PDP-8 FAMILY
PAPER TAPE
SYSTEM USER'S GUIDE

For additional copies order No. DEC-08-NGCC-D from Program Library, Digital Equipment Corporation, Maynard, Mass. Price $3.00

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

1st Printing August 1968
2nd Printing January 1969
3rd Printing March 1969
4th Printing June 1969
5th Printing September 1969
6th Printing January 1970

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

CONTENTS

CONTENTS (Cont)

SECTION 3
DECTAPE

SECTION 4
DISK MONITOR SYSTEM

CONTENTS (Cont)

SECTION 5
SYSTEM DEMONSTRATIONS

ILLUSTRATIONS

ILLUSTRATIONS (Cont)

PREFACE

The purpose of this guide is twofold: (1) To familiarize the new user with the PDP-8 family of computers and many of its input/output devices, and to serve as a useful reference for the experienced user. (2) To furnish precise instructions on how to load, execute, and operate DEC's system and utility programs with and without extended memory.

The first section explains the use of switches and indicators on the computer, Teletype, high-speed paper tape reader/punch, and DECtape transport consoles, and other general information. Precise operating procedures for using DEC's software is found in the second section. Subsequent sections cover the use of paper tape with the DECtape transport unit, the Disk Monitor System, and demonstrations of the system in use. The appendices include a glossary of terms, and other useful information.

Unless specified, flowcharts apply equally to the PDP-8/I, 8/L, 8, and 8/S computers, as does the text even though reference is usually made only to the PDP-8/I.

Details on peripherals not covered in this guide can be found in the PDP-8/I and PDP-8/L User's Handbook (ABM), and other DEC publications referenced herein.

SECTION 1

INTRODUCTION

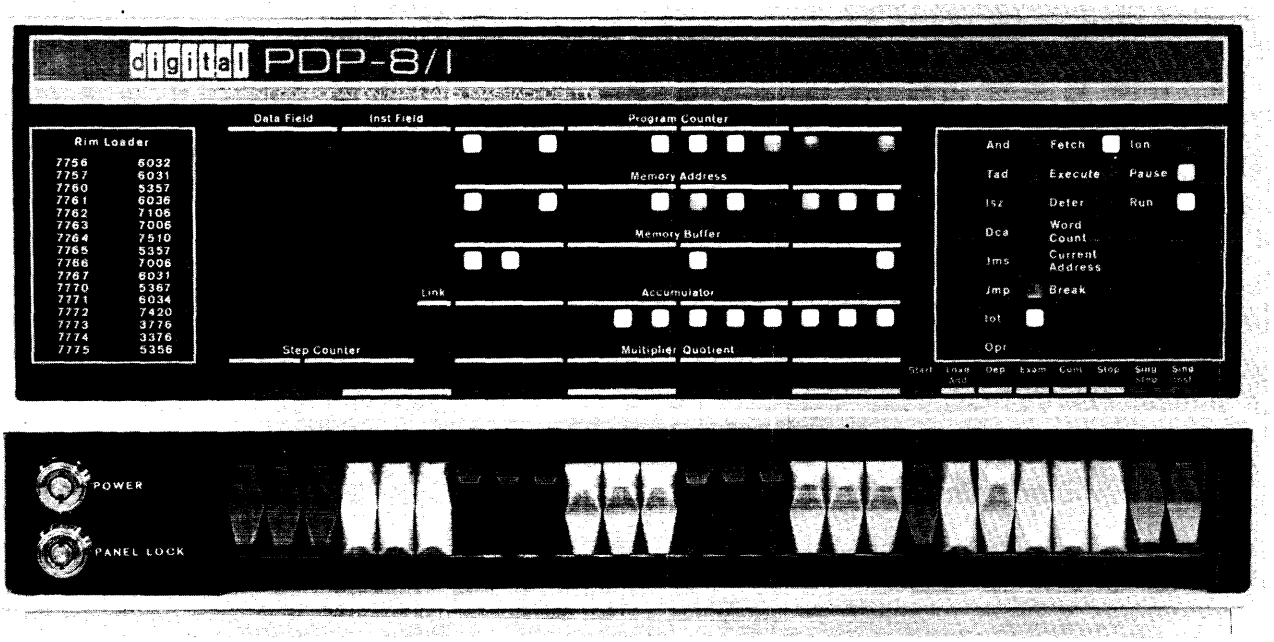


Figure INTRO-1 PDP-8/I Computer Console

CONSOLE SWITCH POSITIONING

PDP-8/I — When the top of a switch is out (push the bottom of the switch in) it represents a binary 1 or is considered set, conversely, when the bottom of the switch is out it represents a binary 0 or is not set.

PDP-8, 8/S, and 8/L — When the switch is up it represents a binary 1 or is considered set, conversely, when the switch is down it represents a binary 0 or is not set.

Note: PDP and Programmed Data Processor are registered trademarks of Digital Equipment Corporation.

INTRODUCTION

This System User's Guide is intended for use at the PDP-8/I computer[1] console when operating with DEC's software. The purpose of the guide is to furnish the user precise instructions on how to load, execute, and operate DEC's system and utility programs with and without extended memory, DECtape, and DECdisk[2].

This section briefly defines each program in Section 2, and familiarizes the new user with the switches and indicators on the computer console, as well as the controls, keys, and switches on the Teletype, high-speed paper tape reader/punch, and DECtape transport consoles. Subsequent subsections describe the various paper tape formats, the procedures for initializing the computer and input/output (I/O) devices, a list of the abbreviations used herein, and other general information. For the experienced user, this section serves as a handy reference.

Section 2 is indexed by system and utility program. The purpose of each program is given, followed by specific core memory requirements and program origin, possible optional hardware, step-by-step instructions (detailed in flowcharts) for loading, executing, and operating the program, summaries of on-line commands, error messages, and other useful information.

Section 3 covers the use of DECtape. The purpose of the TC01/TU55 Bootstrap Loader and Library System are briefly described, and their operation is explained. The five permanent Library System subprograms are defined, and where applicable, system questions are shown, explained, and answered.

Section 4 explains the purpose of the Disk System Builder and briefly describes the library of system and utility programs presently available with the Disk Monitor System.

Section 5 contains three runnable programs to demonstrate the ease with which DEC's system, utility, and service programs can be used. The new user can familiarize himself with his PDP-8/I and DEC-supplied software by duplicating the demonstration programs.

1 Unless specified, reference to the PDP-8/I computer also applies to the PDP-8, PDP-8/S and PDP-8/L computers.
2 DEC's disk system is thoroughly documented in the PDP-8/I Disk Monitor System (DEC-D8-SDAB-D).

The appendices include a list of the USASCII (USA Standard Code for Information Interchange) Character Set, a short glossary of terms, and a summary of the presently available PDP-8/I subroutines.

HOW TO USE THE GUIDE

The user should be acquainted with the material in Section 1 before attempting to operate the system, and then the guide should be made available to the user during system operation.

The programs herein are arranged generally in their order of use, i.e. first the loaders, then the editor, then assemblers and compilers, and then the debugging programs. Therefore, as the user progresses from one phase of operation to the next he will also progress from one section of the guide to the next. By scanning bleeds on the right margins, the user has fast access to the essential operating information for any system or utility program.

SYSTEM AND UTILITY PROGRAMS

All system and utility programs require at least a 4K PDP-8/I computer with an ASR33 Teletype, and can utilize a high-speed paper tape reader/punch, with the following exceptions.

a. The HELP Bootstrap tape of the HELP Loader is read into core using the low-speed (Teletype) reader only.

b. 8K FORTRAN requires at least an 8K PDP-8/I computer with a high-speed paper tape reader/punch.

Each program in Section 2 is briefly defined below.

Read-In Mode (RIM) Loader, used to load into core memory programs punched on paper tape in RIM format (see PAPER TAPE FORMATS). It is primarily used to load the BIN Loader. (See DEC-08-LRAA-D.)

Binary (BIN) Loader, used to load into core memory programs punched on paper tape in BIN format (see PAPER TAPE FORMATS), which includes the user's object programs and all system programs in Section 2, excluding the RIM, BIN, and HELP Loaders. DEC-supplied paper tapes in BIN format are identified by a blue Digital label on the leader portion of the tape (red Digital labels denote USASCII format). (See DEC-08-LBAA-D.)

HELP Loader, used to load into core memory programs punched on paper tape in BIN format. HELP is in two parts: the first part consists of 11g instructions which must be toggled into core using the computer console switches; the second part is the HELP Bootstrap tape which

is read into core using the low-speed (Teletype) reader--the HELP Bootstrap tape contains the RIM and BIN Loaders. (See DEC-08-LHAA-D.)

Symbolic Tape Editor, used to prepare, edit, and generate symbolic (source) program tapes on-line from the Teletype keyboard. (See DEC-08-ESAB-D.)

PAL III Symbolic Assembler, used to translate source programs written in the PAL III language into object programs in two passes through the Assembler. The optional third-pass produces an octal/symbolic listing of the assembled program. (See DEC-08-ASAB-D.)

MACRO-8 Assembler, used to translate source programs written in the MACRO-8 language, containing macros and literals, into object programs in two passes through the Assembler. This Assembler also generates indirect linkages to off-page references. The optional third-pass produces an octal/symbolic assembly listing. (See DEC-08-CMAA-D.)

8K SABR, used with source programs written in the SABR language. SABR is core page independent, it automatically generates off-page and off-field references for direct and indirect statements, it automatically connects instructions on one page to those that overflow onto the next, and it contains an impressive list of pseudo-ops which include external subroutine calling, argument passing, and conditional assembling. The assembled output is punched on paper tape in binary relocatable code. SABR offers an optional second pass to produce a side-by-side octal/symbolic listing of the assembled program. (See DEC-08-ARXA-D)

Dynamic Debugging Technique (DDT-8), used to aid the user in finding mistakes in his program by allowing him to execute small sections at a time, to stop execution where he wishes, and to change portions of his program, all from the keyboard using the symbolic language of the source program. (See DEC-08-CDDB-D.),

Octal Debugging Technique (ODT-8), used for the same purpose as DDT-8 (above) except that the user communicates in the octal representation of the program. ODT-8 requires less core area than DDT-8, and it can be loaded in either the upper or lower portion of core, depending on where the user's program is loaded. (See DEC-08-COCO-D.)

FOCAL (FOrmula CALculator), an on-line, conversational, interpretive program used to solve numerical problems of any complexity; used as a programming tool by students, scientists, and engineers. (See DEC-08-AJAD-D.)

4K FORTRAN, used to compile, debug, and operate a user program written in the 4K PDP-8/I version of the FORTRAN language; compilation requires only one pass through the compiler. (See DEC-08-AFC0-D.)

8K FORTRAN, used to compile, assemble, load, and execute user programs written in the 8K FORTRAN language (a version of FORTRAN II). The system consists of the 8K FORTRAN Compiler which compiles the user's source program into symbolic language, the 8K SABR Assembler which assembles the compiled program into relocatable binary code, and the 8K Linking Loader which loads and executes the relocatable binary program and library of subprograms. (See DEC-08-KFXB-D)

TC01 Bootstrap Loader, used to load and start the DECtape Library System. (See DEC-08-LUAA-D.)

DECtape Library System, is a collection of five programs (INDEX, ESCAPE, UPDATE, DELETE, and GETSYS), used to load named files into core memory, define new named files, delete named files, and to create a new skeleton library system. (See DEC-08-SUB0-D.)

Disk System Builder, used to build a customized Disk Monitor System suited to the user's particular machine configuration. (See DEC-D8-SDAB-D.)

Disk System Program Library, is a collection of system and utility programs. The standard package includes an Editor, Assembler, DDT, FORTRAN system, Peripheral Interchange Program (PIP), and a transparent Loader. (See DEC-D8-SDAB-D.)

Certain programs can operate using extended memory, DECtape, and Disk facilities. They are:

a. Extended Memory - 8K SABR and 8K FORTRAN.
b. DECtape -- DECtape Bootstrap Loader and Library System.
c. Disk - Disk Monitor System programs.

COMPUTER CONSOLE SWITCHES AND INDICATORS

Manual control of the PDP-8/I computer is by means of switches on the computer console. Indicator lamps on the console light to denote the presence of a binary one in specific bits of the various registers and to indicate the status of the computer or of the program being executed. See PDP-8/I and PDP-8/L User's Handbook for details.

The locations of the switches and indicators are shown in Figure INTRO-1. The purpose of each switch and indicator on the computer console is explained below.

SWITCHES

POWER	This key-operated switch applies and removes the computer's primary power supply.
PANEL LOCK	This key-operated switch when turned clockwise disables all console switches except the SR; turned counterclockwise, all console switches function normally.
START	Executes the stored computer program, starting at the address specified in the PC.
LOAD ADDress	Sets the contents of the SR into the PC, and INST FIELD into the IF, and the DATA FIELD into the DF.
DEPosit	Deposits the contents of the SR into the location specified by the PC, and increments the PC by 1. PDP-8/I -- The DEP switch is activated when the top of the switch is depressed. PDP-8, 8/S, and 8/L -- The DEP switch is activated when the switch is raised.
EXAMine	Displays the contents of the location in the PC in the MB, and increments the PC by 1.
CONTinue	Continues program execution at the location specified by the PC.
STOP	Stops program execution.
SINGle STEP	When set, the computer executes instructions one cycle at a time for each depression of CONT.
SINGle INSTruction	When set, the computer executes one instruction at a time for each depression of CONT.
SWITCH REGISTER (SR)	When LOAD ADD is depressed, the contents of the SR is loaded into the PC. When DEP is depressed, the contents of the SR is loaded into the MB and memory. The 12 positions represent a 12-bit binary word, usually read in octal.
DATA FIELD (DF)	Denotes the core memory field of data storage and retrieval when LOAD ADD is depressed.
INSTruction FIELD (IF)	Denotes the core memory field from which instructions are taken when LOAD ADD is depressed.

INDICATORS

PROGRAM COUNTER (PC)	Contents represent the address of the next instruction to be executed.

MEMORY ADDRESS (MA)	Contents represent the address of the word currently being read or written. After depressing DEP or EXAM, the contents represent the address of the word previously read or written.
MEMORY BUFFER (MB)	Contents represent the word being read or written.
ACCUMULATOR (AC)	Indicates the contents of the AC.
LINK (L)	Indicates the contents of the Link.
MULTIPLIER QUOTIENT (MQ)	Activated only with the EAE option. At the start of a multiplication the contents represent the multiplier; at the end the least significant half of the product. At the start of a division the contents represent the least significant half of the dividend; at the end the quotient.
Instruction and Status Indicators	Located to the right of the above indicators. Indicates the type of instruction being executed, and the status of the program being executed.

ASR33 TELETYPE

The ASR33 Teletype is the basic input/output device for PDP-8/I computers. It consists of a printer, keyboard, paper tape reader, and paper tape punch, all of which can be used either on-line under program control or off-line. The Teletype controls (Figure INTRO-2) are described as they apply to the operation of the computer. For off-line operations, see Appendix C. See PDP-8/I and PDP-8/L Users Handbook for details.

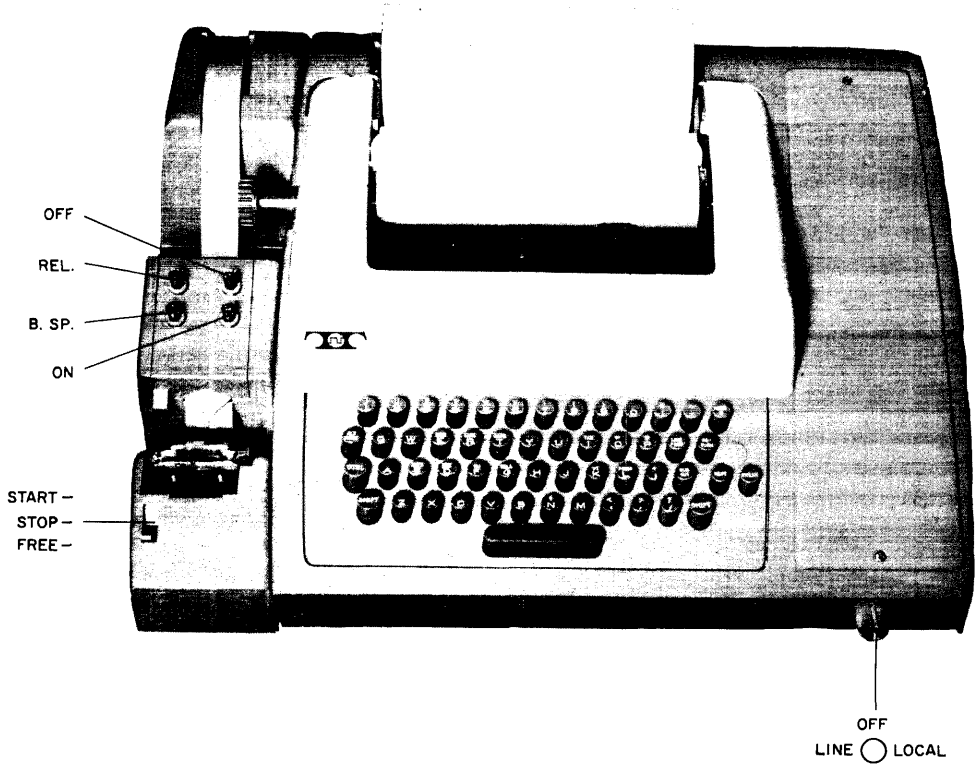


Figure INTRO-2 ASR33 Teletype Console

POWER CONTROLS

LINE	The Teletype is energized and connected to the computer as an input/output device, under computer control.
OFF	The Teletype is de-energized.
LOCAL	The Teletype is energized for off-line operation.

PRINTER

The printer provides a typed copy of input and output at 10 characters per second maximum.

KEYBOARD

The Teletype keyboard is similar to a typewriter keyboard. However, certain operational functions are shown on the upper part of some of the keytops. These functions are activated by holding down the CTRL key while depressing the desired key. For example, when using the Symbolic Editor, CTRL/FORM causes Editor to enter command mode.

Although the left and right square brackets are not visible on the keyboard keytops, they are shown in Figure INTRO-3 and are generated by typing SHIFT/K and SHIFT/M, respectively. The ALT MODE key is identified as ESC (ESCape) on some keyboards.

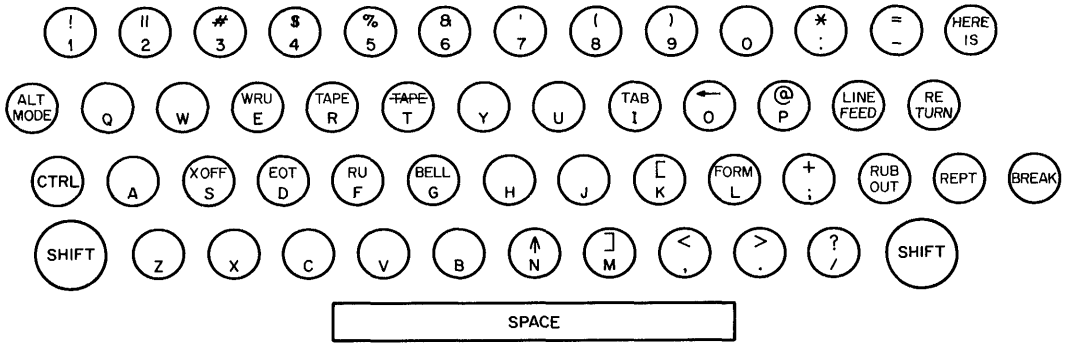


Figure INTRO-3 ASR33 Teletype Keyboard

PAPER TAPE READER

The paper tape reader is used to input into core memory data punched on eight-channel perforated paper tape at a rate of 10 characters per second maximum. The reader controls are shown in Figure INTRO-2 and described below.

START	Activates the reader; reader sprocket wheel is engaged and operative.
STOP	Deactivates the reader; reader sprocket wheel is engaged but not operative.
FREE	Deactivates the reader; reader sprocket wheel is disengaged.

Positioning Tape in Tape Reader

The following procedure describes how to properly position paper tape in the low-speed reader.

a. Raise the tape retainer cover.

b. Set reader control to FREE.

c. Position the leader portion of the tape over the read pens with the sprocket (feed) holes over the sprocket (feed) wheel and with the arrow (printed or cut) pointing outward.

d. Close the tape retainer cover.

PAPER TAPE PUNCH

The paper tape punch is used to perforate eight-channel rolled oiled paper tape at a rate of 10 characters per second. The punch controls are shown in Figure INTRO-2 and described below.

RELease	Disengages the tape to allow tape removal or loading.
Back SPace	Backspaces the tape one space for each firm depression of the B.SP. button.
ON	Activates the punch.
OFF	Deactivates the punch.

HIGH-SPEED TAPE READER AND PUNCH UNITS

A high-speed paper tape reader and punch unit is pictured in Figure INTRO-4 and descriptions of the reader and punch units follow. (See PDP-8/I and PDP-8/L User's Handbook for details.)

READER UNIT

The high-speed paper tape reader is used to input data into core memory from eight-channel fan-folded (non-oiled) perforated paper tape photoelectrically at 300 characters per second. Primary power is applied to the reader when the computer POWER switch is turned on. The reader is under user control from the keyboard through the computer or under program control. However, tape can be advanced past the photoelectric sensors without causing input by pressing the tape feed button (the white rectangular button located in the center of Figure INTRO-4).

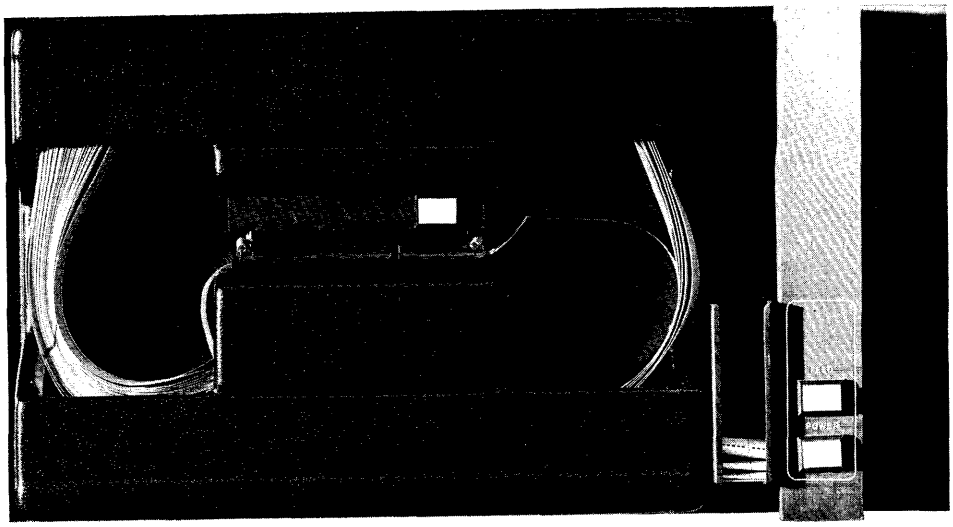


Figure INTRO-4 High-Speed Paper Tape Reader and Punch Units

Loading Instructions

a. Raise tape retainer cover (located beneath the tape feed button).

b. Put tape into right-hand bin with channel one (see PAPER TAPE FORMATS) of the tape toward the rear of the bin.

c. Place several folds of blank tape through the reader and into the left-hand bin.

d. Place the tape over the reader head with feed holes engaged in the teeth of the sprocket wheel.

e. Close the tape retainer cover.

f. Depress the tape feed button (white rectangular button above the reader head) until leader tape is over the reader head.

CAUTION

Oiled paper tape should not be used in the high-speed reader--oil collects dust and dirt which can cause reader errors.

PUNCH UNIT

The high-speed paper tape punch is used to record computer output on eight-channel fan-folded perforated paper tape at 50 characters per second. All characters are punched under program control from the computer. Blank tape (feed holes only, no data) may be produced

by pressing the FEED button (see Figure INTRO-4). Primary power is available to the punch when the computer POWER switch is turned on. Power is applied to the punch when the POWER button is depressed to on (the punch motor can be heard). The two labeled buttons on the punch enclosure are described below.

POWER	This microswitch is depressed to turn the punch ON and OFF.
FEED	While this button is depressed, the punch produces feed-hole-only punched tape for leader/trailer purposes.

CAUTION

Oiled paper tape should not be used in the high-speed punch -- oil collects dust and dirt which can cause reader errors.

DECTAPE CONTROL AND TRANSPORT UNITS

DECtape is a fast, convenient, reliable input/output data storage facility and updating device. The standard DECtape transport unit is pictured in Figure INTRO-5 and descriptions of the control and transport units follow. (See PDP-8/I and PDP-8/L User's Handbook for details.)

