Title
SYSTEM OPERATION OF THE PDP-7 COMPUTERS AT LRL BERKELEY

Authors
Radeloff, J.O.
Robinson, L.B.
Meng, J.D.

Publication Date
1969-05-01

UCRL-18883

ey. 2

SYSTEM OPERATION OF THE PDP-7 COMPUTERS AT LRL BERKELEY

J. O. Radeloff, L. B. Robinson, and J. D. Meng

May 1969

AEC Contract No. W-7405-eng-48

LAWRENCE RADIATION LABORATORY
UNIVERSITY of CALIFORNIA BERKELEY

UCRL-18883

ey. 2

DISCLAIMER

UCRL-18883
UC-32
Mathematics and
Computations
TID-4500 (54th Ed.)

UNIVERSITY OF CALIFORNIA

Lawrence Radiation Laboratory
Berkeley, California

AEC Contract No. W-7405-eng-48

SYSTEM OPERATION OF THE PDP-7 COMPUTERS AT LRL BERKELEY

J. O. Radeloff
Institut für Strahlen und Kernphysik
Bonn, Germany

and

L. B. Robinson and J. D. Meng
Lawrence Radiation Laboratory,
Berkeley, California

May 1969

Printed in the United States of America
Available from
Clearinghouse for Federal Scientific and Technical Information
National Bureau of Standards, U. S. Department of Commerce
Springfield, Virginia 22151
Price: Printed Copy $3.00; Microfiche $0.65

UCRL-18883

SYSTEM OPERATION OF THE PDP-7 COMPUTERS AT LRL BERKELEY

J. O. Radeloff
Institut fur Strahlen und Kernphysik
Bonn, Germany

and

L. B. Robinson and J. D. Meng
Lawrence Radiation Laboratory,
Berkeley, California

May 1969

ABSTRACT

An on-line data-taking system based on a PDP-7 computer has been developed. The system allows a time-shared use of the computer with several counting experiments. The data areas are hardware protected against interference between different users. DEC-tapes are used for storage and retrieval of the programs. Data are stored either on DEC-tape or IBM-compatible magnetic tape.

The operation of the data taking system is described. A software handling program package operating with DEC-tape is also introduced.

SYSTEM OPERATION OF THE PDP-7 COMPUTERS AT LRL BERKELEY

Table of Contents

I. System Summary 1.1
Description of the System 1.1
Conclusion 1.2

II. Using the LRL-PDP-7 Binary Loader and Bootstrap 2.1

III. Use of the PHA 1969 Data Taking System 3.1
A) Loading Procedure 3.1
B) Memory Protection Unit 3.2
C) Commands Via the Teletype 3.4
Table of Commands 3.24
D) Commands Via the Remote Console 3.27
E) Single Parameter Data Taking with one or two ADC's 3.30
F) Gain Stabilizer 3.32
G) Error Diagnostics and Recovery 3.34
Memory Map for PHA 1969 . . . . . 3.36

IV. Use of the LRL-PDP-7 Software Package 4.1
Format of the Symbolic Tape 4.2
A) The Editor 4.4
B) The Assembler 4.10
C) DDT, Quick Save and DDT Q 4.12
D) Copy Symbolic 4.15
E) Debug 4.17
F) Copy Dectape 4.18
G) Dtog L 4.20

V. A New Subprogram for the PHA 1969 System 5.1
A) Creating a New Subprogram 5.1
B) Storing a New Subprogram 5.2

VI. A New User's Main Program 6.1
A) Creating a New User's Program 6.1
B) Storing a User's Program on DEC-tape 6.2

VII. The LRL-PDP-7 Binary Loader System 7.1
A) Binary Program Tape Formats 7.1
B) The Hardware Bootstrap System 7.3
C) The Loader System 7.4
D) Commands to the Loader System 7.5
Table of Memory Locations of Programs 7.11

VIII. Appendices 8.1
A. Programmer's Information PHA 1969 . . . . . 8.1
B. Subprogram "Timed Data Save" 8.16

Acknowledgements . . . . . -iv-

References . . . . . -iv-

ACKNOWLEDGEMENTS

The leadership and stimulating advice provided by Fred Goulding is gratefully acknowledged. Conversations and discussions with the many of the experimenters, here referred to as "users", have been very encouraging and helpful. Advice and suggestions made by members of the Mathematics and Computing Division at LRL Berkeley have been very valuable.

FOOTNOTE AND REFERENCES

* This work was carried out as part of the research program of the Nuclear Chemistry Division of the Lawrence Radiation Laboratory, University of California, which is supported by U.S. Atomic Energy Commission Contract W-7405-eng-48.

1. L. B. Robinson, IEEE Trans. Nucl. Sci, 13, Vol. 1, 161 (1969).

2. L. B. Robinson and J. D. Meng, On-Line Real-Time Operation of a PDP-7 Data-Taking System, UCRL-17220, March 1967.

3. R. Anderson and A. Kray, Lawrence Radiation Laboratory, Livermore, Private Communication.

I. SYSTEM SUMMARY

INTRODUCTION:

In recent years small computers have proved to be very valuable in on-line data taking applications[1]. Their main advantage in research work is the versatility of the whole system as well as reliability. A programming system has been prepared together with the development of external hardware devices and interfaces to make use of a PDP-7 computer (Digital Equipment Corporation) for on-line data taking.

DESCRIPTION OF THE SYSTEM:

At the Lawrence Radiation Laboratory, Berkeley there are actually two separate systems installed each consisting of a PDP-7 computer with an 8192 word memory. The standard peripherals of the computers are a Teletype typewriter and fast paper tape reader and puncher. In addition DEC-tape units (two transports each) and large scale oscilloscope displays (8" x 11") were purchased. With this equipment the computer is capable of doing all required jobs concerned with programming and data processing.

For on-line data taking, however, further equipment is needed [2]. A fast 4096 channel analog-to-digital converter has been developed and interfaced to the computer. Furthermore, remote control consoles have been designed, which allow control of the computer program from up to eight remote stations. The data area of the core memory can be divided into up to eight blocks of switch selected length and each

block is protected against destruction from all others but the assigned control console. By means of these consoles the program parameters, particularly the display can be changed very easily using rotary switches in a fashion similar to that of a multichannel analyzer. For mass storage of data and for communication with the CDC 6600 at the Computer Center IBM-tape units are provided.

The operating programs for the computers are stored on DEC-tape. Thus quick retrieval and reliable storage is achieved. For a "Cold" start a fully automatic hardware bootstrap has been added operating with DEC-tape. A loader system is stored on the beginning of each binary program tape (the "WHAT\" - System), and is started automatically using the hardware bootstrap.

Software systems have been developed and are now available to the experimenter, that allow efficient use of the hardware equipment. According to the two types of operations, a data taking system program (PHA 1969) and a program handling software package are provided. The latter offers the possibility of easy preparation of special data processing programs as required by the experiment. An increasing number of such special programs are presently being made available to the users.

CONCLUSION:

With the presently provided hardware and software it is possible to partially time-share the computer for several users: while one (or more) data counting experiments are going on with high priority, a

second user may get control of the display and all subprogram features. All subprograms use the same area in core memory and therefore only one operation can be performed at a time. A more general way would be to load the required subprogram into the core area which is assigned to the requesting user and thus allow more than one subprogram to reside in core simultaneously [3]. Design of additional hardware as well as further software developments are still in progress for the PDP-7 computers.

CONVENTION:

Throughout this pamphlet switches and typewriter keys are indicated by their names, that is they are spelled with the first letter in capital and all the following small.

Examples: Altmode

Return key

Continue switch

Bootstrap

X

Ø

II. USING THE LRL-PDP-7 BINARY LOADER AND BOOTSTRAP

Introduction:

The LRL-PDP-7 Binary Loader is a short program stored in blocks Ø to 6 on every binary program DEC-tape. It allows information to be read from the tape into core memory or to be written from core onto the tape[a]. The Loader is called and started automatically by the Bootstrap switch.

CALLING A PROGRAM FROM DEC-TAPE:

Prerequisites:

PDP-7 Binary Program Tape present (General User's Tape, PHA 1969 User's Tape or Assembler Tape).

Procedure:

1) Mount the Tape on transport #1, set to WRITE LOCK.
2) Move Tape forward several feet manually.
3) Depress the Bootstrap switch momentarily; Typer types: WHAT \.
4) Type name of the desired program, terminate by pressing either the Return key or the Altmode key*). Tape transport #1 moves, Typer types PROGRAM IN and the called program starts operating. (Run light is on.)

---

a) For writing a program from core onto DEC-tape see sections VI and VII.

*) Older User's Tapes (before January 1969) contain a different Binary Loader: Then, only the Return key will terminate the program name and

5) after Typer says PROGRAM IN, the Continue switch must be depressed to start the called program.

Notes:

a) In case of mistyping the desired program name:

6) Press the Rubout key: the computer provides a linefeed (no carriage return).

7) Type the name again as in 4).

b) In case of incorrect spelling of the program's name (spaces are ignored) - (interchange of two characters is OK except first and last characters) - or if the desired program is not present on the tape, typer says:

...name...UNKNOWN

WHAT \

(Proceed at 4) spelling correctly or get another tape which contains the program.

Remarks:

1) The Bootstrap fails to operate, if:

a) The memory location pointed at by the Address switches contains the HLT instruction. (Press Continue switch)

b) The computer console switches Single Step or Single Instruction are up.

2) If the program is not to be started automatically after read-in, terminate the program's name by depressing both keys Ctrl and Bell instead of Return or Altmode. After the Typer types PROGRAM IN, the program can be started by depressing the Continue switch. (Some programs require to hit the Continue switch twice. These are e.g. Dtog L, Write Loader . . . .).

III. USE OF THE PHA 1969 DATA TAKING SYSTEM

Introduction:

The PHA 1969 software system allows the shared use of a PDP-7 computer for on-line data taking. It provides a live display of the data in core memory and responds to controls of the Remote Control Consoles. The PHA 1969 system also includes some data processing subroutines.

Contents of this Section:

A) Procedure for Loading the PHA 1969 System.

B) Memory Protection Switch Set Up for PHA 1969.

C) Commands to PHA 1969 given via the Teletype Keyboard.

D) Commands to PHA 1969 given via the Remote Control Console.

E) Use of Single Parameter ADC with a PDP-7.

F) The Gain Stabilizer

G) Error Diagnostics and Recovery

Prerequisites:

a) A PHA 1969 User's Tape.

b) If more than one user is served simultaneously by the system, there must be an area of the core memory assigned to each user to be served.

A) Procedure for Loading the PHA 1969 System:

1. Mount a PHA 1969 User's Tape on Transport #1, set WRITE LOCK.
2. Call PHA 1969 via the Bootstrap procedure by typing PHA. Typer types a semicolon.

3. Turn Function switch at Remote Control Console to STOP COUNT and press Command button.

B) Memory Protection Switch Set Up for PHA 1969:

The core memory of the computer has a capacity of $8192_{10} = 20000_8$ words (this holds for both PDP-7 computers at LRL, Berkeley; the "remote memory" at the HILAC is considered as an external device). With the PHA-program residing in core, the area between addresses 4000_8 to 17400_8 is available for data taking and handling. The Memory Protect Unit divides this area into several data areas of switch selected sizes and assignes these to the various Remote Control Consoles. Data within the so assigned areas (legal areas) may be manipulated by the various commands. The display control switches allow the data area to be displayed in groups of up to 1024 channels at once. It is possible to display unassigned data areas, but any attempted operation on an unassigned area will be ignored by the system.

The Memory protection unit allocates memory for data to separate Remote Consoles (and the associated ADC's) in blocks of $(400)_8$ words. ($(400)_8 = (256)_{10}$). The 5 most significant bits of the 13 bit memory address correspond to the sets of 5 switches. Each horizontal set of 5 switches in turn corresponds to the lower boundary of one ADC and Console. The lower boundary for console 4 forms the upper boundary for console 3, while the lower boundary of console 3 forms the upper boundary for console 2, etc. The most significant address bit is at the left side. The switch settings may be expressed as octal computer addresses:

$1\ 111\ 1\ 00\ 000\ 000_2 = 17400_8$

$1\ 011\ 1\ 00\ 000\ 000_2 = 13400_8$

$0\ 000\ 1\ 00\ 000\ 000_2 = 400_8$

$0\ 101\ 0\ 00\ 000\ 000_2 = 5000_8$

Switches

Complete Address

The maximum data area available is $(6400)_{10}$ or $(14400)_8$ words. If all program functions are to be used, only $(5888)_{10}$ data words are available.

1. Set the boundaries of all consoles to 17400 (all switches up).
2. Set the lower boundary of the highest number console to be used at a number B_1 such that $17400 - B_1 = N_1$, where N_1 is the number of data words needed.
3. Set the boundaries of all consoles of ~~lower~~ highest number to B_1.
4. Set the lower boundary B_2 of the next lower number console to be used so that $B_1 - B_2 = N_2$ is the number of data words needed for that console.
5. Repeat steps 3 and 4, using B_2, B_3 and N_3, etc.

The lowest legal value of B is 4000 if all programs are to be available. If a value 3400 is used, "medium" and "short" length subprograms will work. If a value of 3000 is used, only "short" subprograms will respond. If any value of B lower than 3000 occurs, the PHA program will type ADC SWITCH TOO LOW and halt. Correct the error and press the Continue switch.

C) Commands to PHA 1969 given via the Teletype Keyboard:

A short-form list of the commands is given in the table on the pages 3.24 - 3.26. A description and examples of the programming is given in 'PHA Programmers Information" (L. Robinson, LRL, October 1968)*).

The teletype commands are in general of the form

$n_8, n_7, \ldots, n_1$,nXXXX Altmode.

n_8 through n are nine decimal numbers which are separated by commas. Not all commands require all nine numbers to be typed, in fact in some cases missing numbers are interpreted to have specific values. Spaces are allowed before and after the numbers. XXXX is a four letter instruction code. Instructions are described on the following pages 3.6 to 3.23. No action takes place unless the Altmode key is depressed after the instruction is typed.

Hitting the Rubout key erases the current command string. The computer generates a line feed, carriage return sequence and a semicolon is typed.

Hitting the Bell (= Ctrl and G)) stops the current operation except ADC counting. If the typer is typing, it may be necessary to hit "Bell" several times in order to get a response. Typer types a semicolon after carriage return and line feed. (If this does not occur, hit "Bell" once more.)

