Explanation	Characteristics* Monitor Messages
		To which the user may type ↑C to abort log-out; or type one of the following: K) to kill job and delete all unprotected files; L) to list his disk directory; I) to individually save and delete files as follows: After each file name is listed, type: P to save and protect, S to save without protecting, or) to delete. Files with extensions.LST and .TMP will be deleted automatically.	

*Refer to footnote in Table 2-1

2.10 SYSTEM TIMING COMMANDS

All system times are kept in increments of one-sixtieth or one-fiftieth of a second, depending on the power frequency of the country in which the PDP-10 is installed.

Table 2-9

Monitor Commands for System Timing

Command	Abbreviation	Explanation	Characteristics* Monitor Messages
DAYTIME	DA	Types the date followed by the time of day. Time is typed in the format. hh:mm where hh = hours mm = minutes	m
TIME job	TI	Types out 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 format hh:mm:ss.hh where hh = hours mm = minutes ss.hh = seconds to nearest hundredth.	m

Command	Abbreviation	Explanation	Characteristics* Monitor Messages
		Interrupt level and job scheduling times are charged to the user who was running when the interrupt or rescheduling occurred. job — The job number of the job whose timing is desired. If job is omitted, the job to which the console 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. If job = 0, an approximation of the time spent core shuffling (SHFL) is printed, followed by the amount of time spent clearing core (ZCOR), the running time of the null job (NULL), the time during which one or more jobs wanted to run but were swapped out or in the process of being swapped out (LOST), and the total time system has been up (UP).	

*Refer to footnote in Table 2-1

2.11 SYSTEM ADMINISTRATION COMMANDS

The SYSTAT command permits a user to learn how heavily the system is loaded and the status of devices in the sharable device pool. The other commands in this section are restricted to system administrators only.

Table 2-10

Monitor Commands for System Administration

Command	Abbreviation	Explanation	Characteristics*	Monitor Messages
SCHEDULE n	SCH	Changes the scheduled use of the system, depending on n. This command is legal only from the operator's console. n is stored in RH of STATES word in COMMON: 0 = regular time sharing. 1 = no further LOGINS allowed. 2 = no further LOGINS from remote TTY's. If n is omitted, the current value of n is printed.	m	
SYSTAT	SYS	Types out status of the system: system name, time of day, date, uptime, percent null time. Status of each job: job number, project-programmer number (**** if detached), TTY number, program name being run, size of low segment, state (RN = runable, TT = TTY input wait, C = Monitor command mode) and run time. Status of high segments being used: name, directory name, size, number of users in core or on disk. Status of each assigned device: name, job number, how assigned (AS = ASSIGN command, INIT = INIT UUO).		
ASSIGN SYS:dev		To change the systems device to device "dev." The user must be logged in under either [1,1] or [1,2].	m L J	
DETACH dev	DET	To assign the device "dev" to JOB 0, thus making it unavailable. The user must be logged in under [1,1].	m L J	
ATTACH dev	AT	To return a detached device to the Monitor pool of available devices. The user must be logged in under [1,1].	m L J	
CTEST		This command is used by system programmers to test extensions made to the COMPIL CUSP.	L R	

*Refer to footnote in Table 2-1

2.12 MONITOR DIAGNOSTIC MESSAGES

Once a user program has been started, a number of error conditions may arise which cause the job to revert to monitor mode. The error messages typed, and the meanings for each are summarized in the following table.

Table 2-11

Time-Sharing Monitor Diagnostic Messages

Message	Meaning
The typein is typed back followed by ?)	The Monitor command decoder has encountered an incorrect character, such as a letter in a numeric argument. The incorrect character appears immediately before the ?. Example: User types in: CORE ABC Monitor responds: CORE A ?)
ADDRESS CHECK FOR DEVICE dev AT USER adr	Monitor has checked a user address and has found it to be too large (>C(JOBREL)) or too small (<JOBPFI). Some user addresses can be the user's accumulators while others cannot. One of the following addresses may be wrong: buffer buffer header dump mode command list data specified by dump mode command list insufficient core available for setting up Monitor-generated buffers.
BAD DIRECTORY FOR DEVICE -DTAn; UUO AT USER adr	The DECtape directory is not in proper format or had a parity error when read. Many times this error occurs when an attempt is made to use a virgin tape.
DEVICE dev OK?	Device dev is temporarily in an inoperable state, such as LPT off-line. The user should correct the obvious condition and then type a CONT command.

Table 2-11 (Cont)

Time-Sharing Monitor Diagnostic Messages

Message	Meaning
ERROR IN JOB n	A fatal error has occurred in the user's job (or in Monitor while servicing the job). This typeout is normally followed by a 1-line description of the error.
HALT AT USER adr	The user program has executed a halt instruction at loc. adr. Typing CONT will resume execution at the effective address of the halt.
HUNG DEVICE dev; UUO AT USER adr	A device has not generated an interrupt for a timed period and, therefore, is in need of attention.
ILLEGAL DATA MODE FOR DEVICE dev AT USER adr	The data mode specified for a device in the user's program is illegal.
ILLEGAL UUO AT USER adr	An illegal UUO has been executed at user location adr.
ILL INST. AT USER adr	An illegal operation code has been encountered in the user's program.
ILL MEM REF AT USER adr	An illegal memory reference has been made by the user program at adr or adr+1.
INCORRECT RETRIEVAL INFORMATION: UUO AT USER adr	The retrieval pointers for a file are not in the correct format; the file is unreadable. If this typeout occurs, the user should report it on a Software Trouble Report.
INPUT DEVICE dev CANNOT DO OUTPUT; UUO AT USER adr	An illegal OUTPUT UUO has been executed at user location adr.
I/O TO UNASSIGNED CHANNEL AT USER adr	No OPEN or INIT was performed on the channel.
LOOKUP AND ENTER HAVE DIFFERENT NAMES: UUO AT USER adr	An attempt has been made to read and write a file on the disk. However, the LOOKUP and ENTER UUO's have specified different names on the same user channel. This message does not indicate a DECtape error.

Table 2-11 (Cont)

Time-Sharing Monitor Diagnostic Messages

Message	Meaning
MASS STORAGE DEVICE FULL; UUO AT USER adr	The storage disk is full. Users must delete unneeded files before the system can proceed.
NON-RECOVERABLE DISC READ ERROR; UUO AT USER adr	Monitor has encountered an error while reading or writing a critical block in the disk file structure (e.g., the MFD or the SAT table). If this condition persists, the disk must be reloaded using Fail-safe after the standard location for the MFD and SAT table has been changed using the Monitor once-only dialogue.
NON-RECOVERABLE DISC WRITE ERROR; UUO AT USER adr	
NOT ENOUGH FREE CORE IN MONITOR: UUO AT USER adr	The Monitor has run out of free core for assigning disk data blocks and Monitor buffers. If this type-out occurs, the user should report it on a Software Trouble Report.
NOT FOUND	The program file requested cannot be found on the systems device (or on the specified device).
OUTPUT DEVICE dev CANNOT DO INPUT; UUO AT USER adr	An illegal INPUT UUO has been executed at user location adr.
PC EXCEEDS MEMORY BOUND AT USER adr	An illegal transfer has been made by the user program to user location adr.
SWAP READ ERROR	A consistent checksum error has been encountered when checksumming locations JOBDAC through JOBDAC+74 of the Job Data area during swapping.

5.1.1 Data Modes

5.1.1.1 Full-Duplex Software A(ASCII) and AL(ASCII Line)

The input handling of all control characters is as follows. (All are passed to program except as noted below).