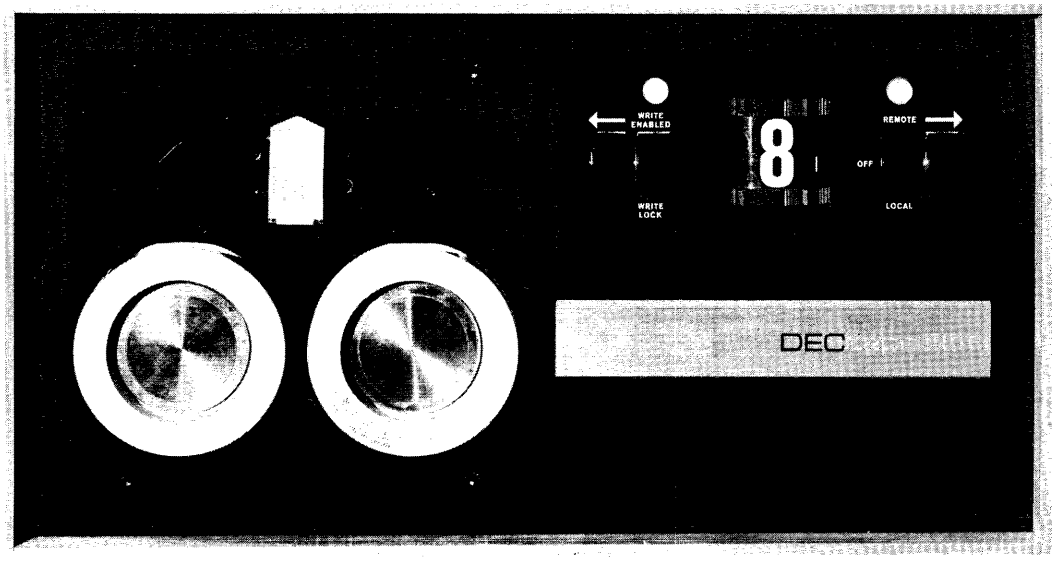


Figure INTRO-5 DECtape Transport Unit

CONTROL UNIT

The DECtape control unit interprets and controls the transfer of data between the computer and the transport unit. The DECtape control unit is usually located inside the rack containing the DECtape transport and can control up to eight separate DECtape transports.

TRANSPORT UNIT

The DECtape transport unit is a bidirectional magnetic tape transport utilizing a 10-track recording head to read and write five duplexed channels. Tape movement can be controlled by commands from the computer program or by commands from the manual operation of switches located on the front panel of the transport; however, manual operation does not transfer data to the computer.

Only Certified DECtapes (prerecorded with timing and marking tracks) should be used. Otherwise, the blank tape must be certified using the DECTOG program (DEC-08-EUFA-D).

Transport Controls

→	When depressed (must be in LOCAL mode), tape feeds onto right spool.
REMOTE	Transport is energized and under program control.
OFF	Transport is de-energized.
LOCAL	Transport is energized and under user control from external transport switches.
Unit Selector	Identifies the transport to the control unit.
WRITE ENABLED	DECtape is available for search, read, and write activities.
WRITE LOCK	DECtape is available for search and read activities only.
←	When depressed (must be in LOCAL mode), tape feeds onto left spool.

The REMOTE and WRITE ENABLED lamps light to indicate the status of the transport.

Operating Procedure

a. Set switch to OFF.
b. Place DECtape on left spindle with DECtape label out.
c. Wind at least four turns of tape on right spool.
d. Set switch to LOCAL.
e. Wind a few turns on right spindle with → switch to make sure tape is properly mounted.
f. Dial correct unit number on unit selector (number 8 is equivalent to 0).

g. Depress REMOTE switch.

h. Depress WRITE ENABLED or WRITE LOCK switch.

i. DECtape transport is now under program control.

INITIALIZING THE SYSTEM

Before using the computer system, it is good practice to initialize all units. To initialize the system, ensure that all switches and controls are as specified below.

a. Main power cord is properly plugged in.

b. Teletype is turned OFF.

c. Low-speed punch is OFF.

d. Low-speed reader is set to FREE.

e. Computer POWER key is ON.

f. PANEL LOCK is unlocked.

g. Console switches are set to

DF=000 IF=000 SR=0000

SING STEP and SING INST are not set

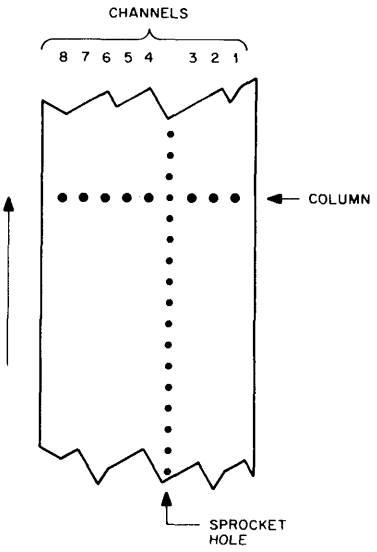
h. High-speed punch is OFF.

i. DECtape REMOTE lamps OFF.

The system is now initialized and ready for your use.

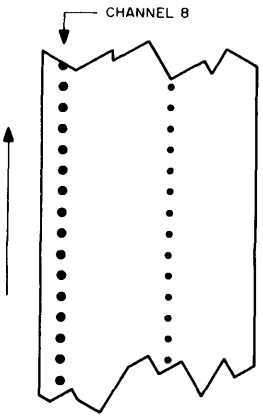
PAPER TAPE FORMATS

Data are recorded (punched) on paper tape by groups of holes arranged in a definite format along the length of the tape. The tape is divided into channels which run the length of the tape, and into columns which extend across the width of the tape as shown in the adjacent diagram. The paper tape readers and punches used with the PDP-8/I computers accept 8-channel paper tape. The various formats are briefly explained and identified below.



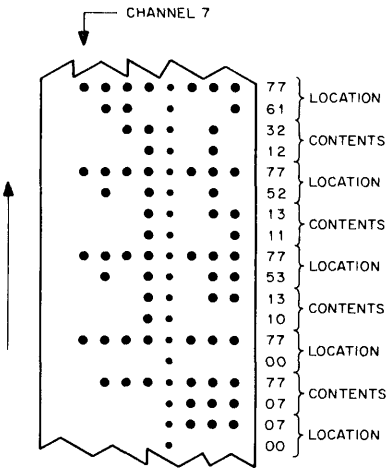
Leader/Trailer Format

Leader/trailer tape is used to introduce and conclude the object program when punched on paper tape. Leader/trailer tape can be recognized by a consistent channel 8 punch only as shown in the adjacent diagram.



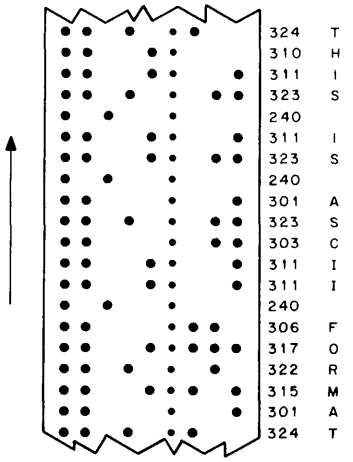
RIM Format

Paper tape punched in RIM format can be identified by the absence of a channel 8 punch, and by a channel 7 punch in every fourth column. The channel 7 punch indicates the start of a line of coding, and that (the first) column and the second column contain the location and the third and fourth columns contain the contents of the location.



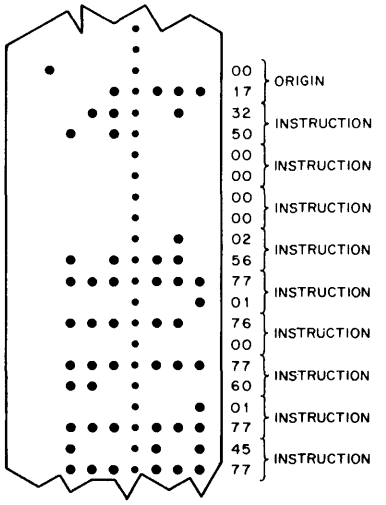
USASCII Format

USASCII (USA Standard Code for Information Interchange) format uses all eight channels to represent a single character (letter, number, or symbol) as shown in the adjacent diagram.



Binary Format

Binary format can be recognized by the absence of a channel 8 punch, an occasional channel 7 punch, and frequent sections of blank tape. The channel 7 punch denotes an origin of a program or subprogram or a change in origin, and subsequent columns contain the instructions (two columns per instruction) or data of succeeding locations.



ABBREVIATIONS

The abbreviations listed below are used throughout the guide.

Abbreviations	Meaning
AC	Accumulator
ADDR	Address
B. SP.	Back Space
BIN	Binary
CLC	Current Location Counter
CONT	Continue
CR	Carriage Return
CR/LF	Carriage Return-Line Feed
CTRL/L	Control/L (represents holding down the CTRL key while depressing the L key or the key following the slash)
DEC	Digital Equipment Corporation
DEP	Deposit
DF	Data Field
EAE	Extended Arithmetic Element
EXAM	Examine
IF	Instruction Field
INST	Instruction
L	Link
LF	Line Feed
LOAD ADD	Load Address
LOC	Location
LSP	Low-Speed Punch
LSR	Low-Speed Reader
HSP	High-Speed Punch
HSR	High-Speed Reader
KBRD	Keyboard
PC	Program Counter
PROG	Program
MA	Memory Address
MB	Memory Buffer
MQ	Multiplier Quotient
MRI	Memory Reference Instruction
REL	Release
RIM	Read-In Mode
SA	Starting Address
SHIFT/P	Shift/P (similar to CTRL/L)
SING INST	Single Instruction
SING STEP	Single Step
SR	Switch Register
SW	Console Switches
TTY	Teletype
USASCII	USA Standard Code for Information Interchange

SECTION 2

SYSTEM PROGRAMS

PURPOSE

The RIM Loader is used to load into core memory programs punched on paper tape in RIM format, e.g., the Binary Loader. (See DEC-08-LRAA-D for details.)

STORAGE REQUIREMENTS

RIM requires locations 7756-7776 (21_8 locations). Starting Address=7756.

LOADING

RIM is loaded (toggled) into core memory using the console switches. RIM can use either the low- or high-speed readers when loading RIM coded program tapes into core. The locations and corresponding instructions for both input devices are listed below.

	Instruction	
Location	Low-Speed Reader	High-Speed Reader
7756	6032	6014
7757	6031	6011
7760	5357	5357
7761	6036	6016
7762	7106	7106
7763	7006	7006
7764	7510	7510
7765	5357	5374
7766	7006	7006
7767	6031	6011
7770	5367	5367
7771	6034	6016
7772	7420	7420
7773	3776	3776
7774	3376	3376
7775	5356	5357
7776	0000	0000

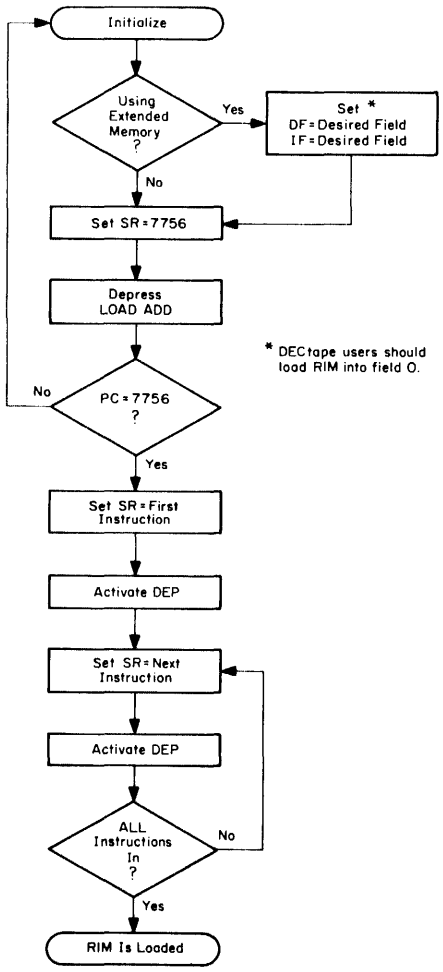


Figure RIM-1 Loading the RIM Loader

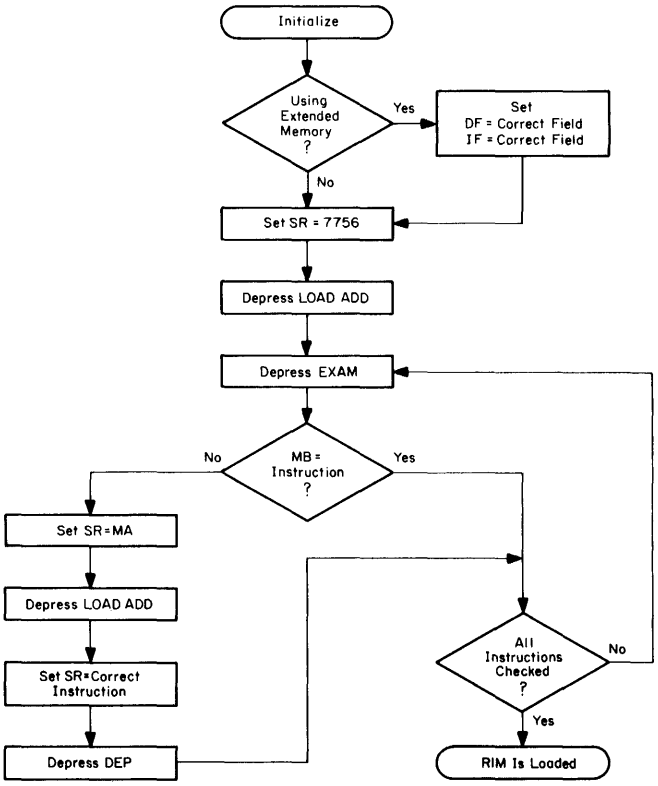


Figure RIM-2 Checking the RIM Loader

PURPOSE

The BIN Loader is used to load into core memory binary coded programs punched on paper tape. When in core, BIN can be destroyed only by the user's program because DEC's programs (excluding Disk Monitor) do not use the last page of core (location 7600-7777). (See DEC-08-LBAA-D for details.)

STORAGE REQUIREMENTS

BIN occupies locations 7625-7752 and 7777 (123_8 locations). Starting Address=7777

LOADING

RIM is used to load BIN into core. BIN must be loaded into the same field as RIM, and the input device (low- or high-speed reader) must be that which was selected when loading RIM.

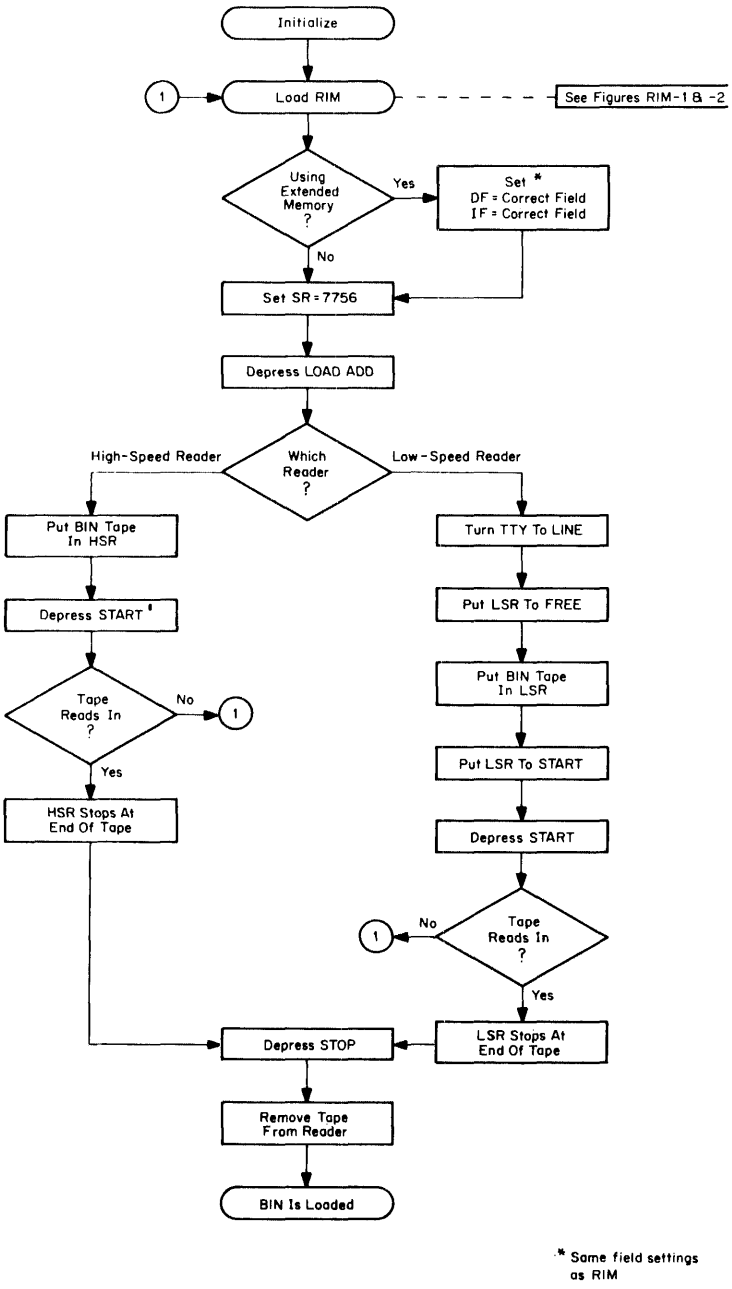


Figure BIN-1 Loading the BIN Loader

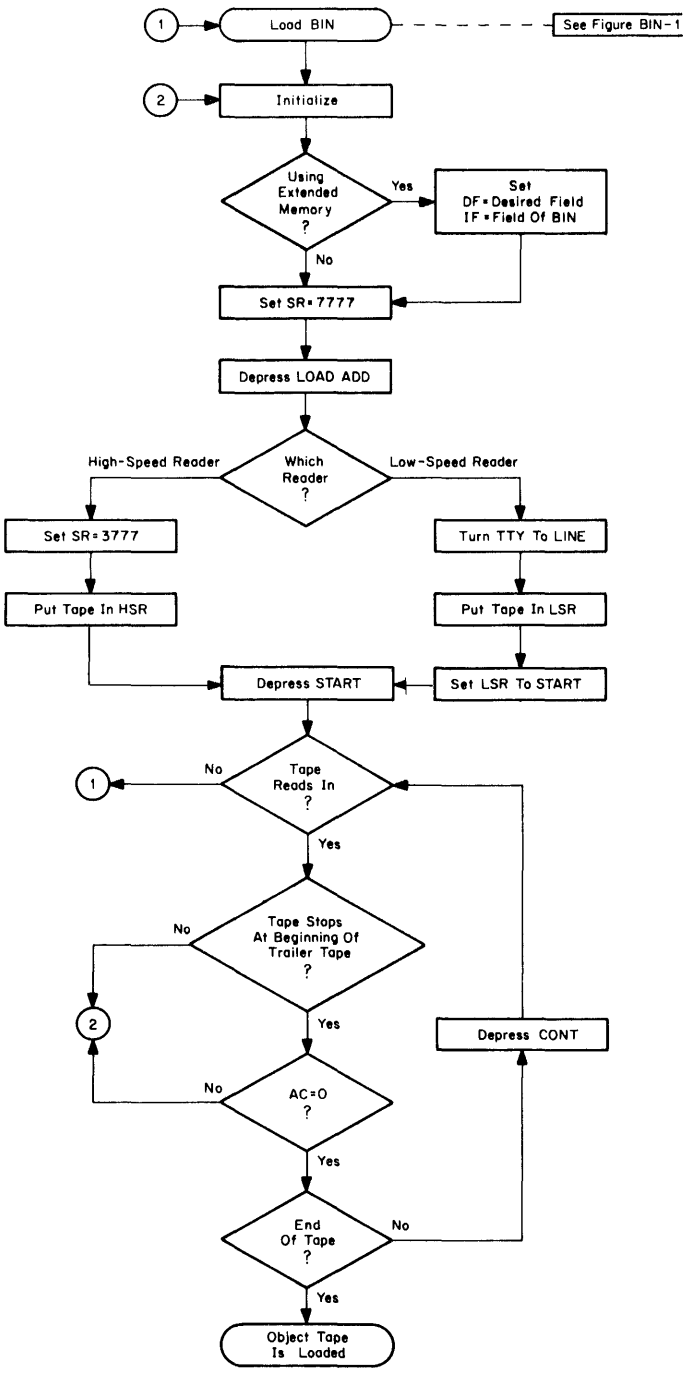


Figure BIN-2 Loading A Binary Coded Object Tape Using BIN

PURPOSE

The HELP Loader is used to quickly load into core memory the RIM and BIN Loader programs. (See DEC-08-LHAA-D for details.)

STORAGE REQUIREMENTS

HELP uses locations 0005-0036 (32_8 locations) to load the HELP tape into core. The HELP tape contains the RIM and BIN Loaders.

LOADING

HELP is in two parts: The first part consists of the 11_8 instructions shown below, which are toggled into core using the console switches. The second part is the HELP Bootstrap Loader punched on paper tape, which is loaded into core using the low-speed reader.

Location	Instruction
0027	6031
0030	5027
0031	6036
0032	7450
0033	5027
0034	7012
0035	7010
0036	3007
0037	2036
0040	5027

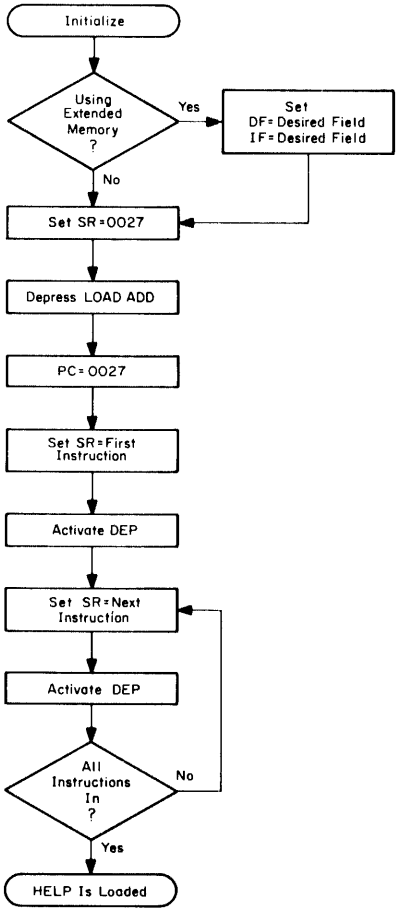


Figure HELP-1 Loading the HELP Loader

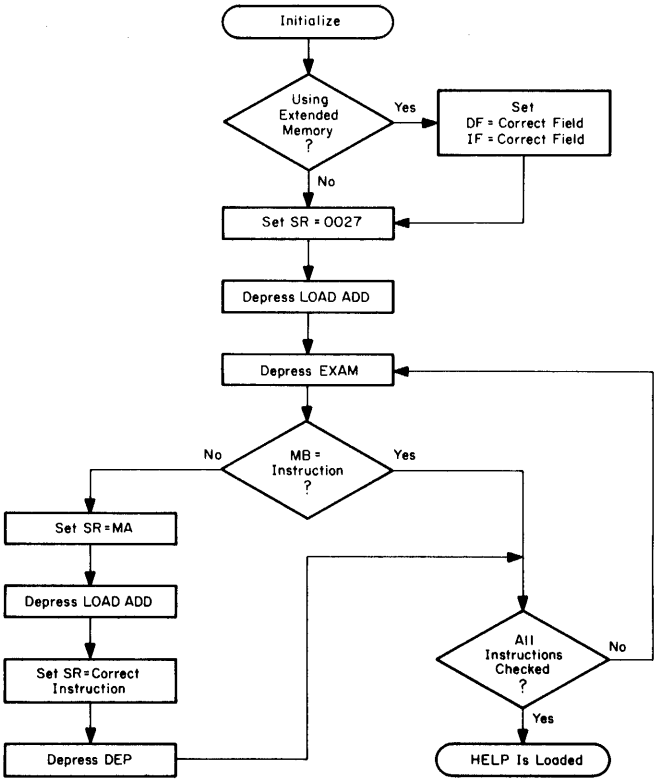


Figure HELP-2 Checking the HELP Loader

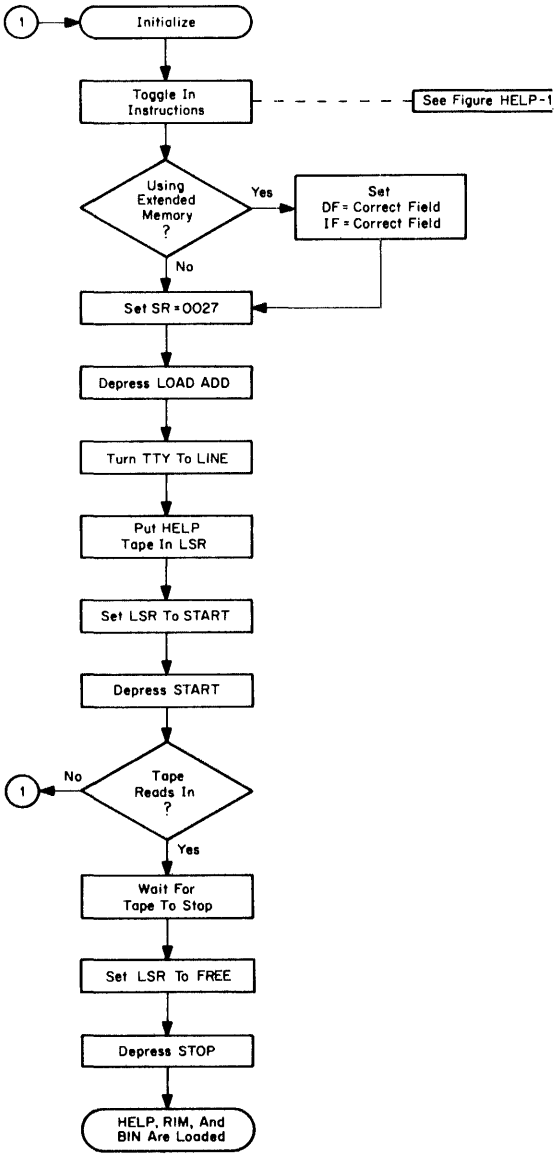


Figure HELP-3 Loading the HELP Bootstrap Tape Into Core

PURPOSE

The Symbolic Tape Editor is used to prepare, edit, and generate symbolic program tapes on line. (See DEC-08-ESAB-D for details.)