Typing KILK Altmode stops all operations and reinitializes data areas from memory protection switches. This command must be given after the settings of the switches have been changed. The typer types: ADC 1, 2 ... DATA LOCATIONS and in a second row 8 five digit numbers. These numbers

* Appendix A

are the lower boundaries of the 8 remote consoles typed out as octal core addresses. A value of 20000_8 ($=8192_{10}$) means, there is no usable area assigned to that console. Type a number before command KILK to suppress field printout.

A command which has not been programmed will cause the teletype to type ?. If the program is on tape, a pause will occur while the program is called from Tape 1. If the program is already in memory it will start at once.

The instruction list can be divided into several functional groups. These are:

Data Taking	3.6
Storage and Retrieval on DEC-Tape	3.8
IBM - Compatible Magnetic Tape	3.12
Display Manipulation	3.15
Data Processing	3.16
Spectrum Stripping	3.17
Data Analysis	3.18
Printing and Plotting	3.20
Miscellaneous	3.22

On the following pages the currently available*) features and subprograms of the PHA 1969 system are described.

*) March 1969

DATA TAKING:

The PHA system program is written with special emphasis on flexible features for data taking with the computer. To provide this flexibility insertion of special purpose data taking subroutines is made easy.

At the moment several subprograms are available for single parameter work (one pulse height spectrum from one detector or up to three spectra from several detectors, which are not in coincidence). A program for multiparameter data taking (up to four parameters) is in preparation.

Straight-forward pulse-height analysis (one spectrum, one ADC, no automatic special features) can be done with the system program PHA alone (no further subroutines). See Chapter E) of this section: USE OF SINGLE PARAMETER ADC WITH PDP-7.

.	n_2,	n_1,	n	ADCX		several ADC measurements

1. More than One ADC in use:

   Connect the ADC-MULTIPLEX GATES one by one to convenient Remote Console BNC's, thus the Console numbers to be used are determined (see Chapter E) for cables to be used). Set the memory protection switches (see Chapter B) of this section) such that the assigned data areas do not overlap in order to control the counted data independently (e.g. for later storage on DEC-tape).

   Typing now the above command starts simultaneous data taking with ADC's - n_2 and n_1 for n seconds. The ADC's are not numbered themselves. The numbers, n_2 and n_1 are assigned to them by the connection to the appropriate remote control console (see Chapter E for the procedure).

2. One ADC controlled by several Remote Consoles:

Connect the ADC-MULTIPLEX GATE to the highest numbered Remote Console BNC to be used. Set up the memory protection switches (see Chapter B)) such that upper boundary of the highest numbered console is not higher than the lower boundary + 4096 channels of the lowest numbered console. This allows access to the data by means of the Group Switches rather than shifting the origin by means of the Toggle switches.

Typing now the above command starts data taking with the spectrum going into the data area assigned to the highest numbered console for n seconds. The display (and all other features) can be controlled (and operated) from either console n_1, n_2.

NOTE: The legal data area after an ADCX-command is the composition of the data areas of consoles n_1, n_2. ERASE (Function switch set to ERASE, then press Command button) will operate on this whole area. Many other operations, as for instance DEC-tape storage or creation and removal of markers, will operate on the displayed area, i.e. starting at the current display origin.

			n	TEMB		timed data taking	
			n	TEMG		with IBM-tape storage	

Both commands cause data taking with a properly hooked up ADC (see Chapter E)) for n-1 second,then:

store 1024 channels of data on IBM-tape,

print out the live time in milliseconds,

print out the real time in seconds,

print out the area under two or four peaks with a calculated background subtracted (the printed value is 0.1 of the actual area),

erase the data in core memory and restart counting after 1 second (n seconds is the cycle time, one second being used for storage and printing).

Both commands can be used with two ADC's using the ADCX-command before.

For a detailed description of the operating procedure see the write-up: "Timed Data Save", L. B. Robinson; LRL Berkeley, March 1969.*) Both commands call a subprogram into core which gains semi-permanent status. That is (a part of) the subprogram stays in memory (addresses 3400 - 4000) and is protected against over-writing by other long subprograms (see memory map on page 3.36). The semi-permanent program can be stopped and the core area released by typing $ (no Altmode required).

NOTE: In fact the programs are laid out to operate with two ADC's. Therefore in any case there have to be 8 markers defined for TEMB or 9 markers for TEMG.

STORAGE AND RETRIEVAL ON DEC-TAPE:

Data are stored on DEC-tape in "runs" of 1024 channels (or less). The storage format allows up to 120_{10} such runs to be stored on one DEC-tape. Each run is stored in five 256_{10} word blocks, the first of which

---

* Appendix B.

contains general information and identification. Previously stored data may be over-written by new data at any time; but a protection is provided to save useful data from being overwritten.

The identification block is of the following format:

WORD NUMBER	USE
1	Unused
2	Absolute core address of data origin.
3	Absolute core address of this run.
4	Negative channel count (< 1025).
5	Users special run number (binary).
6 - 31	Teletype I D message.
32 - 49	Unused.
50 - 51	Display and plotting parameters.
52 - 60	Unused.
61 - 64	4 decimal numbers n_3, n_2, n_1, n typed ahead of the storage command stored in binary form.
65 -112	8 sets of six marks previously stored.
113 -224	Mics. program constants.
225 -248	Gain stabilizer settings.
249 -256	Unused.

The identification block is an image of the core area (program variables) 17400 through 17777 as it appears during the write operation.

Data storage and retrieval always starts at the current CRT display origin, and checks that the displayed area really belongs to the currently active remote console. In case of error, the typer types: DT ER and no data transfer takes place.

When Dectape data are retrieved, only the first 32_{10} I.D. words are read back. When copying Dectape to IBM tape, the first 100_{10} words are used. The other identification words can easily be read back if needed by future subprograms. For example, gain stabilizer settings could be read back.

Since tape 1 is also used for program storage, data must not be written into the tape's program area. Data storage is automatically inhibited for blocks $(0 - 163)_{10}$ on tape 1, which is the area used for program storage. Thus run numbers 0 - 31 cannot be read or written on tape 1.

	n_2,	n_1,	n	TAPE		write onto tape			
		n_1,	n	GETE		read from tape			
	1024,	3,	next	- " -		default			*)

Transfers run n of n_2 channels, starting at current display origin to or from tape transport # n_1. ID message OK is stored automatically. Previously present data are not protected against overwriting; neither on the tape nor in core memory (all data in core areas not assigned to the currently active console are protected however). These two commands use a minimum amount of program area: < 3000.

---

* If numbers are omitted while typing the command, the computer inserts figures noted as default.

	n_2,	n_1,	n	WRTZ		write onto tape			
		n_1,	n	GETZ		read from tape			
	1024,	3,	next	- " -		default			*)

Transfers run n of n_2 channels, starting at current display origin to and from tape transport #n_1. Before writing onto tape takes place, an identification message of up to 75 characters may be typed, following the typewriters request ID: . Terminate with carriage return.

n_3,	n_2,	n_1,	n	SAVD		multirun write tape		
n_3,	n_2,	n_1,	n	READ		multirun read tape		
4,	1024,	3,	next	- " -		default		
e.g. n_3,	,	n_1,	n	- " -		is allowed (n_2=1024 is inserted)		

Transfers n_3 runs of n_2 channels ($\leq$ 1024) starting at run No. n to or from tape transport # n_1. Before writing onto tape takes place, an identification message of up to 72 characters may be typed following the typewriters request ID: and terminated by carriage return. The several runs transferred with only one command are distinguished by an automatically included identification AA, AB, AC, ... etc.

Data on tape as well as data in core memory are protected against overwriting. In case of previous data present, the typer says ZERO OLD DATA FIRST! To clear data in core set Function switch on the desired console to Z or ERASE and press Command button. To clear data on tape use ZERO command.

* If numbers are omitted while typing the command, the computer inserts figures noted as default.

In case typer types DT ER (DEC-tape error) either a too small area of core was assigned to the active console or a parity checksum error was encountered. Change memory protection switches and use KILK command or use TEST command (see under Miscellaneous, see also OLDZ there) respectively.

		n_1,	n	ZERO		clear protection of previous data on tape.			
		1,	ø	- " -		default.			*)

Clears protection of n_1 runs (previously stored with SAVD command) starting at run No. n of tape on transport #3.

IBM COMPATIBLE MAGNETIC TAPE:

The magnetic tapes at the PDP-7's are written (and read) as 7 channel, 556 bits per inch at a speed of 30 inches per second. Lateral and longitudinal parity bits are provided automatically by the hardware.

Runs of 1024 channels are written in two records: the first containing 100_{10} words of identification and information, the second containing the data. The identification record consists of the first 100_{10} words as described for the DEC-tape storage. The first word is used to contain the run number. The data record contains 1030_{10} words, the last six of them being meaningless. The reason for storing six more words on the tape is to make the record fit into an integral number of 60 bit words of the CDC-6600 computer.

*) If numbers are omitted while typing the command, the computer inserts figures noted as default.

			n	WRTI		write a	run			

Writes the next two records as run n of 1030_{10} channels starting at the display origin. Data are written in binary format.
(Write ring necessary!)

			n	GETI		read a	run			

Searches forward for run n on the tape until an End-of-File mark is found. If it finds the desired run, both identification and 1024 channels of data are read from the tape, the run number and the identification is typed out and the data stored in core memory starting at the current display origin.

			n	BAKI		backspace				

Causes the tape to backspace n records. As each run consists of two records, n should be twice the number of runs to be backspaced.

				EOFI		write	End-of-File			
				REWI		rewind	tape			

After several runs (or records) have been written on tape, the whole file is closed by an end-of-file mark. Rewinding of the tape can be done either by the above command (tape control stays in remote mode) or manually.

		n_1,	n	IBMI		magnetic tape			
		n_1,	n	IBMY		instruction command			
		N-O-C,	3	- " -		default			*)

Performs an IBM-tape operation whose function is determined by n according to the following table:

1	read a BCD record to display area
3	read a binary record to display area
4	rewind
129	write display area as BCD record
131	write display area as binary record
133	write an end-of-file mark.

The maximum number of words transferred is given by n_1 or, if omitted, determined by the Number-of-Channels switch at the currently active remote control console. If, in case of a read, the record on tape is shorter than the specified channel number, transfer will stop at the end of the record.

				SCIN		read SCIPP-tape: 7 channels			
				SCON		read SCIPP-tape: 6 channels			

Data written on magnetic tape by a Victoreen 1600-channel-analyzer (SCIPP) are of a format that cannot immediately be handled by the present PDP-7 programs. The above stated commands provide the possibility to read such data and convert them into ordinary binary channel by channel while they are read into core. The commands will read one file consisting of 16 records with 100 channels each.

* If numbers are omitted while typing the command, the computer inserts figures noted as default.

DISPLAY MANIPULATION:

In general, the display is handled (i.e. the parameters are changed) by means of the various rotary switches and toggle switches at the remote control consoles. In addition to those features, there are a few typewriter commands included in the system to provide more flexibility.

n_7,	n_6,	n_5,	n_4,	n_3,	n_2,	n_1,	n	SETS	set markers

Places up to eight markers at the specified channel numbers. The numbers may be typed in any order. The channel numbers refer to the origin of the assigned data area. All previous markers are removed.

		n_1,	n	MOVS		move a	marker		

Moves the n^{th} marker to channel No. n_1. This command is especially useful when several markers are present, as the toggle switches at the console allow only the last created marker (or the highest marker displayed if any command is executed) to be moved. It is not necessary to have the marker in the displayed area, neither before nor after the movement. Markers 1 to 7 can be handled by this command.

			n	ADDZ		shift	display	origin	right
			n	SUBZ		shift	display	origin	left
			ø	- " -		default			

*)

Shifts the display origin right or left by n channels, starting from the current display origin. Press the Command button with any setting on the Function switch to restore normal display origin.

* If numbers are omitted while typing the command, the computer inserts figures noted as default.

			n	BASZ		set Y-baseline		

Sets the Y-baseline to n counts. Press Remove toggle switch to restore normal baseline.

DATA PROCESSING:

A number of commands are provided to process data present in core memory. Such are: background subtraction, normalization, smoothing, and insertion of data points by typing.

			n	ADDX		add a constant		
			n	SUBX		subtract a constant		

Adds (or subtracts) n counts to the contents of each channel in the displayed area.

			n	MULS		normalize data		

Multiplies the contents of each channel in the displayed area by $\frac{n}{1000}$.

			n	J		smooth data		
	n = setting of Number-of-Channels switch if default							

*)

Smoothes n channels starting at display origin according to the formula

$$\tilde{N}_i = \frac{N_{i-1} + 2N_i + N_{i+1}}{4}$$

---

* If numbers are omitted while typing the command, the computer inserts figures noted as default.

n_1,	n	SETX	insert data

Replaces the contents of channel No. n_1 with n and places a marker at the next channel. If there were markers present before the command was given, then the lowest of these markers is taken to channel (n_1 + 1). If n_1 is not typed, then the lowest marked channel is taken instead. Into channel zero data can be inserted only if a marker is placed there.

SPECTRUM STRIPPING:

The term spectrum stripping refers to the correction of one spectrum by another on a channel-by-channel basis.

n_1,	n	ADDO	read from tape and add	
n_1,	n	SUBO	read from tape and subtract	
3,	same	-"-	default	*)

Adds or subtracts data on transport # n_1 from data in core memory, channel by channel, starting at the display origin. Before any operation takes place the program checks, that the legal area is not smaller than the number of channels in the required run on tape.

n_2,	n_1,	n	MULO	read from tape and multiply	
1000,	3	same	-"-	default	*)

Reads run n from the tape on transport # n_1, by data from tape, then multiplies spectrum in core by spectrum on tape, channel by channel, then divides by n_2 with 36-bit precision.

---

* If numbers are omitted while typing the command, the computer inserts figures noted as default.

	n_2,	n_1,	n	DIVO		read from tape and divided				
	1000,	3,	same	- " -		default				

*)

Reads run n from the tape on transport #1, multiplies the present data in core by n_2 and divides spectrum in core by spectrum from tape, channel by channel, starting at display origin.

DATA ANALYSIS:

With a measured spectrum present in the display area, a number of analyzing procedures can be performed on-line: such as integrating the area under a specified (marked) peak, fitting a background curve, calibrating the energy scale (X-axis).

		n_1,	n	H		energy	calibration		

Set markers to center of two known peaks (one marker each) in spectrum, then apply H command to assign energies n_1 and n to the peaks. The typer will type out the channel numbers together with the corresponding energies for the two marked channels and for channel zero.