000	NULL	Ignored on input, suppressed on output.
001	↑A	Echoes as ↑A. Passed to program.
002	↑B	Complements switch controlling echoing, not passed to program. Used on local-copy dataphones and TWX's.
003	↑C	The Teletype mode is switched to Monitor mode the next time input is requested by the program. Two successive ↑C's cause the mode to be switched to Monitor mode immediately.
004	↑D (EOT)	004 passed to program. Not echoed, so typing in a "Control D" (EOT) will not cause a full duplex dataphone to hang up.
005	↑E (WRU)	No special action.
006	↑F	Complements switch controlling translation of lower case letters to upper case. Used when lower case input is desired to programs. Not sent to program, but program can sense the state of this switch by the TTCALL UUO.
007	↑G (Bell)	007 passed to program, and is a break character.
010	↑H (Back-space)	Acts as a RUBOUT, unless either DDT mode or full character set mode is true, or the ↑F switch is on. In these cases, 010 is sent to the program.
011	↑I (TAB)	011 passed to program. Echoed as spaces if Teletype is a model 33 (determined by ↑P switch). Spaces are not passed to program.
012	↑J (Line-feed)	Is a break character. No other special action.
013	↑K (Vertical Tab)	013 passed to program. Echoes as four linefeeds, if a model 33. Is a break character. Linefeeds are not passed to program.
014	↑L (Form)	014 passed to program. Echoes as 8 linefeeds on a 33. Is a break character. Linefeeds are not passed to program.
015	↑M (Carriage Return)	If Teletype is in paper-tape input mode, 015 is simply passed to program. Otherwise supplies a linefeed echo, and is passed to program as a CR and LF, and is a break character (due to LF).
016	↑N	No special action
017	↑O	Suppresses output until an INPUT, or an INIT, or OPEN UUO occurs. Not passed to program. Typed as ↑O followed by carriage return-linefeed.

020	↑P	Does not appear in the input buffer. Some Teletype units (usually Models 35 and 37) have horizontal tab, vertical tab, and form feed mechanisms while other units (usually Model 33s) do not. If the user finds that his particular Teletype unit does not have these mechanisms, he should type ↑P. Otherwise, tabs will not be printed at all or spaces will be substituted for a tab depending upon the Monitor's assumption.
021	↑Q (XON)	Starts paper-tape-mode, as described above. Passed to program.
022	↑R (TAPE)	No special action.
023	↑S (XOFF)	Ends paper-tape mode, as described above. 023 is passed to program.
024	↑T (NO TAPE)	No special action.
025	↑U	Deletes input line back to last break character. Typed back as ↑U followed by carriage return-linefeed.
026	↑V	No special action.
027	↑W	No special action.
030	↑X	No special action.
031	↑Y	No special action.
032	↑Z	Acts as end-of-file on Teletype input. Echoes as ↑Z followed by carriage return-linefeed. Is a break character. Appears in buffer as 032.
033	↑[(ESC)	This is the ASCII altmode these days, but is translated to 175 before being passed to the program, unless in full character set mode (bit 29 in INIT). 175 is the 1963 altmode. Echoes as a dollar sign. Always, is a break character.
034	↑\	No special action.
035	↑]	No special action.
036	↑↑	No special action
037	↑←	No special action.
040-137		Printing characters, no special action.
140-174		"Lower case" ASCII. Translated to upper case, unless ↑F switch is set. Echoes as upper case if translated to upper case.
175 and 176		Old versions of altmode. See description of "ESC" (033).
177		RUBOUT or DELETE: A) Completely ignored if in papertape mode (XON) B) Is a break character, passed to program if either DDTmode or fullcharacter-set mode is true. C) Otherwise (ordinary case) causes a character to be deleted for each rubout typed. All the characters deleted are echoed between a single pair of backslashes. If no characters remain to be deleted, echoes as a carriage return-linefeed.

DECTAPE EDITOR (EDITOR)

Editor creates, adds to, or deletes from sequentially numbered source files recorded in lines of ASCII characters on a DECtape. Editor edits the source file; (the input and output files are the same). Fresh source files have editing space in each physical DECtape block. If the user has more edits for a block than will fit in it, an extra block in the DECtape is used and appropriately linked to the preceding and following logical blocks of the file. Editor provides a simple method of creating or modifying Macro or FORTRAN IV source programs.

Requirements

Minimum Core:	1K
Additional Core:	Not used
Equipment:	One DECtape unit for the reel containing the file(s) to be modified

Initialization

.R EDITOR)	Loads the DECtape Editor program.
*	Editor is ready to receive a command.

Commands

Initialize a File For Processing

Command	Function
Sn tA)	Select DECtape n and zero the directory.
Sn,filename.ext tA $	Select DECtape n, zero the directory, and create a file called filename.ext.

Command	Function
Sn, filename.ext)	Select DECtape n and locate filename.ext for processing.
Sn, filenome.ext $	Select DECtape n ond add a new file called filenome.ext.

NOTE

All the obave cammands place Editor in the Command mode; i.e., the next typein is assumed to be ane of the commands given below.

Insert a Line

Innnnn)	Insert the following typed line ot line number nnnnn af the currently open file; nnnnn con be specified os a line sequence number or as a paint (.). A point refers to the last line typed. If the line number already exists in the file, the line is reploced.
nnnnn aaao.....o)	
nnnxx $)	
*	

Insert Multiple Lines

Innnnn, increment)	Insert the following typed lines, beginning at line number nnnnn af the currently open file; nnnnn con be specified as a line sequence number or as o point (.). Each time a line is entered, nnnnn is increosed by the specified increment, and the result becomes the line number for the next insertian. Type $ after last line insertian.
nnnnn aaoa...aaa)	
nnnxx bbbb...bbb)	
nnnxx $)	
*	

Delete a Line

Dnnnnn)	Delete line number nnnnn from the currently apen file; nnnnn con be specified as o line sequence number ar os o point (.).
*	

Command	Function
Delete o Series of Lines	
Dmmmmm,nnnnn)	Delete lines mmmmm through nnnnn from the currently open file.
*	
Print o Line	
Pnnnnn)	Print line number nnnnn af the currently open file; nnnnn con be specified as o line sequence number or as o point (.).
nnnnn aaa...aao)	
*	
Print o Series of Lines	
Pmmmmm,nnnnn)	Print lines mmmmm thraugh nnnnn af the currently apen file.
mmmmm oaa...aao)	
.	
.	
nnnnn bbb...bbb)	
*	
Close the Current File	
E) (End af file)	Closes the currently open file. Another file can be opened on the same ar a different DECtape vio on Sn command, or o return can be made to Monitor ta terminate Editar.
*	

Examples

.R EDITOR)	
*S1,VECTOR $	Select DECtape 1 and create o new file on DECtape 1 called VECTOR.
*I20,20)	Begin inserting at line sequence number 20, and increment this number by 20 each time o line is inserted. Switch to Text Mode.

```
00020 DEFINE VMAG (A,B)
00040 <MOVE 0,A)
00060 FMP 0)
00080 MOVE 1,A+1)
00100 FMP 1,1)
00120 FAD 1)
00140 MOVE 1,A+2)
00160 FMP 1,1)
00180 FAO 1)
00200 JSR FSQRT)
00220 MOVEM B)
00240 $ $)
```

Editor responds with first line sequence number. Operator types line of coding to be inserted, followed by a carriage return.

Typing $ terminates insertions and returns Editor to command mode.

```
*I20)
```

Change line number 00020.

```
00020 DEFINE VMAG (A,B,C))
```

```
*ILR*)
```

ILR indicates that the indexing increment has resulted in the next line number being equal to the line number of an already existing line (00040). Note that the indexing increment remains at 20 until explicitly changed.

```
*I90)
```

Insert a line between lines 00080 and 00100.

```
00090 MOVE 1,C)
```

```
*ILS*)
```

ILS indicates that the indexing increment has resulted in an existing line (00100) being skipped, since the next line addressed would be 00110.

```
*D180)
```

Delete line 00180.

```
*P20,220)
```

Print lines 00020 through 00220.

```
00020 DEFINE VMAG (A,B,C))
00040 <MOVE 0,A)
00060 FMP 0)
00080 MOVE 1,A+1)
00090 MOVE 1,C)
00100 FMP 1,1)
00120 FAO 1)
00140 MOVE 1,A+2)
00160 FMP 1,1)
00200 JSR FSQRT)
00220 MOVEM B)
```

```
*E)
```

Close the currently open file.

```
*^C)
```

Return to the Monitor.

Editar Diagnostic Messages

Message	Meaning
?DDE*	Device Data Error due to o write errar ar WRITE LOCK switch. Editar must be restarted.
?DEC*	DECtape directary is full.
?FAU*	Filename Already Used. A filename assigned to o new file already exists on the DECtape.
?ILC*	Illegal Cammand.
ILR *ILS*	Insert Line Replocement, Insert Line Skip. The line sequence increment specified for the insert functian will couse the next existing line to be either reploced (R) or skipped (S). This is o warning message anly ond does nat necessarily indicate an error.
?NCF*	Not a Current File.
?NFO*	No File Open. A cammond requiring on octive file has been given but na file is currently apen.
?NLN*	Nonexistent Line Number. A print (P) or delete (D) cammand refers ta a nonexistent line.
?UNA*	Unit Nat Availoble. The DECtape specified in an Sn cammand is assigned to another jab.