STORAGE REQUIREMENTS

Editor requires locations 0-1577 (1600_8 locations). Starting Address=0200.

LOADING

BIN is used to load Editor into core memory. The loading of the user's symbolic tapes is performed by Editor itself under keyboard control.

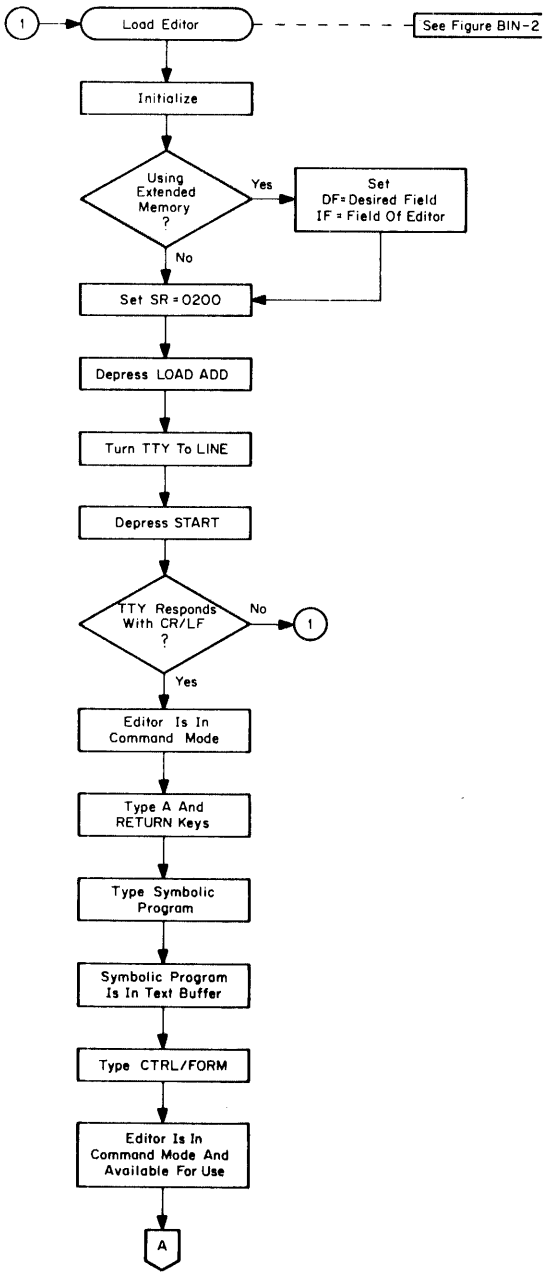


Figure EDIT-1 Generating a Symbolic Program On-Line Using Editor

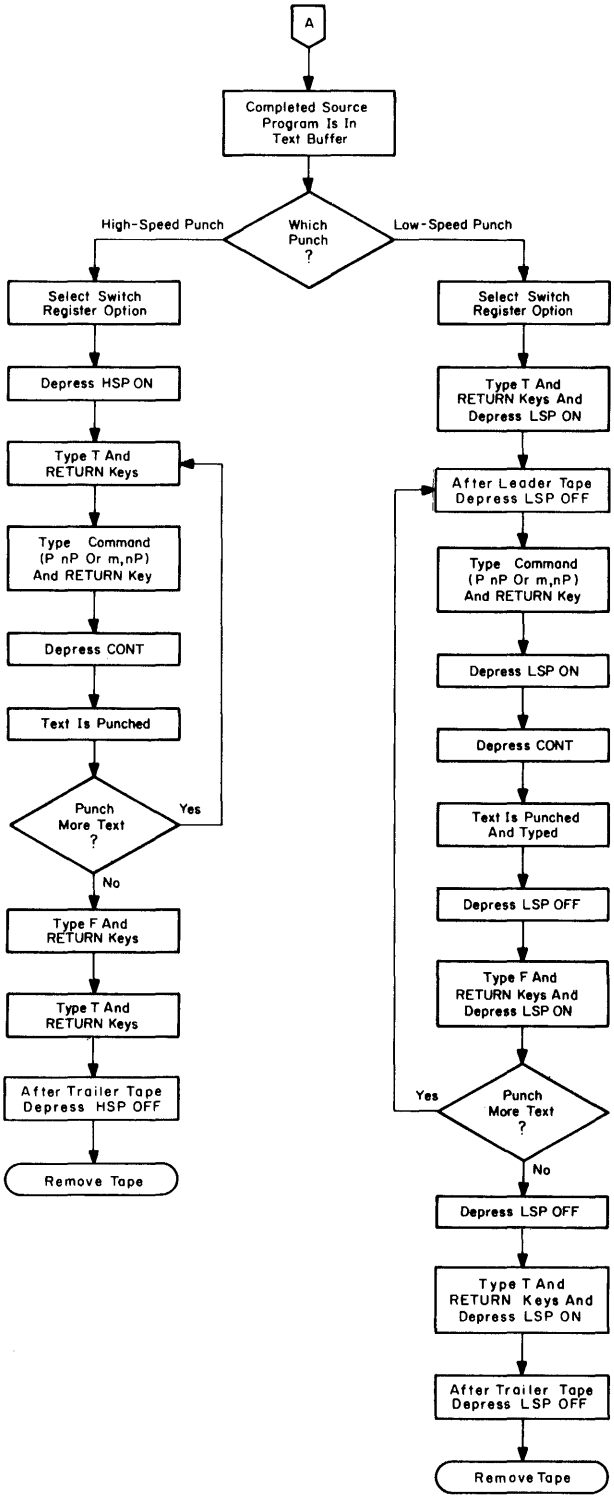


Figure EDIT-2 Generating a Symbolic Tape Using Editor

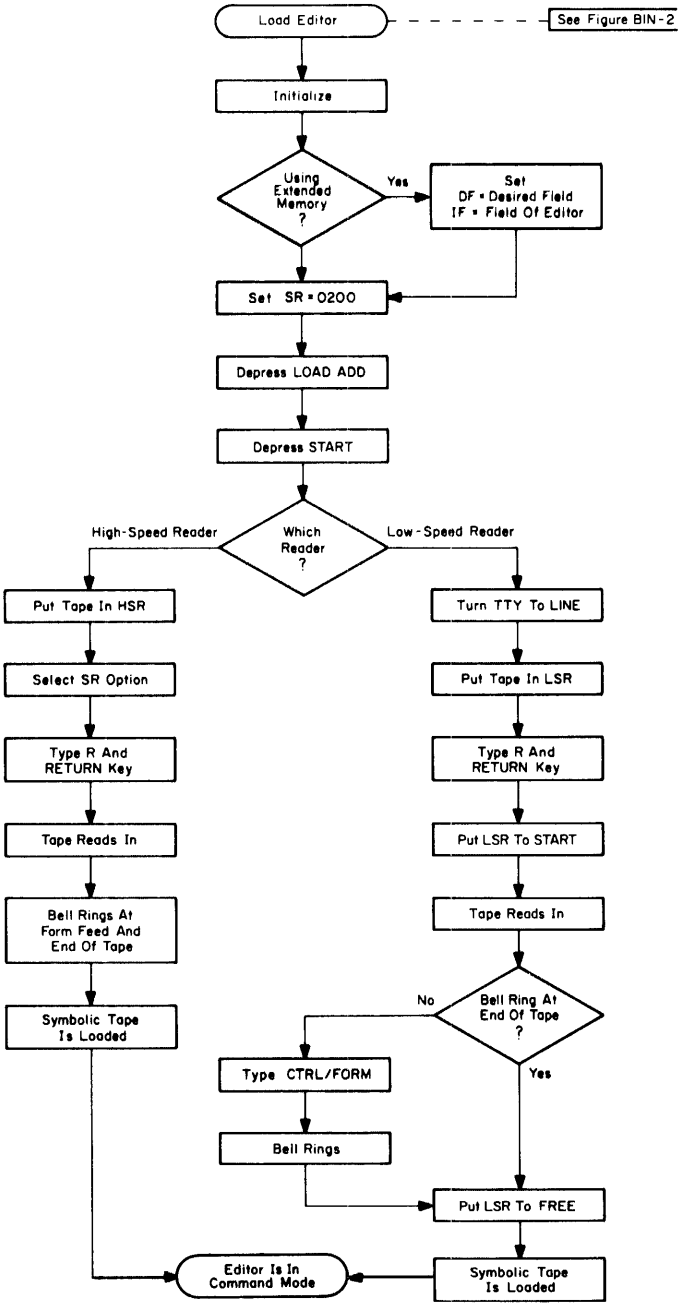


Figure EDIT-3 Loading a Symbolic Tape Using Editor

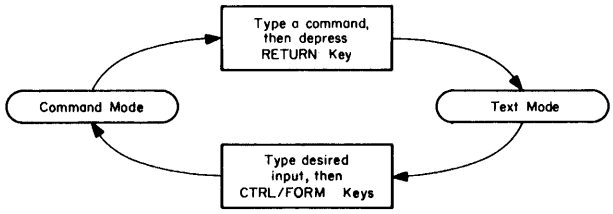
OPERATING MODES

Editor is always in one of the following modes.

Command Mode: All characters typed on the teleprinter are interpreted as commands to the Editor.

Text Mode: All characters typed or tapes being read in are interpreted as text to be put into the text buffer in the manner specified by the previous command and the SR options.

Transition between modes:



RESTART PROCEDURES

If the user stops the computer for any reason, he may restart it at location 0177 without disturbing the text in the buffer. A CR/LF will follow.

If no CR/LF is desired, restart at location 0200.

CAUTION

If Editor is restarted at location 0176, all text currently in the buffer is wiped out, and the text buffer is re-initialized for a new program.

SPECIAL KEY FUNCTIONS

Key	Command Mode	Text Mode
RETURN	Execute preceding command	Enter line in text buffer
←	Cancel preceding command (Editor responds with ? CR/LF)	Cancel line to the left margin
RUBOUT	Same as ←	Delete to the left one character for each depression; a \ (back-slash) is echoed (not used in READ command)
CTRL/FORM		Return to command mode (bell rings)

Key	Command Mode	Text Mode
.	Used as argument alone or with + or - and a number (.=, .+5L, .-2S) (a space is equivalent to a +)	Legal text character.
/	Value equal to number of last line in buffer; used as argument (/-5G, /L)	Legal text character
LINE FEED	List next line	Used in SEARCH command to insert CR/LF into line
ALT MODE (ESC)	List next line	
>	List next line	
<	List previous line	
=	Used with . and / to obtain their value (.=27)	
:	Same as = (gives value of any legitimate argument)	
CTRL/TAB		Produces a tab which on output is interpreted as ten spaces or a tab/rubout, depending on SR option.

SEARCH FEATURE IN COMMAND MODE

Following a nS command, Editor waits for the user to specify the search character which when typed is not echoed. When Editor locates and types the search character, typing stops and all or any combination of the following operations may be carried out.

a. Type new text and terminate line with the RETURN key

b. ← delete entire line to the left

c. RETURN delete entire line to the right

d. RUBOUT delete from right to left one character for RUBOUT typed (a \ is echoed for each RUBOUT typed)

e. LINE FEED insert a CR/LF, thus dividing line into two

f. CTRL/FORM search for next occurrence of search character

g. CTRL/BELL change search character to next character typed by the user

SWITCH REGISTER OPTIONS

Switch Register options are used with input and output commands to control the reading and punching of paper tape.

SR Bit	Position	Function
0	0	Input text as is
	1	Convert all occurrences of 2 or more spaces to a tab
1	0	Output each tab as 10 spaces
	1	Tab is punched as tab/rubout
2	0	Output as specified
	1	Suppress output*
10	0	Low-speed punch
	1	High-speed punch
11	0	Low-speed reader
	1	High-speed reader

COMMANDS

Input	R	Read incoming text from tape reader into core
	A	Append incoming text from keyboard into core
Editing	L	List entire text buffer
	nL	List line n
	m,nL	List lines m through n inclusively
	nC	Change line n
	m,nC	Change lines m through n inclusively
	I	Insert before first line
	nI	Insert before line n
	K	Delete entire text buffer
	nD	Delete line n
	m,nD	Delete lines m through n inclusively
	m,n$kM	Move lines m through n to before line k
	G	Print next tagged line (if none, Editor types ?)
	nG	Print next tagged line after line n (if none, ?)
	S	Search buffer for character specified after RETURN key and allow modification (search character is not echoed on printer)
	nS	Search line n, as above
	m,nS	Search lines m through n inclusively, as above
Output	P	Punch entire text buffer
	nP	Punch line n
	m,nP	Punch lines m through n inclusively

*Bit 2 allows the user to interrupt any output command and return immediately to command mode.

F	Punch leader tape, a Form Feed, and trailer tape
T	Punch about 6 inches of leader/trailer tape
nN	Do P, F, K, and R commands n times

where m and n are decimal integers, and m is smaller than n; k is a decimal number; P and N halt to allow user to select SR option. Press CONT to execute command.

Commands are executed upon depressing the RETURN key.

BUFFER OVERFLOW

Editor has storage for about 5000_{10} characters (approximately 60 heavily commented lines or 340 uncommented lines). When the text buffer is exceeded, operation continues, but a bell rings for every location used beyond the buffer limit. The user may expand the text buffer by changing location 0001 to contain the address of the last location (should not be greater than location 7570) used prior to buffer overflow. Very large programs should be divided into sections.

DIAGNOSTICS

a. Editor checks commands for nonexistent information and incorrect formatting, and when an error is detected Editor types a ? and ignores the command. However, if an argument is provided for a command that doesn't require one, the argument is ignored and the command is executed properly.

b. Corrections and additions to the user's program may be either typed in from the teleprinter keyboard or read in from the paper tape reader.

c. Since Editor does not recognize extraneous and illegal control characters, a tape containing these characters can be corrected by merely reading the tape into Editor and punching out a new tape.

PURPOSE

The PAL III Symbolic Assembler is used to translate symbolic (source) programs into binary (object) programs. PAL III is a two-pass assembler with an optional third pass which produces a program assembly listing.

Pass 1: Assembler reads the source tape and defines all symbols used. The user's symbol table and any error diagnostics are typed out.

Pass 2: Assembler reads the source tape and generates the object tape using the symbols defined during Pass 1. Ignore meaningless characters typed when using the low-speed punch, but note any error diagnostic typed.

Pass 3: Assembler reads the source tape and types and/or punches the program assembly listing.

A PAL III program is assembled in the System Demonstration section of this guide.

(See DEC-08-ASAC-D for details.)

STORAGE REQUIREMENTS

PAL III requires locations 0-2735 (2736_8 locations)

Symbol Table Capacity: LSR allows 590 user symbols
HSR allows 495 user symbols

Starting Address=0200

LOADING

BIN is used to load PAL III into core. PAL III is used to read in the symbolic tapes during assembly.

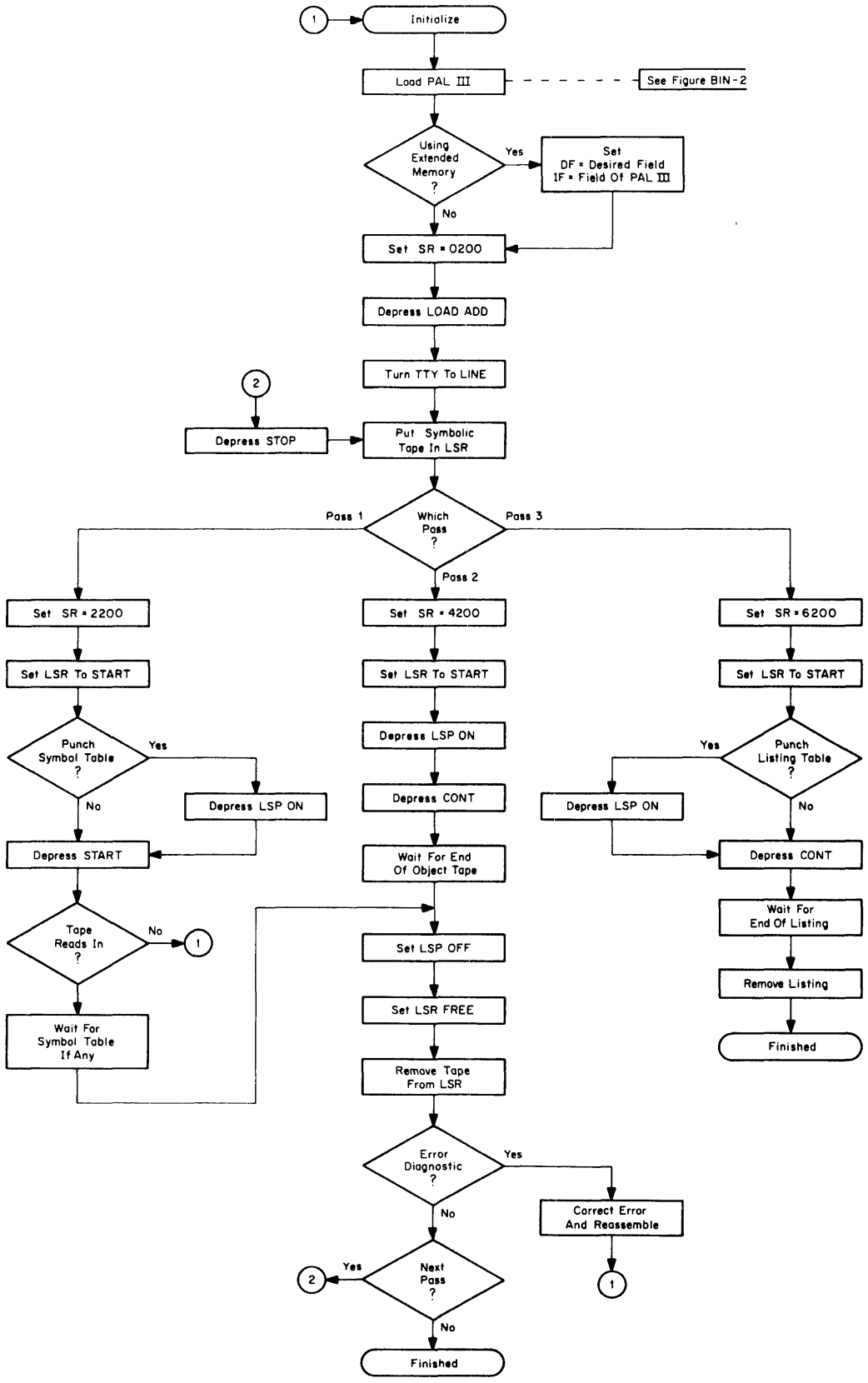


Figure PAL-1 Assembling with PAL III Using Low-Speed Reader/Punch

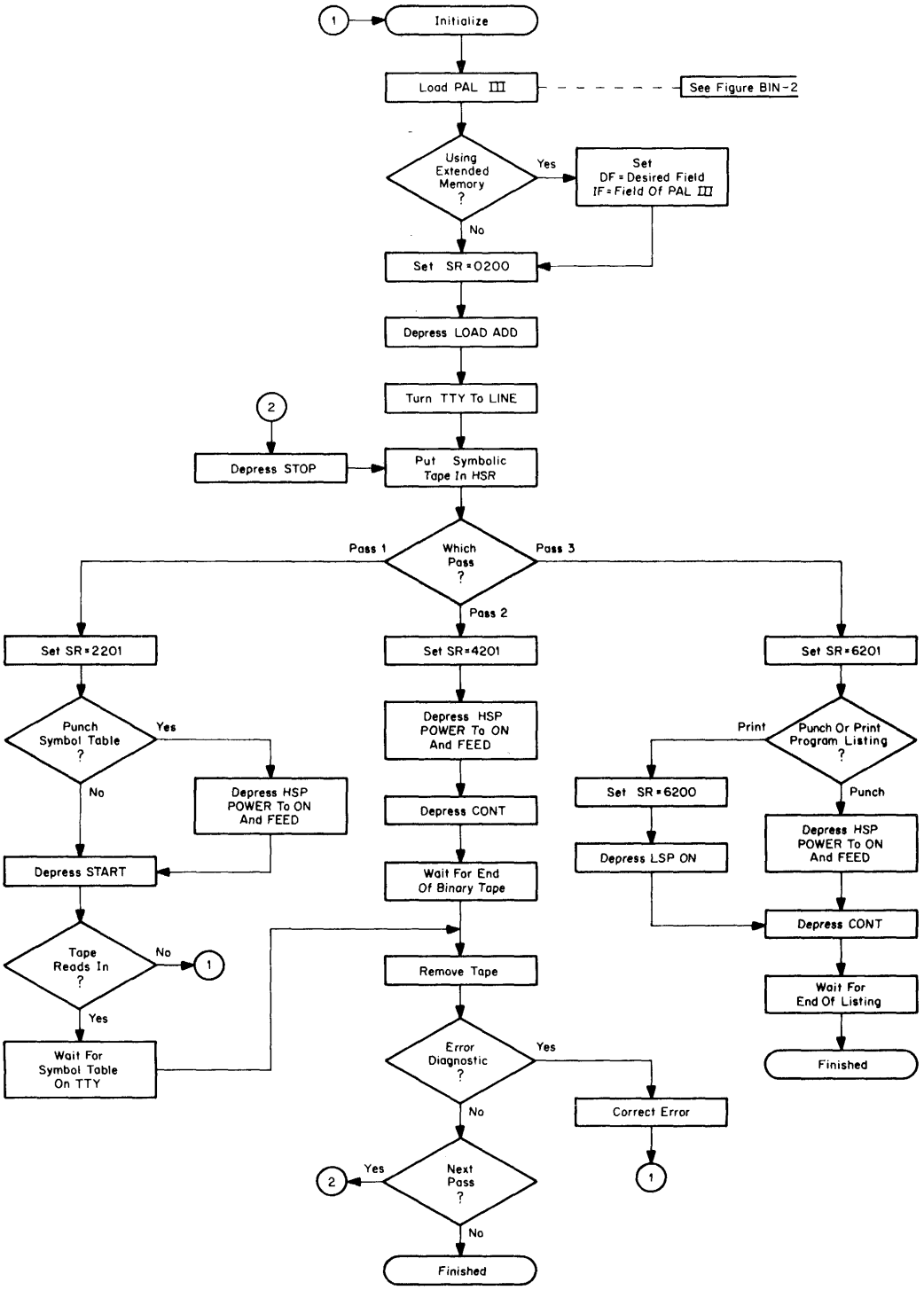


Figure PAL-2 Assembling With PAL III Using High-Speed Reader/Punch

OUTPUT CONTROL

Output is controlled by the setting of switch register bit 11 as shown below.

Pass 1: Bit 11=0 Type and punch symbol table on TTY
11=1 Punch symbol table on HSP

Pass 2: No effect; binary tape will be punched on the HSP if it is turned on.

Pass 3: Bit 11=0 Type and punch program listing on TTY
11=1 Punch program listing on HSP

DIAGNOSTICS

Format: xx yyyyyy AT nnnn

where xx is the error message (see below), yyyyyy is the symbol or octal value of the symbol of the error occurring AT location nnnn.

Pass 1:	IC	Illegal Character
	RD	ReDifinition
	DT	Duplicate Tag
	ST	Symbol Table full
	UA	Undefined Address
Pass 2:	IR	Illegal Reference

PURPOSE

The MACRO-8 Symbolic Assembler is used to translate symbolic (source) programs into binary (object) programs. MACRO-8 is a two-pass assembler with an optional third pass which produces a program assembly listing.

Pass 1: Assembler reads the source tape and defines all symbols and macros used and places them in respective tables.

Pass 2: Assembler reads the source tape and generates the object tape using symbols and macros defined during Pass 1. The Assembler then types and/or punches the user's symbol table, for use with DDT-8, followed by any error diagnostic.

Pass 3: Assembler reads the source tape and types and/or punches the program assembly listing.

See DEC-08-CMAA-D for details.

STORAGE REQUIREMENTS

MACRO-8 requires locations 0-7577 (7600_8 locations).

Symbol Table Capacity: 227_8 symbols (expandable to 524_8 symbols using the switch options)

Starting Address=0200.

LOADING

BIN is used to load MACRO-8 into core. There are two versions of MACRO-8:

Low version: Uses the low-speed reader for all input and the low-speed punch for all output.

High version: Uses the high-speed reader for all input, the high-speed punch for binary output, and the Teletype and low-speed punch for output of error diagnostics, symbol table, and third-pass assembly listing.