Set now one marker to an unknown peak, set Q-rotary switches to 6-1-1 and depress Q-toggle switch. Typer will type out the channel number and the corresponding energy (if more than one marker is present, the lowest one is used). Choose energy units such that 262, 143 is never exceeded.

---

* If numbers are omitted while typing the command, the computer inserts figures noted as default.

n_7,	n_6,	n_5,	n_4,	n_3,	n_2,	n_1,	n	SETM	set markers
						n_1,	n	MOVM	move a marker

The commands SETM and MOVM use exactly the same programs as SETS and MOVS. See page 3.15 for detailed description.

n_4,	n_3,	n_2,	n_1,	n	SAVM		store a set of 6 markers	
0,	0,	0,	0,	-"-	-"-		default	*)

After applying SETM (or SETS or the Create and shifting toggle switches) SAVM will store a set of six markers defining peak # n and its background. The two center markers define the peak itself whereas the two markers each to the right and left are defining the amount and the trend of the background. To determine the amount of the background more accurately an averaging width may be specified around each of the four background markers. n_4 (through n_1) $\neq 0$ cause the program to take the amount of the background at the lowest (etc.) marker as the average of the counts in an area n_4 channels up and down about the marked channel (thus $2n_4 + 1$ channels contribute to the average).

n_i must not exceed 31, i = 1, 2, 3, 4 and

n must not exceed 8.

			n	TESM		analyze one peak	

Analyzes peak number n. The program calculates a 3rd order polynomial to fit the outer 4 of 6 markers about the peak, which have been

* If numbers are omitted while typing the command, the computer inserts figures noted as default.

stored previously by means of the SAVM command. Then it types out the area under the peak defined by the center two markers and with the background subtracted. If the points given for the background curve cannot be fitted, the program will exit after about 20 seconds. For an immediate exit, strike any typewriter key.

			n	CALM		analyze n peaks	

Analyzes the first n peaks in the same manner as TESM. SAVM should be used previously to store n sets of markers.

			n	AVRM		average		

Averages the counts over 2 n + 1 channels centered at each of up to 6 markers. This operation is useful in testing the linearity of an ADC, after feeding it with pulses randomly distributed in height. The averages are typed out.

PRINTING AND PLOTTING:

Data can be printed out at the teletyper either in octal or decimal form or can be plotted in various modes at the Calcomp plotter.

		n_1,	n	P		print in decimal	

The program prints out the contents of all channels between (and including) channels n_1 and n, or if no numbers are typed, between the lowest two CRT markers. The numbers are typed in decimal form with suppression of leading zeros. Each row of five points is preceeded by the channel

number of the first point and a colon.

The printing can be interrupted by hitting "Bell" (Ctrl and G). If no response, hit again.

			n_1,	n	OCTZ	print in octal	

The program prints out the contents of n channels starting at address n_1. n_1 has to be typed as the absolute core memory address in decimal. The contents are typed in octal, no leading zeros suppressed. Each row of eight contents is preceded by the core address (in octal!) of the first point and a colon. If n_1 is omitted, display origin is used.

				HIST		histogram, no marks
				HIMT		histogram, with marks
				PLOT		point plot, no marks
				PLMT		point plot, with marks

Use of the Calcomp Plotter:

1) Turn power switch at the plotter on.

2) Reset the plotter pen to the right-hand border of the paper; since the plotter is controlled incrementally, there is no way for the program to know where the pen is initially. The program assumes the pen to be at the right border.
The plotter will plot the data exactly to the scale shown on the CRT display.
NOTE: In LOG-mode the Display-Scale rotary switch determines the number of decades arranged within the full paper width of 10 inches.

4) Type one of the plotter commands (or use the appropriate Q switch setting).

In point plot mode a small cross (+) is plotted at each data point. This operation is comparatively slow as for each point the pen has to be lowered onto the paper and raised again. In a plot with marks, every tenth, hundredth and thousandth channel is marked with respect to the data origin.

MISCELLANEOUS:

			n	OLDZ		read old DEC-tape			*)
			next	-"-		default			

Reads run n of 1024 channels from DEC-tape at transport #3 into core memory starting at the display origin and without performing a checksum test (parity check). Run identification is typed out. The command is mainly used to read data from old DEC-tapes which have been produced with the 1967 PHA program.

		n_1,	n	TEST		read DEC-tape			*)
		3,	0	-"-		default			

Reads block n of 256 words from DEC-tape at transport # n_1 into core memory starting at the display origin. Remember: five blocks are used for storage of one run of 1024 channels; see Data Storage and Retrieval on DEC-tape.

				VERI		read IBM-tape			

Reads one file (one or more runs) from IBM-tape into core memory starting at the display origin. The program reads one run after the other, typing out the run number and identification.

---

* see footnote on page 3.19

	n_2,	n_1,	n	COPY		copy	DEC-tape	onto	IBM-tape

Copies runs n_2 through n_1 (incl.) from DEC-tape at transport # 3 onto IBM-tape starting at IBM run number n. Upon finishing the Execute light remains lit (IBM program status "busy"). Additional runs can be transferred by repeating the command. The program status and Execute light can be cleared by either typing $ (Shift and 4 keys) or by hitting Bell (Ctrl and G keys).

As each run is read from DECtape, the identification line and the IBM run number is typed out. The data is written on IBM tape, then the tape is backspaced and read back for a word by word comparison. If this test fails, a sample of the error words are typed out and the transfer is repeated. In case of three successive failures, the typer types BAD IBM TAPE.

DATA TAKING

			DECIMAL NUMBERS	COM-MAND	DESCRIPTION	LENGTH OF SUBPROGRAM
	ADC NUMBERS,		Counting Time	ADCX	Take PHA Data	Short
			Counting Time	TEMB	Timed Data Save	Semiperm
			Counting Time	TEMG	Timed Data Save	Semiperm
DEC-tape						
	Channels Per run	Tape No.	Run No.	TAPE	Write one run (No I.D.)	Short
		Tape No.	Run No.	GETE	Read one Run	Short
		Tape No.	Run No.	GETZ	Read one run	Medium
	Channels Per run	Tape No.	Run No.	WRTZ	Write one run with I.D.	Medium
No. of Runs	Channels Per run	Tape	Run No.	READ	Multi Run Read	Medium
"	"	"	"	SAVD	Multi Run Store	Medium
		No. of Runs	First Run	ZERO	Remove protection from a tape.	Medium
IBM-tape						
				EOFI	Write End of File	Medium
				REWI	Rewind	Medium
			Run No.	WRTI	Write a Run	Medium
			Run No.	GETI	Read a Run	Medium
			No. of Records	BAKI	Backspace	Medium
			Function	IBMY	IBM Function	Long
			Function	IBMI	Read Binary	Medium
				SCIN	Read SCIPP Tape 7 Characters.	Medium
				SCON	Read SCIPP Tape 6 Characters.	Medium

DISPLAY

			DECIMAL NUMBERS	COM-MAND	DESCRIPTION	LENGTH OF SUBPROGRAM
		Up to 8 Channel No.		SETS	Set CRT Markers	Short
		Channel No.	Marker Number	MOVS	Move CRT Marker	Short
			Channel Count	ADDZ	Shift Display Origin Upward	Medium
			Channel Count	SUBZ	Shift Display Origin Down	Medium
			n	BASZ	Set Y Baseline=n	Medium

DATA PROCESSING

			Count	ADDX	Add to Displayed Data	Short
			Count	SUBX	Sub from Displayed data	Short
		Channel Number	Data Point	SETX	Set Data at lowest marker	Short
			N	MULS	Multiply Data $\times \frac{N}{1000}$	Short
				J	Smooth Displayed Data	Short

SPECTRUM STRIPPING

		Tape No.	Run No.	ADDO	Read and	Medium
		Tape No.	Run No.	SUBO	Operate	Medium
	Extra Multi.	Tape No.	Run No.	DIVO	With Data from	Medium
	Extra Divisor	Tape No.	Run No.	MULO	Dectape	Medium

DATA ANALYSIS

		Up to 8	Channel Numbers	SETM	Insert CRT Marks	Long
		Channel No.	Mark No.	MOVM	Move one Mark	Long
			1=No 0=Yes	DISM	B.G. Display Cont	Long
4 Averaging Widths			Peak No.	SAVM	STORE 6 points to define peak	Long
			Averaging Width	AVRM	Type Average for Marked Channels	Long
			Peak No.	TESM	Analyze 1 peak	Long
			m	CALM	Call markers & analyze m peaks	Long
		First Energy	Second Energy	H	STORE calibrated Energies	Short

PLOTTING AND PRINTING

			DECIMAL NUMBERS	COM-MAND	DESCRIPTION	LENGTH OF SUBPROGRAM
		Number of Channel	X Scale	PLOT	Plot Points	Medium
		"	"	HIST	Plot Histogram	Medium
		"	"	PLMT	Points with Channel Marks	Medium
		"	"	HIMT	Histogram with Channel Marks	Medium
		First Channel	Last Channel	P	Print Data	Short
		First Add. (Absolute)	Word Count	OCTZ	Print in Octal	Medium

MISCELLANEOUS

	Channels Per run	Tape No.	Block No.	TEST	Read one block DEC-tape	Medium
			Run No.	OLDZ	Read old format (Tape 3) DEC-tape	Medium
				VERI	IBM-tape	Medium
	First Run No.	Last Run No.	First IBM Run No.	COPY	Copy DEC-tape to IBM	Long
				KILK	Reinitialize the system	Short

REMOTE MEMORY

			First Channel	DISR	Display Remote Memory	Long
			m	INCR	Increment Remote Memory Channel m	Long
	L	m	n	TESR	Test Remote Memory Add m to n channels starting at L	Long
No. of Runs	Channels Per run	Tape No.	First Run No.	PUTR	Store on DecTape from Remote Memory	Long

D) COMMANDS TO PHA 1969 GIVEN VIA THE REMOTE CONTROL CONSOLE:

1. To get control of the CRT display, and computer program, press the Command button of a remote console. The computer will then display the data area assigned to the console, and execute the operation selected by the Function switch:

   a) ERASE - erases all data in the memory assigned to the console.

   b) START - STOP - enables and disables corresponding ADC as indicated by ADC light. If no manual STOP is given, counting will stop after 262,144 seconds, (approximately three days).

   c) Z(ero) - erases only the data displayed (if assigned to the console.

   d) SAVE - allows one remote console to take exclusive control of the computer for 60 seconds. This allows time to select and store a spectrum on Dectape, prepare and start a plotting operation, etc.

   e) PLOT - plots a histogram of the displayed data to the same scale as the CRT display. Although only 1024 channels can be displayed, up to 4096 channels can be plotted by appropriate setting of the Number of Channels switch. Every 10th, 100th and 1000th channel is marked.

   f) DISPLAY - sets the data origin for display, printout, etc. corresponding to the ADC, whose number is selected by the Sector switch. Does not call any function at present.

   g) A - undefined as yet.

2. Use of the Display Scale and Multiplier switches should be obvious. In LOG mode, the Display Scale switch gives the number of decades.

3. Toggle Switches:

   a) REMOVE - remove highest or most recently created*) marker. When the last marker is removed, it will be placed at the lowest displayed channel. Also removes any YBIAS (see below).

   b) CREATE - generate a new marker at the left hand side of display.

   c) LEFT, RIGHT - move highest or most recently created*) marker, left or right.

   d) FAST - move 16 times faster.

   e) ORIGIN - if depressed, motion applies to X display origin.

   f) YBIAS - if depressed, motion applies to Y display origin. (Data is unchanged.)

   g) If both ORIGIN and YBIAS are depressed, motion applies to X scale.

4. Q Switch:

   If the toggle switch labeled Q is depressed, a program will be started corresponding to the setting on the 3 small rotary switches above.

N.B.: For instructions initialized by the Q switch, the switch should be held down until the EXECUTE light goes out.

---

*) If no subprogram is called to perform an operation, then the most recently created marker is affected, the highest otherwise.

Q_A (LEFT HAND SWITCH)

Q_A = 1 - Initialize gain stabilizer (see detailed description on page 3.32).

Q_A = 3 - Transfer data from remote pulse height analyzer

Q_A = 4 - Write 1024 channels, starting at display origin, on Dectape #1 as one of 16 runs (run number selected by Q_B, Q_C switches).

Q_A = 5 - Read a run back into the display area (run number selected by Q_B, Q_C switches).

Q_A = 6 - Print out channel number and energy corresponding to the lowest CRT mark. (Energy Calibration previously given using H command on teletype.)

Q_A = 7 - Unused.

Q_A = 8 - Add, Multiply, Subtract or Divide, channel by channel, from selected Dectape Run on Tape #1 into 1024 channels starting at lowest displayed channel in core memory. Arithmetic function is selected by the position of Function switch.

SAVE: ADD

DISPLAY: SUBTRACT

START: DIVIDE

Z: MULTIPLY

Q_A = 2 : 16 special functions are selected according to the position of switches Q_B, Q_C.

Q_B,Q_C = 1,1 Overlap displays starting at first two CRT markers. (Use YBIAS, LEFT, RIGHT to move 2nd display with respect to the first.)

Q_A, Q_C = 2,1 - Stop overlapping.

= 3,1 - Expand display scale. (Data is unchanged.)

= 4,1 - Contract display scale.

= 1,2 - Type out data between first two markers.

= 2,2 - Smooth displayed data: $N_i' = \frac{N_{i-1}+2N_i+N_{i+1}}{4}$

= 3,2 - Plot a histogram, no channel markers.

= 4,2 - Plot points, no channel markers.

= 1,3 - Plot points with channel markers.

= 1,4 - Display channel number and content of lowest marked channel on CRT as two digital numbers.

= 4,4 - START a timed ADC count for a time in seconds equal to the setting on the Display Scale and Multiply switches (LOG illegal).

NOTE: When a function is initialized, the EXECUTE light goes on until it is completed. The BUSY light goes on when the computer is unable to service the request, or monentarily during any error printout.

If the execute light does not go out press Ctrl and strike Bell(G) on the keyboard.

E) USE OF SINGLE PARAMETER ADC WITH PDP-7:

The ADC stores counts in PDP-7 memory without direct program intervention. Counting is enabled, when the ADC indicator light on the corresponding Remote Console is lit. Counting can be inhibited by the ADC

front panel Flag switch. The ADC can be externally gated if the ENABLE switch is set to REMOTE. "Inhibit" pulses overlapping or "abort" pulses following (within 20 μs) can prevent an input signal from being counted.