LINE EDITOR FOR DISK (LINED)

LINED is similar to DECtape Editor, but operates on disk files instead of DECtape files. LINED is called in by typing

```
.R LINED
*
```

LINED responds with an asterisk to indicate it is ready to accept a command string.

Commands

LINED commands are very similar to those of Editor; consequently, only a brief summary is given here.

Command	Function
S filename.ext)	Select an existing file for editing.
S filename.ext $	Select (create) a new file for editing.
Innnnn)	Insert a line at line number nnnnn.
Innnnn,increment)	Insert multiple lines, starting at line number nnnnn and incrementing the line number each time.
Dnnnnn)	Delete the line at line number nnnnn.
Dmmmmm,nnnnn)	Delete lines mmmmm through nnnnn.
Pnnnnn)	Print line number nnnnn on the Teletype.
Pmmmmm,nnnnn)	Print lines mmmmm through nnnnn on the Teletype.
E)	End (close) the current file.
$	If in Insertion Mode, ignore current text line and return to LINED command level. If in Command Mode, print the next line.

NOTE

Files are written with standard protection (055). All blocks are assumed to have integral number of lines. Use the /A switch (line blocking) with PIP to put each file on disk.

Diagnostic Messages

The diagnostic messages for LINED are similar to those for Editor. The diagnostic message ?DEC* is not included, and the following message is added:

Message	Meaning
?CCL*	CCL error. Error occurred while referencing CCL command file.

Monitor Commands

To call in LINED and open a new file for creation:

Monitor Command	Equivalent CUSP Commands
.CREATE filename.ext)	.R LINED) *S filename.ext $

To call in LINED and open an existing file for editing:

Monitor Command	Equivalent CUSP Command
.EDIT filename.ext)	.R LINED) *S filenome.ext

TEXT EDITOR AND CORRECTOR (TECO)*

TECO edits files recorded in ASCII characters an any standard device. It can perform simple editing functians as well as highly saphisticated search, match, and substitute operatians, and aperate upon arbitrary length character strings under contral af cammands which are themselves character strings (and cantains the mechanisms necessary ta explait this recursiveness).

Requirements

Minimum Care:	4K
Additianal Care:	Takes advantage af any additianal care available. Each 1K additional care augments the basic 6,200+ – character buffer by 5K additianal characters.
Equipment	One input device and ane autput device.

NOTE

TECO autamatically requests mare care ta expand its buffer under any of the fallawing canditians:

*PDP-10 TECO is based on PDP-1 TECO, developed at MIT by Daniel Murphy, and PDP-6 TECO, developed at MIT's project MAC by Stewart Nelsan and Richard Greenblatt.

a. An insert by way of the "I" command or "X" (Q Register) will overflow the present memory boundaries.

b. The command acceptance routine needs more core.

c. The total number of characters in the Data Buffer falls below 3000, and an input command from a peripheral device (other than the user console) is executed. Thus, TECO maintains a Data Buffer of at least 3000 characters.

If TECO is successful at obtaining more core, the following message will be typed:

```
*10000 <IJS>$$
[5K CORE]                    nk Core, where n is
                             new core size of job
*
```

If TECO is unsuccessful at obtaining the core request, the following message is typed:

```
STORAGE CAPACITY EXCEEDED
?
*
```

Initialization

```
*R TECO)
*
```

Loads the Text Editor and Corrector program.

TECO is ready to accept a command.

Basic Commands

NOTE

When typing command strings to TECO, the following points should be noted.

$ — One $ is used to terminate the text within a command string, where applicable; two successive $s terminate the entire command string sequence and generate a RETURN, LINE-FEED.

(RUBOUT) — The RUBOUT key can be used to erase the preceding typed-in character(s) of a command string. Each character erased is echoed back on the teletype (e.g., ABD (RUBOUT) DC...). Successive RUBOUTs can be used to erase more than one character.

N.B. To erase a carriage return (which generates RETURN, LINE-FEED), two RUBOUT's are required, one RUBOUT to erase the LINE-FEED and one to erase the RETURN.

Two successive †Gs ((BELL) s) can be used to wipe out the entire command string currently being typed.

TECO commands in the form †x (where x is any character) can be entered by either holding down the CTRL key while striking the x key or typing up-arrow (shift N) followed by the x character. These alternatives are not true where †x is a character within a text string (such as in a Search argument); in this case, the CTRL key must be used.

A carriage return, line feed, ( ) is ignored in a TECO command string as long as it does not appear within a particular command, such as Insert. Examples of this are given on the following pages.

Select The Input Device

Command	Function
ERdev:filename.ext [proj,prog] $	Edit Read. Selects the input device and file (if specified).

dev:* DTAn: (DECtape)
PTR: (paper tape reader)
DSK: (disk)
MTAn: (magnetic tape)
CDR: (card reader)

filename.ext (DSK: or DTAn: only)

[proj,prog] (DSK: only)

NOTE

Device TTY: cannot be used here. See I (Insert) command.

Specified only if file is located in other than user's area.

EBdev:filename.ext $	Edit Backup. Selects an input and file to be edited (the input device, which will also be the output device for the edited file, must be the disk*). EB is intended to be used to keep a backup of a file during a debugging session, without the user having to invent a new name for each version of the file.

*If dev: is not specified, DSK: is assumed.

Command	Function
	For example, the command string sequence EBPROG1.MAC $ . . editing . . EF $$ results in: a. Reading from file PROG1.MAC on disk b. Creating a new output file, which is initially given a temporary name of TECOnn.TMP, where nn is the user's job number in octal; incorporating the job number in the filename solves the problem of identifying temporary files belonging to multiple 100,100 users c. Performing a number of RENAMEs following the EF command so that the input file becomes PROG1.BAK, any previous PROG1.BAK file is deleted, and the new output file becomes PROG1.MAC. An ER command can be given following an EB command and before the EF command, but an intervening EW command is illegal and results in the error message ?22 (see Table 2-5). Even though an ER command may be given, the name of the final output file is still taken from the EB command.

Select The Output Device

EWdev:filename.ext [proj,prog] $	Edit Write. Selects the output device and file (if specified).
EZdev.filename.ext [proj,prog] $	Edit Zero. Selects the output device and file (if specified), and rewinds the tape (if magnetic tape) or zeros the directory (if DECtape).

dev:*		
	DTAn:	(DECtape)
	DSK:	(disk)
	MTAn:	(magnetic tape)
	PTP:	(paper tape punch)
	LPT:	(line printer)

*If dev: is not specified, DSK: is assumed.

Command	Function
	filename.ext (DSK: or DTAn: only) [proj,prog] (DSK: only) Specified only if file is located in other than user's area.

Terminate Output; Close File

EF	End File. Terminate output on the current output file and close the file without selecting a new output file.
EX	Exit. The EX command finishes the current edit operation by writing out all remaining pages of a file, performing an EF (End of File) command, and then exiting to the Monitor. Thus, EX is equivalent to the command string 1000000PEF (BELL)
EG	Exit and Go. The EG command executes an EX command followed by the last Monitor command (COMPILE, LOAD, EXECUTE, or DEBUG) typed by the user. (See Chapter 9.)

Magnetic Tape Positioning

EM	Edit Magtape. Rewind the currently selected input magnetic tape.
nEM	Depending upon the value of n, perform one of the following operations on the currently selected input magnetic tape.

n	Operation
1	Rewind tape to load point.
3	Write end of file.
6	Skip one record.
7	Backspace one record.
8	Skip to logical end of tape.
9	Rewind and unload tape.
11	Erase 3 in. of tape.
14	Advance tape one file.
15	Backspace tape one file.

NOTE

Throughout TECO, all numbers in command strings are interpreted as decimal.

Input Commands

Command	Function
Y	Yank. Read from current input device into buffer until a. A FORM character is read (i.e., a page has been input), or b. The buffer is more than 2/3 full and one of the following is encountered (1) Line Feed (2) Form Feed c. or a point 128 characters from the end of the buffer is reached. NOTE The FORM character, if read, does not enter the buffer. Any data previously residing in the buffer is destroyed. The pointer is positioned immediately before the first character in the buffer. Representative buffer size for 5K TECO: Total buffer capacity = approx. 11,200 characters 2/3 buffer capacity = approx. 7,460 characters 1 line-printer page = 7,200+ characters (120 char./line) (60 lines) 7,800+ characters (130 char./line)
A	Append. Read from the current input device and append the incoming data to information already residing in the buffer. Terminate reading on the same conditions as in Y. NOTE No previous data is destroyed. The pointer is not moved.