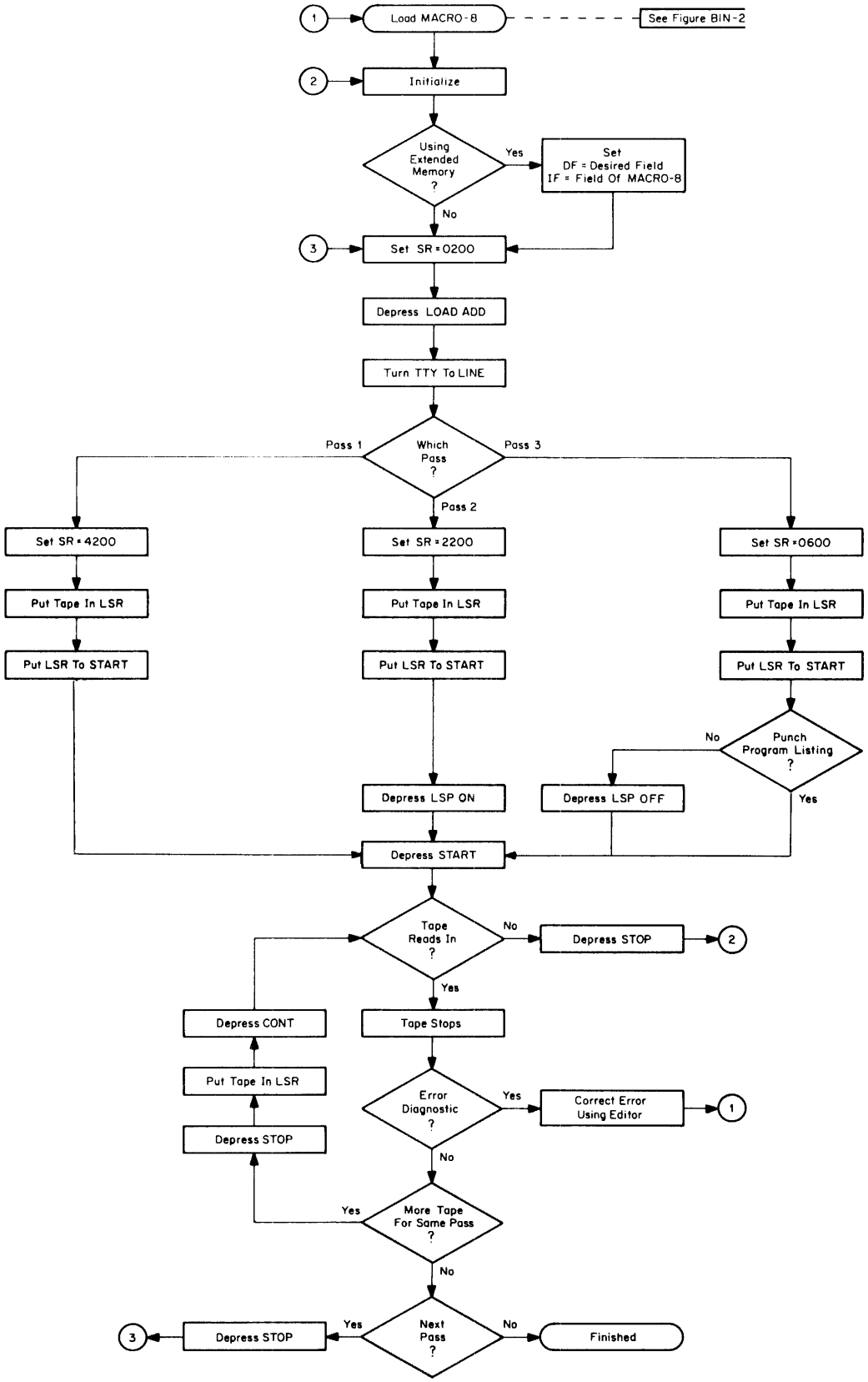


Figure MACRO-1 Assembling a MACRO-8
Source Program Using the Low-Speed Reader/Punch

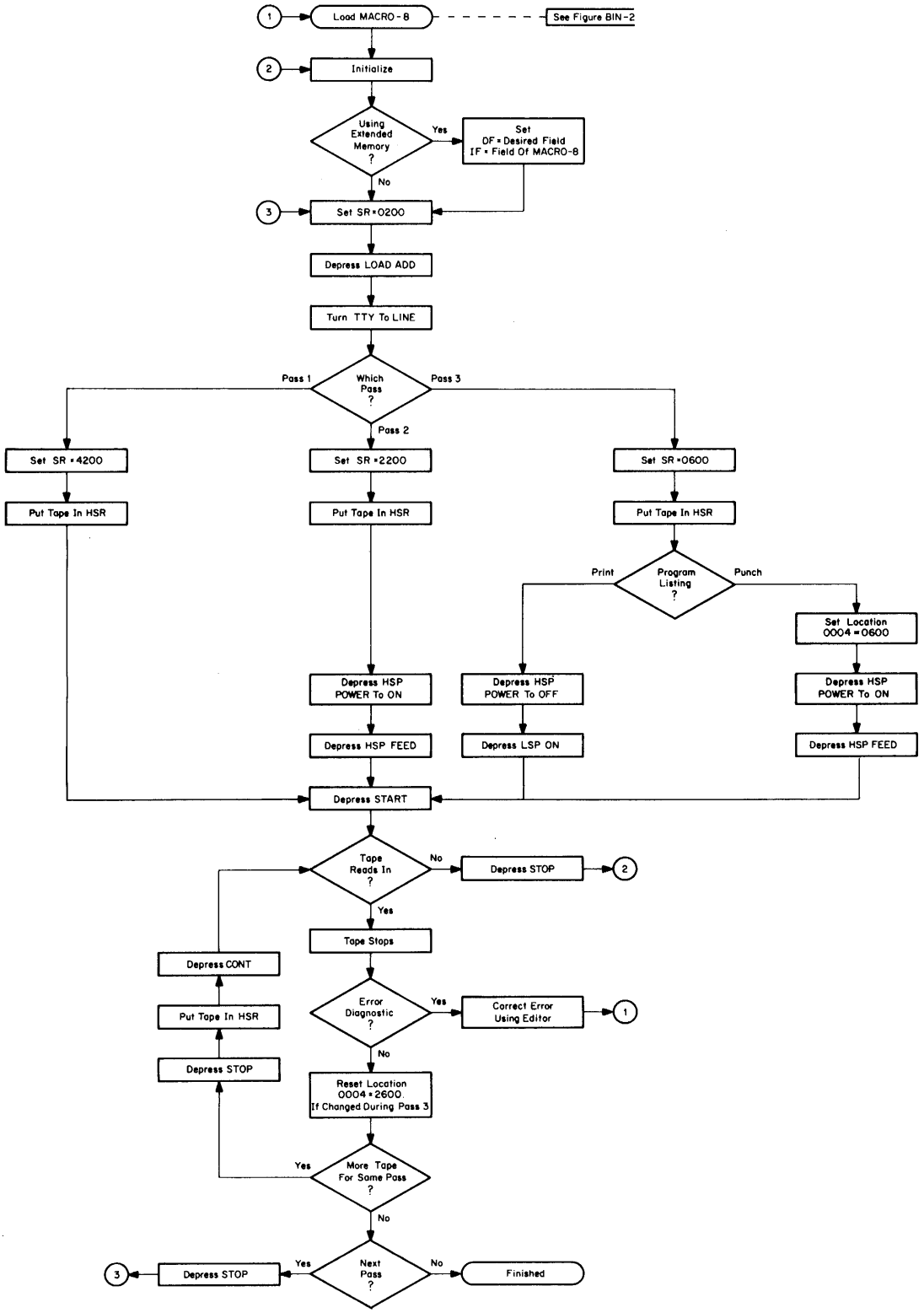


Figure MACRO-2 Assembling a MACRO-8 Source Program Using the High-Speed Reader/Punch

SYMBOL TABLE MODIFICATION

There are 1134_8 locations available for the user's symbols and the macro table. There are three ways to increase the size of this storage area.

a. Prior to Pass 1, set bit 10 = 1 to add 100_8 locations; the double precision integer and floating-point processors are deleted.

b. Prior to Pass 1, set bit 11 = 1 to add 175_8 locations; the macro and number processors are deleted.

c. Use the pseudo-ops EXPUNGE and FIXTAB to remove unnecessary instruction mnemonics.

SWITCH REGISTER OPTIONS

Bit	Result
0-11 = 0	Enter next pass.
0 1	Erase symbol table excluding permanent symbols and enter Pass 1; depress STOP then CONT.
1 = 1	Enter pass 2 to generate another binary tape.
2 = 1	Enter pass 1 without erasing defined symbols.
3 = 1	Enter pass 3.
10 = 1	Delete double precision integer and double precision floating-point processors; this increases the symbol table size by 100_8 symbols.
11 = 1	Delete macro and number processors; this increases the symbol table size by 175_8 symbols.

Bits 10 and 11 are sensed whenever pass 1 is entered. Therefore, MACRO-8 would have to be reloaded to handle subsequent programs that use macros, double precision integers, or floating-point numbers.

In the high version, the high-speed punch may be used as the output device by changing the contents of location 0004 from 2600 to 0600. This is useful for long third pass listings, since the punched output from the high-speed punch can be subsequently listed off line. It is advised that this change not be made until pass 3, so that pass 1 and 2 error diagnostics will be printed.

DIAGNOSTICS

Format: ERROR CODE ADDRESS

where ERROR CODE is a two-letter code listed below, and ADDRESS is either the absolute address of the error or the address of the error relative to the last symbolic tag on that page.

Error Code	Explanation
BE	MACRO-8 internal tables have overlapped
IC	Illegal character
ID	Illegal redefinition of a symbol

Error Code	Explanation
IE	Illegal equal sign
II	Illegal indirect address
IM	Illegal format in a macro definition
LG	Link generated to off-page address*
MP	Missing parameter in macro call
PE	Current, nonzero page exceeded
SE	Symbol table exceeded
US	Undefined symbol
ZE	Page zero exceeded

*This is to inform the user of off-page references which may not be an error. This diagnostic can be suppressed to speed up pass 2 assembly by setting location 1234=7200.

PURPOSE

The 8K SABR Assembler (Symbolic Assembler for Binary Relocatable programs) is an advanced one-pass assembler for use with source programs written in the SABR language. SABR is core page independent, it automatically generates off-page and off-field references for direct and indirect statements, it automatically connects instructions on one page to those that overflow onto the next, and it contains an impressive list of pseudo-ops which include external subroutine calling, argument passing, and conditional assembling. The assembled output is punched on paper tape in binary relocatable code. SABR offers an optional second pass to produce a side-by-side octal/symbolic listing of the assembled program. (See DEC-08-ARXA-D for details.)

STORAGE REQUIREMENTS

8K SABR requires locations:

0-777 in field Ø
0600-1100 and 2000-2427 in field 1
Starting address: 0200 (field Ø)

8K Linking Loader requires locations:

0-777 in field Ø
0-177 and 6200-7577 in highest available field
0-177 in all available fields
Starting address: 0200 (in highest field)

8K Library Subprograms require locations:

0020-0032 in field where used

LOADING

The Binary Loader is used to load the 8K SABR and 8K Linking Loader into core. The 8K Linking Loader is used to load the assembled program and 8K Library Subprograms into core.

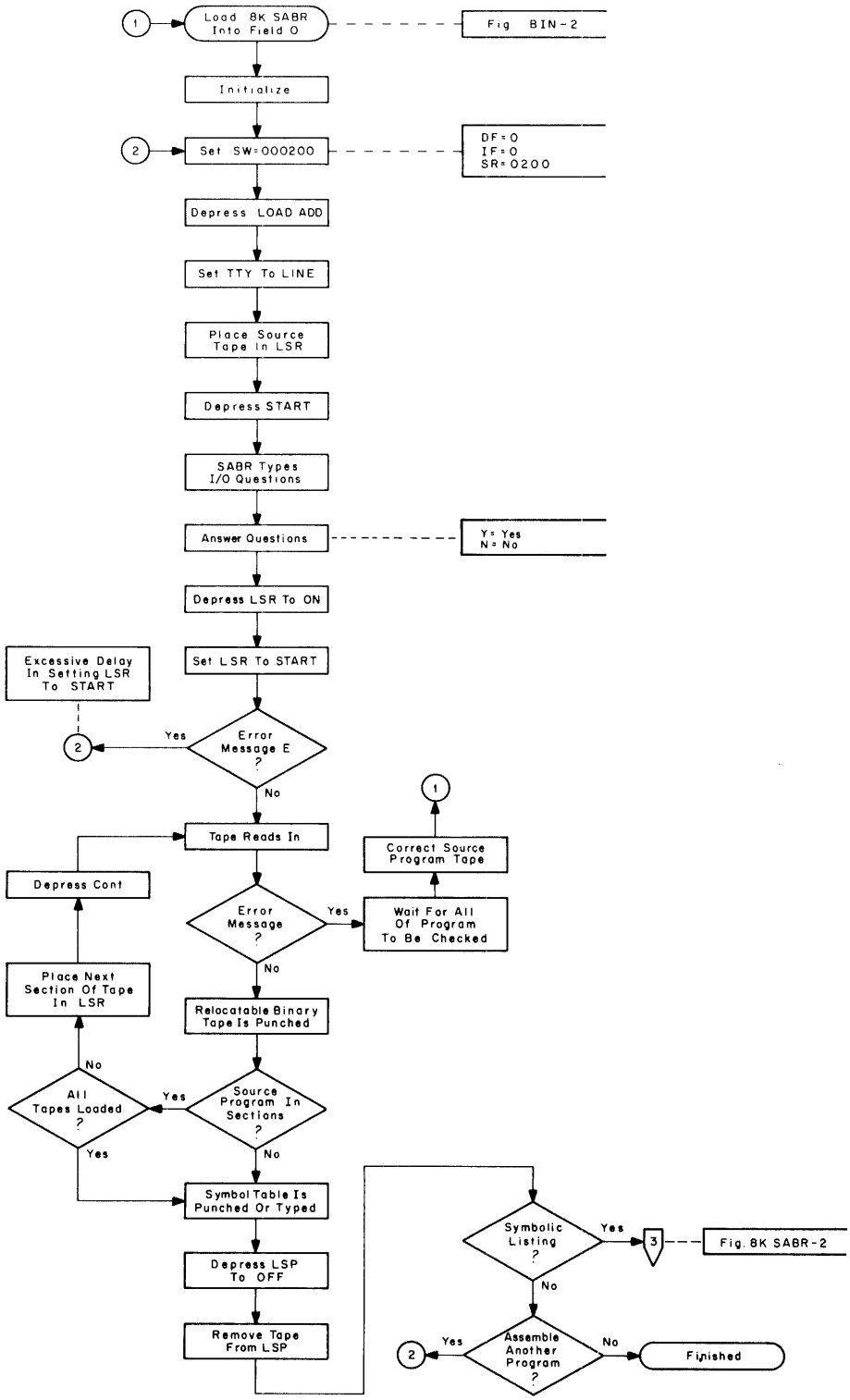


Figure 8K SABR-1 Assembling an 8K SABR Source Program Using the Low-Speed Reader/Punch.

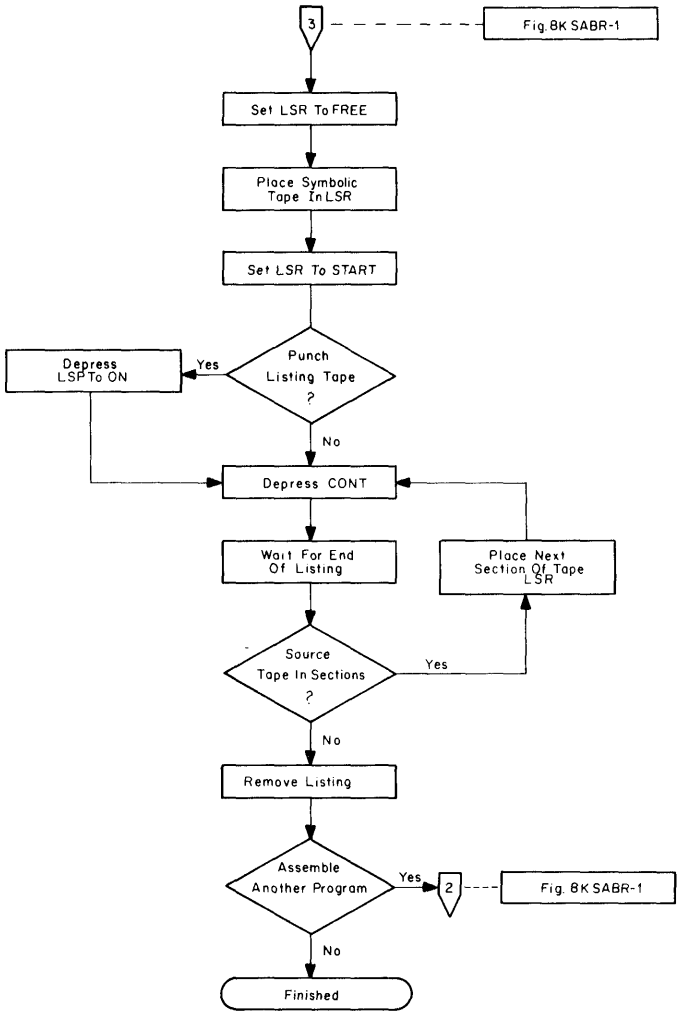


Figure 8K SABR-2 Listing an Assembled Program Using the Low-Speed Reader/Punch.

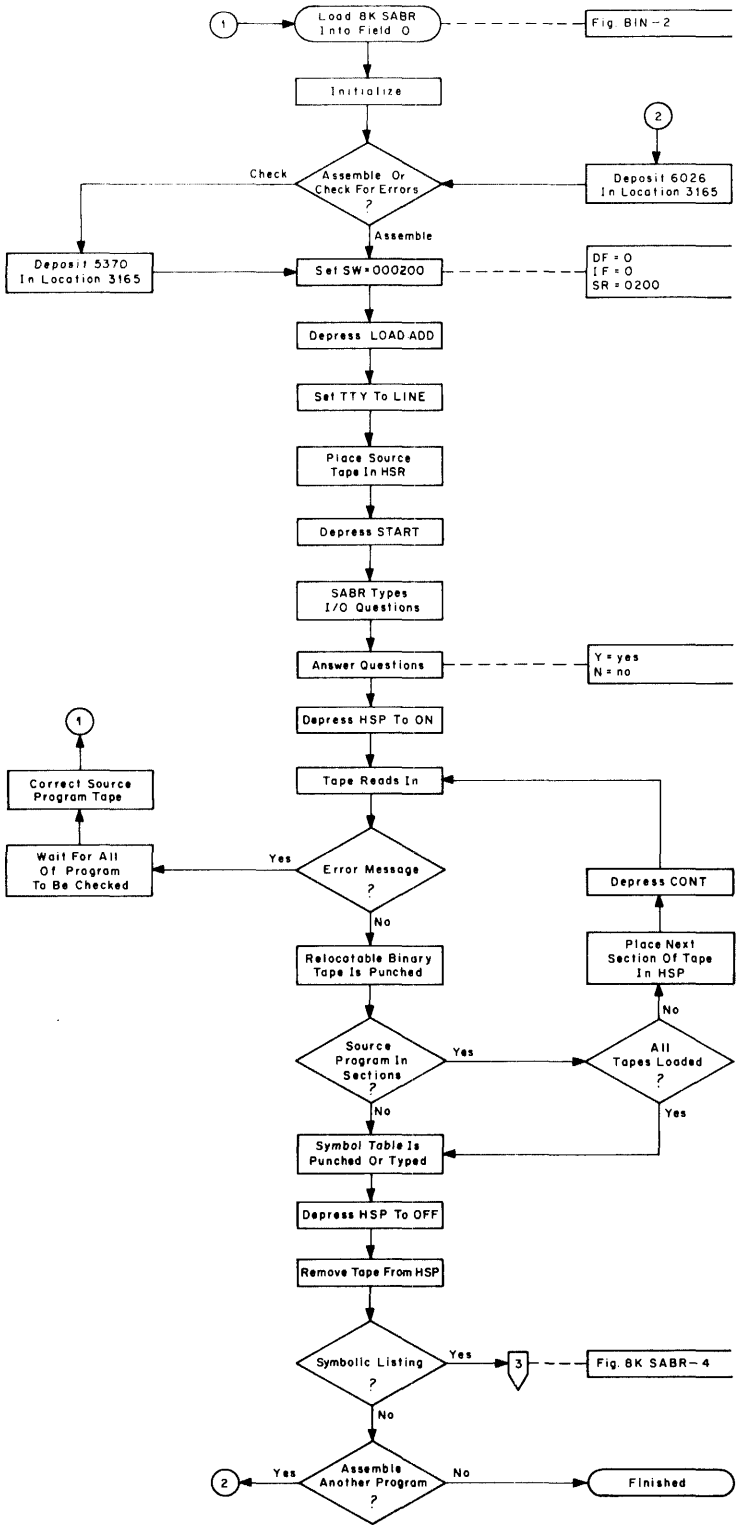


Figure 8K SABR-3 Assembling an 8K SABR Source Program Using the High-Speed Reader/Punch.

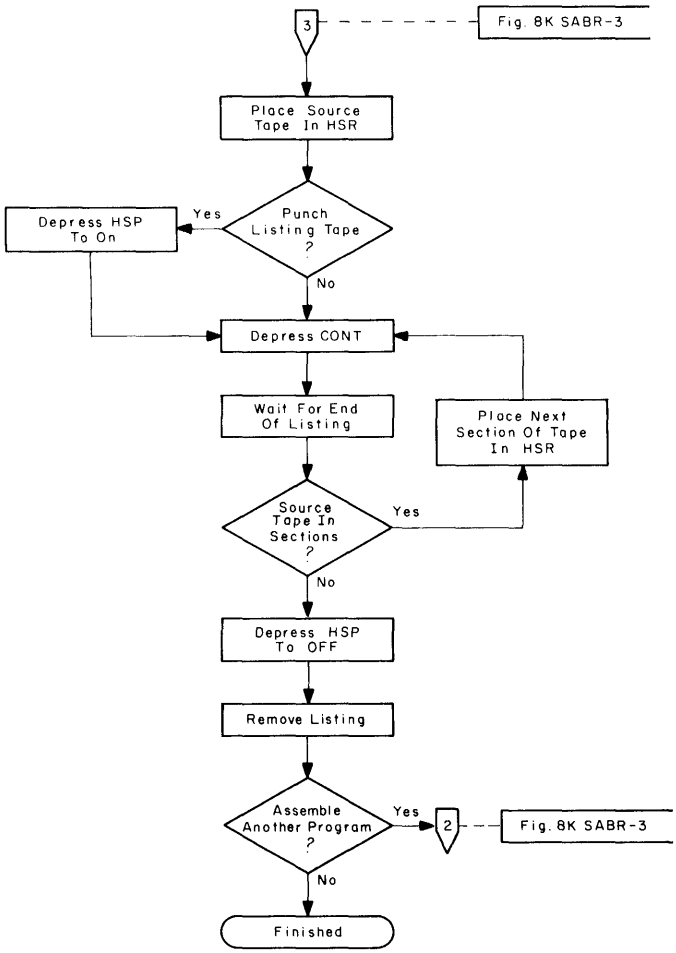


Figure 8K SABR-4 Listing an Assembled Program Using the High-Speed Reader/Punch

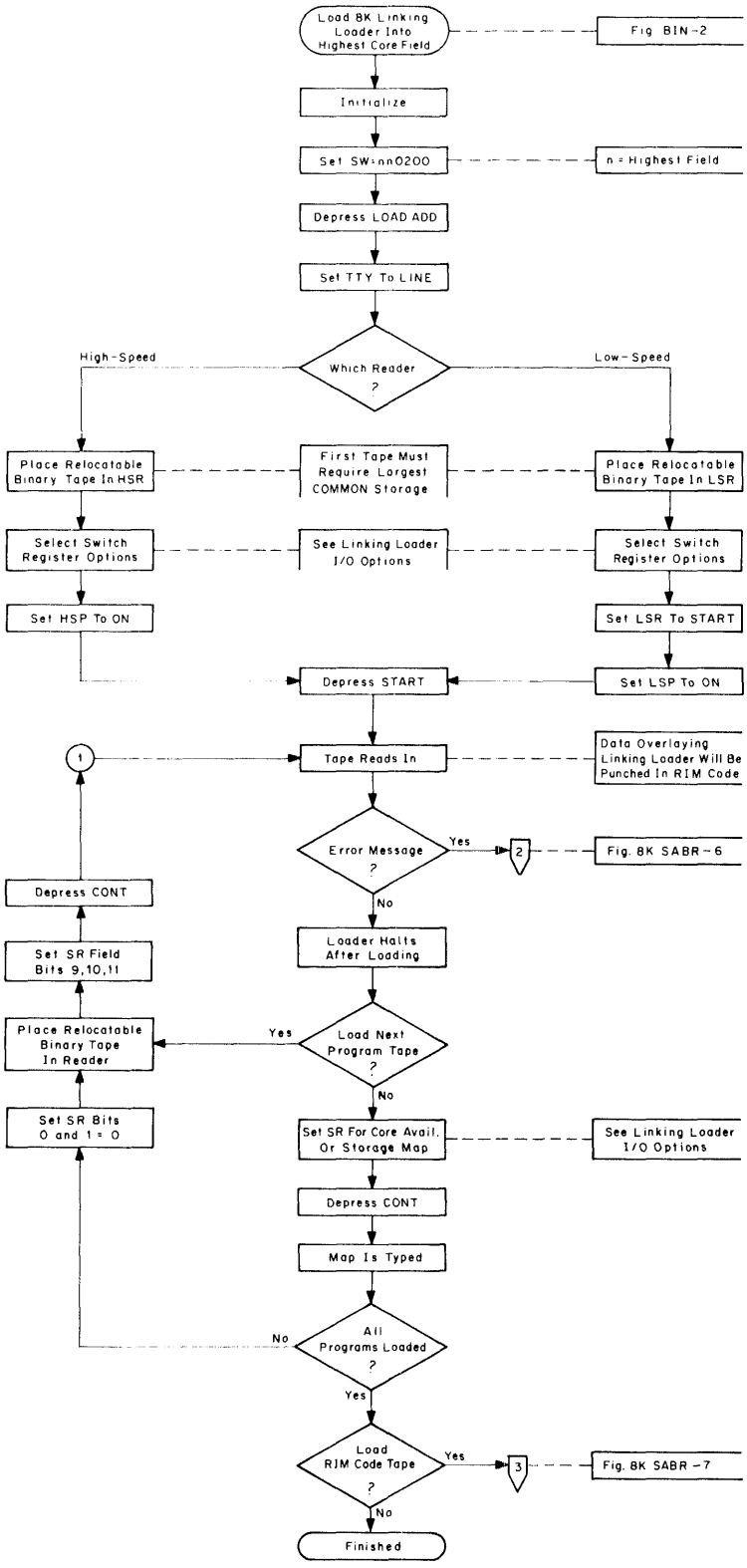


Figure 8K SABR-5 Loading Programs and Subprograms Into Core Using the 8K Linking Loader.