1. Connect analog signal (0 to + 8 V, 1 to 10 μs pulses) to Input BNC connector ADC. Use a 125 Ω cable (red) to carry analog signal from remote amplifier.
2. Select number of channels for ADC by plugging in the appropriate program plug.
3. Provide a jumper cable from output ADC-MULTIPLEX-GATE to one of the numbered Remote Console BNC's. Any cable, black, green or red may be used. This determines, which remote station will control the ADC. (Remote Consoles are labeled one through eight on their front panel.)
4. Be sure the ADC-Flag switch is set to NORMAL.
5. Check the settings of the boundary switches at the Memory Protection Unit (see Chapter B of this section). Provide only as much memory space as is needed corresponding to the program plug at the ADC.
6. Mount PHA 1969 User's Tape on Dectape transport #1 and call the PHA-program as described in Chapter A of this section.
7. Mount a Scratch tape or data tape on transport #3, move tapes forward several feet and set WRITE ENABLE on both transports. (The program area on DEC-tape #1 is protected automatically against overwriting.)

8. Use the remote console at the experimental location to control ADC start and stop count, erasure and data storage.
9. When the ADC START command is given, channel Ø will count the live time in milliseconds with overflow in channel 1 for every 262.144 seconds.

NOTES:

1. Two ADC's can be controlled from one remote console, and two consoles can be used to control one or two ADC's:

   After Step 7):

   Type the command n_1, n_2, n_3 ADCX (Altmode). This causes ADC n_1 and ADC n_2 to count for n_3 seconds. It also allows count-control and data erasure for ADC n_1 and ADC n_2 from either console n_1 or console n_2.

2. Routing signals may be applied to the ADC routing inputs so that counts to one ADC can be routed to up to 64 different groups of channels. The route can be advanced by a positive pulse to the ADC ADD 1 input (rear panel).
3. A positive ADC BUSY signal is available at the ADC gate connector.

F) THE GAIN STABILIZER (11X5600 P-1):

Introduction:

The variable gain unit allows gain control under PDP-7 program control. A voltage stretcher (range: 0 - 8 V; drift: 1 mV/sec.) stores a computer generated voltage several times per second. The output of the stretcher

in turn controls the gain of an amplifier over a range of ± 2%. The variable gain inverting amplifier will accept signals of up to ± 4 V and is normally inserted between preamp and main amplifier.

An associated PDP-7 program in PHA 1969 will control up to eight such stabilizers.

Use:

1. Connect CRT Y to Control IN of G.S. (11X5600).
2. Connect PG10 and PG11 of remote console to "sample" inputs of G.S. 1 and G.S. 2 respectively.
3. With G.S. Control Switch at OUT, count until a suitable peak is located. Stop count, set Control Switch to IN. Set Feedback gain to 64.
4. Put first CRT marker on top of peak, second marker at lower edge (less than 31 channel separation). Remove all lower markers.
5. Set Q switch (Left) to 1.
   Set Q switch (Middle) to 1 to set up G.S. 1.
   Set Q switch (Middle) to 2 to set up G.S. 2.
6. Depress toggle switch Q. A gain control point will appear at right center of display.
7. Start counting. Gain Control point will move until peak is accurately centered. Feedback gain may be reduced so long as control point stays on scale.

8. To remove gain control point, repeat 5 and 6 with only one CRT marker.
9. Always switch control to OUT when gain control point is not present.
10. The zero balance can be adjusted so that no gain change occurs with variable gain IN or OUT. (Hint - set control markers to an area where no counts occur when making this adjustment, or hold toggle switch Q depressed!)

G) ERROR DIAGNOSTICS AND RECOVERY:

A number of error conditions can prevent the program from performing certain functions. In most cases a message is typed out on the teletype.

During Startup:

"TURN TRAP SWITCH ON" - trap switch on PDP-7 control panel must be up to enable memory protection.

"ADC FIELD TOO LOW" - the lowest allowed address on any of the memory protection switches is $(3000)_8$, some programs require $(4000)_8$. Set protection switches and press Continue.

"MEM. BUSY" - the memory needed by a subprogram has been assigned to data: (or see below). Readjust memory protection switches so no data area falls below $(4000)_8$ and type KILK and press Altmode key.
Caution - ADC counting is stopped by KILK.

(OR) - the memory needed has been assigned to a quasi permanent subprogram. Terminate the previously used program by typing $.

Caution - Be sure a long experiment is not in progress using the quasi permanent program. (Most such programs control IBM tape.)

"MEM. ILLEGAL" "NO MEMRY" - An attempt has been made to read or write a data area which is not assigned to the remote console currently in control. Check that area on CRT display belongs to you and has room for the number of channels being transferred.

Other Symptoms:

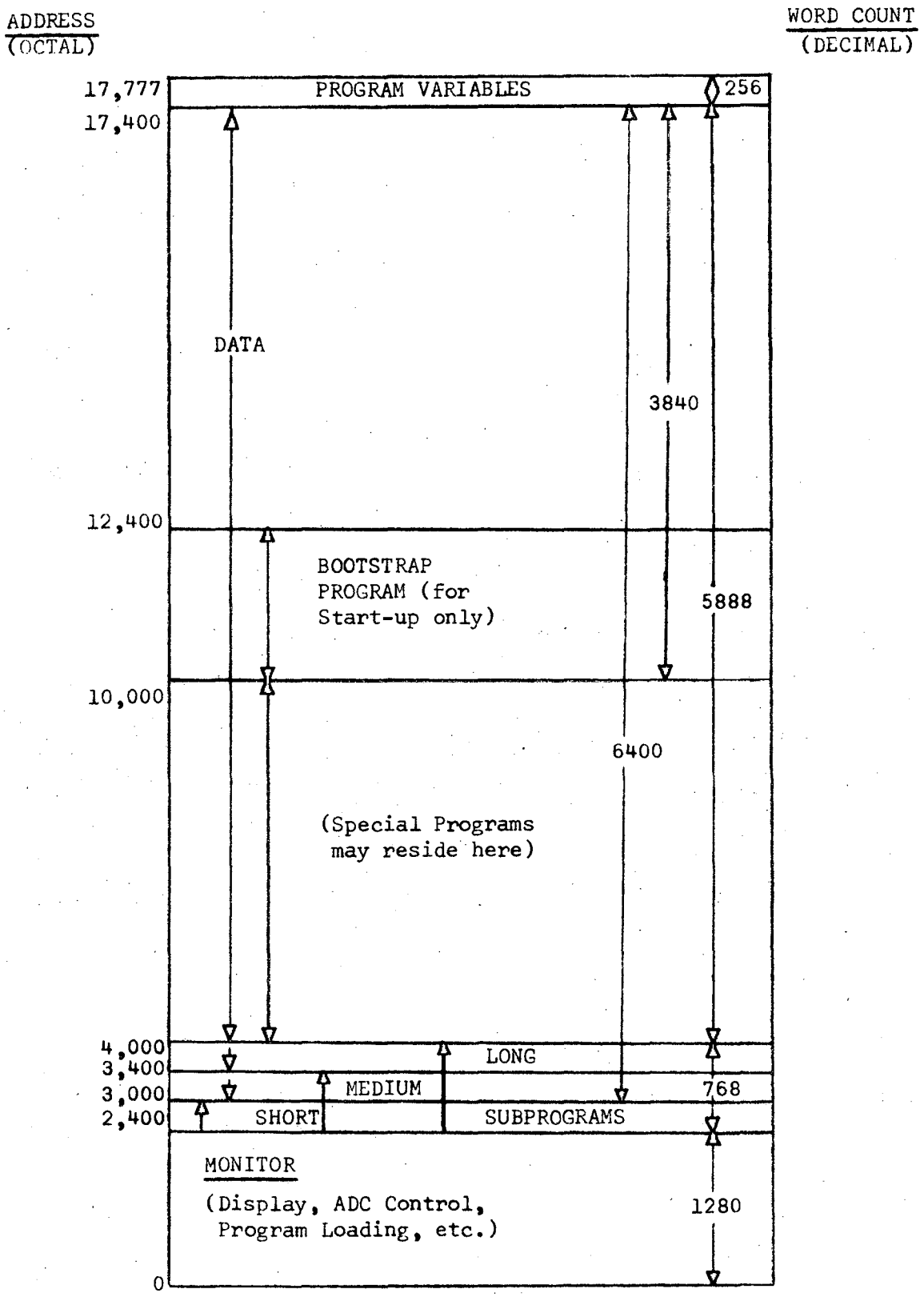
BUSY LIGHT - stays on. A subprogram has not completed its operation properly. Press CTRL and then strike G (Bell) until light goes out.

COMPUTER HALT - (Run light is off, no CRT display). This usually indicates a hardware fault or a bad subprogram. Set Address switch to $(4)_8$ and press START switch on computer panel.

ERRATIC CRT DISPLAY - possible program damage. Reload the program. If valuable data is in memory between $(10000)_8$ and $(12400)_8$ load the EMERGENCY paper tape: set address switches to Ø and press READ-IN switch.

The EMERGENCY - paper tape contains the PHA Main Program. The program starts automatically and rescue operations may be performed by using subroutines present on DEC-tape #1.

PDP-7 MEMORY MAP FOR "PHA 1969"



IV. USE OF THE LRL-PDP-7 SOFTWARE PACKAGE

The software package consists of a number of programs for preparing, debugging and handling data processing (user's) programs. The user's programs are considered to be prepared in machine language: either in symbolic (mnemonic) or in numerical (octal) form. The DEC-Fortran-Compiler for the PDP-7 is not taken into account here.

The LRL-PDP-7 software package includes the following programs:

A) Editor

B) Assembler

C) DDT, Quick Save, DDT Q

D) Copy Symbolic

E) Debug

F) Copy Dectape

G) Dtog L

All the above programs are arranged and written to operate with DEC-tape. The only exceptions are the output of the Assembler, and the input of DDT, which is in the form of punched paper tape. (Both the Editor and Assembler may be operated with paper tape instead of DEC-tape simply by setting Accumulator switch 9 down.)

For greater convenience, however, DEC-tape input/output routines have been added to Digital Equipment's Editor and Assembler programs. The programs of the Software Package are stored on the "Assembler Tapes" (in

binary form). See section VII for details of the tape format. The Assembler expects to find a program prepared in symbolic language on DEC-tape, the "Symbolic Tape". The Editor is used to prepare (and correct, if necessary) this Symbolic Tape.

DDT is Digital Equipment's debugging program and Quick Save is a powerful aid for its use. It allows storage and retrieval of the user's program together with the symbol list of DDT on DEC-tape for up to ten programs.

Copy Symbolic is a tool to make copies of symbolic programs from one Symbolic Tape onto another and provides a printout of the Index. list of program names currently present on the Symbolic Tape.

Debug is an octal debugging program with some features exceeding those of DDT.

Copy Dectape allows any DEC-tape to be copied and verified in whole or in part.

Dtog L writes the Timing and Mark tracks and block numbers on a DEC-tape.

FORMAT OF THE SYMBOLIC TAPE:

To keep track of the many programs, which may reside on one Symbolic Tape, the Editor stores an Index of the names of these programs. These are usually stored on the Assembler Tape to reduce search time on the tapes. Because of this Index, there is in general only one Assembler Tape associated with each Symbolic Tape.

The Editor stores symbolic programs from core memory onto a Symbolic Tape using block numbers 110_8 and up. A space of storage four blocks in length is assigned to each program. This space can be doubled by setting Accumulator switch 6 up. Blocks 100 - 107 (octal, incl.) are reserved to contain the Index, although this is usually stored (at the same block numbers) on the associated Assembler Tape.

On a Symbolic Tape, blocks Ø - 77_8 should not be used. This space could be used to contain the Loader (see section VII) and some binary programs. However, if too many binary programs were stored in this space the Loader would overwrite symbolic programs which had been stored previously (see Chapter E-7 of section VII).

The Index is a listing of the names of the symbolic programs, which are present on the Symbolic Tape. The Index is stored in blocks 1ØØ - 1Ø7 (octal, incl.) on the Assembler Tape. It will accept the names of up to 128 programs. A space of 16_{10} words for identification is assigned to each symbolic program name.

WORD NO.	CONTENT	MEANING
Ø	171717	Heading
1	Two octal digits per character	name
14_8		
15	ØØØØØØ	Terminator for name
16		last block number of program on Symbolic Tape
17		first

Caution! Up to 36_{10} characters (including all spaces) may be used for a program name. The 37^{th} character would wipe out the terminator and the Editor and Assembler will be confused when reading the name. It is advisable to choose for the "name" of a symbolic program its own "title" (first line of the symbolic listing). Only the "name" is stored in the Index and used to retrieve the program. A listing of the program, however, will show its "title" only.

A) THE EDITOR:

Introduction:

The purpose of the Editor is the preparation of a source program written in symbolic language in order to make it accessable for the Assembler (Chapter B). The symbolic programs are stored on the Symbolic Tapes. The commands and procedures for its operation are, with a few exceptions, the same as described in DEC's Programming Manual DIGITAL-7-1-S. The exceptions result from the changes of the Editor to make it work with DEC-tape.

Prerequisites:

a) An Assembler Tape

b) A Symbolic Tape; either a new tape (Mark and Time tracks written) or a tape with some symbolic programs stored on it. (Note that the index on the assembler tape must be made to correspond to the programs on the symbolic tape.)

Procedures:

1) Mount Assembler Tape on Transport #1, mount Symbolic Tape on Transport #3; set to Write Lock.

2) Set Accumulator switches 1, 9, 10, 15, 16 up, all others down.

3) Call the Editor via the Bootstrap procedure (see section II) by typing EDIT T.

4) Use the facilities of the Editor as described in the DIGITAL-7-1-S Programming Manual. HINT: Watch the Link light; in command mode the light is off; in text mode the light is on.

5) Special Commands:

Due to the change of the input/output routines of the Editor to work with DEC-tape, 2 commands have been changed in meaning and 3 have been added. Those changed are: P for "punch" (write on DEC-tape), R for "read"; those added are: U for "update", H for "help" and E for "end".

a) To read a program from a Symbolic Tape into core memory:

Type K followed by carriage Return to clear the Editors buffer from previous programs.

Type R followed by carriage Return, typer asks WHAT NAME:

Type name of desired program (spaces are significant!) and terminate by carriage return. Tape #3 spins and typer provides a line feed upon completion of read in.

Apply Editor commands as described in the DIGITAL-7-1-S Manual for listing, changing, etc.

b) To store a symbolic program on a Symbolic DEC-tape:

Set WRITE ENABLE on both transports.