Output Commands

PW	Punch, Wait. Output the entire buffer to the selected output device, with a FORM character appended as the last character. Do not alter the contents of the buffer or move the pointer.

Command	Function
P	Equivalent to a PW command followed by a Y command (i.e., output the current contents of the buffer followed by a FORM character, and then read in more data from the input device).
nP	Perform the P command n times.
m,nP	Output the m+1 through the nth character from the buffer to the current output file. Do not append a FORM character at the end. Do not alter the contents of the buffer or move the pointer.

Editing Commands

Move The Pointer

nJ	Jump. Move pointer to the right of the nth buffer character and give the pointer symbol (.) the value of n. If n is omitted, set pointer in front of the first buffer character (same as 0J).
nC	Continue. Set the pointer to the right of the nth character beyond the pointer's present position (equal to .+nJ). If n is omitted, 1 is assumed.
nR	Reverse. Set the pointer to the left of the nth character prior to the pointer's present position (equal to .-n). If n is omitted, 1 is assumed.
nL	Lines. +n – Move the pointer to the right, stopping after it has passed over n LINE-FEED characters. -n – Move the pointer to the left, stopping after it has passed over n+1 LINE-FEED characters, then move to the right of the last LINE-FEED character passed over. If n is omitted, assume 1L.

Delete Text

nD	Delete. Delete n characters. +n – Delete n characters just to the right of the pointer. -n – Delete n characters just to the left of the pointer. If n is omitted, 1 is assumed.
nK	

Command	Function
nK	Kill. +n - Move the pointer to the right, stopping after it has passed over n LINE-FEED characters. Delete all characters the pointer passes over.
	-n - Move the pointer to the left, stopping after it has passed over n+1 LINE-FEED characters, then move it to the right of the last LINE-FEED character passed over. Delete all characters between this point and the pointer's previous position.
	If n is omitted, 1 is assumed.
m,nK	Delete the m+1 through the nth characters of the buffer. Set the pointer where the deletion occurred.

Insert Text

Itext... $	Insert. Insert the text following the "I" up to, but not including, the $ character, beginning at the current pointer position. Move the pointer to the right of the inserted material.
nI	Insert at the pointer location a character with an ASCII code of n (n must be a decimal value). Move the pointer to the right of the inserted character.
n\	Insert at the current pointer location the ASCII text representation of the decimal value of the expression n. Move the pointer to the right of the inserted text.
↵ text... $	Insert at the current pointer location a (↵) character and the following text up to, but not including, the $ character. Move the pointer to the right of the inserted text.
@I/text/	Insert at the current pointer location the text which follows. The text is delimited by a character, /, which can be any character not appearing in the text.

Type Text

NOTE

T commands do not move the pointer.

Command	Function
nT	Type. Type out the string of characters beginning at the current pointer position and terminating after the nth LINE-FEED character is encountered.
	+n – Typeout n lines to the right of the current pointer position.
	-n – Typeout n lines to the left of the current pointer position.
	If n is omitted, the value is assumed to be 1.
m,nT	Type out the m + 1 through the nth characters of the buffer.

Stand-Alone Examples (Elementary)

Open an Input File

ERDTA5:SOURCE.MAC $	Open the input file called SOURCE.MAC located on DTA5.
ERDSK:SRCE.MAC[12,24] $	Open the input file called SRCE.MAC located in area 12,24 on the disk.
ERPTR: $	Open an input file on the paper tape reader.

Open an Output File

EWDTA3:EDITED.MAC $	Open an output file on DTA3 and call it EDITED.MAC.
EZDTA1:DEBUG.MAC $	Zero the directory on DTA1, open an output file on it, and call the file DEBUG.MAC.

Read a Page

Y	Read a page into the buffer from the current input file, destroying the previous contents of the buffer.
A	Read a page into the buffer, appending the data to the end of the information currently in the buffer.

Output Data

Command	Function
PW	Output the entire buffer, followed by a FORM character.
6P	Execute the WRITE and READ cycle six times.
12,50P	Write out the 13th through the 50th characters of the buffer.

Pointer Positioning

Y18J	Read in a page of information and position the pointer after the 18th character of the buffer.
5R	Then, move the pointer left to between characters 13 and 14.

Delete Text

J19C3D or	Move the pointer to the right of the 19th character in the buffer and then delete the next three characters to the right (characters 20, 21, and 22).
19,22K	Delete the 19+1 (20th) through the 22nd characters of the buffer.

Insert Text

J2LITAG: MOVE 1, AMT $	Move the pointer to a position following the second line of the buffer; insert the text TAG: MOVE 1, AMT between the second and third lines of the buffer.
69\	Insert the digits 69 in ASCII at the current pointer position (same as 169 or $ or 541571).
	NOTE Unless a \key is present, \ is typed with a SHIFT L.
→ERROR IN JOB $	Insert a tab followed by the text ERROR IN JOB at the current pointer position.

Command	Function
G1#ERDSK:PROG1#)	Insert the text ERDSK:PROG $ at the current pointer position.

NOTE

The use of delimiters is the only method for inserting a $ in the text.

Typing Text

3T	Type out the first three lines of the buffer.
25,100T	Type out the 25+1 (26th) through the 100th character of the buffer.

Examples (Basic)

*R TECO)	
*ERDTA1:SCFILE.MAC$$)	Open the file called SCFILE.MAC on DTA1 for input.
*EWDTA2:EDFILE.MAC$$)	Open on output file on DTA2 and call it EDFILE.MAC.
*Y0,20T$$)	Read a buffer of information from the input file and type the first 20 characters of the
aaaaa.....aaaoo)	buffer.
*3LT$$)	Move the pointer to the right, stopping when three LINE-FEED characters have been en-
bbbbb.....bbbbb)	countered; type the text of the fourth line in the buffer.
*ITHIS IS A SAMPLE INSERT)	Insert the text THIS IS A SAMPLE INSERT)
$$)	between the third and fourth lines of the buffer, and position the pointer after the inserted material.
*10PT$$)	Write out the current buffer to the output device; read in and write out the next nine
ccccc.....ccccc)	pages of data; read in the 11th page of data and position the pointer at the beginning of the buffer; type out the first line of the buffer.
*0$$)	Delete this line from the file; position the pointer at the beginning of the (now) first line in the buffer.

Command	Function
*EX$$)	Writes out all remaining pages of the file, performs an EF (End of File) command, and exits to the Monitor.
EXIT) ↑C)	Kill the job, deassign all devices, release core.

Advanced Commands

Search Commands

Summary

S text $ (Search) - Searches for text in current buffer only.

N text $ (Nonstop Search) - Searches for text through successive buffers by repeatedly writing out current buffer and reading in next buffer (similar to P command, but a form character is not inserted after outputting each buffer).

-text $ - Searches for text through successive buffers by repeatedly reading in new bufferful of information (Y command).

NOTES

All searches begin at the current location of the pointer.

Each search command can be preceded by the modifier characters (: and/or @).

: causes the search command to have a numeric value at completion;

0 if the search has failed (the requested text was not found) or -1 if the search was successful (the requested text was found).

@ indicates that the text to be matched is delimited by some character (same as in the @I command).

A numeric argument can appear following the modifiers (if any) but must precede the command. If the numeric argument is n, TECO searches for the nth occurrence of the text. If n is not used, the value of n is assumed to be 1.

If search is successful, the pointer is positioned to the right of the matched text. If the search fails, the pointer is positioned at the beginning of the buffer.

Use of speciol chorocters within text:

↑S – Match any separator character (any chorocter nat a letter, number, periad, dallar sign, or percent symbol).

↑X – Match any (arbitrory) character. Used when the cantents of some position within the text is unimportant.

↑Nx – Match any character except x.

↑Q – Takes the next character literolly, even if it is ane af faur speciol characters. For example, S ↑Q↑X $ – Find the chorocter ↑X.

See note under TECO, Basic Commands.