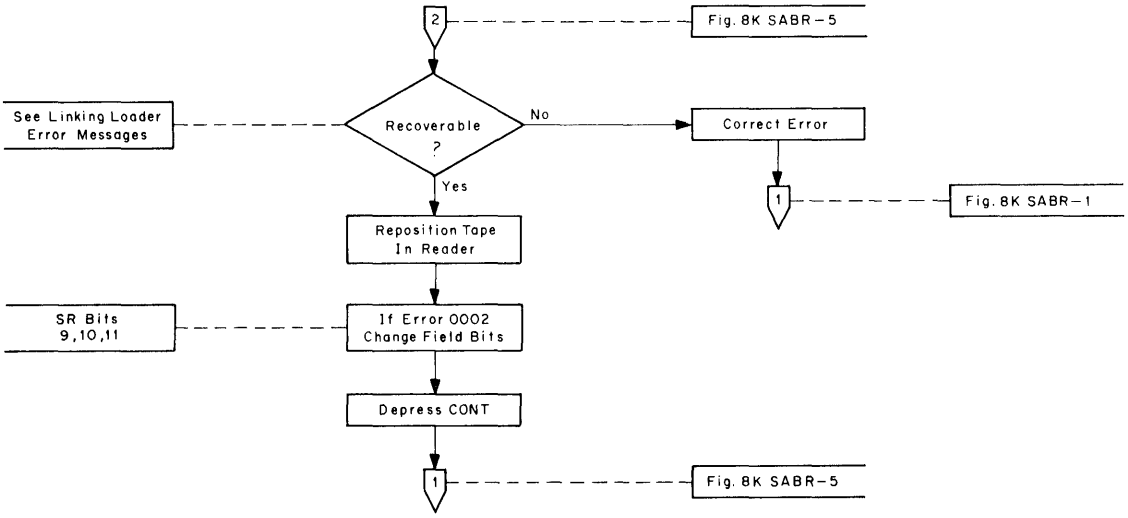


Figure 8K SABR-6 Recovering From Linking Loader Error Message

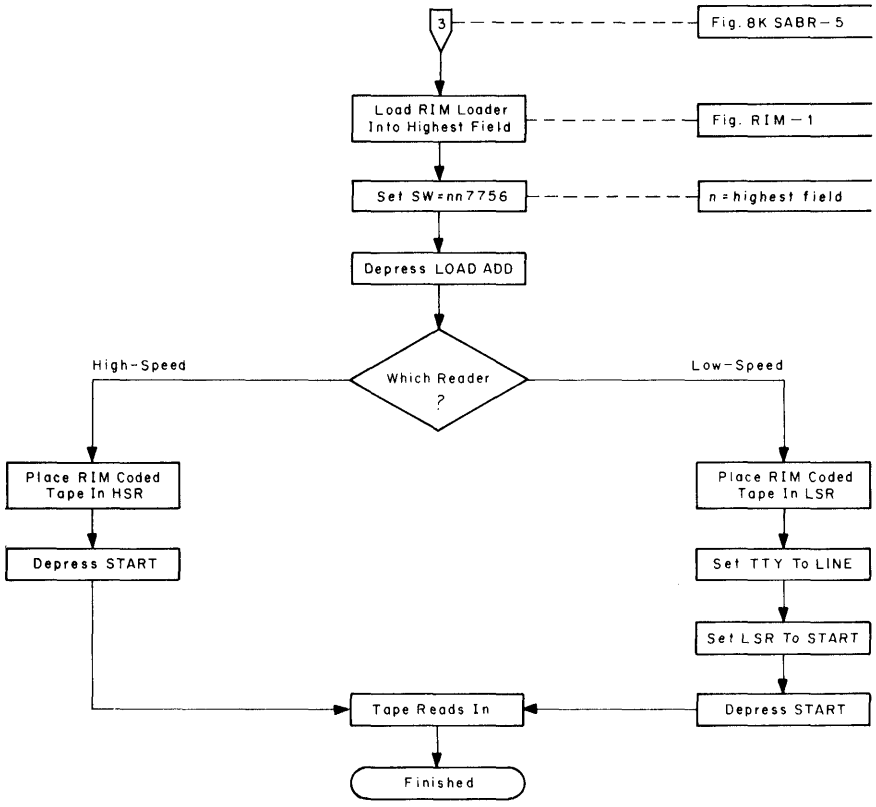


Figure 8K SABR-7 Loading RIM Coded Tape Using RIM Loader.

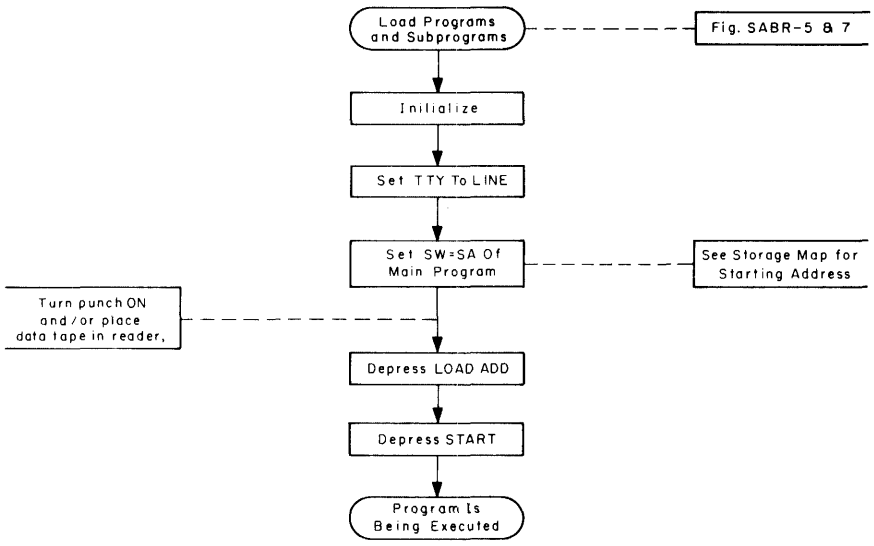


Figure 8K SABR-8 Executing SABR Programs Using the Run-Time Linkage Routines.

SYMBOL TABLE

The symbol table is listed at the end of assembly; each symbol is listed with its relative address. Special symbols are identified as follows:

ABS	The address is absolute.
COM	The address is in COMMON.
OP	The symbol is an operator.
EXT	The symbol is an external, and thus may or may not be defined. If not defined, there is no difficulty, it is in another program.
UNDF	The symbol is not an external and has not been defined in the program.

RESERVED LOCATIONS

Locations reserved for special use are below.

Library Linkage Routines require:

0200-0777	in field zero
0007, 0033-0073	in every field
0020-0032	in field where routines reside

I/O Handler Routines require:

0176-0177	in field where routines reside

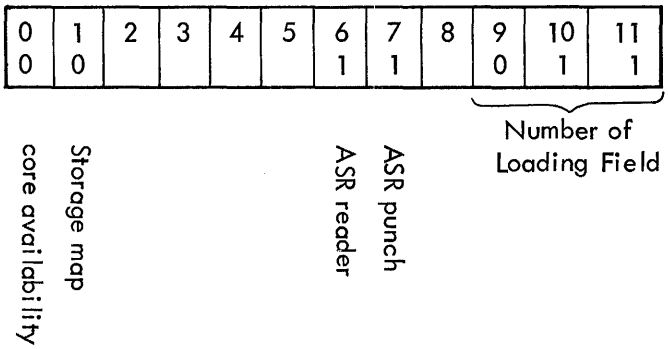
Available to user, in every field:

0000-0006	interrupts, debugging, etc.
0010-0017	auto-index registers
0020-0022	floating-point AC (use with care)
0023-0032	arbitrary
0074-0177	arbitrary

LINKING LOADER I/O OPTIONS

When loading programs and subprograms using the 8K Linking Loader, the the following switch register options are read to perform the following I/O options.

Switch Register*



*All other switch register bits are irrelevant.

Any time the Linking Loader halts, the user may access memory directly via the DEPosit and EXAMine console switches. After this is done the Linking Loader may be restarted via the console switches at location 7200 (in the highest field, where the Linking Loader resides).

8K LIBRARY SUBPROGRAMS

These subprograms are contained on two tapes, organized as shown below. Each subprogram is separated from the other by a noticeable length of blank tape. Load only those subprograms required by the relevant user program.

Tape 1.	IOH	contains	IOH, READ, WRITE
	FLOAT	contains	FAD, FSB, FMP, FDV, STO, FLOT, FLOAT, FIX, IFIX, IFAD, ISTO, CHS, CLEAR
	INTEGER	contains	IREM, ABS, IABS, DIV, MPY, IRDSW
	UTILITY	contains	TTYIN, TTYPUT, HSIN, HSOUT,OPEN CKIO
	ERROR	contains	SETERR, CLRERR, ERROR
Tape 2.	SUBSC	contains	SUBSC
	POWERS	contains	IIPOW, IFPOW, FIPOW, FFPOW,EXP, ALOG
	SQRT	contains	SQRT
	TRIG	contains	SIN, COS, TAN
	ATAN	contains	ATAN

ERROR MESSAGES

All assembly and execution time errors are fatal. Therefore, punched output by the assembler should be suppressed until the source program is correct. Always examine the assembly symbol table listing for undefined symbols before loading and executing the assembled program.

Do not attempt to load and execute a program which has an assembly error. Unpredictable results will occur. Do not attempt to proceed after an execution time error by depressing CONT. Unpredictable results will occur.

SABR

During assembly, error messages are typed as they occur.

Format:

C AT LOC +0004

where C (an illegal character) occurred at the 4th instruction after location tag LOC. Line counts include comment and blank lines.

During listing, error messages are typed in the address field of the instruction line containing the error.

Error Code	Explanation
A	Too many or too few ARGs to a CALL statement.
C	Illegal character.
E	No END statement.
I	Illegal syntax: pseudo-op with improper argument, quote mark with no argument, text string not terminated, MRI with improper address, or illegally combined microinstruction.
M	Multiple defined symbol (appears only during assembly)
S	Either: symbol table overflow, common storage is full, or more than 64 different user-defined symbols in a core page.
UNDF	Undefined symbol (appear only in symbol table listings).

All errors are fatal; correct error and reassemble.

LINKING LOADER

During loading, the error message is typed as it occurs, and the partially loaded program or subprogram is ignored (removed from core). Format:

ERROR xxxx

where xxxx is the error code number.

Error Code	Explanation
0001	Symbol table overflow (more than 64 subprogram names)
0002	Current field is full.
0003	Program with largest common storage was not loaded first.
0004	Checksum error in input tape.
0005	Illegal relocation code.

To recover from errors 2, 4, and 5, reposition the tape in the reader at the leader code of the program or subprogram and depress CONTinue. With error 2, load into a different field.

LIBRARY PROGRAM

During program execution, error messages are typed as they occur.

Format:

"xxxx" ERROR AT LOC nnnn

where xxxx is the error code and nnnn is the location of the error.

Error Code	Explanation
"ALOG"	Attempt to comput log of negative number.
"ATAN"	Result exceeds capacity of computer.
"DIVZ"	Attempt to divide by zero.
"EXP"	Result exceeds capacity of computer.
"FIPW"	Error in raising a number to a power.
"FMT1"	Multiple decimal points.
"FMT2"	E or . in integer.
"FMT3"	Illegal character in I, E, or F field.
"FMT4"	Multiple minus signs.
"FMT5"	Invalid FORMAT statement.
"FLPW"	Negative number raised to floating power.
"FPNT"	Floating-point error.
"SQRT"	Attempt to square root a negative number.

When an error occurs, execution stops; correct error and reassemble.

To pinpoint the location of the error:

1. From the Storage Map, determine the next lowest numbered location (external symbol) which is the entry point of the program or subprogram in error.

2. Subtract in octal the entry point location from the error location in the error message.

3. From the assembly symbol table, determine the relative address of the external symbol found in step 1 and add that relative address to the result of step 2.

4. The sum of step 3 is the relative address of the error, which can then be compared with the relative address of the numbered statements in the program.

When multiple error messages are typed, the location of the last error message is relevant--the other errors are to subprograms called by the statement at the relevant location.

PURPOSE

The Dynamic Debugging Technique for the PDP-8 computers facilitates program debugging by allowing the user to examine core memory locations (registers) and change and correct their contents, place and remove strategic halts and automatically restore and execute the instructions replaced by the halts, and much more. Communication is via the Teletype keyboard using defined commands and the symbolic language of the source program or octal representation, with DDT-8 performing all translation to and from the binary representation. (See DEC-08-CDDB-D for details.)

STORAGE REQUIREMENTS

DDT-8 requires locations 0004 and 5237-7577 (2341_8 locations)
Permanent Symbol Table requires locations 5237-5000
External Symbol Table is allotted locations 5000-3030 (250 symbol capacity)
Starting Address = 5400

LOADING

BIN is used to load DDT-8 and the object program into core.

The user should have at the console the Pass 3 listing of his object program so that the listing can be updated to reflect any debugging change made to the program.

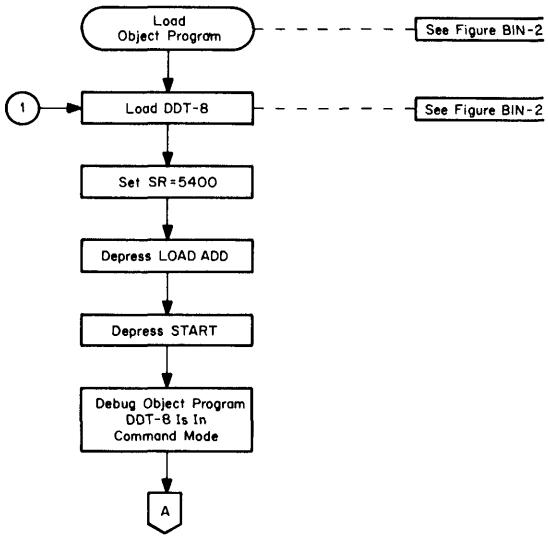


Figure DDT-1 Loading and Executing DDT-8

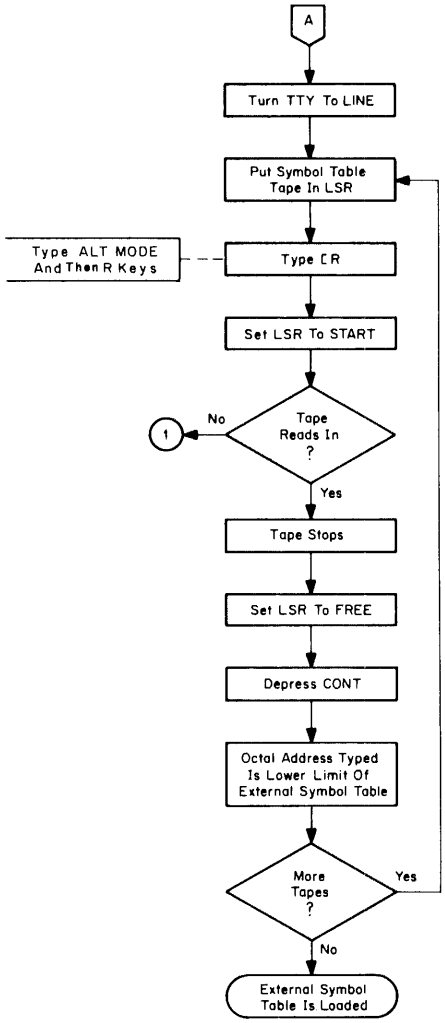


Figure DDT-2 Loading External Symbol Table Tapes (LSR Only)

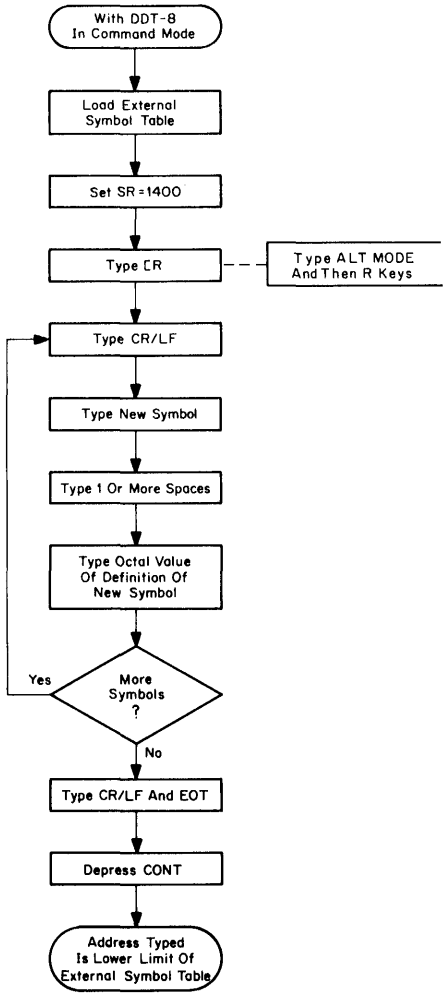


Figure DDT-3 Appending New Symbols to External Symbol Table

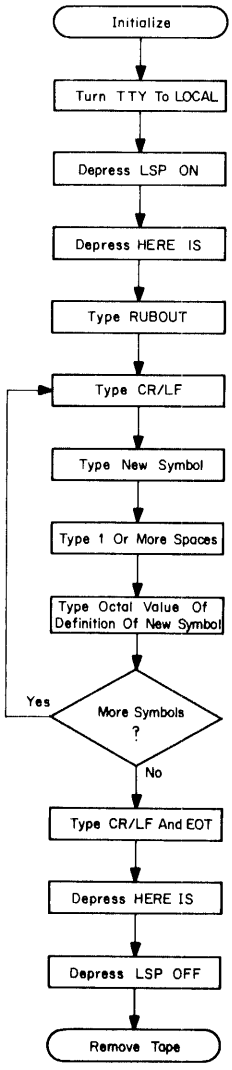


Figure DDT-4 Generating New External Symbol Tape Off-Line
(TTY and LSP Only)

RESTART PROCEDURE

Restart at location 5400 and DDT-8 will be in control.

If the user wishes to restart DDT-8 before he has punched a complete tape with checksum, he must restart at location 5401 to perserve the checksum.

EDITING NOTES

a. Do not open any symbol table location.

b. To enter a combined operate class and IOT instruction into an open location, the combination must contain no more than two mnemonics, the second of which must be CLA. Any other combination is ignored.

c. The symbol table tape is loaded using the LSR only.

d. Each user symbol occupies four locations in the symbol table area.

e. Input is interrupted when symbol table storage is full.

COMMANDS

Mode Control	Explanation
[O	Sets DDT-8 to type out in octal mode.
[S	Sets DDT-8 to type out in symbolic mode.
Input	
[R	Read symbol tape into external table from LSR, or define new symbol from keyboard.
Program Examination and Modification	
k/	Open location k (k may be octal or symbolic).
RETURN	Close location currently open; enter modification, if any.
LINE FEED	Close location currently open and open next sequential location; enter modification, if any.
↑ (SHIFT/N)	Close location currently open and open location address therein; enter modification, if any.
Breakpoint Insertion and Control	
[B	Remove current breakpoint.
k[B	Insert a breakpoint at location k.
k[G	Go to location k and start program execution.
n[C	Continue from breakpoint, execute breakpoint n times and return control to user. If n is absent, it is assumed to be 1.
Word Search	
N[W	Begin word search for all occurrences of expression N masked by the contents of [M between the limits imposed by [L and [U. [M, [L, and [U are locations within DDT-8 which may be opened, modified, and closed exactly as any general register k in the user's program.

Output

[T	Punch leader/trailer code.
a;b[P	Punch binary tape from memory bounded by addresses a and b.
[E	Punch end of tape (i.e., checksum and trailer).

Address Tags

Command	Explanation
[A	Accumulator storage (at breakpoints)
[L	Lower limit of search
[U	Upper limit of search
[M	Mask; used in search
[Y	Link storage (at breakpoints)

The left bracket ([) which precedes a command letter is printed in response to typing the ALT MODE (or ESC) key, i.e., type ALT MODE (or ESC, whichever is on your keyboard) key and then the desired command letter key.

Special Characters

Character	Explanation
(space)	Separation character
+ (plus)	Specifies address arguments relative to symbols
- (minus)	Same as +
. (period)	Current location; used in address arguments
= (equal)	Type last quantity as an octal integer
RETURN	Make modifications, if any, and close register
LINE FEED	Make modifications, if any, close location, and open next sequential location.
/ (slash)	Location examination character; when following the address location, the location is opened and its contents printed
↑ (up-arrow)	When following a location printout, the location addressed therein is opened
← (back-arrow)	Delete the line currently being typed

DIAGNOSTICS

DDT-8 checks for the errors listed below and types a ? when any is detected. All data between the error point and the previous tab or carriage return is ignored.

a. Undefined or illegal symbol
b. Illegal character
c. Undefined control command
d. Off-page addressing

PURPOSE

The Octal Debugging Technique for the PDP-8/I is a debugging program which facilitates communication with and alteration of the object program. Communication with the program is from the Teletype keyboard, using octal numbers. (See DEC-08-COCO-D for details.)

STORAGE REQUIREMENTS

ODT-8 requires 600_8 consecutive locations, and 1 location on page 0 for breakpoint location.

Low version: locations 1000-1577
High version: locations 7000-7577

Breakpoint is initially at location 0004.

Starting Address is 1000 (low) or 7000 (high)

LOADING

BIN is used to load ODT-8 and the object program into core. The user should have at the console the octal listing of his object program so that the listing can be updated to reflect any debugging change made to the program.