Type P followed by carriage Return, typer asks WHAT NAME:

Type name of program followed by carriage Return. The name should be identical to the first (title) line of the program and must not exceed 36 characters, spaces are significant.

Typer types COMPLETE upon successful storage.

c) To update an existing symbolic program on DEC-tape with a changed version residing in core memory:

Set WRITE ENABLE on both transports. (If the name is not to be changed, WRITE ENABLE on transport #3 is sufficient).

Type U followed by carriage Return. Typer types WHAT NAME:

Type name of program terminated by carriage Return. Typer says TYPE R TO RENAME ...(name)...

Strike space bar if program name is not to be changed.

Typer says COMPLETE upon successful storage.

Notes:

1) To delete an incorrectly typed program name, strike the Rubout key and start typing the full name again.
2) If in case of a U or P command typer says PROGRAM TOO LONG, set Accumulator switch 6 up and attempt the storage procedure again by using a new name (and title line) for the program and the P command instead.

3) If typer again says PROGRAM TOO LONG, then - there is no help other than to split the program into smaller pieces (with different names). If typer says COMPLETE upon the second attempt, you may rewrite the program under its original name as follows:

   a. Kill the Editors buffer by typing K and Return.
   b. Recall the program fragment by typing R and Return. Again Kill the Editor buffer.
   c. Append a dummy one line program (e.g. SPARE 1).
   d. Now use U to update the original program but rename it SPARE 1.
   e. Kill the Editor buffer a third time.
   f. Read the complete program by typing the new name.
   g. Use U command to update the (new) program and rename it to its original name.

   The SPARE 1 program space on tape may be used later on for storage of a shorter program via the updating and renaming facility.

4) ..."name 2"... TO BE OVERWRITTEN BY ...name 1... occurs if an attempt is made to store (Print or Update) a program with "name 2", that has not been read from the Symbolic tape previously and if "name 2" does exist already in the Index. "name 1" is the name of the program that has been handled by the Editor last before the current one.

d) To get a list of the programs present on the Symbolic Tape:

Type H followed by carriage Return. Typer will type out the names (not the title lines) of symbolic programs as stored in the Index (of the Assembler Tape); transport #1).

In fact, there are two ways to get this listing, see Copy Symbolic for the second way.

e) To copy the Index from the Assembler Tape onto the Symbolic Tape:

Type E followed by carriage Return. This command copies the Index (blocks $1\emptyset\emptyset$ - $1\emptyset7_8$) from tape on transport #1 to tape on transport #3. During operation memory locations are used as buffer (inter-mediate storage), where parts of the Editor reside. Thus the Editor is destroyed and the E command should be given only at the "end" of an editing job.

Note:

No special command is provided to delete a program on the Symbolic Tape. The only way is to replace the out-dated program via the U command and the use of its renaming facility.

Remarks:

1) The 4 blocks of storage for a symbolic program give space enough for approximately 130 lines of text. (This figure depends on the length of the lines, and therefore, the number of remarks. Actually the number of characters including the

tabulator spaces is important.) It is advisable to keep the program's lengths within this bracket, rather than use Accumulator switch 6 up permanently. In order to split a larger program into pieces, the commands

n, m U
n, m P

are provided. In both cases only the contents of lines n to m (incl.) are stored on the Symbolic Tape. In the case of n, m U, all the previously stored program (with the requested name) is replaced.

2) The Assembler requires that each separate program stored on DEC-tape must end with a PAUSE statement.

3) Upon a stop (Run light off) the Editor can be restarted at location 20_8, without loss of information.

4) With the Loader, Editor and Assembler stored (in binary form) in blocks Ø - 77 and the Index stored in blocks 100 - 107 on a Symbolic Tape, the editor can be run with only one tape transport in use: set Accumulator switch 8 up (in addition to 1, 9, 10, 15 and 16 up) and the transport to #3.

5) The Editor can be operated with paper tape input and/or output (P and R commands) by setting Accumulator switch 9 down.

B) THE ASSEMBLER

Introduction:

The Assembler transforms a (properly) written symbolic language program into binary form so that it may be run at the computer. The output of the Assembler is punched paper tape in DEC's FF format. For a detailed description of the programming language used by the Assembler (instructions, pseudo-instructions, symbols and definitions) see Digital Equipment's Programming Manual DIGITAL-7-3-S.

Prerequisites:

a) An Assembler Tape

b) A Symbolic Tape, with the symbolic program to be assembled stored on it.

c) A "Master" program list stored on the Symbolic Tape.

A "Master" program is a list of names of symbolic programs, which are to be assembled together. The "Master" program has a title line, which should be the same as its name. It is stored under its name like any other symbolic program on the Symbolic Tape. The title line is followed by the names of the program modules (start typing each line immediately, no spaces, tab or slash) which are to be assembled together. The pseudo-instruction PAUSE at the end of each program module causes the Multi-Assembler to look for the name of the next module to be assembled. The "Master" program must list only program names, with no termination.

Procedures:

1) Mount Assembler Tape on transport #1, mount Symbolic Tape on transport #3; set WRITE LOCK.
2) Set Accumulator switches 1, 9, 10, 15 and 16 up, all others down.
3) Call the Assembler via the Bootstrap procedure (see section II) by typing ASEM. Typer asks MULTI-ASEM WHAT?
4) Type name of a "Master" program (spaces are significant, Rub-out deletes an incorrectly typed name). Depress Return key. The assembly starts.
5) Watch the paper tape puncher for folding the punched paper properly.
6) Assembly goes to completion automatically. (That is, you are at the end of step 4 on page 32 of the 7-3-S manual.) A tape leader (sprocket holes only) of approximately one foot length is punched.

Notes:

a) The Assembler can be operated with paper tape input by setting Accumulator switch 9 down.
b) The starting address for MULTI ASSEM WHAT? is 7770_8.

Remarks:

1) To assemble a single program module (without a "Master" list): restart the Assembler at location 20_8. Typer asks ASSEM WHAT. Type name of the program followed by carriage return. When assembly halts with all accumulator lights lit, press Continue switch to complete assembly. The completion is indicated by the punching of program's name in readable form and a strip of blank tape.

2) In order to extend a user's main program by additional sub-programs to be assembled later, it is convenient to save the Assembler's symbol table after the assembly of the main program. (This is done for the PHA 1969 User's Program where the whole assembler, including the expanded symbol table, is stored as ASEM SUB PHA on the Assembler Tapes.)

The programmer may store his own sub-program assembler via the ADD feature of the Loader (see section VII): first address = 12400, last address = 27777 (whereby all memory except the Loader is stored on DEC-tape), starting location = 7770

C) DDT, QUICK SAVE AND DDT Q

Introduction:

DDT is Digital Equipment's debugging program as described in the DIGITAL-7-4-S Programming Manual. Quick Save is an aid for fast storage of binary programs on DEC-tape and recall into core memory. Quick Save allows storage and recall of up to ten versions of a program (or ten different programs) on tape. This is particularly useful when debugging a program that destroys either itself or DDT.

With DDT Q, both DDT and Quick Save are called into core at the same time.

Prerequisites:

a) An Assembler Tape with block numbers 400_8 and up unused.

b) A program to be tested in binary form either in core memory or as a paper tape in DEC's binary ff format.

Procedures for DDT:

1) Mount Assembler Tape on transport #1, set to WRITE LOCK.

2) Call DDT (or DDT Q) via the Bootstrap procedure (see section II) by typing DDT (or DDT Q).

3) Use the facilities of DDT as described in the DIGITAL-7-44S Manual to load, execute, examine or change the binary program.

4) DDT starts at location 16ØØØ.

Procedures for Quick Save:

1) Mount Assembler Tape on transport #1 set WRITE ENABLE.

2) Have the program to be saved (in binary) in core memory; (for instance by using the command DEBUG $ of DDT to read an ff binary paper tape; the output of the Assembler).

3)* Call Quick Save (or DDT Q) via the Bootstrap procedure (see section II) by typing QUICK SAVE (or DDT Q). When Quick Save is started, the typer says:

W(R) Ø --- 9 A (D)

* Somethimes the DDT symbol table overlaps the dectape loader (which resides between 10000 and 12140). In this case, load DDT, then quick save, then use DDT to load program for debugging then go to 6.

4) Type WØA to store all of memory (with the exception of addresses 73ØØ through 7777_8) on DEC-tape (transport #1) in blocks number $4ØØ_8$ - 437_8.

5) Computer halts if all accumulator switches are down, otherwise starts DDT at 16000.

6) Debug program and store an improved version by again starting Quick Save at 73ØØ and typing W1A, etc.

Commands for Quick Save:

The commands for Quick Save consist of three characters (no termination!); the first: W or R for write (on tape) or read (from tape into core). The second: a figure from Ø to 9 indicating the location on tape (Ø for blocks 4ØØ - 437, 1 for 440 - 477 etc.). The third: A or D for all of memory (except locations 73ØØ - 7777_8) or the DDT half of memory only (1ØØØØ - 17777_8).

Remarks:

1) No provision is made for Quick Save to look, if the requested blocks on tape are already in use. The user himself must be cautious not to overwrite useful data (see SET BLOCK PROTECT in section VII, D-8).

2) No command is provided to clear the blocks used by Quick Save on tape. Reuse is possible by simple overwriting of previous contents.

D) COPY SYMBOLIC

Introduction:

Copy Symbolic has the purpose of duplicating programs from one Symbolic Tape onto another. It also provides a listing of the program names from the Index of the Assembler Tape.

Prerequisites:

a) An Assembler Tape.

b) A Symbolic Tape with its Index updated (see E Command in Editor).

c) An empty tape (or second Symbolic Tape) to transfer onto.

Procedure for a Numbered Listing of the Index:

1) Mount Assembler Tape on transport #1, mount Symbolic Tape on transport #4, set to WRITE LOCK.

2) Call Copy Symbolic via the Bootstrap procedure (see section II) by typing COPY SYMBOLIC. Typer types

    PROGRAM IN.
    FROM 2 to 4 COPY NO:

3) Restart computer at location 4. Typer types a numbered list of names of symbolic programs: the Index from the Symbolic Tape (transport #4).

Procedure for Copying a Symbolic Program:

1) Get a numbered listing of the Index.
2) Set transport with Symbolic Tape to #2, WRITE LOCK.
3) Mount empty tape (time and mark tracks written) or second Symbolic Tape on transport #4, set WRITE ENABLE.
4) Start Copy Symbolic at location 100_8. Typer types FROM 2 TO 4 COPY NO:
5) Type the number of the program to be transferred (see listing for the number) then strike the Space bar. Typer types NO:
6) Type number of next program to be transferred, followed by a Space.
7) Several symbolic programs may be transferred one after another. At last the question NO: must be answered by typing Ø followed by Space. This assures that the updated Index is written from core onto tape #4.

Notes:

1) An incorrectly typed number can be deleted by striking the Rubout key.
2) If the name of a program to be transferred exists already in the Index of tape #4, the typer says:

   XXX TO BE OVERWRITTEN BY ZZZ

   IF OK TYPE Y. Type anything but Y if the version on tape #4 is still needed. Recall the Editor and use it to rename the program on one of the tapes.

Remarks:

Instead of starting at 100_8 the operation may be started at 103_8. This avoids the reading of the Index of the second Symbolic Tape into core. Thus the updating of the Index due to the copying starts from zero and, after step 7, none of the old symbolic programs can be retrieved from the tape on #4.

E) DEBUG

Introduction:

Debug is a program for debugging binary programs. Its performance and features are described in the manual "PDP-7 Debug System" (D. Zurlinden and M. Myers, Computer Center LRL, November 1965).

Prerequisits:

a) An Assembler Tape.

b) A User's program (assembled) residing in core memory.

Procedure:

1) Load User's program into core via any applicable method.

2) Mount Assembler Tape on transport #1, set WRITE LOCK.

3) Call Debug via the Bootstrap procedure (see section II) by typing DEBUG.

4) Apply commands as described in the manual: PDP-7 Debug System.

Remarks:

In contrast to DDT, Debug does not give any **symbolic** printout. It has, however, some powerful features as for instance:

Set any desired part of memory to any value including zero,

List any desired part of the program in octal,

Punch a paper tape that can be loaded using the READ IN switch,

Search memory for non-zero words.

F) COPY DECTAPE

Introduction:

To copy a DEC-tape or parts of it onto another, use Copy Dectape. Copy Dectape handles tapes with a block length up to 256_{10} - 400_8 data words. The program consists of two parts: (1) it permits copying data from one tape onto another and (2) it permits comparison of data on two tapes.

Prerequisites:

a) An Assembler Tape or Utility Tape that contains Copy Dectape.

b) A DEC-tape to be copied.

c) A tape with mark and time tracks written.

Procedure:

1) Mount Assembler Tape on transport #1, set to WRITE LOCK.

2) Call Copy DEC tape via the Bootstrap procedure (see section II) by typing COPY DECTAPE. Typer types TYPE C TO COPY, V TO VERIFY.

3) Type one of the two letters. Then typer types

COPY
COMPARE DEC TAPE 2 TO 4 OCTAL BLOCK:

4) Mount tape to be copied on transport #2, mount the tape to be modified on transport #4, set WRITE ENABLE.

5) Type first and last block number of blocks to be copied in octal, each terminated by hitting the Space bar.

Notes:

1) In case of errors the program stops the tapes and types out unique error messages. If a difference is found in Verify mode, the block number and the contents of the two different words are typed out.

2) To copy a tape completely, type block numbers Ø and 11ØØ each followed by Space.

3) To stop tape motion and restart the program, strike the space bar at any time.

Remarks:

1) The program copies data and forward checksums (parity check).

2) Block numbers range from Ø through $576_{10} = 11ØØ_8$ for tapes with the normal block length of $256_{10} = 4ØØ_8$ data words. (Block numbers, checksum and other control marks are stored in extra words on the tape, so that the total length of one block is 266_{10} words.)

3) It is possible to add new blocks of information to an old tape which is partly filled with useful data, s-nce only the block numbers requested will be modified on tape #4.

G) DTOG-L

Introduction:

Dtog-L is a short program that writes Mark and Time tracks on a DEC-tape. The program is developed out of Digital Equipment's program DECtog (see manual DIGITAL-7-2Ø-IO).

Prerequisites:

1) An Assembler Tape
2) A tape, either new or whose previous contents are to be completely erased.