Search Commands Summary

Command	Action at End of Buffer	Action at End of File	Values Success	Values Fail	Typeout? if Failure
S	Failure	N/A	N/A	N/A	Yes
:S	Failure	N/A	-1	0	No
N	Performs a P command (but a FORM character is not inserted) and resumes search	Failure	N/A	N/A	Yes
:N	Performs a P command (but a FORM chorocter is not inserted) and resumes search	Failure	-1	0	No
=	Performs a Y command (read only) and resumes search	Failure	N/A	N/A	Yes
:=	Performs a Y command (read only) and resumes search.	Failure	-1	0	No

Q-Register Commands

Q registers are provided for storing quantities, command strings, or buffer contents for later use. Thirty-six Q registers, labeled 0 through 9 and A through Z, are available.

Command	Function
nUi	Use. Ploces the numeric volue n in Q-register i.
Qi	Q-register. Represents the current value in Q-register i.
%i	Adds 1 to the value in Q-register i ond represents the new value.
m,nXi	Xfer. Copies chorocters m+1 through the nth choracter of the buffer into Q-register i. Does not olter buffer contents or pointer.
nXi	Copies the buffer chorocters between the current pointer position ond the nth LINE-FEED chorocter in Q-register i.
Gi	Get. Inserts the text contoined in Q-register i into the buffer beginning ot the current pointer locotion. Set the pointer to the right of the insertion.
[i	Pushes the contents of Q-register i onto the Q-register pushdown list.
]i	Pops the top entry of the Q-register pushdown list into Q-register i. The Q-register pushdown list is cleared each time two successive $s ore typed.

Macro, Iterotion, ond Conditionol Commands

Mi	Macro. Perform the text in Q-register i os o series of commands.
<>	Iteration brockets. When > is encountered, command interpretotion is sent back to <.
n <>	Perform the commonds within the iterotion brockets n times.
;	If not in on iteration, on error results. If mast recent search failed, send command interpretotion to just beyond the matching >on the right; otherwise, no effect.
n;	If not in on iteration, on error results. If the value of n is 0 or greater, send command interpretotion just past the matching > to the right; otherwise, no effect.
! tag !	Tog definition. Tog is the nome of the locotion in which it oppears in o command string.
O tag $	Go to the nomed tog, which must oppear in the current macro or command string.
n"G	If n is Greater than or equol to 0, perform commands up to next !. Otherwise, skip to next !.

Command	Function
n"L	If n is Less than or equal to 0, perform commands up to next '. Otherwise, skip to next '.
n"N	If n is Not equal to 0, perform commands up to next '. Otherwise, skip to next '.
n"E	If n is Equal to 0, perform commands up to next '. Otherwise, skip to next '.
n"C	If n is a symbol Constituent (a letter, number, period, dollar sign, or percent symbol), perform commands up to next '. Otherwise, skip to next '.

NOTE

The double quotation mark (") and the single quotation mark (') symbols are matched in the same way as the left parenthesis symbol, (, and right parenthesis symbol,).

Numeric Values and Arguments in Command Strings

Many command string formats permit arguments with numeric values. The following characters may appear in a command string to develop these values in any instance where a numeric value is permissible.

0 through 9	Represent their corresponding numeric values.
B	Beginning. Equivalent to 0.
Z	Equivalent to the number of characters in the buffer.
.	Equivalent to the number of characters to the left of the current pointer position (or in other words, equal to the current pointer position).
Qi	Q-register. Equivalent to most recent numeric value placed in Q-register i.
nA	ASCII. Equivalent to ASCII value of character to right of pointer; n is used to differentiate this argument from an Append command (A) and has no other significance.
tH	Equivalent to value of elapsed time in 60ths of a second since midnight.
tF	Equivalent to the value of the console data switches.

Command	Function
tE	Has the value of the form feed switch. If, during the last Y or A command execution, data transmission was terminated by a form feed character, tE has a value of -1, otherwise, the value is 0.
t t	(On Teletype Models 33 and 35, hold down both the CTRL and SHIFT keys and type N.) Equivalent to the ASCII value of the next character in the command string; this character is not interpreted as a command.
tT	Typed Character. Stops command execution until user types a character on the Teletype; tT then becomes equivalent to the ASCII value of the character typed.
\	Equivalent to the value represented by the digits (or minus sign) immediately following the current pointer position. The value is terminated by the first nonnumeric character encountered. The pointer is positioned immediately following the value.
m+n m-n	Add Subtract } Take one or two arguments. A space is equal to +.
m*n m/n	Multiply Divide (truncates) } Take one or two arguments.
m&n	Logical AND; bitwise AND of binary representations m and n.
m#n	Logical IOR; bitwise inclusive OR of binary representations m and n.
( )	Operators +, -, *, /, #, and $ are normally performed left to right. This sequence can be overruled by use of parentheses.

NOTE

TECO does not assume that multiplication and division are always performed before addition and subtraction. Thus, to obtain the equivalent of a + (b * c), one must use the parentheses; otherwise, (a + b) * c is assumed.

n=	Causes the value of n to be typed out.
H	Abbreviation for B, Z. (0 through the last location of the buffer; in other words, the whole buffer).

NOTE

If a command takes two numeric arguments, a comma is used to separate them.

TECO Termination Commands

Command	Function
↑C	Returns control to the Monitor without waiting for any I/O operations to finish.
↑G (BELL)	Returns control to the Monitor after completing all current output requests.

Stand-Alone Examples (Advanced)

Search

```
J3SMOVE  $  IM  $
```

Within the current buffer, search for the third occurrence (3S) of the text MOVE, position the pointer immediately after it, and insert an M at that point.

Search for a Special Character

```
S↑NA  $
```

Search for any character except A within the current buffer.

```
S↑S  $
```

Search for any separator character within the current buffer.

Q-Registers, Macros, Iterations, and Conditionals

```
J0UN <S↓
      $;%N>QN =
```

Count the number of LINE-FEED characters in the buffer as follows:

1. Position the pointer at the beginning of the buffer (J),
2. Place 0 in Q-register N(0UN),
3. Perform a search for a LINE-FEED character (S LINE-FEED $); if one is found, add 1 to Q-register N (;%N). Go back (< >) and repeat this cycle until the end of the buffer is reached and the test fails (;); at this point type out the contents of Q-register N(QN=).

```
J<SJUMPA  $; -4DIRST  $ >
```

Whenever JUMPA appears in the current buffer replace it with JRST.

Command	Function
	1. Position the pointer at the beginning of the buffer (J).
	2. Search for JUMPA; when found, backspace the pointer four positions and delete the four characters passed over (;-4D).
	3. Replace these four characters with the characters RST (IRST).
	4. Repeat this routine (< >) until the test fails (end of the buffer has been reached) and exit (;) to >.

Placing a Command in a Q-Register for Later Execution

@I# JØUN <S! - $) ;%↑>QN = #HXP	1. Insert the text 'JØUN <S)! $; %N >QN=' into the buffer (@I# .....#) 2. Copy the contents of the buffer into Q-register P (HXP).
To Execute the Command:	
ERDTA3:FN.EX $ YMP	1. Read in a page of a file to search. (ERDTA3:FN.EX $ Y) 2. Execute the command stored in Q-register P (MP).

Reading in Text to be Inserted in Several Places in a File and Storing it in a Q-Register

ERPTR: $ YHXP)	1. Assume that the text to be inserted is on paper tape. Open an input file on the paper tape reader (ERPTR:); read the text into the buffer (Y); copy the contents of the buffer into Q-register P (HXP).
ERDTA4: TXTEDT $)	2. Open the input file to be edited and the output file to contain the edited version.
EWDSK: TXTEDU $) YNCALC: $ GP)	3. Read a page from the input file and initiate a search for the text CALC:. When found, insert the text stored in Q-register P at that point (GP).
NTOT: $ GP	4. Search for the text TOT: and, when found, insert the text stored in Q-register P after it.