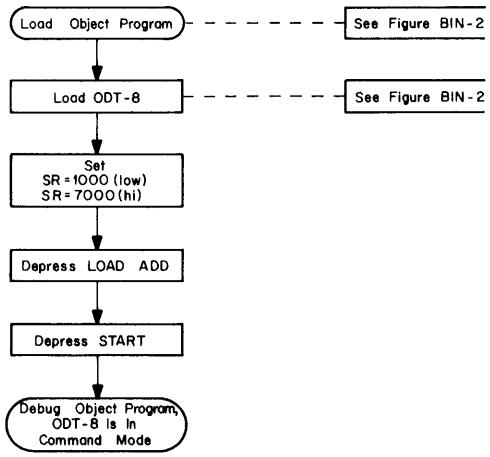


Figure ODT-1 Loading and Executing ODT-8

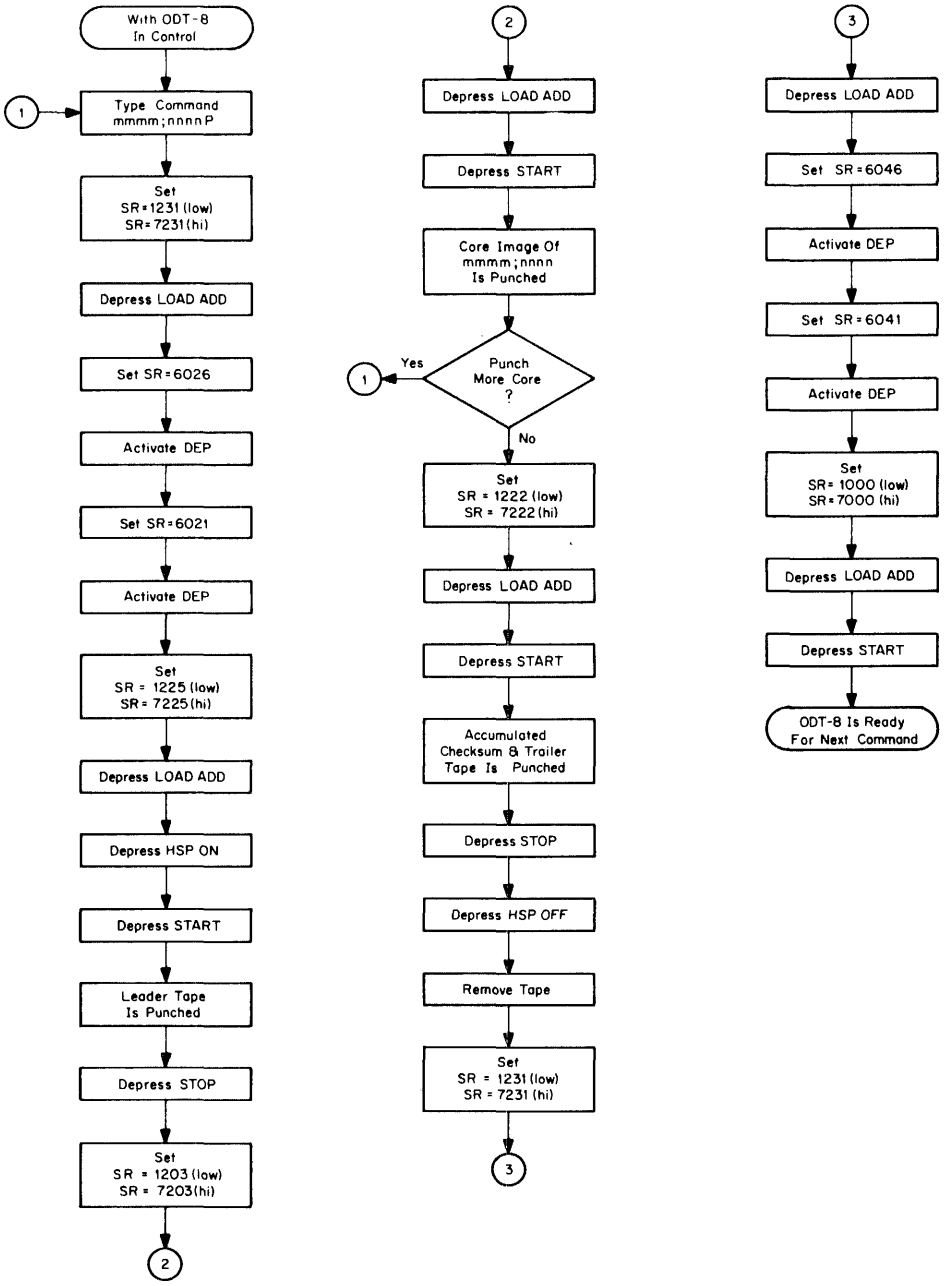


Figure ODT-2 Generating Binary Tape Using High-Speed Punch

COMMANDS

Command	Explanation
/	Reopen latest opened location.
nnnn/	Open location nnnn.
RETURN	Close previously opened location.
LINE FEED	Close location and open next sequential location.
↑ (SHIFT/N)	Close location, take contents of that register as a memory reference and open it.
← (SHIFT/O)	Close location, open indirectly.
nnnnG	Transfer program control to location nnnn.
B	Remove the breakpoint and restore original contents of that location.
nnnnB	Establish the breakpoint at location nnnn.
A	Open location containing AC.
C	Proceed from the breakpoint.
nnnnC	Continue from the breakpoint and iterate past the breakpoint nnnn times; stop at breakpoint
M	Open the search mask. Initially set to 7777 LINE FEED Open lower search limit LINE FEED Open upper search limit
nnnnW	Search defined upper and lower limits of core for nnnn of search mask.
T	Punch leader/trailer tape.
mmmm;nnnnP	Punch binary core image of locations mmmm through nnnn.
E	Punch checksum and trailer tape.

RELOCATING THE BREAKPOINT

ZPAT (the breakpoint symbol) is initially set to location 0004. The breakpoint location can be relocated to any location on page 0 by setting ZPAT equal to the desired location.

RESTRICTIONS

Although ODT-8 is relocatable to any page, it will not operate outside the field in which it is located.

ODT-8 will not turn on the program interrupt. However, it does turn off the interrupt when a breakpoint is encountered. This prevents disrupting interrupts.

The user's program must not use or reference any location occupied or used by ODT-8.

The breakpoint location must not be used by the user program.

DIAGNOSTICS

When ODT-8 detects an error it types a ? followed by a carriage return-line feed. ODT-8 checks for the following conditions.

a. Only legal control characters and octal digits are acceptable, any other character causes the character or whole line to be ignored.

b. G typed alone is an error; control will be transferred to location 0000.

c. Typing a P command with the punch ON is an error; ASCII characters will be punched on the binary tape.

d. Octal numbers must be from 1 to 4 digits; more than four digits is an error.

e. An illegal character (neither a valid control character nor a 1- to 4-digit octal number) causes the current line to be ignored.

PURPOSE

FOCAL (FOrmula CALculator) is an on-line, conversational, interpretive program used by students, engineers, and scientists in solving virtually any mathematical problem, and much more. Additional segments (overlays) expand FOCAL's power, flexibility, and capability:

Utility Segments

4-WORD	increases calculating accuracy to 10 digits.
8K	provides space for large user programs on an 8K computer.
EXTEN.BIN	removes or restores extended functions.

Graphics Segments

CLINE	permits the interaction of an oscilloscope (34D).
PLOTR	permits the interaction of an incremental plotter (CALcomp plotter).
GRAPH	permits the interaction of a storage tube display (KV8/I).

Multi-User Segments

QUAD	permits four users to share FOCAL on an 8K computer,
LIBRA	permits seven users to share FOCAL on an 8K computer with disk (not covered herein, see DEC-08-AJAD-D for details).

FOCAL may be combined with one or more segments as shown below

Segment	Combination														
4-WORD	x	x	x	x	x	x									
8K		x	x				x	x	x	x	x				
EXTEN.BIN			x	x				x							
CLINE									x			x			
PLOTR										x			x		
GRAPH											x			x	
QUAD					x										x
LIBRA						x									

A FOCAL program calling a machine language subprogram is used in the System Demonstration section of this guide.
(See DEC-08-AJAD-D for details.)

STORAGE REQUIREMENTS

FOCAL occupies locations 1-330 and 4600-7576 (6300_8 locations)
Extended functions occupy locations 4600-5377 (1000_8 locations)
Starting Address = 0200

OPTIONAL EQUIPMENT

PDP-12, LINC-8, or LAB-8 computer; analog-to-digital converter; storage tube display; incremental plotter; DECdisk; extended memory.

LOADING

The BIN Loader is used to load FOCAL and segments into core. FOCAL accepts user programs from the Teletype keyboard, low-speed tape reader, or high-speed tape reader.

Initial Dialogue (alternate responses)

```
CONGRATULATIONS!!
YOU HAVE SUCCESSFULLY LOADED 'FOCAL, 1969' ON A PDP-8 COMPUTER.

SHALL I RETAIN LOG, EXP, ATN?:  YES

PROCEED.
*

CONGRATULATIONS!!
YOU HAVE SUCCESSFULLY LOADED 'FOCAL, 1969' ON A PDP-8 COMPUTER.

SHALL I RETAIN LOG, EXP, ATN?:  NO

PROCEED.
*

CONGRATULATIONS!!
YOU HAVE SUCCESSFULLY LOADED 'FOCAL, 1969' ON A PDP-8 COMPUTER.

SHALL I RETAIN LOG, EXP, ATN?:  NO
SHALL I RETAIN SINE, COSINE?:  NO

PROCEED.
*
```

(about 700_{10} locations for user programs)

(about 900_{10} locations for user programs)

(about 1100_{10} locations for user programs)

If the above functions are not retained, they may be restored by loading the first section of the EXTEN.BIN overlay tape. If they are retained and are no longer desired, they may be removed by loading the last section of the EXTEN.BIN overlay tape. This overlay must not be used with the graphics and multi-user segments.

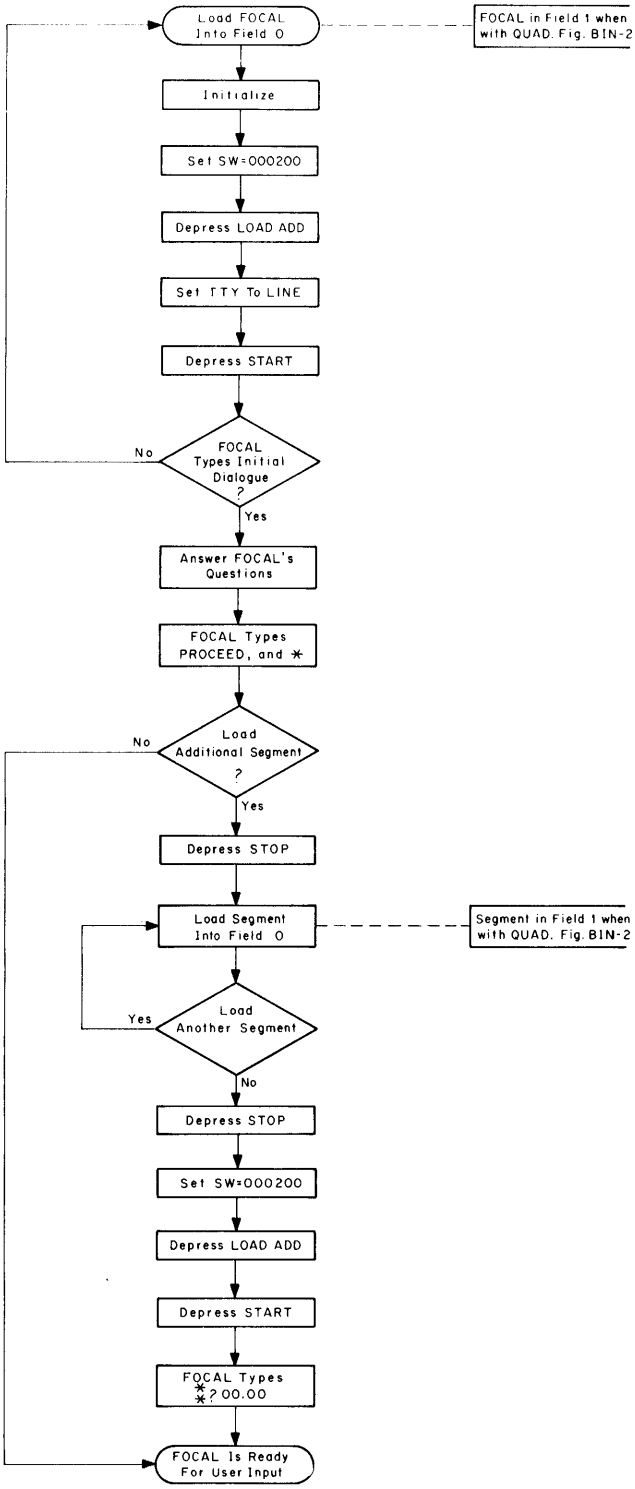


Figure FOCAL-1 Loading and Starting FOCAL (and Segments).

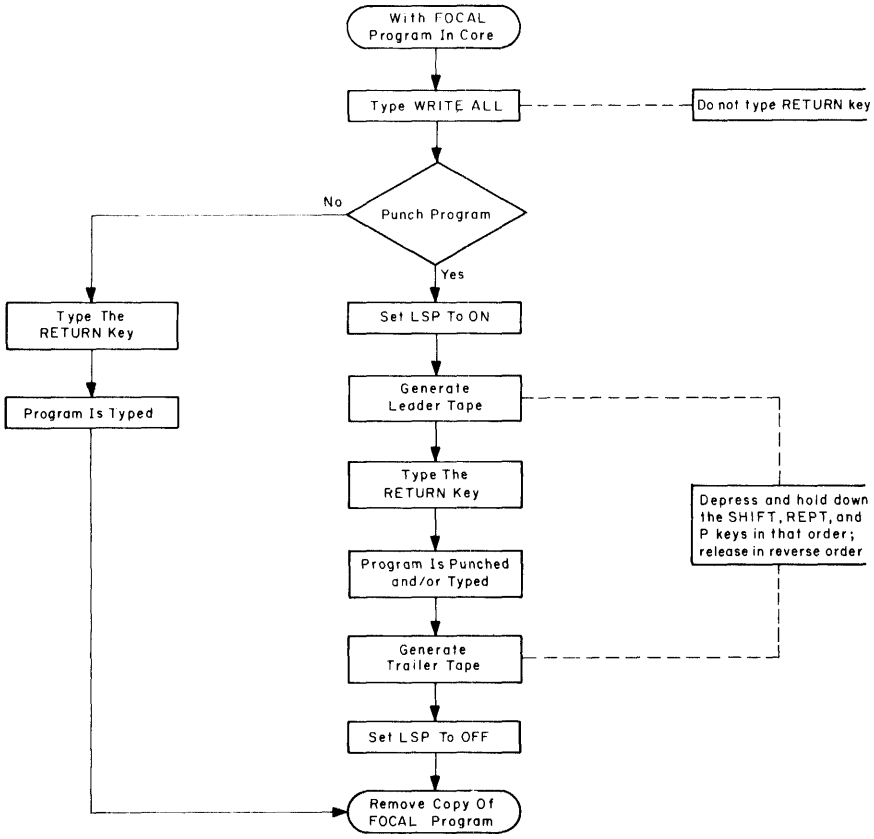


Figure FOCAL-2 Saving a FOCAL Program (Teletype Console)

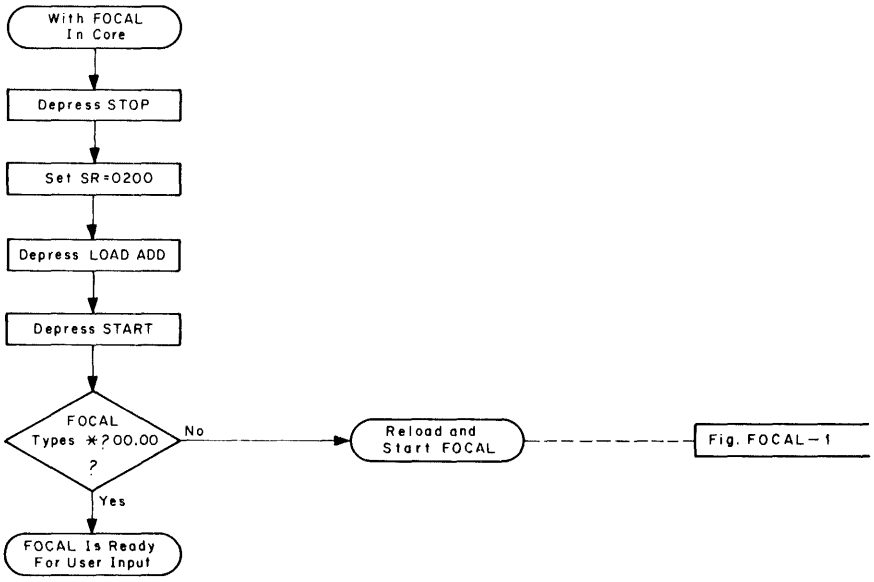


Figure FOCAL-3 Restart Procedure, Computer Console

COMMANDS

Command	Abbreviation	Example of Form	Explanation
ASK	A	ASK X, Y, Z	FOCAL types a colon for each variable the user types a value to define each variable.
COMMENT	C	COMMENT	If a line begins with the letter C, the remainder of the line will be ignored.
CONTINUE	C	C	Dummy lines
DO	D	DO 4.1	Execute line 4.1; return to command following DO command.
		DO 4.0 DO ALL	Execute all group 4 lines; return to com mand following DO command, or when RETURN is encountered.
ERASE	E	ERASE	Erases the symbol table.
		ERASE 2.0	Erases all group 2 lines.
		ERASE 2.1	Deletes line 2.1.
		ERASE ALL	Deletes all user input.
FOR	F	FOR i=x,y,z; (commands) FOR i=x,z; (commands)	Where the command following is execu ted at each new value. x=inital value of i y=value added to i until i is greater thc z.
GO	G	GO	Starts indirect program at lowest numbe ed line number.
GO?	G?	GO?	Starts at lowest numbered line number and traces entire indirect program until another ? is encounted, until an error is encountered, or until completion of program.
GOTO	G	GOTO 3.4	Starts indirect program (transfers contro to line 3.4). Must have argument.

IF	I	IF (X) Ln, Ln, Ln
		IF (X) Ln, Ln; (commands)
		IF (X) Ln; (commands)
			Where X is a defined identifier,a value or an expression, followed by three lin numbers.
			If X is less than zero, control is trans- ferred to the first line number.
			If X is equal to zero, control is trans- ferred to the second line number.
			If X is greater than zero, control is transferred to the third line number.

Command	Abbreviation	Example of Form	Explanation
LOCATIONS*	L	L	FOCAL types four locations indicating start and end of text area, end of variable list, and bottom of pushdown list. (void with PLOTR)
MODIFY	M	MODIFY 1.15	Enables editing of any character on line 1.15 (see below).
QUIT	Q	QUIT	Returns control to the user.
RETURN	R	RETURN	Terminates DO subroutines, returning to the original sequence.
SET	S	SET A=5/B*C;	Defines identifiers in the symbol table.
TYPE	T	TYPE A+B-C;	Evaluates expression and types out = and result in current output format.
		TYPE A-B, C/E;	Computes and types each expression separated by commas.
		TYPE "TEXT STRING" '	Types test. May be followed by ! to generate carriage return-line feed, or # to generate carriage return.
WRITE	W	WRITE WRITE ALL	FOCAL types out the entire indirect program.
		WRITE 1.0	FOCAL types out all group 1 lines.
		WRITE 1.1	FOCAL types out line 1.1.

*Before using the LOCATIONS command deposit 5177 into location 7600.

SPECIAL CHARACTERS

Mathematical operators:

↑	Exponentiation	Order of precedence is as listed; properly paired enclosures are evaluated first; otherwise evaluation is from left to right.
*	Multiplication	
/	Division	
+	Addition	
-	Subtraction	

Control characters:

%	Output format delimiter	
!	Carriage return and line feed	
#	Carriage return	
$	Type symbol table contents	
( )	Parentheses	(expression enclosures)
[ ]	Square brackets	
< >	Angle brackets	
" "	Quotation marks	(text string)
? ?	Question marks	(trace feature)
*	Asterisk	(high-speed reader input)

Terminators:

SPACE key (names)	(nonprinting)
RETURN key (lines)	
ALT MODE key (with ASK statement)	
Comma (expressions)	
Semicolon (commands and statements)	

CTRL/key combinations:

CTRL/BELL	Used with MODIFY
CTRL/C	Keyboard restart of FOCAL
CTRL/L	Used with MODIFY

RESTART FROM KEYBOARD

Type the CTRL/C keys. FOCAL will type ?01.00 indicating a keyboard restart, and an asterisk on the next line indicating it is ready for user input.

CTRL/C indicates holding down the CTRL key while depressing the C key.

INPUT, PROGRAM

Low-Speed Tape Reader

When loading a long program tape into FOCAL the user can suppress the echo (printing) feature by changing the content of location 2475 to 7000. This will cause only asterisks to be typed as the tape is being read; there will not be a carriage return-line feed at the end of the line.

Entries from the keyboard will not echo unless each entry is preceded by a TYPE command. Output will be typed in the normal manner.

To restore the echo feature, depress the STOP key on the computer console and deposit 4277 into location 2475.

High-Speed Tape Reader

**

The second * was typed by the user. Input is from the high-speed reader until occurrence of next *. FOCAL types * for each line number read in from the reader.

1.10;

User typed 1.10*;. Input is taken from the high-speed reader until occurrence of next *.

If there is no tape in the high-speed reader or when an end-of-tape condition occurs, FOCAL automatically switches back to keyboard input.

OUTPUT, DATA

Output Format

TYPE % x.y	where x is the total number of digits, and y is the number of digits to the right of the decimal point.
TYPE %6.3, 123.456	FOCAL types: = 123.456
TYPE %	Resets output format to floating point.

Symbol Table

TYPE $	Other statements may not follow on this line.

OVERLOAD RECOVERY

When the program and symbol table areas become too large the error diagnostic ?06.54 will be typed out. The user should then do one of the following.

a. Type ERASE and depress RETURN.

b. Restart at location 2216, if ?06.54 follows a legitimate command. This erases all variables.

c. As a last resort, restart at 2213. This erases the text.

MODIFY

After a MODIFY command, the user types a search character, and FOCAL types out the contents of that line until the search character is typed. The user may then perform any of the following operations.

a. Type in new characters. FOCAL will add these to the line at the point of insertion.

b. Type a CTRL/L. FOCAL will proceed to the next occurence of the search character.

c. Type CTRL/BELL. After this, the user may change the search character.

d. Type RUBOUT. This deletes characters to the left, one character for each time the user strikes the RUBOUT key.

e. Type ← . Deletes the line over to the left margin, but not the line number .

f. Type RETURN. Terminates the line, deleting characters over to the right margin.

g. Type LINE FEED. Saves the remainder of the line from the point at which LINE FEED is typed over to the right margin.

THE TRACE FEATURE

Special Character	Example of Form	Explanation
?	?...?	Those parts of the program enclosed in question marks will be printed out as they are executed.
	?...	If only one ? is inserted, the trace feature becomes operative, and the program is printed out from that point until another ? is encountered, until an error is encountered, or until program completion.

ESTIMATING PROGRAM SIZE

FOCAL requires five words for each identifier stored in the symbol table, and one word for each two characters of stored program. This may be calculated by

$$5s + \frac{c}{2} .1.01 = \text{length of user's program}$$

where

s = Number of identifiers defined

c = Number of characters in indirect program

The following routine allows the user to find out how many core locations are left for his use (applicable with 4K systems only).

```
*FOR I=1,300; SET A(I)=I
?06.54                                   (disregard error code)
*TYPE %4, I*5, " LOCATIONS LEFT "
=+ 705 LOCATIONS LEFT *
```

SUMMARY OF FUNCTIONS

FSQT( )	Square Root	FCOS( )	Cosine*
FABS( )	Absolute Value	FATN( )	Arc Tangent*
FSGN( )	Sign Part	FLOG ( )	Logarithm*
FITR( )	Integer Part	FDIS(x,y)	Scope Function
FRAN( )	Random Number	FADC( )	A-D Input
FEXP( )	Exponential*	FNEW( )	User Function
FSIN( )	Sine*	FCOM( )	LIBRA Common Storage
		FX( )	Graphics (KV8/I)

*Extended functions; may be restored using the EXTEN.BIN overlay (not used with graphics segment).