Procedure:

1) Mount Assembler or Utility Tape on transport #1, set WRITE LOCK.
2) Call Dtog-L via the Bootstrap procedure (see section II) by typing DTOG L.
3) Mount new tape on transport #4, set WRITE ENABLE. Do Not move tape forward: writing of the end zone starts at the rest position of the tape.
4) Set switch for Writing Mark and Time tracks on. The switch is located internally in the computer at the right hand side from

the console. It is marked RELTM.

5) Press Continue, or start program at location 116. Typer types END, after tape motion.

6) Set Mark and Time track switch off.

7) Start program at location 117. Tape #4 moves backwards till beginning of tape, then forward till the end and again backwards. Typer types END a second time.

8) Depress the Stop switch at the console to deselect the tape Transport!

Remarks:

1) The program writes the Time track and the Mark track. The Mark track contains different identification codes for the several types of words within each block (e.g. data, blocknumber, checksum).

2) Into the information tracks the program writes the block numbers from Ø through $11ØØ_8$. The block length is 266_{10} words total for 256_{10} words of data.

3) The program also writes the reverse checksums as 777777 (18 bits!) and it fills the data words of each block with the so called virgin tape pattern (i.e. a number increasing with the relative position of the word within each block).

V. A NEW SUBPROGRAM FOR THE PHA-1969-SYSTEM

Introduction:

For a description of the operation and the organization of the PHA-1969-System see section III and Appendix A of this pamphlet. Introducing a new operation into the system is easily possible by adding a subprogram, which then is called by the system upon request from keyboard or remote console switches.

A) CREATING A NEW SUBPROGRAM

1) Write the subprogram in symbolic language. Use mnemonic instructions as described in the PDP-7 User's Handbook; use pseudo-instructions, specific characters and spaces to control the Assembler as described in the programming manual DIGITAL-7-3-S. For further hints see the manual "PHA 1969" (L. Robinson, LRL, October 1968)*). Especially, no numerical origin statement is necessary for the new subprogram.

2) Edit the symbolic program and store it on a Symbolic Tape. For the commands and usage of the Editor see the programming manual DIGITAL-7-1-S and Chapter A) of section IV.

3) Assemble the symbolic program using the subprogram assembler (ASEM SUB PHA). This assembler contains all the tags and symbols in its symbol table, which are used in the basic program PHA-1969. Thus the subprogram, by using tags of the basic program, has access

---

* Appendix A.

to contents of the tagged memory cells. See Chapter B) of section IV for the procedures to use the subprogram assembler.

Note:

With the statements - XXXXXX = .
KBΣ/ JMS YYYYYY
XXXXXX/

inserted in your subprogram, the letter Σ (in this description Σ stands as a representative for a letter) is inserted into the dispatch table and given the program number that you type. Thus after storage is complete, the subprogram can be called by the PHA basic program by typing Σ followed by Altmode. XXXXXX may be any 6 letter code and YYYYYY stands for the tag of the entry of the new subprogram. Of course you choose a letter not yet used in the dispatch table for your own purpose.

B) STORING A SUBPROGRAM ON A PHA-1969 USER's TAPE

1) Mount PHA-1969 User's Tape on transport #1.
2) Place the ff binary paper tape of the previously assembled subprogram in the paper tape reader.
3) Call Store Subprogram via the Bootstrap procedure (see section II) by typing STORE SUB PROGRAM

   Typer types PROGRAM IN, then starts printing a listing of tags and corresponding two digit numbers (octal). These are the currently used subprogram entries with the program numbers (the contents of the dispatch table).

   This printout can be suppressed by depressing the Stop switch, then restarting at address 4.

Typer types LOAD SUB PROGRAM. THEN START AT 5

When program is loaded[1], set address switches to 5 and press Start switch. Typer types -

```
PROG. LIMIT XXXXX XXXXX
TYPE PROG. NO. (OCTAL):
```

4) Set WRITE ENABLE at transport #1.

5) Type a program number in octal (two digits) followed by a space.

Typer types DONE upon successful storage.

Notes:

1) With the subprogram's paper tape present in the paper tape reader, loading is performed automatically.

2) The "PROG. LIMIT" numbers are the core addresses (incl., in octal) occupied by the subprogram. It is advisable to save these numbers by writing them onto the paper tape.

3) Possible program numbers are Ø1 through 27 (octal), 3Ø and 34. Look up the listing of the dispatch table to find out which program numbers are already in use and choose an unused one for your subprogram.

4) The storage program will accept binary programs loaded to core in any fashion. They could even be typed in manually with DDT or Debug, or loaded from magnetic tape.

VI. A NEW USER'S MAIN-PROGRAM

Introduction:

A User's Main-Program is a data processing program which operates on its own, i.e. it is independent of other programs (e.g. independent of PHA 1969). For new subprograms to be used in connection with PHA 1969 see section V.

A) CREATING A NEW MAIN-PROGRAM:

1) Write the program in symbolic language. Use mnemonic instructions as described in the PDP-7 User's Handbook; use pseudo-instructions, specific characters and spaces to control the Assembler as described in the programming manual DIGITAL-7-3-S.

   The most convenient locations in core for the new program, are in the lower half of memory, (i.e. locations $100 - 7777_8$). See page 7.11 for core assignments of the programs of the Software Package.

2) Edit the symbolic program and store it on a Symbolic Tape. See the programming manual DIGITAL-7-1-S together with Chapter A) of section IV for the commands and usage of the Editor. Save an up-to-date listing of the symbolic program.

3) Assemble the program (translate it from symbolic into binary form) using the procedures of Chapter B) of section IV.

4) Read the Assembler's output (ff binary paper tape) into core memory by using the DEBUG$-command of DDT(Q). Before running the program, store it by using the WØA-command of DDT Q.

5) Run your program. - (Don't be surprised, if the program does not perform correctly the first time. Even great geniuses have their weak moments!)

6) Apply the techniques of DDT Q as described in the manual DIGITAL-7-4-S and Chapter C) of section IV for debugging the program.

7) After a satisfactory version of the program has been obtained save it for easy reference by use of a W9A-command in DDT Q or Quick Save. Update the symbolic program on the Symbolic Tape, get a final assembly and a printout of the tags, and test this version again. Save the **binary program paper** tape prepared by the assembler.

B) STORING THE NEW MAIN-PROGRAM ON A GENERAL USER's TAPE

1) Call DDT from an Assembler Tape or General User's Tape and apply ZERO$ and DEBUG$ to clear the core memory and load the new program from paper tape.

2) Mount a General User's Tape on transport #1, set WRITE ENABLE*).

3) Apply the ADD-operation of the Loader as described in section VII.

---

*) For the preparation of a new tape as a General User's Tape see Chapter D-7 of section VII.

VII. THE LRL-PDP-7 BINARY LOADER SYSTEM

Introduction:

The LRL-PDP-7 Binary Loader System has the purpose of transferring binary programs between the core memory of the computer and DEC-tape. It consists of two parts: a Bootstrap loading system and a DEC-tape handling system. The complete Loader is stored in blocks Ø through 6 on the User's Tapes (binary program tapes) and on the Assembler Tapes.

Contents of this Section:

A) Binary Program Tape Formats

B) The Hardware Bootstrap System

C) The Loader System

D) Commands to the Loader System

A) BINARY PROGRAM TAPE FORMATS

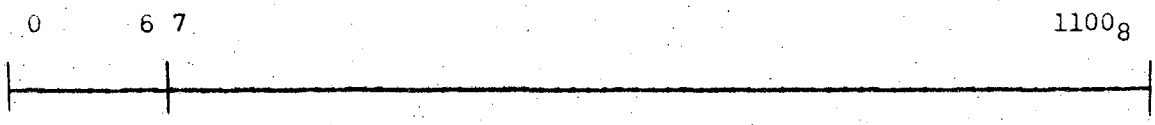
The User's Tapes (General User's Tape, User's Tape PHA 1969) contain 577_{10} blocks of 256_{10} words length numbered from Ø through 576. Blocks Ø through 6 contain the Loader System and the rest of the tape contains the user's programs. The first 13_{10} words of each of these blocks contain an identification: i.e. the title of the program and the addresses in core, where the program is to be stored upon a call.

In addition to this, the Assembler Tapes contain an Index in blocks $64 - 71_{10}$ ($100 - 107_8$). This Index tells the Editor and Assembler what symbolic programs are present on the corresponding Symbolic Tape and

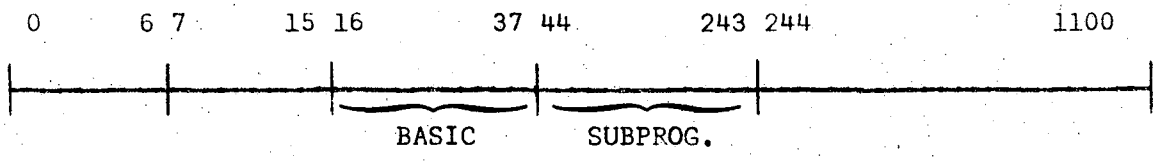
where to find them. Because of this interconnection each Assembler Tape corresponds to only one Symbolic Tape and vice versa.

The PHA 1969 System finds its main program (Basic Program) and subprograms in blocks 16 - 234_8 of the tape. The main program contains a calling system to call subprograms from the tape for each operation to perform. This calling system is made as short as possible (to save memory space for data taking) and is therefore simpler than the calling system of the Loader.

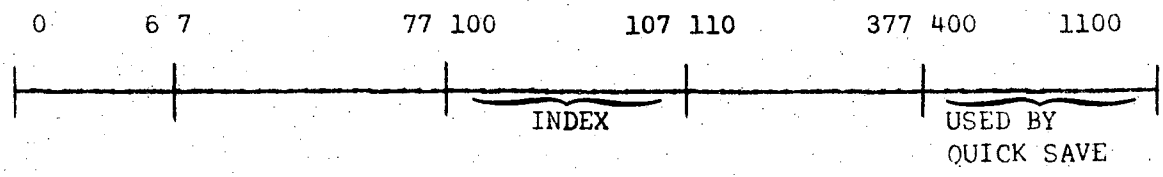
General User's Tape.



PHA 1969 User's Tape.



Assembler Tape.



B) THE HARDWARE BOOTSTRAP SYSTEM

The Bootstrap system (Black switch) allows the computer to be started "cold". Operating the Bootstrap switch loads a small program into core memory by hardware, via the Data Break Channel. The program occupies memory locations 12322 - 1234Ø incl., 17 and Ø. The computer is started as soon as the Bootstrap switch is depressed and a data break request generated. This request can be served only after completion of the current instruction. The computer starts at the location pointed at by the address switches and performs the instruction, whichever it may find there. At the end of this instruction the loading takes place and at last the Program Counter is set to zero at which location a jump to 12322 is loaded. Thus the Bootstrap program is started.

The hardware loaded program selects DEC-tape transport #1, searches backwards for the tape end zone and then reads (forward direction) the contents of block zero of the tape into memory locations 12137 - 12334. The last word it reads is a JMP 12140, which causes control to be transferred to the newly loaded program from Block Zero. The tape stays in read-forward mode.

The program from block zero handles the read-in of the Loader program from blocks 1 - 6 incl. and exits into the "WHAT\" system. The Loader program occupies the memory locations, 10000 - 12127 incl.

During reading of blocks 1 - 6 a parity check*) is performed, which, in case of an error, halts the tape and the computer displaying the two checksums in the AC- and MQ-lights.

---

*) The parity check or "checksum" is a PDP-9 compatible 6-bit figure on tapes generated later than January 1969.

Remarks:

As the computer always performs one complete instruction before the hardware bootstrap program is loaded and operated, caution is necessary, that this instruction does not do any damage to other programs or data. The simplest failure occurs if this instruction is a HLT: then the computers halts and the data break request is not served. It is advisable to operate the Bootstrap switch only with transport #1 in the WRITE LOCK condition, or set the address switches to a location whose content is known to cause no trouble (NOP etc.).

C) THE LOADER SYSTEM

The Loader System with the exception of the bootstrap part resides on the binary program tapes (General User's, PHA 1969 User's, and Assembler Tapes) in blocks NO. 1 - 6 incl. It is loaded into core memory via the bootstrap procedure*) to occupy locations 10000 - 12127 incl.

The purpose of the Loader is to control the transfer of binary programs from and to core memory and DEC-tape. In order to do so, the Loader must know what programs are present on the tape and where they are. This information is stored in the Loader in two buffers (areas of words). One of these buffers simply keeps track of what blocks of the tape are occupied. The second buffer contains identification words for the programs; one word for each program. Up to 63_{10} different program names can be stored in the identification buffer. (Of course no further program name is accepted,

---

*) See Chapter B) of this section.

if the present programs are so long that they need all the available space on the tape; even if their total number is smaller than 63).

The identification buffer of the Loader contains only one identification word for each program. The program's storage blocks on the tape contain all further information about the program: the complete name, first and last locations in core memory, the starting address and the relative block-number of the program. The first 13_{10} words of each block are reserved for this purpose.

Due to this construction, it is not necessary to have the programs stored on tape in consecutive blocks. A program may be distributed over the tape, part of it at the beginning and the rest somewhere else. In fact the Loader stores a new program on tape wherever space is available.

D) COMMANDS TO THE LOADER SYSTEM

The Loader System responds to the following commands:

PRINT	LIST DEC TAPE
ADD	DELETE
REWRIT	MODIFY
WRITE LOADER	SET BLOCK PROTECT

The commands are given in the same way as programs are called from the tape: i.e. type the name followed by carriage Return or Altmode (see section II). The programs for the commands PRINT, ADD, DELETE, REWRIT and MODIFY are included in the Loader (blocks Ø - 6) and directly available after WHAT \. The programs for the commands LIST DECTAPE, WRITE

LOADER and SET BLOCK PROTECT are stored separately on the tape and must be read into core like every other program. Of course they must be present on the tape in order to be successfully called. WRITE LOADER and SET BLOCK PROTECT will work only if the new Loader (after January 1969) is present in core memory.

In detail the functions are:

1) To get a listing of the names of all programs present on the tape, call:

   PRINT. The computer returns to WHAT\.

2) To get a listing of the names and addresses (incl., octal) of all programs present on a DEC-tape, call:

   LIST DECTAPE. The computer returns to WHAT\.

   In addition the program types out the block numbers (incl., octal) at which the programs are stored on the tape.

3) To store a program on DEC-tape, which is present in core memory, call:

   ADD*). Set the tape transport to WRITE ENABLE. The computer asks for the name of the program and the first and last locations and the starting address. The name may be a maximum of 24 characters long. The addresses must be typed in octal. All answers are to be terminated by a carriage Return.