Command	Function
.R TECO) *ERMTA1:$EM14EM$$)	Select MTA1 for input; rewind the tape (EM) and advance the tape one file (14EM).
EZDTA1:REVFIL$$)	Select DTA1 for output; zero the directory; open a file and call it REVFIL.
*YNTAXRT$0LT) IX1$$)	Read in the first page from the input file; search for the text TAXRT; if it cannot be found, write the buffer out, read in the next page, search again, etc.; continue this cycle until either TAXRT is found or end of file is reached. If TAXRT is found, position the pointer at the beginning of the line containing it, type the line, and place the line in Q-register 1.
oooo...TAXRT aaaa.....oaaaa	
*JNTXRTE$0LT: 0LT)	Search the buffer for the text TXRTE; if not found, write out the buffer, read the next page, search again; continue this cycle until either TXRTE is found or end of file is reached. If TXRTE is found, position the pointer at the beginning of the line containing it, type the line, and insert the contents of Q-register 1 immediately before that line.
G1$$) bbb...TXRTE bbb.....bbbbb	
*NTXTEND:$) J<SA;1A-47"G1A-58"L-DIB$">) PWEF$$)	Read pages from the input file and write them on the output file until end of file (marked by the text TXTEND;) is found. At that point, move the pointer to the beginning of the buffer (J), and search for all As in the buffer (SA); if the character following the A is a digit, 0 through 9 (ASCII codes 48_{10} through 57_{10}), change the A to a B (1B); continue searching and modifying until end of buffer is reached; write out the last page and write end of file on the output device.
*↑G$$) EXIT) ↑C)	Return control to the Monitor after all output requests have been completed.

TECO Diagnostic Messages

Message	Meaning
?n	n is a decimal number associated with one of the list of error messages given in Table 2-5. TECO ignores the remainder of the command string and returns to the idle state. At this point, the user can type back ?, causing TECO to type out the command string terminated by the bad command.

Error List for ?n Messages

n_{10}	Meaning
1	TECO attempted to read commands beyond the terminating $$. This error is probably due to an unterminated @I or @S command, or to an unsatisfied O command.
2	Same as 1.
3	An attempt was made to supply more than two arguments to a command, either by the use of two commas or by "H,".
4	Too many right parentheses.
5	= command with no argument.
6	U command with no argument.
7	Q, U, X, or G command specifies an illegal Q-register (i.e., other than A through Z or 0 through 9).
8	In an X command, the second argument is not greater than the first.
9	In a G command, the Q-register does not contain text.
10	In a G command, the data in the Q-register is not in correct form (this is an internal error).
11	In an E command (e.g., ER, EW, EF, etc.), c is illegal.
12	File not found on LOOKUP.
13	Blank filename specified for directory device.
14	Project-programmer number specified does not have UFD.

Error List for ?n Messages

n_{10}	Meaning
15	Protection foilure on disk.
16	File connot be occessed becouse it is currently being written.
17	LOOKUP or ENTER returned error type 6 (not defined).
18	LOOKUP or ENTER returned error type 7 (no device).
19	Directory full on ENTER.
20	Requested I/O device not ovailoble.
21	Not ossigned.
22	EW commond between on EB commond ond its EF.
23	EM command given, but no input file open.
24	nEM command, where n is not in the range 1 to 16.
25	Internol error: EF after EB, but no input file is open.
26	Illegol chorocter in filename.
27	Illegol choracter in proiect-programmer number.
28	Attempt to read on input page when no file has been opened for input.
29	I/O error on input device.
30	Attempt to output o page when no file has been opened for output.
31	Two orguments were supplied for an L command.
32	Attempt to move pointer beyond page.
33	A 2-argument command has its second orgument less thon the first orgument.
34	Attempt to search for too long o chorocter string.
35	Seorch command did not find the required string.
36	In on M command, the Q-register does not contoin text.
37	In on M command, the dato in the Q-register is not in correct form (this is on internol error).
38	Unmatched right ongle bracket.

Error List for ?n Messages

n_{10}	Meaning
39	; encountered when not in iteration.
40	" command with no numeric orgument, or "x where x is nat G, L, E, N, ar C.
4T	This is the number typed aut at the end of the ? command's dump of the command string in error. Refer ta the number of the previous error.
42	A character has been encauntered as an undefined cammond.
43	A 1D commond, when DDT has nat been looded with TECO.
44	Not enough core available from the Monitor.
45	A RENAME ottempted with either o blonk nome or one alreody in use. Presumably due to a fault in the EB command.

Debugging Aids – As on oid in debugging macros ond iterations, TECO can be set in the trace mode by typing ? as any character other than the first in o cammond string. When in Trace Mode, TECO types out each cammand as it is interpreted, interspersed with requested output. Typing a second ? in the same monner takes TECO out of Troce Mode; the ? can be typed each time it is desired to change the current mode.

The user can also type comments an his teletype sheet as he executes TECO by typing:

t Atext tA

This causes all text entered to be printed on the teletype (with the exception of terminating tA choracter).

NOTE

Since the terminator tA is not o command, it must be typed by holding the CTRL key down while typing A. It cannat be entered as "up arrow, A."

If DDT hos been loaded olong with TECO by the Linking Loader, control can be transferred to DDT by using the command 1D.

Monitor Commands

To call in TECO and open a new file for creation:

Monitor Commands	Equivalent CUSP Commands
.MAKE filename.ext	.R TECO
* (text input commands) $$	*EWDSK:filename.ext $$
*EX $$	*(text input commands) $$
	*EX $$

To call in TECO and open an already existing file for creation:

Monitor Commands	Equivalent CUSP Commands
.TECO filename.ext	.R TECO
*(editing) $$	*EBDSK:filename.ext $ Y $$
*EX $$	*(editing) $$
	*EX $$

PERIPHERAL INTERCHANGE PROGRAM (PIP)

PIP transfers data files from any standard I/O device to any other standard I/O device and, additionally, performs simple editing and magnetic tape control functions. PIP1, a compact version of PIP, performs a subset of PIP functions. PIP handles all data formats, and eliminates the need for a satellite computer to handle off-line data conversions.

Requirements

PIP	Minimum Core: 3K	Additional Core: 1K if disk is one of the I/O devices; any core above that required is used for extra I/O buffers.
	Equipment Handled:	DECtape, disk, magnetic tape, paper tape reader, paper tape punch, card reader, line printer, and teletype.
PIP1	Minimum Core: 1K	Additional Core: Any core greater than 1K is used for extra input buffers.
	Equipment Handled:	DECtape, disk, magnetic tape, paper tape reader, paper tape punch, card reader, line printer, and teletype.

Initialization

.R PIP) or .R PIP1)	Loads PIP (or PIP1) into core.
*	PIP is ready to receive a command; an asterisk is typed after each requested action has been completed.

General Command Format

destination-dev:filename.ext =source-dev:filename.ext,...source-n)

destination-dev:
source-dev:

The destination device, to which the data is to be transferred; the source device(s), from which the data is to be read

NOTE

If logical device SYS (the CUSP device) is a DECtape, it must not be modified using the /R or /D switches or any other request requiring it to be initialized for input and output at the same time.

DTAn:	(DECtape)
PTR:	(paper tape reader)
PTP:	(paper tape punch)
DSK:	(disk)
CDR:	(card reader)
MTAn:	(magnetic tape)
LPT:	(line printer)
TTY: or TTYn:	(Teletype)

If more than one file is to be transferred from a magnetic tape, card reader, teletype, or paper tape reader, dev: is followed by a comma for each file after the first; these devices can also be followed by * or *.* to indicate all files are to be transferred.

filename.ext (DSK: and DTAn: only)

The filename and filename extension to be assigned to the file on the destination device; the filename and filename extension of the file(s) to be read from the source device.

An asterisk can be used for source files as follows.

filename.*	- Transfer all files having the specified filename.
*.ext	- Transfer all files having the specified extension.
.	- Transfer all files.
*	- Transfer all files with null extensions.

The destination descriptors and the source descriptors are separated by the left arrow symbol (←).

Disk File Descriptor Format

DSK:filename.ext[proj,prog] <protection>

[proj,prog] — Project-programmer number assigned to the disk area to be used, if other than the user's project-programmer number.

<protection> — Protection value to be assigned to the destination file. If omitted, the standard protection is assigned.

NOTE

Standard protection (055) designates that the owner is permitted to read or write, or change the protection of, the file while others are permitted only to read the file.

Standard Assumptions

Unless otherwise changed by switches, all files which are on directory devices and which have a filename extension of .REL, .SAV, .DMP, or .CHN are copied in binary; all other files are assumed to be in ASCII line mode. Magnetic tape files, unless otherwise changed by switches, are read in odd parity and written in odd parity at 556 bpi.