CALCULATING TRIGONOMETRIC FUNCTIONS

Function	FOCAL Representation	Argument Range	Function Range
Sine	FSIN(A)	$0 \le \lvert A \rvert < 10\uparrow 4$	$0 \le \lvert F \rvert \le 1$
Cosine	FCOS(A)	$0 \le \lvert A \rvert < 10\uparrow 4$	$0 \le \lvert F \rvert \le 1$
Tangent	FSIN(A)/FCOS(A)	$0 \le \lvert A \rvert < 10\uparrow 4$ $\lvert A \rvert \neq (2N+1)\pi/2$	$0 \le \lvert F \rvert < 10\uparrow 6$
Secant	1/FCOS(A)	$0 \le \lvert A \rvert < 10\uparrow 4$ $\lvert A \rvert \neq (2N+1)\pi/2$	$1 \le \lvert F \rvert < 10\uparrow 6$
Cosecant	1/FSIN(A)	$0 \le \lvert A \rvert < 10\uparrow 4$ $\lvert A \rvert \neq 2N\pi$	$1 \le \lvert F \rvert < 10\uparrow 6$
Cotangent	FCOS(A)/FSIN(A)	$0 \le \lvert A \rvert < 10\uparrow 4$ $\lvert A \rvert \neq 2N\pi$	$0 \le \lvert F \rvert < 10\uparrow 440$
Arc sine	FATN(A/FSQT(1-A↑2))	$0 \le \lvert A \rvert < 1$	$0 \le \lvert F \rvert \le \pi/2$
Arc cosine	FATN(FSQT(1-A↑2)/A)	$0 < \lvert A \rvert \le 1$	$0 \le \lvert F \rvert \le \pi/2$
Arc tangent	FATN(A)	$0 \le A \le 10\uparrow 6$	$0 \le F < \pi/2$
Arc secant	FATN(FSQT(A↑2-1))	$1 \le A < 10\uparrow 6$	$0 \le F < \pi/2$
Arc cosecant	FATN(1/FSQT(A↑2-1))	$1 < A < 10\uparrow 300$	$0 < F < \pi/2$
Arc cotangent	FATN(1/A)	$0 < A < 10\uparrow 615$	$0 < F < \pi/2$
Hyperbolic sine	(FEXP(A)-FEXP(-A))/2	$0 \le \lvert A \rvert < 700$	$0 \le \lvert F \rvert \le 5*10\uparrow 300$

Function	FOCAL Representation	Argument Range	Function Range
Hyperbolic cosine	(FEXP(A)+FEXP(-A))/2	0≤|A|<700	1≤F<5*10↑300
Hyperbolic tangent	(FEXP(A)-FEXP(-A))/ (FEXP(A)+FEXP(-A))	0≤|A|<700	0≤|F|≤1
Hyperbolic secant	2/(FEXP(A)+FEXP(-A))	0≤|A|<700	0<F≤1
Hyperbolic cosecant	2/(FEXP(A)-FEXP(-A))	0<|A|<700	0<|F|<10↑7
Hyperbolic cotangent	(FEXP(A)+FEXP(-A))/ (FEXP(A)-FEXP(-A))	0<|A|<700	1<|F|<10↑7
Arc hyperbolic sine	FLOG(A+FSQT(A 2+1))	-10↑5<A<10↑600	-12<F<1300
Arc hyperbolic cosine	FLOG(A_FSQT(A 2-1))	1≤A<10↑300	0≤F<700
Arc hyperbolic tangent	(FLOG(1+A)-FLOG(1-A))/2	0≤|A|<1	0≤|F|<8.31777
Arc hyperbolic secant	FLOG((1/A)+FSQT((1/A 2)-1))	0<|A|≤1	0≤F<700
Arc hyperbolic cosecant	FLOG((1/A)+FSQT((1/A 2)+1))	0<|A|<10↑300	0≤|F|<1400
Arc hyperbolic cotangent	(FLOG(X+1)-FLOG(X-1))/2	1<A<10↑616	0≤F<8

ERROR MESSAGES

Format: ?nn.nn @ nn.nn (error code @ line number)

Code	Meaning
?00.00	Manual start given from console.
?01.00	Interrupt from keyboard via CTRL/C.
?01.40	Illegal step or line number used.
?01.78	Group number is too large.
?01.96	Double periods found in a line number.
?0.1:5	Line number is too large.
?01.;4	Group zero is an illegal line number.
?02.32	Nonexistent group referenced by 'DO'.
?02.52	Nonexistent line referenced by 'DO'.
?02.79	Storage was filled by push-down-list.
?03.05	Nonexistent line used after 'GOTO' or 'IF'.
?03.28	Illegal command used.
?04.39	Left of "=" in error in 'FOR' or 'SET'.
?04.52	Excess right terminators encountered.
?04.60	Illegal terminator in 'FOR' command.
?04.:3	Missing argument in display command.
?05.48	Bad argument to 'MODIFY'.
?06.06	Illegal use of function or number.
?06.54	Storage is filled by variables.
?07.22	Operator missing in expression or double 'E'.
?07.38	No operator used before parenthesis.
?07.:9	No argument given after function call.
?07.;6	Illegal function name or double operators.
?08.47	Parentheses do not match.
?09.11	Bad argument in 'ERASE'.

Code	Meaning
?10.:5	Storage was filled by text.
?11.35	Input buffer has overflowed.
?20.34	Logarithm of zero requested.
?23.36	Literal number is too large.
?26.99	Exponent is too large or negative.
?28.73	Division by zero requested.
?30.05	Imaginary square roots required.
?31.◁	Illegal character, unavailable command, or unavailable function used.

Note: The above error codes apply only to FOCAL, 1969.

PURPOSE

4K FORTRAN (FORmula TRANslator) for the PDP-8 computer is used to compile, debug, and operate a user program written in the PDP-8 version of the 4K FORTRAN language. Compilation requires only one pass. A 4K FORTRAN program using a PAL III subprogram is in the System Demonstration section of this guide. (See DEC-08-AFC0-D for details.)

STORAGE REQUIREMENTS

Compiler and symbol table requires locations 0003-7577 (7574_8 locations)

Starting Address=0200

Symbolprint requires locations 0600-0777 (200_8 location)

Starting Address=0600

Operating System requires:

locations 0-5177 for paper tape I/O (5200_8 locations)
locations 0-5777 for DECtape I/O (6000_8 locations)

Starting Address=0200

LOADING

BIN is used to load the Compiler, Symbolprint, and Operating System into core. The user's program is loaded by the appropriate 4K FORTRAN system program above.

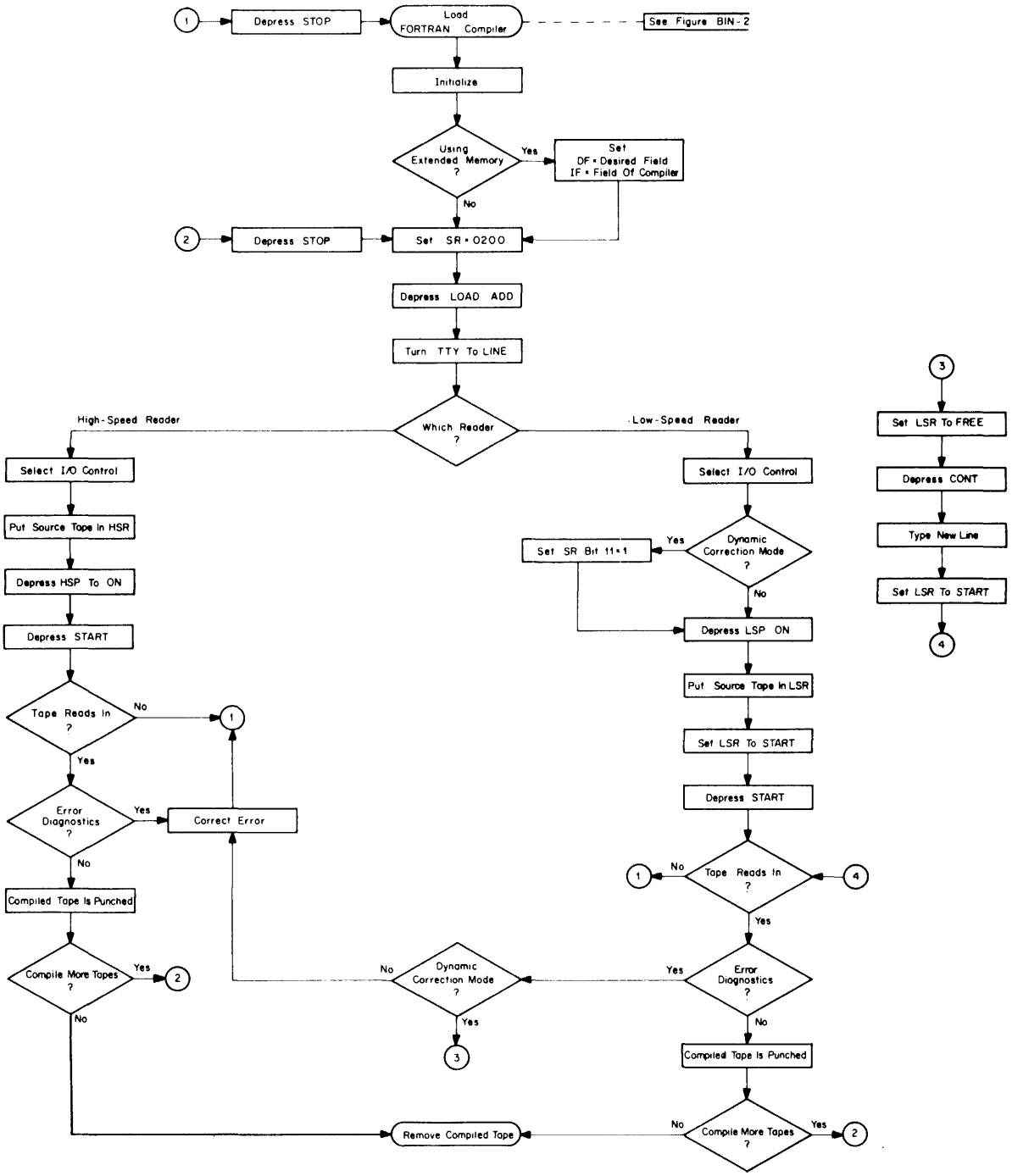


Figure 4K FORTRAN-1 Compiling a FORTRAN Program

SYMBOLPRINT

The Symbolprint program is used to print out a memory map of the compiled source program. The memory map is useful when debugging the program. Symbolprint is run immediately after compiling a source program and before compiling another or loading the Operating System.

NOTE

Symbolprint destroys the Compiler's DECtape I/O processors. Therefore, the Compiler must be reloaded to compile a source program containing DECtape I/O statements.

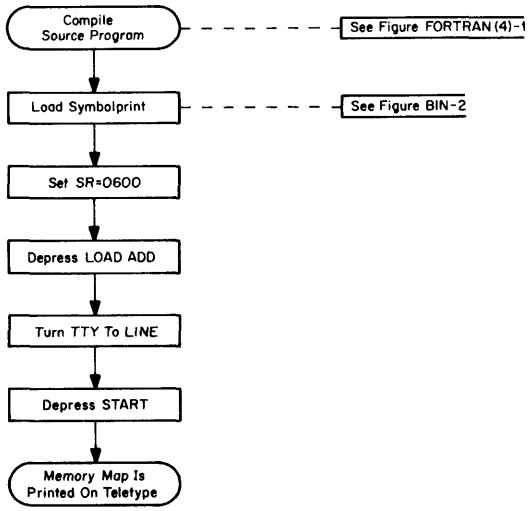


Figure 4K FORTRAN-2 Generating a Memory Map Using Symbolprint

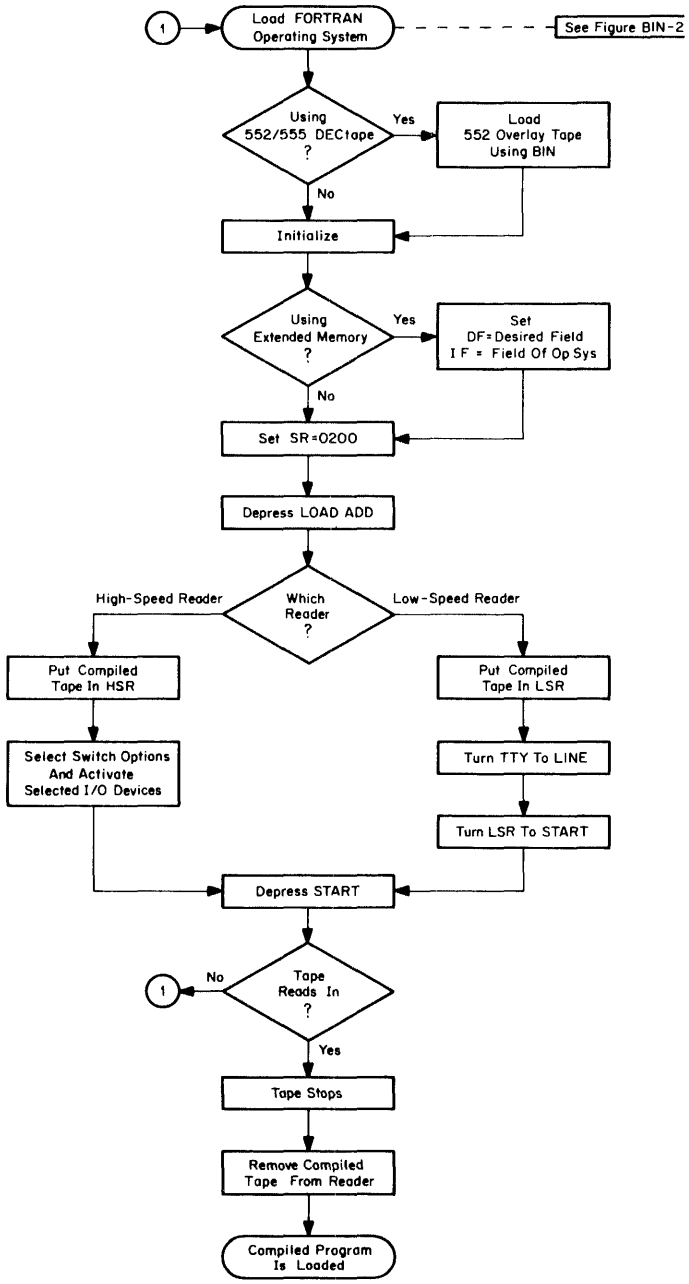
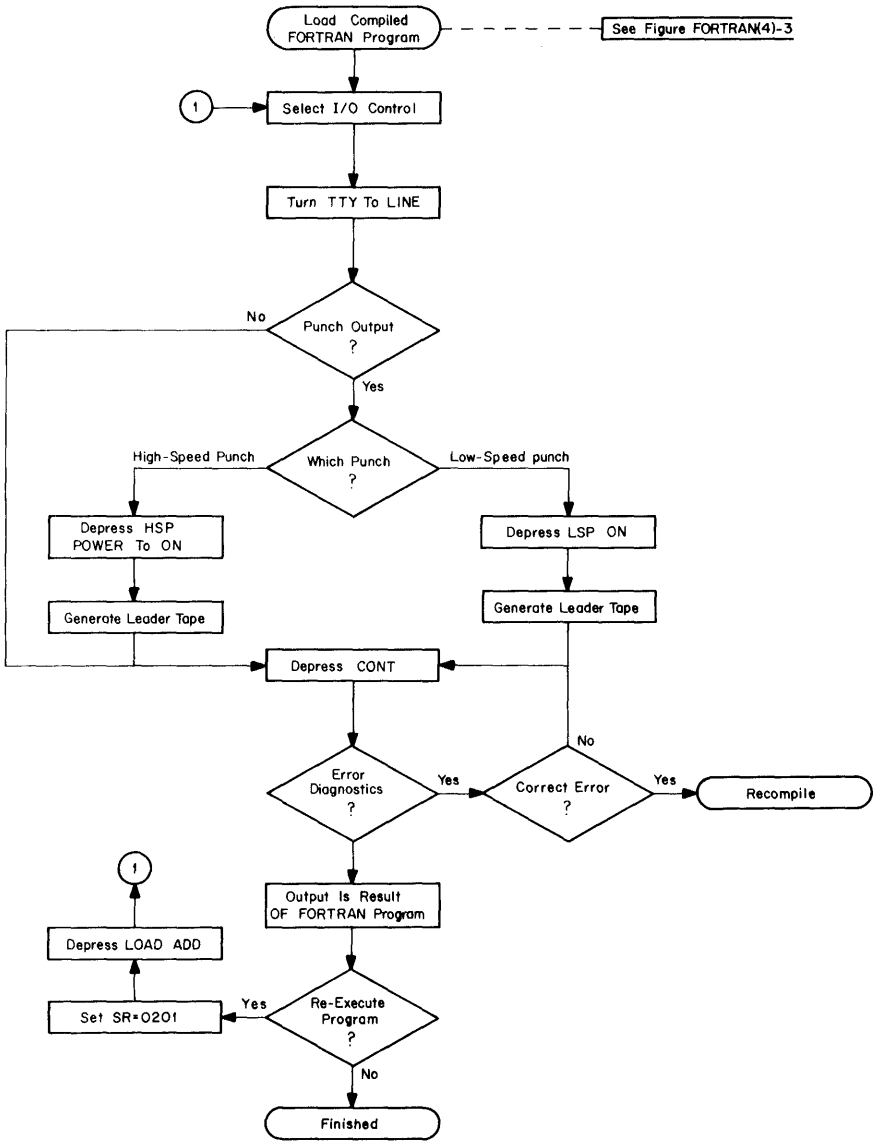


Figure 4K FORTRAN-3 Loading a Compiled FORTRAN Program



NOTE

If the FORTRAN program has been debugged, the internal stack overflow/underflow test can be removed to speed up program execution by setting location 0404 = 7000.

Figure 4K FORTRAN-4 Executing a Compiled FORTRAN Program

INPUT/OUTPUT CONTROL

The selection of I/O devices for both the Compiler and Operating System is controlled by setting the console switches as shown below.

SR Bit	Set To	Results
0	0	The program contains only paper tape I/O statements.
	1	The program contains DECtape I/O statements.
1	0	Compiler: Use low-speed reader for input of source program.
		Operating System: Use low-speed reader for input of object program and the keyboard for ACCEPT statements.
	1	Use the high-speed reader.
2	0	Compiler: Use Teletype and low-speed punch for compiler output (interpretive code) and diagnostics.
		Operating System: Use Teletype and low-speed punch for TYPE statements.
	1	Use the high-speed punch (diagnostics still appear on the printer).

I/O selections cannot be changed without reloading the compiler.

DECTAPE I/O STATEMENTS OPTION

The DECtape I/O statements are the READ and WRITE Statements. The I/O statements option must be set before compiling or running a program.

SR Bit	Set To	Results
0	0	Delete DECtape I/O processing routines.
	1	Use DECtape I/O processing routines.

DYNAMIC ERROR CORRECTION

When compiling in the dynamic correction mode (see Note, below) the user can correct a statement, which the compiler has determined contains a source-language error, by reentering the offending line from the keyboard. To implement the dynamic correction mode set SR bit 11 = 1 as shown in Figure 4K FORTRAN-1. If an error is detected, the diagnostic is typed out in the normal fashion and the computer halts. To correct the statement:

a. Set LSR to FREE

b. Depress CONT

c. Type the new line in its entirety (excluding the statement number if any) followed by a carriage return-line feed.

d. Set LSR to START and compilation will continue.

To leave the dynamic correction mode, restart the compiler at location 0200 with SR bit 11 = 0.

NOTE

This feature applies only to the low-speed paper tape reader.

DIAGNOSTICS

Compiler

Format: xxxx yy zz

where xxxx is statement number of last numbered statement, yy is numbered statement, yy is number of statements since the last numbered statement, and zz is error code (numbers in octal).

Code	Explanation
00	Fixed- and floating-point modes are mixed.
01	Two operators next to each other.
02	Compiler error; reload compiler.
03	Illegal comma in arithmetic statement.
04	Too many operators in a statement.
05	Function argument is in fixed mode.
06	Variable subscript in floating-point mode, or an operator is missing.
07	More than 64_{10} variable names in program.
10	Program too large.
11	Unpaired parentheses.
12	Illegal character.
13	Error in statement format.
14	Program too large, or duplicate statement numbers.
15	Subscripted variable defined prior to DIMENSION statement, or subscripted variable not in DIMENSION statement, or operator missing in fixed-mode expression.
16	Statement too long.
17	Floating-point operand should be fixed-point.
20	Referenced statement number not in program.
21	More than 40_{10} numbered statements in program.
22	Too many incompleted operations in statement.
23	More than 20_{10} statements referenced before being defined.
24	Illegal attempt to compile READ or WRITE statement.

Operating System

Format: "TILT" nn
where nn is the error code.

Code	Possible Cause	Action Taken When CONT is Depressed
11	Attempt to divide by zero.	Quotient set to + or - largest number representable in computer and execution continues.
12	Floating-point exponent on input greater than + or - 2047.	System executes next instruction.
13	Illegal operation code.	System executes next instruction.
14	Transfer to location 0 or 1.	No recovery possible; recompile.
15	Nonformat statement used as FORMAT statement.	Next instruction is executed.
16	Illegal FORMAT statement	Rest of statement is examined.
17	Attempt to fix large floating-point number.	Ignored.
20	Attempt to square root a negative number.	Square root of absolute value is taken.
21	Attempt to raise a negative number to a power.	Absolute value is raised to power specified.
22	Attempt to find logarithm of 0 or negative number.	Attempts to find logarithm of absolute value.
31	Select error: system halts with called unit in bits 0-2 of AC (0-3 with DECtape 552/555).	Recovered by correcting logical unit and depressing CONT.
32	Physical tape error.	System halts with error status in AC.
33	DECtape buffer exceeded.	Ignored.
34	DECtape control switch set incorrectly.	Ignored.
76	System stack overflow.	No recovery possible. Recompile.
77	System stack overflow.	Same as above.

PURPOSE

The 8K FORTRAN System is used to compile, assemble, load, and execute user programs written in the 8K FORTRAN language (a version of FORTRAN II). The system consists of the 8K FORTRAN Compiler which compiles the user's source program into symbolic language, the 8K SABR Assembler which assembles the compiled program into relocatable binary code, and the 8K Linking Loader which loads and executes the relocatable binary program and library of subprograms. (See DEC-08-KFXB-D for details.)

STORAGE REQUIREMENTS

8K FORTRAN Compiler requires locations 0-7577 (field 1).

Starting Address = 1000 (field 1).

8K SABR Assembler requires locations 0-777 (field Ø) and 0600-2200, 2000-2427 (field 1).

Starting Address = 0200 (field Ø)

8K Linking Loader requires locations 0-777 (field Ø); 0-177, 6200-7577 (highest field); 0-177 (all fields).

Starting Address = 0200 (highest field)

8K Library Subprograms require locations 0020-0032 in field where used.

EQUIPMENT REQUIREMENTS

PDP-8/I, -8/L, -8, -8/S, or -5 computer with 8K or more core and high-speed paper tape reader and punch unit. The PDP-5 requires a PDP-8 extended memory control modification.

LOADING

The Binary Loader is used to load the 8K FORTRAN Compiler, 8K SABR Assembler, and 8K Linking Loader into core. The 8K Linking Loader is used to load the relocatable binary program and 8K Library Subprograms into core.

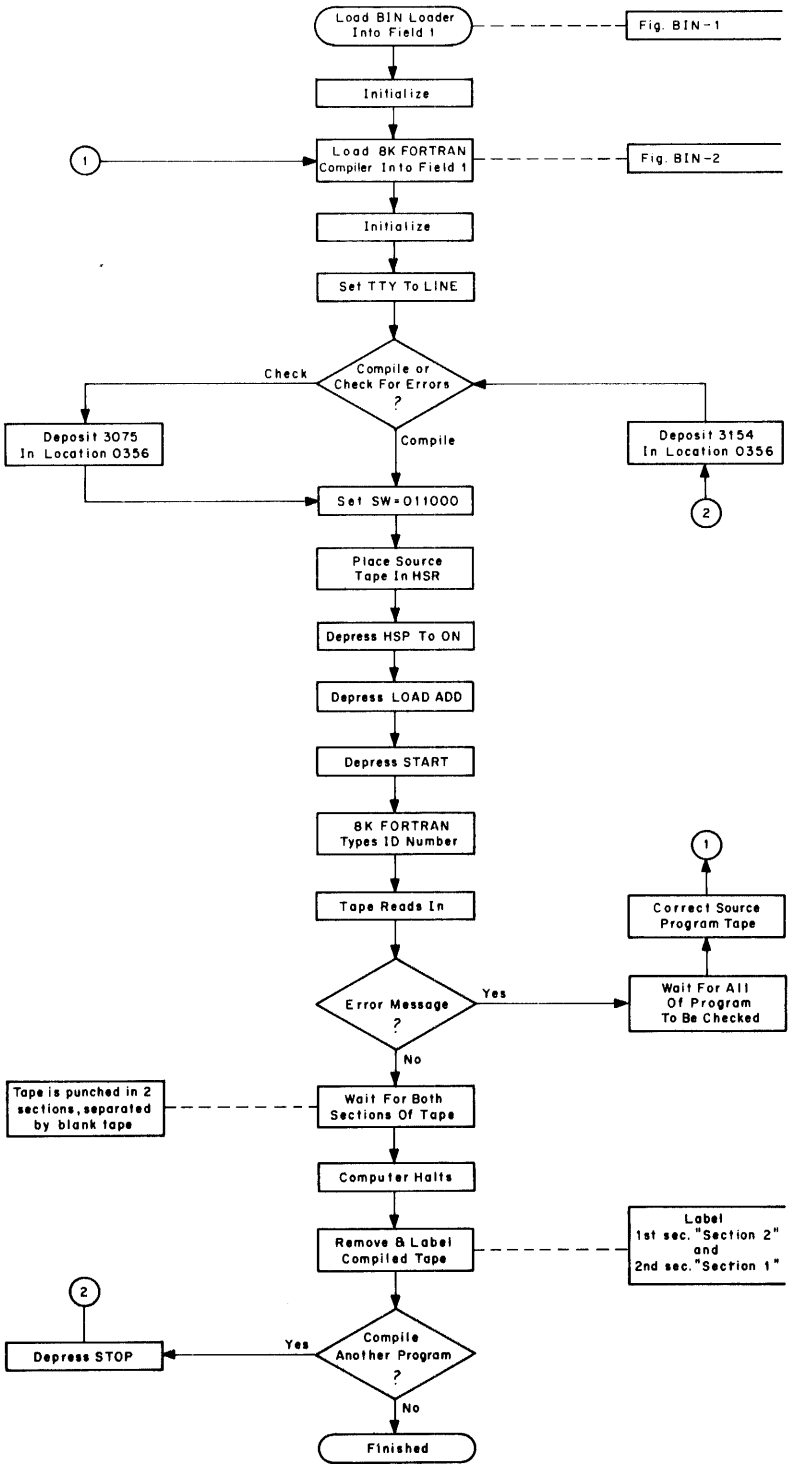


Figure 8K FORTRAN-1 Compiling an 8K FORTRAN Source Program (High-Speed Reader/Punch Only)