---

*) If before "adding" a program, the Loader has to be recalled by the bootstrap procedure, caution is necessary, that the instruction pointed at by the address switches does not do any damage. Set the address switches properly, for instance to the start location of the program to be "added".

The Loader places the new program at the first available space on tape going forward from the present location and then rewrites its own Index Buffer. It is therefore advisable to recall the Loader via the bootstrap procedure before requesting the ADD operation. The computer returns to WHAT\.

It is convenient to label a tape with, e.g. a title, the user's name and date. To do so, perform an ADD operation on the new tape with all locations defined to Ø. This procedure should be done with a new tape (with the exception of the Loader, no programs stored previously). Otherwise upon a PRINT request the label would not occur on the first place.

4) To delete a program on DEC-tape, call:
DELETE*). Set the tape transport to WRITE ENABLE. Computer asks for program's name; terminate by carriage Return. Only the identification and storage buffers in the Loader are updated. The program is not cleared to zero on the tape, but it can no longer be found by the Loader. The tape spins during execution of the delete command in order to find out which blocks on tape are occupied by the program. The computer returns to WHAT\.

---

*) See footnote on page 7.6

5) To update a program on DEC-tape, call:
REWRIT*). Set tape transport 1 to WRITE ENABLE. The computer asks for the program's name, which cannot be changed while updating. The Loader assumes that the first, last and starting locations do stay the same. Upon completion the computer returns to WHAT \.

6) To perform a more complex updating of a program on DEC-tape, call:
MODIFY*). The operation is a combined DELETE and ADD. However, the program's name cannot be changed while updating.

7) To store the Loader on a DEC-tape, call:
WRITE LOADER. The program halts after typer types PROGRAM IN. Mount the new tape on transport #1, set WRITE ENABLE. Depress the Continue switch to run the program. The Loader is stored on the tape in blocks Ø through 6 and blocks 7 through 1100_8 are erased to zero. The reverse checksums (18 bit 777777) are not effected. The program returns to WHAT \.

8a) To prepare a new Assembler Tape call:
SET BLOCK PROTECT**). Mount the tape to be prepared (mark and time tracks written) on transport #1, set WRITE ENABLE. Type Y (for yes) after the first question of the typer. The program writes the Loader and clears all the rest of the tape.

---

*) See footnote on page 7.6

**) See footnote on page 7.9

The new Loader is directed not to ADD any programs into blocks 100 - 107_8 (the area of the Index of the names of symbolic programs) and into blocks 400_8 - 1100_8 (the area for the use of Quick Save).

8b) To prepare a new PHA 1969 User's Tape, call: SET BLOCK PROTECT*). Mount the tape to be prepared (mark and time tracks written) on transport #1, set WRITE ENABLE. Type anything but Y after the first question of the typer and answer the second question by typing Y (for yes). The program writes the Loader (block Ø - 6) and clears all the rest of the tape. The new Loader is directed not to ADD any programs into blocks 40 - 243_8 (the area of the PHA 1969 subprograms).

Note:

To store the PHA basic program into blocks 16_8 etc., as is required to make Store Sub Program work properly, procede as follows:

After completion of step 8b) ADD a title to the tape (PHA 1969, your name and the date) typing the core locations as all Ø. Then store a dummy program on tape by ADDing: RUBBISH, first Loc.=Ø, last Loc.=25ØØ, start loc.=Ø. This will occupy blocks 10 - 15_8. Now ADD the PHA program (basic only) and afterwards delete RUBBISH.

---

*) Copy DEC-tape might be used as well, if the whole tape is copied.

8c) To preserve any area on a program tape against access of ADD, call:

SET BLOCK PROTECT. Mount the program tape (User's Tape or Assembler or Utility Tape) on transport #1, set WRITE ENABLE. Answer first and second questions both by typing anything but Y. Answer the following questions by typing the block numbers (inclusive, octal; both the same is legal) terminated by Return. Typer says DONE. If any block of the area to be protected has already been used via an ADD (or else 8a) or 8b)) the typer will so state. The tape is not cleared nor the Loader written.

Note:

This feature will work only if the User's Tape contains the new Loader (after January 1969).

	PROGRAM			BUFFERS	
	FIRST LOC.	LAST LOC.	START LOC.	FIRST LOC.	LAST LOC.
LOADER	10000	12130	10000		
WRITE LOADER	7000	7577	7000		
SET BLOCK PROTECT	6400	7577	6400		
LIST DECTAPE	100	1377	101	1416	17777
DEBUG	17000	17677	17000		
DDT	14200	17777	16000	downward	14200
EDIT T	Ø and 12400	5464 and 15577	20	5465 and 14500	up and 15577
ASEM	Ø and 14335	7777 and 17777	7770 or 20	downward and 10000	17777 and 11477
ASEM SUB PHA	Ø 12400	7777 17777	7770 or 20		
COPY SYMBOLIC	Ø	1377	100/103 or 4	1377	17777
QUICK SAVE	7300	7777	7300	Ø and 10000	7277 and 17777
COPY DECTAPE	Ø	1077	100	1100	17777
DTOG L	Ø	7577	116 and 117	all of memory	
PHA	Ø	2377	4 5	Memory Protection applies and	
SUB PROGRAMS OF PHA	2400	4000		17400	17777
STORE SUB PROGRAM	17600 Ø	17777 2000	3 4 5		

VIII. APPENDICES

A. Programmer's Information PHA 1969

This is a revised version of earlier data handling programs written for the PDP-7. It's main advantage over previous versions is an increase in the amount of both data space and program space for the 8192 word PDP-7. The maximum available data area has been increased to 6400_{10} words of core memory from the previous 4864_{10} and the flexibility of data taking and analysis programs has been simultaneously improved. Addition of special purpose programs has also been made easier.

PHA 1969 allows simultaneous use of the PDP-7 for data taking during plotting, printing, data storage and recall, and simple data analysis.

In earlier versions, all of the program resided in main memory at once, forcing programming compromises to conserve the severely limited data space. In "PHA 1969", the core memory holds only a basic program loader, CRT display routines, ADC counting, gain control and remote-console control routines, and a Dec-Tape read routine. A buffer of about $(500)_{10}$ words is then used in common for special calculating and input-output routines, which are automatically called from DEC-tape when needed. The length of this buffer can be extended up to nearly $(4000)_{10}$ words for special programs by temporarily restricting the data area.

Loading and running of programs occurs automatically when a keyboard or switch command is given. The monitor program prevents loading of programs which overlap a data area. Thus up to $(6400)_{10}$ words could be used for data, using only minimum length data storage programs; other longer routines would refuse to load if accidentally requested, without affecting data taking and display.

BASIC FUNCTIONS:

The basic system resident in core memory includes ADC-START, ADC-STOP, DATA ERASE, 8 Gain-Stabilizer controls, plus subroutines for teletype message output, instruction input, and remote console display functions. A real-time clock, Dectape subroutines and interrupt handling routines are also present in memory at all times.

The only vital components of the system are the Dectape handling routines and program loader. The others are included for ease in programming of nuclear spectroscopy data-handling problems.

PROGRAMMING:

PHA 1969 has spaces provided for up to $(31)_8$ subprograms. Each is stored and retrieved by number only. Since less than $(20)_8$ of the spaces have been used, subprograms can readily be added to this system. New routines are stored on Dectape almost automatically by a program on the System Dectape called STORE SUB-PROGRAM. This program expects to have a new subprogram loaded to core addresses between $(2351)_8$ and $(7777)_8$. Programs extending beyond $(3777)_8$ can be stored only as program numbers 30, 34. Other subprogram numbers must be from 1 to 27 (octal).

Operator control of subprograms is from the keyboard or from switches on a remote console. The "Monitor" program (see "Call Sub-Program") finds the subroutine address in the dispatch table, and calls the subroutine by a "JMS I" instruction if the subprogram is in core memory; or loads core memory from Dectape and then calls it.

STORE SUB-PROGRAM expects to find a pointer (or pointers) to the new subprogram in dispatch table locations, and will assign these dispatch locations to the new program if not already used. For ecample, a subprogram called CALCUL is to be loaded and started whenever the keys J and then ALT MODE are typed. The subprogram might start at location 2600. An entry would be placed at location 52. (Symbolic name: "KBJ" is assigned in PHA R POINTERS to location 52.)

```
2600/  CALCUL,  0
                |
                |
                |
                (body of subprograms)
       JMP I CALCUL  /Final Exit
KBJ/   CALCUL  /Entry Point
```

Procedure:

A) TO PREPARE A NEW SUBPROGRAM:

1. Create the new program in PDP-7 machine symbolic code. End it with a PAUSE statement.

2. Call assembler program ASEM SUB from Dectape. (ASEM SUB was created by calling the Dectape Assembler, assembling the basic PHA 1969 system, then storing the assembler as ASEM SUB. The "Dectape Assembler" is identical to the basic PDP-7 assembler provided by DEC, but the Input is from Dectape instead of paper tape. It recognizes all tags in the basic PHA 1969 system, so that calls to basic system subroutines or interrupt exits can easily be included in the new subprogram. ASEM SUB will automatically locate the new subprogram at the end of the basic system, so no origin statement is necessary.

3. START ASEM SUB at 20 to assemble a single program or at 7770 to assemble a master list from Dectape.

   If ACS bit 9 is up, the assembler will take its symbolic input from Dectape #3 using the address index from Dectape #1. If bit 9 is down, assembler input is from paper tape (ASCII code). See LRL Dectape Program Editor and Assembler, November 1966.
   To insert a keyboard call to "NAME", e.g. from letter "J": insert the following code anywhere in the subprogram.

```
CURREN = .          (or any 6 letter code)
KBJ /  NAME
CURREN/
```

B) TO PLACE A NEW SUBPROGRAM ON USER'S TAPE

1. Call STORE SUBPROGRAM - if program is in paper-tape ff binary format, place it in tape reader.
2. Press continue, or start at 3, for a printout of all previously used entry points.

   or

3. Start at 4 to store the program. Typewriter types a message.
4. If program is in paper tape reader, loading starts at once. Otherwise program can be loaded by any suitable method such as DDT. STORE SUBPROGRAM occupies locations below $(2000)_8$ and only words in the dispatch table, or at addresses above both it and the basic system should be altered during program loading. Currently, the highest address in the basic system is 2350. By deleting certain functions, a shorter basic system could be achieved. In any case, the first free address after the loader would be the lowest possible address for subprograms.
5. Set WRITE ENABLE on Dectape 1, and start at address 5. Answer program-number question with a number 1 --- 27, 30, 34 (octal). When storage is finished typewriter types DONE.

C) TO MODIFY AN EXISTING SUBPROGRAM

1. Call QUICK SAVE using "WHAT" Library call, on User's Master Tape.
2. START at 10000; call STORE SUB PROGRAM.
3. START 7300 - Type RØD (loads DDT including all PHA 1969 tags).
4. START at location 4. Message is Typed.

5. START DDT at 16000.

6. Load binary paper-tape version of existing subprogram with DDT by typing DEBUG$ (Return). (This step could be omitted if a short feature is to be added under a new subprogram number.)

7. Type patches, added dispatch entries, etc., using DDT. (Type "LASWOD:" to find the last used address in basic system.)

8. Start at Location 5, with Tape #1 write Enabled.

9. When computer asks for program number, type an octal number between 1 and 27 - followed by space. Higher numbers in general should be used. A low number can be used at the cost of losing a currently existing subprogram from the tape.

10. If any dispatch location used by the new subprogram is already used, the Loader will so state. These dispatch commands will, however, be stored with the subprogram, and will become active when the subprogram is called.
    Each new subprogram must have at least one dispatch location that has not been used before, otherwise the subprogram could never be called.

11. Some dispatch locations are used by several subprograms. The subprogram most recently in memory and which uses that location will be recalled when such a dispatch location is used. Thus, each typewriter key and each console switch position should normally call only one function. Otherwise the function called by one key will be different, depending on past history.

12. When finished, call COPYDT. Type Ø Return 6ØØ Return, to copy user's Master Tape as Tape 2 to user's operating tape as Tape 4.

Programming Notes:

MEMORY ASSIGNMENTS

The basic system resides in the memory area 1 - 2350. Subprograms are loaded immediately above; in most cases reaching to below 3400. Some subprograms use memory as high as 3777.

The memory area from 3000 to 17377 can all be used for data; subprograms that need memory that is currently assigned (by memory protect switches) to data will not load if accidentally called.

The area 17400 to 17777 is used as a general buffer area. See PHA R POINTERS for exact assignments.

When data is stored on Dectape, the first block stores location 17400 to 17777. When reading back, only the message buffer 17400 to 17437 is used. When copying to IBM tape, 17400 to 17543 is used as a buffer.

PROGRAMMING NOTES

SPECIAL LOCATIONS IN BASIC PROGRAM:

A number of system constants can be used, and in some cases modified, by sub-programs.

-Decimal numbers typed on the keyboard n_8, n_7 ---, n are stored respectively in locations DELTA8, DELTA7, --- DELTA, (See PHA R POINTERS).

-The most recent keyboard entry is stored in KEYINP.

-The preceeding 3 non-decimal characters are stored in LETTER.

-Current data origin and display origin of active Remote console are found in ORGDAT and ORGDIS. ORG1 to ORG1 + 7 holds the data origins as set on the memory protect box.

-Addresses in core memory of Marked channels are found in MARTAB to MARTAB +11. These are sequenced in ascending order by JMS ORDERN whenever an operation is initiated from keyboard or switches.

-NUMSWT contains the Number of Channels switch reading on currently active remote console.

-NUMBER holds the number of channels being displayed.

-QUEUE is loaded with a bit corresponding to an ADC which has timed out.

-CONSV1, CONSV2 hold the bit code from the currently active remote console.

-ADCREG holds bits corresponding to currently enabled ADC's.

- PRSTAT, DTSTAT, TTSTAT, IBSTAT, CLSTAT hold the status of sub-program, Dectape, teletype, IBM tape and Calcomp. Any zero bit indicates an active or busy status.

Special Information Words:†) Dectape and IBM Tape.

Word $(62)_8$ - X and Y display scales (see subprogram Dectape Save).

$(63)_8$ - Zero offset; display scales.

X Scale occupies lowest 9 bits

:1 for 1024 channel display

:2 for 512 channel display; etc.