Examples

Command	Function
.R PIP↵	Loads and starts PIP.
*LPT:←DTA1:FILE1↵	Transfer the file named FILE1 from DTA1 to the line printer.
LPT:←DTA1:↵	Transfer all files with null extensions from DTA1 to the line printer.
*DTA2:FILE2←DTA1:FILE1.TMP↵	Transfer the file named FILE1.TMP to DTA2 and give it the name FILE2.

Command	Function
*DTA2:FILE3←DTA1:FILE1,FILE2)	Transfer the files named FILE1 and FILE2 from DTA1 to DTA2, combining them as one file under the name FILE3.
*DTA2:FILE3←DTA1:FILE1,DTA3:FILE2)	Transfer the file named FILE1 from DTA1 and the file named FILE2 from DTA3 to DTA2, combining them as one file under the name FILE3.
*DSK:FILE1←MTA1:)	Transfer the next file from the present position of MTA1 to the user's area on the disk, call it FILE1, and assign the standard protection of 055.
DSK:FILE1<177>←MTA1:)	Transfer all files from MTA1 (starting at the current position of the read head) to the user's area on the disk, combining them into one file called FILE1, and assign protection 177.
*DSK:FILE1[1,3]←MTA1:,)	Transfer the next two files from the present position of MTA1 to area 1,3 on the disk, combining them into one file called FILE1, and assign the standard protection (055).
*PTP:←PTR:,,,,)	Transfer five files from the paper tape reader, combining them as one file on the paper tape punch.
*^C)	Return to Monitor.
.	Dot indicates that user is at Monitor level.

Switches

Nonmagnetic-tape switches, when used, are preceded by a slash (if more than one is specified, they may be enclosed by parentheses instead) and can appear anywhere in the command string; however, if the command string contains commas, the switches must be specified prior to the first comma.

Magnetic tape switches are enclosed by parentheses and must appear immediately following the device or file to which they refer.

Switches are used to specify:

a. Particular files for transferral or deletion;
b. Editing;
c. Mode of transfer;
d. Directory manipulation (DECtape and DSK); and
e. Magnetic tape control.

A listing of PIP switches can be obtained by typing

*output-dev:/Q↵

where output-dev: may be either LPT: or TTY:

PIP Switch Options

Switch	Meaning
A	Line blacking
B	Process file in Binary mode.
C	Suppress trailing spaces and Convert multiple spaces to tabs.
D	Delete the file.
E	Treat Ending (card) columns 73 through 80 as spaces.
F	List the directory in short form for DSK: or DTAn: only. (Filenames and extensions only.) Useful when disk directory cannot be listed with /L sw.
G	Ignore I/O errors.
H	Process file in image binary mode.
I	Process file in Image mode.
L	List the directory (DSK: or DTAn: only)
M	Magnetic tape switches. A string of one or more magnetic tape switches begins with an M and is enclosed in parentheses.

#nA	Advance the tape n files.	E	Even parity.
#nB	Backspace the tape n files.	F	Mark End of File.
#nD	Advance the tape n records.	T	Skip to logical end of Tape.
#nP	Backspace the tape n records.		
2	200 bpi density.		
5	556 bpi density.		
8	800 bpi density.	U	Rewind tape and Unload.
A	Advance tape one file.	W	Rewind the tape.
B	Backspace tape one file.		

NOTE

MTA switches always apply to the device or file immediately preceding the switches. MTA switches should be used only in specific situations. For a more detailed treatment, see PIP Programmer's Reference Manual.

PIP Switch Options

Switch	Meaning
N	Delete the sequence Numbers from a file.
O	Same as /S switch except sequence numbers are incremented by 1 (One).
P	FORTRAN output file format assumed as input. Convert format control characters for line Printer listing.
Q	Print this list of switches.
R	Rename the file.
S	Add Sequence numbers to the file or resequence a file already containing Sequence numbers; Sequence numbers are incremented by 10.
T	Suppress Trailing spaces.
U	Copy blocks 0, 1 and 2 of a DTA file. Commonly used to transfer TENDMP.
V	Count unmatched angle brackets < >.
X	Copy specified files only.
Y	Perform a RIM DTA to PTP conversion. Source extension: — Destination format: .RTB — RIM Loader, RIM10B File, XFERWD .SAV — RIM10B File only. .RMT — RIM10
Z	Zero out the directory (DTAn: or DSK: only).

NOTE

Switches A, B, N, S, and Z are available for use in PIP1. Y is an optional switch obtained by setting RIMSW = 1 at assembly time (see PIP, OPR file for explanation of PIP assembly and loading procedures).

Examples

```
.R PIP )
*DSK:/X-DTA1:*.* )
*DSK:(DX)-DTA1:FILE1,*.REL )
*MTA2:/S-CDR: )
*LPT:/P-DTA1,FILE1 )
*DTA2:FILE1/I-PTR: )
*TTY:/L-DTA1: )
 nnnn FREE BLOCKS LEFT )
 filename.ext no. blocks creation date
*DTA1:/Z- )
*MTA2:(M8E)-MTA1:(ME8) )
*MTA2:(MW)- )
*LPT:-MTA1:(M2W),(MA),, )
*MTA1:(M#4A)-CDR: )
*^C )
```

Load and run PIP.

Transfer all files from DTA1 to DSK, keeping them separate and retaining their filenames.

Transfer all files, except FILE1 and any files with the extension. .REL, from DTA1 to DSK, keeping them separate and retaining their filenames.

Transfer a file from the card reader to MTA2 and add sequence numbers.

Take FILE1 (a FORTRAN output print file), interpret the carriage control characters, and print the file using specified carriage control.

Initialize both DTA2 and the paper tape reader in image mode and transfer one file from the paper tape reader to DTA2, calling it FILE1.

List the directory of DTA1 on the teletype.

Zero the directory of DTA1.

Transfer a file from MTA1 to MTA2 in 800 bpi, even parity mode.

Rewind MTA2.

Set MTA1 to 200 bpi, odd parity, rewind the tape, and transfer the first, third, fourth, and fifth files to the printer.

Advance MTA1 four files before transferring a file from the card reader.

Return to the Monitor.

PIP Diagnostic Messages

Message	Type	Meaning
?4K NEEDED -	S	4K is not currently available but is needed when a disk is present in the system.
?DECTAPE I/O ONLY	S	I/O device for copy block 0 (/U) must be a DECtape.
?DEVICE dev:DOES NOT EXIST	I/O	Either device name has been misspelled or there is no such device.
?DEVICE dev:NOT AVAILABLE	I/O	The device has been assigned to another job.
?DIRECTORY FULL	FR	There is no room for an entry in a DECtape directory.
DISK DIRECTORY READ	I/O	This message is nonfatal if the /G switch is used; otherwise, it is fatal and is preceded by a ?. A second message follows (see Table 6-3).
?DISK OR DECTAPE INPUT REQUIRED	I/O	This command requires a directory device for input.
?DTA TO PTP ONLY	RIM	DTA input and PTP output must be specified for /Y.
FAILURE DURING (/X,/Z,/D,/R) REQUEST	S	Each file requested does exist, but one or more was unavailable for processing. This message is never fatal.
?FILE filename.ext ILLEGAL EXTENSION	RIM	Extension for /Y request must be .RMT, .RTB, or .SAV.
?FILE filename.ext ILLEGAL FORMAT	RIM	1. Zero-length file; or 2. Requisite job data info not available; or 3. Block overlaps previous block (RIM 10) or 4. EOF found when data was expected, or 5. A pointer word was expected but not found in the source file.
?filename.ext (3) FILE WAS BEING MODIFIED	FR	Disk file named is currently being processed by another job.
?filename.ext (0) FILE WAS NOT FOUND	FR	Filename.ext not found during LOOKUP.

PIP Diagnostic Messages

Message	Type	Meaning
?filename.ext (0) ILLEGAL FILE NAME	FR	Indicates that 1. No filename was specified for DTA output file; or 2. A reject occurred on a /R request for disk file; or 3. Illegal filename was specified for a /R request on DTA.
?filename.ext (1) NO SUCH PROJECT-PROGRAMMER NUMBER	FR	The project-programmer number specified for a DSK file is incorrect.
INPUT DEVICE dev: FILE filename.ext	I/O	This message is nonfatal if the /G switch is used; otherwise, it is fatal and is preceded by a ?. A second message follows (see Table 6-3).
?LINE TOO LONG	S	A line >140 characters was detected in the source file.
?LOAD POINT BEFORE END OF (MB) OR (MP) REQUEST	S	Load point on a magnetic tape file has been reached before the tape has been backspaced the number of files or records specified in (M#nB), (M#nP).
?NO BLOCK 0 COPY	C	/U given but PIP assembled without provision for this.
?NO FILE NAMED filename.ext	FR	No such file found during PIP directory search.
?NO FILE NAMED QPIP	S	The data file for the /Q switch is not available.
OUTPUT DEVICE dev: FILE filename.ext	I/O	Followed by a second message (see Table 6-3).
?PIP COMMAND ERROR	C	1. Illegal format for command string; or 2. Nonexistent switch requested; or 3. Filename.ext other than * (or *.*) requested for a non-directory device; or 4. The illegal switch combination RX.
?filename.ext (2) PROTECTION FAILURE	FR	Same as FAILURE DURING ... message except that the processing halts.
?filename.ext (4) RENAME FILENAME ALREADY EXISTS	FR	Tried to rename file with already existing name.
?filename.ext (5) RENAME ERROR	FR	LOOKUP or ENTER not done.
?filename.ext (6)	I/O	Error not yet defined.
?filename.ext (7)	I/O	Error not yet defined.

PIP Diagnostic Messages

Message	Type	Meaning
?TERMINATE /X MAX of 999 FILES PROCESSED	S	The /X switch specified for nondirectory device source files has processed the maximum number of files (999).
?TOO MANY REQUESTS FOR dev:	C	Conflicting parity/density requests have been given for a magnetic tape.
?TRY PIP		During a PIP1 run, a switch or function which is not present in PIP1 has been requested.

NOTES

All fatal diagnostic messages are preceded by a question mark (?).

Message types are:

C	Command string error
FR	File reference error
I/O	I/O error
RIM	Readin Mode specification error
S	Other types of errors.

Table 6-3
Secondary PIP I/O Diagnostic Messages

Message	Device	Meaning
BINARY DATA INCOMPLETE	PTR	Length of block disagrees with word count (nonfatal if the /G switch has been specified).
BLOCK TOO LARGE	DTA	DTA link number $>1101_8$.
CHECKSUM OR PARITY ERROR	All	Read or write error (nonfatal if the /G switch has been specified).
INPUT BUFFER OVERFLOW	All except DTA	Block too large for buffer (nonfatal if the /G switch has been specified).
DEVICE ERROR	All	The data control unit has detected the loss of data (nonfatal if the /G switch has been specified).

Secondary PIP I/O Diagnostic Messages

Message	Device	Meaning
PHYSICAL EOT	MTA	The end of tape has been reached (nonfatal if the /G switch has been specified).
WRITE (LOCK) ERROR	DTA,DSK,MTA	Attempt has been made to write on a write-locked file.
7-9 PUNCH MISSING	CDR	Binary card lacks 7-9 punch (nonfatal if the /G switch has been specified).

IK Version of PIP (PIP1) Limitations

The following limitations apply to PIP1:

a. Z and MW requests ignore all source devices.

b. B switch included since REL,SAV,DMP, and CHN files are not automatically copied in 36-bit bytes.

c. Error messages assume all I/O devices are DECtape.

d. Neither project-programmer numbers nor protection can be specified for disk files.

e. The * cannot be used for filenames or extensions.

f. SAV files cannot be successfully copied with PIP1.

Monitor Commands

The following Monitor commands perform PIP-type operations.

Desired Result	Monitor Command	Equivalent CUSP Commands
To type the contents of a file on the TTY.	.TYPE dev:filename.ext)	.R PIP) *TTY:=dev:filename.ext)
To list the contents of a file on the line printer.	.LIST dev:filename.ext)	.R PIP) *LPT:=dev:filename.ext)
To type the directory of a device on the TTY.	.DIRECT dev:)	.R PIP) *TTY:=dev:/L)
To list the directory of a device on the line printer.	.DIRECT dev:/L)	.R PIP) *LPT:=dev:/L)

Desired Result	Monitor Command	Equivalent CUSP Commands
To delete a file.	.DELETE dev:filename.ext	.R PIP) *dev:filename.ext/D~)
To rename a file.	.RENAME dev:newfn=oldfn	.R PIP) *dev:newfn/R=oldfn)

NOTE

If dev: is omitted in the Monitor commands, DSK: is assumed.

FILE UPDATE GENERATOR (FUDGE2)

FUDGE2 updates files cantoining ane ar more relacatable binary pragrams, and permits the user to manipulate individuol programs within program files.

Requirements

Minimum Care:	2K
Additianal Care:	Dynomically allocates its buffers to utilize as much core as is made availoble.
Equipment:	Two input devices, one for the master file and one far the transaction file; one output device for the updated file. The input device(s) and autput device can be the same device (DSK:). The two input devices can be the same DECtape.

Initializotion

.R FUDGE2)	Loads the File Update Generator program.
*	FUDGE2 is ready to receive o command.

MONITOR COMMANDS

NAME	ABBREVIATION	ARGUMENTS				
		1	2	3	4	5
ASSIGN	AS	dev	ldev			
ASSIGN SYS		dev				
ATTACH	AT	dev				
ATTACH	AT	job	proj prog			
CCDNT	CC					
CDMPILE	COM	list				
CDNT	CON	lh	rh	adr		
CDRE	COR					
CREATE	CREA					
CREF	CREF	dev				
CSTART	CS	list				
CTEST		list				
D(deposit)	D	dev				
DAYTIME	DA	dev				
DOT	DD	adr				
DEASSIGN	DEA	file	ext			
DEBUG	DEB	list				
DELETE	DEL	dev				
OETACH	DET	dev	file	ext	proj prog	core
OIRECT	OI					
E(examine)	E					
EOIT	ED	list				
EXECUTE	EX	list				
FINISH	F					
GET	G	file	ext			
HALT	↑C					
KJOB	K	file	ext	core		
LIST	LI	dev	job			
LDAD	LOA					
LOGIN	LOG	arg				
MAKE	M					
PJOB	P	dev	file	ext	proj prog	core
R	R	dev	file	ext	core	
REASSIGN	REA	n				
REENTER	REE	dev	file	ext	core	
RENAME	REN	adr				
RESOURCES	RES					
RUN	RU	core				
SAVE	SA	file	ext			
SCHEOULE	SCH					
SSAVE	SS					
START	ST					
SYSTAT	SYS					
TALK	TA	dev				
TECO	TE	file	ext			
TIME	TI	job				
TYPE	TY	list				

Key:

adr	octal address	lh rh	octal value of left and right half words
core	decimal number of 1K blocks	proj prog	project-programmer numbers
dev	physical device name	list	a single file specification or a string of file specifications
ldev	logical device name	arg	a pair of file specifications or a string of pairs of file specifications
ext	filename extension	n	scheduled use of the system.
file	filename		
job	job number assigned by Monitor		

See Book 3, Chapter 2 for further explanation of commands.

These abbreviations are accurate and unique as of now, but their accuracy and uniqueness may be changed in the future by the addition of new commands.

Make contact with your computer facility by whatever means the facility has established (e.g., acoustic coupler, telephone, or data phone).

↓

Turn the little plastic knob on the right-hand side of the Teletype to ON.

↓

Type ↑C on the Teletype (i.e., hold down the CTRL key while striking C). This establishes communication with the Time-Sharing Monitor. The Monitor signifies its readiness to accept commands by responding with a period (.).

↓

Type LOGIN, or LOG, followed by a carriage return. The system will respond with an informative message like the following:

JOB n NAME OF SYSTEM
#

JOB n is the job number the system has just assigned to you. NAME OF SYSTEM is usually the Monitor name and version number.

↓

Type your project-programmer numbers after the number sign, followed by a carriage return.

↓

The time-sharing system will then type
PASSWORD:

↓

Type your secret password followed by a carriage return. The system will keep the password secret by not printing it on the paper.

↓

If the project-programmer numbers and the password match the project-programmer numbers and password stored in the system accounting file, the system responds with the time, date, TTY number, ↑C, and a period.
Example:

1301 8-Aug-69 TTY23
↑C

↓

Now the time-sharing system is ready to accept any commands you wish to type in. You may direct it to load and start a program from the System Library (.R prog), start a program already loaded in core (.START), or perform any of a variety of other operations. (See inside of back cover for a summary of Time-Sharing Monitor commands.)