Y Scale occupies upper 8 bits*)

8* BIT OCTAL CODE	COUNTS FULL SCALE	*8 BIT OCTAL CODE	LOG
1	- 1	4	- one cycle
2	- 2	10	- two cycle
3	- 5	14	- three cycle
5	- 10	20	- four cycle
6	- 20	24	- five cycle
7	- 50	30	- six cycle
11	- 100		
12	- 200		
13	- 500		
15	- 1000		
⋮			
33	- 5×10^6		

†) This is to allow automatic setup of plotting scales on data sent to the computer center.

Programming Notes:

DATA TAKING SUBPROGRAMS

There are several special purpose locations in the basic program which can be loaded by a subprogram for calls at regular intervals, or for data-input interrupts. These are labeled CLFUN2, CLFUN3, and ADCFLG, SPFLAG, DSBACK, GETWY and BRIGHT.

The first 2 are found in subroutines ADC CLOCK, and are executed once per second. The last 2 are in subroutine INTEX and are executed during the flag search cycle that occurs for interrupts.

Instructions loaded in GETWY, BRIGHT (Display Subroutines) and DSBACK (PHA Main) will be executed with interrupt ON. I/O should not be initiated from these locations unless the entry point is protected from a second call before the first is done. If an exit via DISMIS is to be used, first load location Ø with contents of DISGOP. Either GETWY or BRIGHT is executed just before each CRT display point. DSBACK is executed once for each 6 display sweeps.

PROGRAM CONTROLLED DATA-TAKING: (Quasi Permanent Subprograms)

Normally, each subprogram is overwritten by the next one to be called. However, some applications such as program-controlled data-taking demand that parts of a subprogram remain permanently in memory. It is still desirable that other functions such as printout, plotting, and data-storage remain available.

A subprogram can gain quasi-permanent status by limiting the area of memory available to other subprograms.

Since it is then secure against being overwritten, JMS calls to the subprogram can safely be inserted at the high priority locations CLFUN3 and ADCFLG. These subroutines should be short; not lasting for many milliseconds at one time. If they use an interrupt exit, TYPEXT for typing as an example, the subroutine should inhibit its

entry point before starting the operation; otherwise a second call may occur before the first has been completed. Normally any such quasi-permanent subprogram should be written to terminate itself and remove any special entry points when a $ sign is typed. (See following examples.)

A typical quasi-permanent subprogram could run as follows. References to locations in the basic program are underlined. Standard PDP-7 assembler language is used throughout.

```
KBJ/  SUBPRO

3600/          /Origin of quasi-permanent program.
     SUBPRO,0 /Called from keyboard by typing J ALTMODE.
     LAC (SUB2)
     DAC MINORG  /Quasi-permanent status.
     LAC (JMS SUB2)
     DAC CLFUN3
     LAM-143
     DAC SUBCNT  /Set up a 100 second delay interval.
     JMP I SUBPRO

SUB2, 0          /Entered once per second from "ADC CLOCK".
     ISZ SUBCNT
     JMP I SUB2
     LAM-143     /Repeat every 100 seconds
     DAC SUBCNT
      |
      |    Operation - (data storeage, type out, etc.) which
      |                lasts less than 100 seconds.)
      |
     JMP I SUB2
```

LOW PRIORITY ENTRY:

CLFUN2, SPFLAG can be used with few precautions, as the entries are killed when a new sub-program is called. A suitable format is as follows:

```
SUBPRO, 0                  /Entered by a keyboard instruction.
    LAC (JMS SUB2)
    DAC CLFUN2
    LAM
    DAC SUBTIM
    JMP I SUBPRO
```

At the next one second interrupt, the instruction JMS SUB2 will be executed.

```
SUB2, 0
    ISZ SUBTIM
    JMP I SUB2
    LAM - 1750
    DAC SUBTIM             /Set a 1000 second interval.
    LAW DTSTAT             /Test I/O device, Dectape in this example.
    JMS STATST
    JMP SUBXIT             /Device-busy-return, from STATST
    JMS PRSETS             /Device free, set program status busy
     |
     |  Operation such as Data Storage on Dectape.  It could use
     |  system Dectape sub-routines, which are always in core memory.
     |
    JMS PRDONE             /Clear Program Status
    LAW DTSTAT             /Clear Dectape Status
    JMS STAFIN
    JMP I SUB2
    SUBXIT, LAM-3          /Device Busy
            DAC SUBTIM     /Try again in 4 seconds.
            JMP I SUB2
```

This program will be overwritten if any sub-program function is called by a keyboard or switch command.

A TYPICAL SUB-PROGRAM TO READ DATA FROM AN INTERRUPTING ADC

In this example, the program is initialized by typing Q ALTMODE; data taking starts when "(" is typed. Data taking stops when ")" is typed. The program space will be released if "$" is typed, or if the master reset KILL Altmode is typed.

If a sub-program requiring memory above address DATFUN is called (while this sub-program is operating) "MEM. BUSY" will be typed. Likewise, if any ADC data field (as defined by the switches in the Memory Protect box) overlaps the area required by this sub-program, the program cannot be called, and MEM. BUSY will be typed.

```
KBQ/    JMS DATSET
        (type Q Altmode to call the program.)

3300/
    DATSET, 0
        LAC (JMS DATFUN)
        DAC CLFUN3          /Enter DATFUN once per second.
        LAC (DATFUN)
        DAC MINORG          /Protect this program
        JMP I Datset

    DATFUN, 0               /Entered once per second.
        LAC KEYINP          /Most recent keyboard input.
        SAD (244            /Keyboard $ code
        JMP ADEXIT          /Kill sub-program for $
        SAD (250            /"(" keyboard code
        JMS ASTART          /start data taking
        SAD (251            /")" keyboard code
        JMS ASTOP           /Stop data taking
        JMP I DATFUN
```

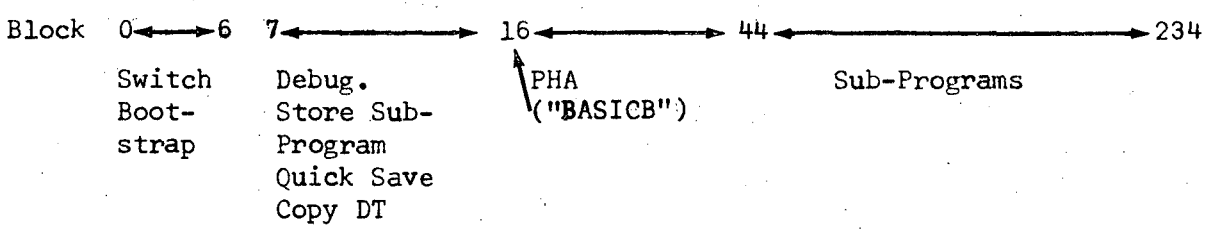
```
ADEXIT, LAC MINSAV   /Final Exit
        DAC MINORG   /Restore normal field limit
        JMS ASTOP
        LAC NOPCOD   /NOP code
        DAC CLFUN3   /Delete entry point
        JMP I DATFUN

ASTART, 0
        LAC (JMS ADCTES)
        DAC ADCFLG   /High priority interrupt
        EXION        /Enable ADC interrupt
        JMP I ASTART

ASTOP, 0
        LAW
        DAC ADCFLG
        EXIOF        /Disable ADC interrupt *
        JMP I ASTOP

ADCTES, 0
        ADCSF        /Test ADC flag
        JMP I ADCTES
        A1READ       /Read ADC
        A1REST       /Reset ADC
           |
           Analyze and/or store
           ADC CODE
           |
        JMP I ADCTES
```

* Pressing CNTRL and Bell keys would also stop data input as the CAF, caused by this operation, produces EXICF.

PROGRAMMING INFORMATION - DECTAPE FORMAT

PHA 1969 USER'S TAPE

1. DECTAPE ULILIZATION:



2. SUB-PROGRAMS:

a) First Block Numbers = (Prog. No.) x 4 + $(16 + 22)_8$

b) Tape Format:

Word	Content	
1	Dispatch Table Address	First Entry
2	Dispatch Table Contents	
3	ID Code (=252725) + Prog. No.	
4	Dispatch Table Address	Second Entry
5	Dispatch Table Content	
6	ID Code + Prog. No.	
7	etc.	
171	LAW DTCONR (See "Dectape Subroutines")	
172	- (Word Count +3) for sub-program	
173	ID Code + Prog. No.	
174	LAW DATA (Storage pointer for Dectape)	
175	First Address of sub-program.	
176	ID Code + Prog. No.	
	Body of Sub-Program	
$(2000)_8$	- END	

Read in Sequence:

```
Wait,----
      MMRD   /read a word
      DAC I DATA
      1SZ DATA
      1SZ DTCONR
      JMP Wait(for next
      EXIT         word)
```

B. Subprogram "Timed Data Save"

TYPICAL SUBPROGRAM

- This is a Subprogram of PHA 1969.
- Stores 1024 channels of data on IBM tape, erases data in memory and restarts counting every t seconds. (Counts for t-1 seconds.)
- Started by command t TEMB (Altmode) or t TEMG.
- Stopped by typing $.
- Prints out live time in milliseconds for two ADC's, real time in seconds and the areas under two peaks in the spectrum, divided by 10.
- Background is a triangle for NaI, and two straight line segments for Ge spectra.

USE: (Operation of ADC's 1, 2 using consoles 1, 2 is described here. Other combinations can be used.)

1. Mount pre erased IBM tape (write ring in).
2. Set memory protect switches as shown on Switch Setting TDS1 card.
3. Mount 512 channel ADC program plugs in ADC'(s).
4. Connect ADC multiplex gates to consoles 1 and 2.
5. Call program PHA from PHA 1969 tape.
6. Type 1, 2, 10 ADCX (Altmode). (This allows control of both ADC's from either console 1 or 2.)
7. Set 8 markers on CRT display: [9 Marks for TEMG]
   a. Channel Ø ADC 1 (Live Time 1)
   b. Lower Limit of peak for Area 1.
   c. Upper limit of background triangle.
   d. Upper Limit of peak for Area 1.
   e. Channel Ø ADC 2.

f. Lower Limit of peak for Area 2.
g. Upper Limit of background triangle.
h. Upper Limit of peak for Area 2.
i. Upper Limit TEMG.

} See sketch for TEMG

8. Set Q switches to 2-3-4 and press Q switch. Execute light should come on and go out. Marked channel numbers will be printed out. Release Q switch. (This stores the markers.)

9. Type t TEMB (or t TEMG) (Altmode): t is counting and storage cycle time in seconds; counting time per cycle is t-1 seconds. (Computer must be under control of console whose data address is lowest.)

10. Set FUNCTION switch to START and press COMMAND button to start counting. This can be done during the printout that follows TEMB command.

11. To terminate experiment, press shift key and type $.

12. To rewind tape, type REWI (Altmode). Any run can then be recalled by typing n GETI where n is the run number.

13. To continue after rewinding, type n GETI to read the last run, then type t TEMB again. Storage will continue from end of last run.

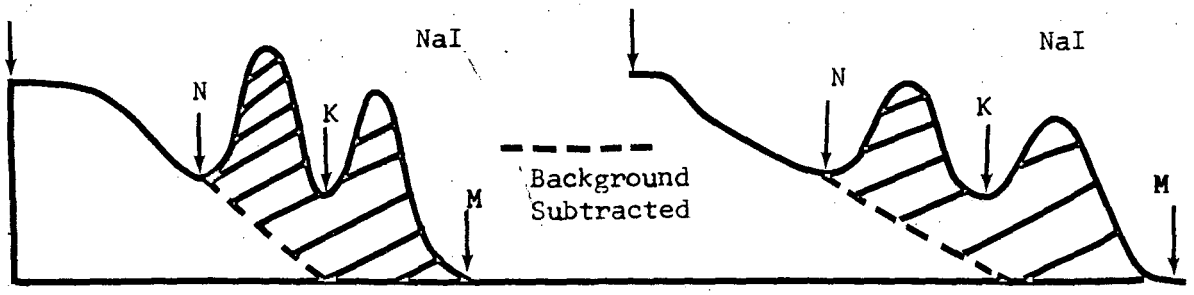
14. Markers can be recalled by setting Q switches to 2-2-4 and pressing Q switch.

15. Type n, m MOVM to move marker m to channel n. Then go to 9 (above).

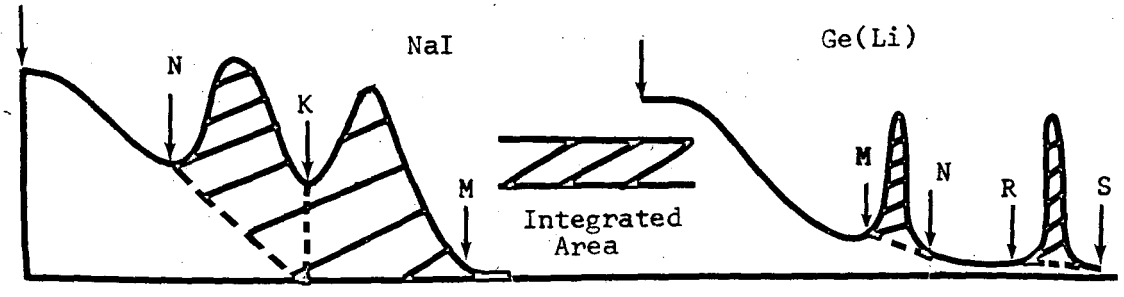
16. Type all new markers by n_1, n_2, ... n_8 SETM. Then go to 9.

NOTE: Typed instructions (except $) are only executed when Altmode key is pressed.

MARKER ↓ SETTING FOR TEMB (8 MARKS)



TEMG \ (9 MARKS)



BACKGROUND CALCULATION

A) NaI Spectra:

2nd. marker is at lower edge of area to be integrated. (Channel N).

4th. marker is at upper edge of area.

3rd. marker at channel K

Area is integrated over M - N channels.

Background: $\frac{C(N) + C(N) + C(N-1) + C(N+1)}{4} = A$

$$\text{Background} = A \times \frac{(K-N)}{2}$$

Note: C(N) means counts in channel N.

B) Ge Spectra:

Separate Integration

for N - M channels and for S - R channels.

Background for 1st. peak:

$$A = \frac{C(M-1) + 2 \cdot C(M) + C(M+1) + C(N-1) + 2 \cdot C(n) + C(N+1)}{8}$$

$$B.G. = (N-M) \cdot A$$

Similar procedure for second peak

AREA2 is sum of two Ge peaks!

TECHNICAL INFORMATION DIVISION
LAWRENCE RADIATION LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720