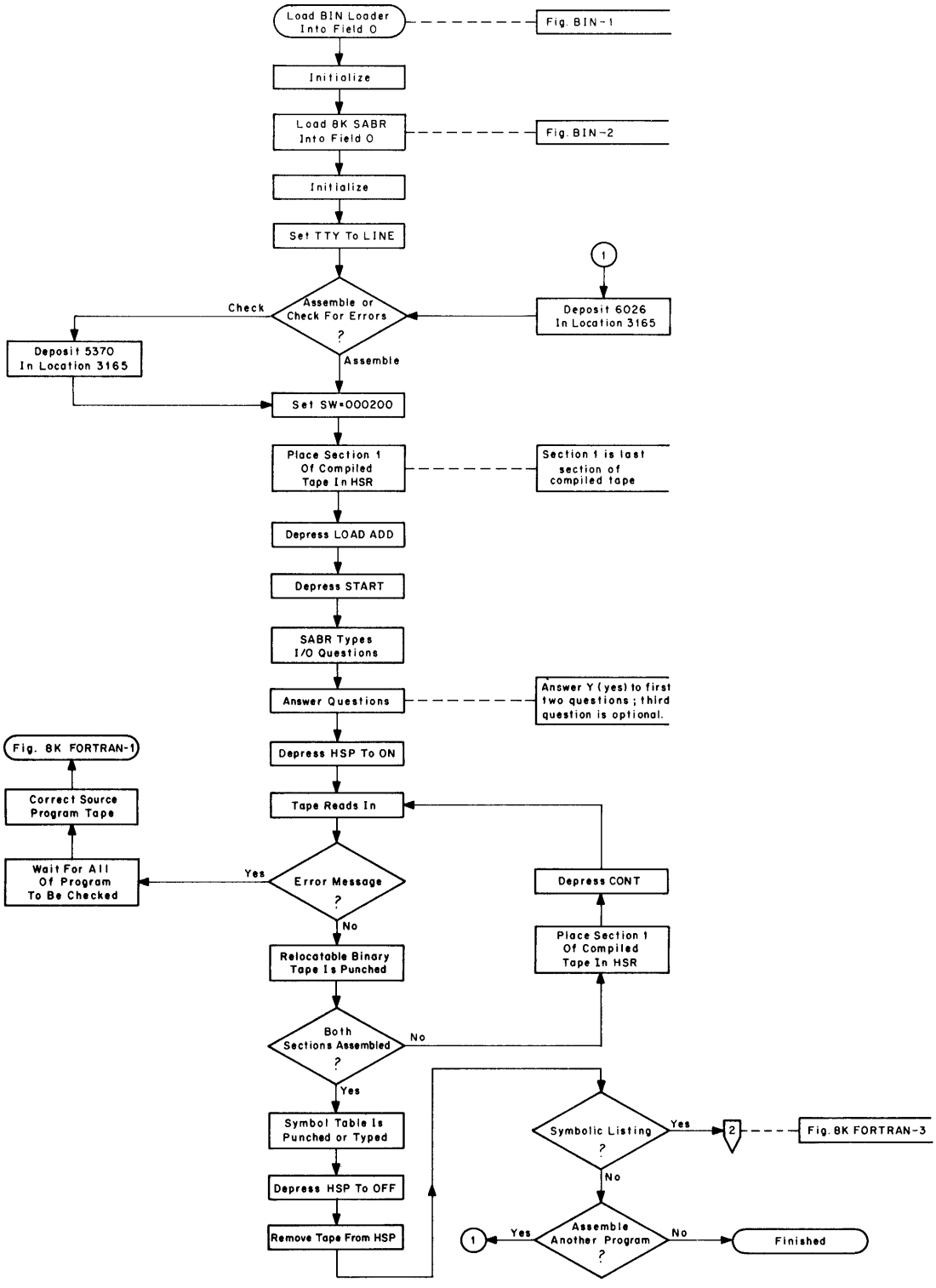


Figure 8K FORTRAN-2 Assembling (Methods 1 & 2) an 8K FORTRAN Compiled Program into a Relocatable Binary Program. (High-Speed Reader/Punch Only)

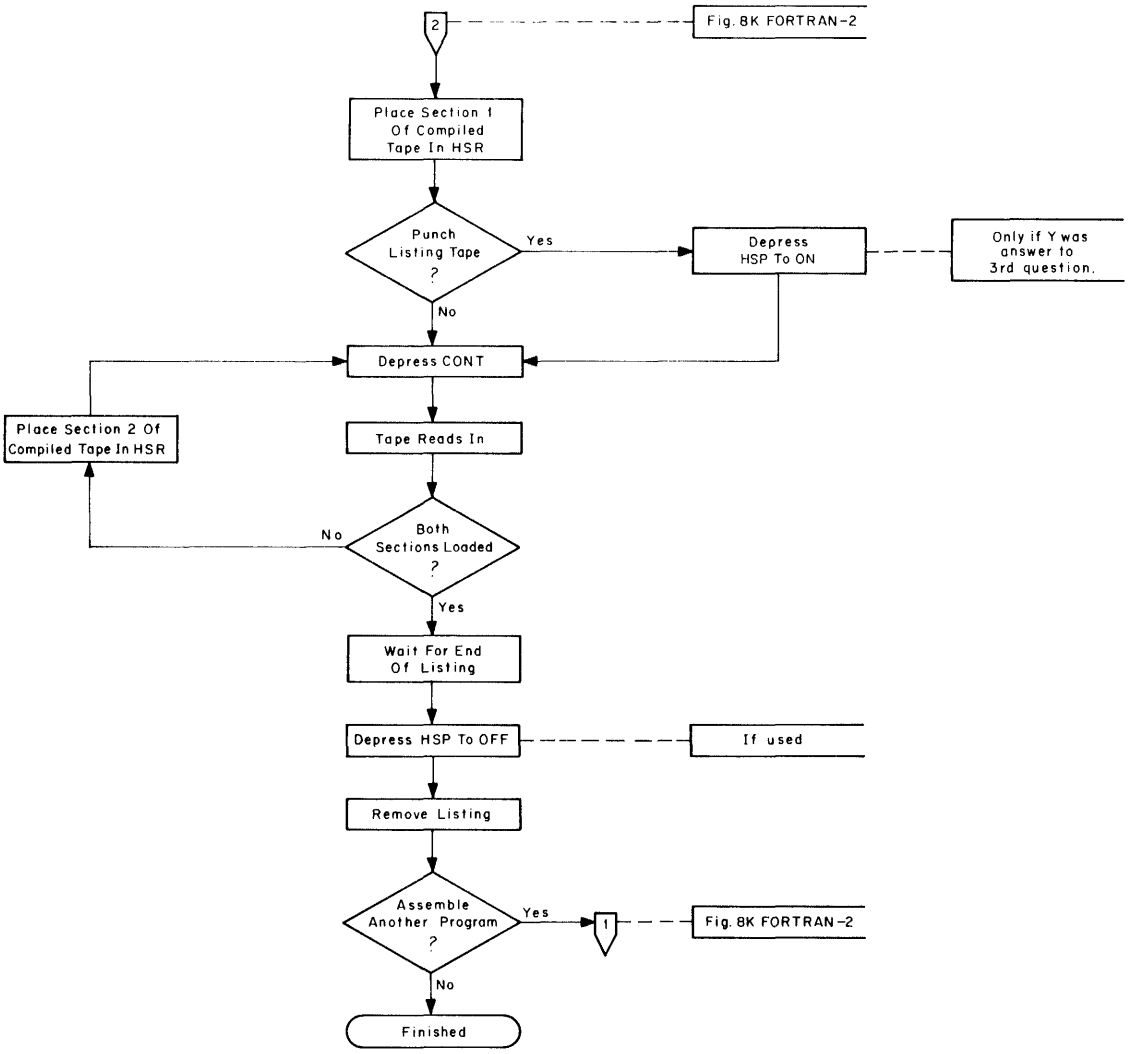


Figure 8K FORTRAN-3 Listing an Assembled 8K FORTRAN Program (High-Speed Reader/Punch Only)

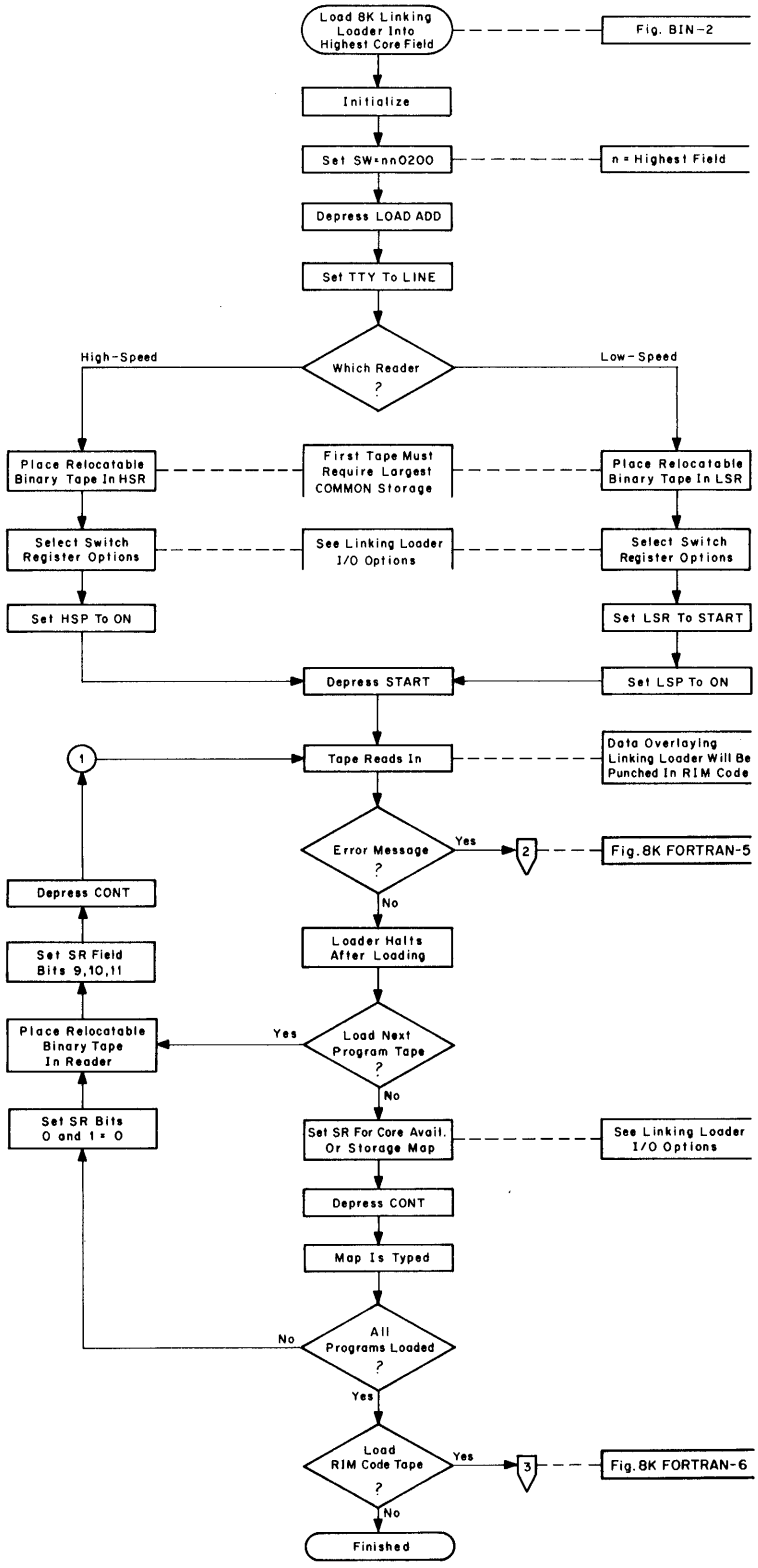


Figure 8K FORTRAN-4 Loading Programs and Subprograms Into Core Using the 8K Linking Loader.

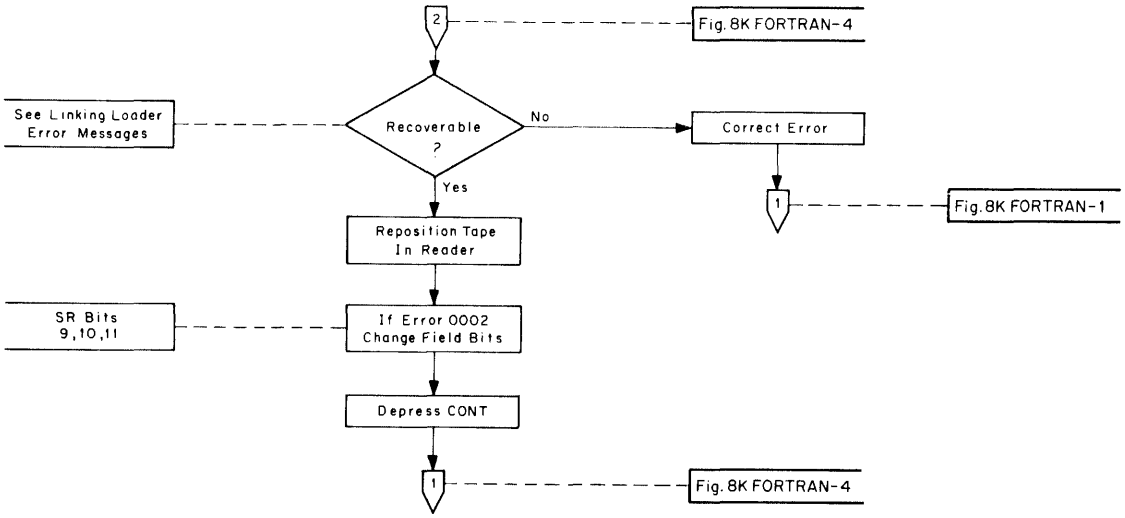


Figure 8K FORTRAN-5 Recovering From Linking Loader Error Message.

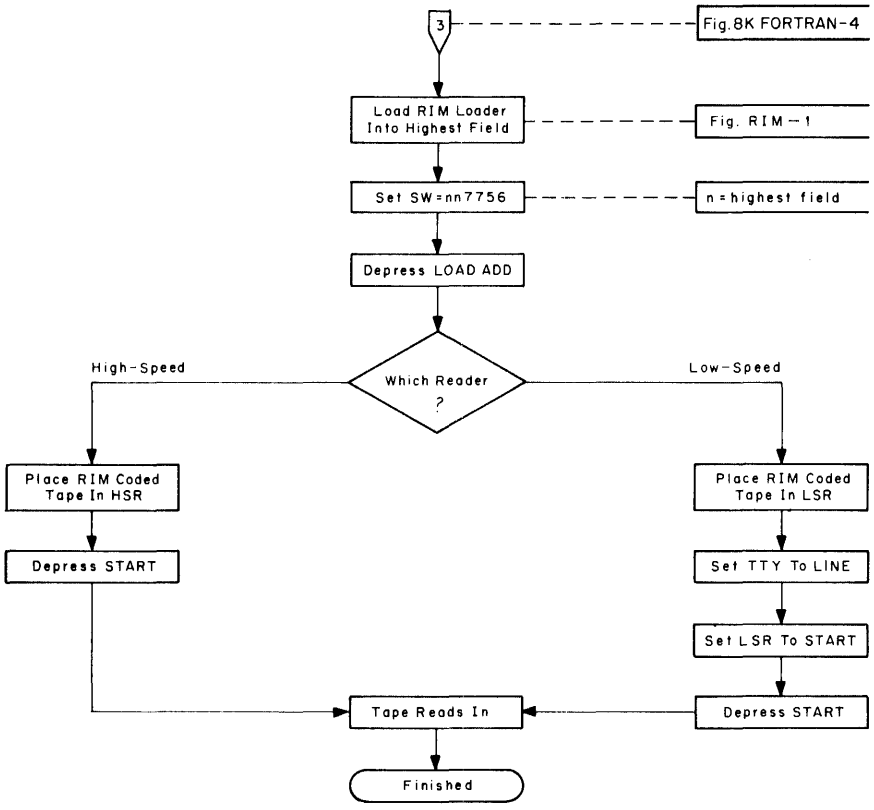


Figure 8K FORTRAN-6 Loading RIM Coded Tape Using RIM Loader.

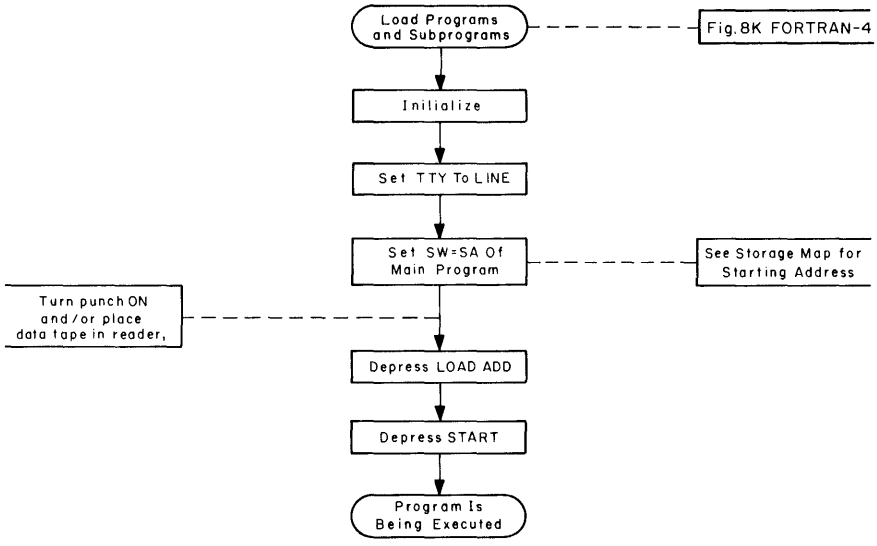


Figure 8K FORTRAN-7 Executing SABR Programs Using the Run-Time Linkage Routines.

SYMBOL TABLE

The symbol table is listed at the end of assembly; each symbol is listed with its relative address. Special symbols are identified as follows.

ABS	The address is absolute.
COM	The address is in COMMON.
OP	The symbol is an operator.
EXT	The symbol is an external, and thus may or may not be defined. If not defined, there is no difficulty, it is defined in another program.
UNDF	The symbol is not an external and has not been defined in the program.

RESERVED LOCATIONS

Locations reserved for special use are below.

Library Linkage Routines require:

0400-0777	in field zero
0007, 0033-0073	in every field
0020-0032	in field where routines reside

I/O Handler Routines require:

0176-0177	in field where routines reside

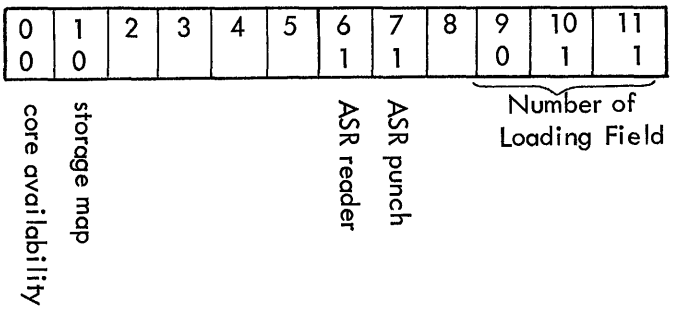
Available to user, in every field:

0000-0006	Interrupts, debugging, etc.
0010-0017	auto-index registers
0020-0022	floating-point AC (use with care)
0023-0032	arbitrary
0074-0177	arbitrary

LINKING LOADER I/O OPTIONS

When loading programs and subprograms using the 8K Linking Loader, the following switch register options are read to perform the following I/O options.

Switch Register*



*All other switch register bits are irrelevant.

Any time the Linking Loader halts, the user may access memory directly via the DEPosit and EXAMine console switches. After this is done the Linking Loader may be restarted via the console switches at location 7200 (in the highest field, where the Linking Loader resides).

8K LIBRARY SUBPROGRAMS

These subprograms are contained on two tapes, organized as shown below. Each subprogram is separated from the other by a noticeable length of blank tape. Load only those subprograms required by the relevant user program.

Tape 1.	IOH	contains	IOH, READ, WRITE
	FLOAT	contains	FAD, FSB, FMP, FDV, STO, FLOT, FLOAT, FIX, IFIX, IFAD, ISTO, CHS, CLEAR
	INTEGER	contains	IREM, ABS, IABS, DIV, MPY, IRDSW
	UTILITY	contains	TTYIN, TTYPUT, HSIN, HSOUT, OPEN, CKIO
	ERROR	contains	SETERR, CLRERR, ERROR
Tape 2.	SUBSC	contains	SUBSC
	POWERS	contains	IIPOW, IFPOW, FIPOW, FFPOW, EXP, ALOG
	SQRT	contains	SQRT
	TRIG	contains	SIN, COS, TAN
	ATAN	contains	ATAN

ERROR MESSAGES

All compilation, assembly, and execution time errors are fatal. Therefore, punched output by the compiler and assembler should be suppressed until the source program is correct. Always examine the assembly symbol table listing for undefined symbols before loading and executing the assembled program.

Do not attempt to load and execute a program which has an assembly error. Unpredictable results will occur. Do not attempt to proceed after an execution time error by depressing CONT. Unpredictable results will occur.

8K FORTRAN Compiler

During compilation, error messages are typed as they occur. Format:

```
/  A=B+M(6) + N(1)
             ↑
MIXED MODE EXPRESSION
```

and the arrow (↑) points to the incorrect statement, indicating that the error is the character directly above the arrow or is somewhere to the left of the arrow. These error messages are self-explanatory:

ILLEGAL CONTINUATION
ILLEGAL STATEMENT NUMBER
ILLEGAL STATEMENT
ILLEGAL CONSTANT
ILLEGAL EQUIVALENCING
SYNTAX ERROR (usually illegal punctuation)
ILLEGAL OR EXCESSIVE DO NESTING
ARITHMETIC EXPRESSION TOO COMPLEX
ILLEGAL ARITHMETIC EXPRESSION
ILLEGAL VARIABLE
MIXED MODE EXPRESSION
EXCESSIVE SUBSCRIPTS
SYMBOL TABLE EXCEEDED

If an error occurs in a series of continuation lines, all remaining lines in that statement will be printed with ILLEGAL CONTINUATION.

8K SABR Assembler

During assembly, error messages are typed as they occur. Format:

```
C  AT \10  +0004
```

where C (illegal character occured AT the 4th instruction after location tab \1Ø, which would correspond to statement 1Ø in the source program.

During listing, the error code is typed in the address field of the instruction line.

Error Code	Explanation
A	Too many or too few ARGs to a CALL statement.
C	Illegal character.
E	No END statement.
I	Illegal syntax: pseudo-op with improper argument, quote mark with no argument, text string not terminated, MRI with improper address, or illegally combined microinstruction.
M	Multiple defined symbol (appears only during assembly)
S	Either: symbol table overflow, common storage is full, or more than 64 different user-defined symbols in a core page.
UNDF	Undefined symbol (appears only in symbol table listings).

All errors are fatal; correct error and recompile.

LINKING LOADER

During loading, the error message is typed as it occurs, and the partially loaded program or subprogram is ignored (removed from core).

Format:

ERROR xxxx

where xxxx is the error code number.

Error Code	Explanation
0001	Symbol table overflow (more than 64 subprogram names).
0002	Current field is full.
0003	Program with largest common storage was not loaded first.
0004	Checksum error in input tape.
0005	Illegal relocation code.

To recover from errors 2, 4, and 5, reposition the tape in the reader at the leader code of the program or subprogram and depress CONTinue. With error 2, load into a different field.

LIBRARY PROGRAM

During program execution, error messages are typed as they occur.

Format:

"xxxx" ERROR AT LOC nnnn

where xxxx is the error code and nnnn is the location of the error.

Error Code	Explanation
"ALOG"	Attempt to compute log of negative numbers.
"ATAN"	Result exceeds capacity of computer.
"DIVZ"	Attempt to divide by zero.
"EXP"	Result exceeds capacity of computer.
"FIPW"	Error in raising a number to a power.
"FMT1"	Multiple decimal points.
"FMT2"	E or . in integer.
"FMT3"	Illegal character in I, E, or F field.
"FMT4"	Multiple minus signs.
"FMT5"	Invalid FORMAT statement.
"FLPW"	Negative number raised to floating power.
"FPNT"	Floating-point error.
"SQRT"	Attempt to square root a negative number.

When an error occurs, execution stops; correct error and reassemble.

To pinpoint the location of the error:

1. From the Storage Map, determine the next lowest numbered location (external symbol) which is the entry point of the program or subprogram in error.

2. Subtract in octal the entry point location from the error location in the error message.

3. From the assembly symbol table, determine the relative address of the external symbol found in step 1 and add that relative address to the result of step 2.

4. The sum of step 3 is the relative address of the error, which can then be compared with the relative address of the numbered statements in the program.

When multiple error messages are typed, the location of the last error message is relevant--the other errors are to subprograms called by the statement at the relevant location.

SECTION 3

DECTAPE

PURPOSE

The TC01 Bootstrap Loader is used to load the DECtape Library System programs into core memory. See DEC-08-LUAA-D and Section 4 of this manual for details.

STORAGE REQUIREMENTS

The TC01 Bootstrap Loader requires locations 7600-7623 (24_8 locations). Starting Address=7600

REQUIRED EQUIPMENT

DECtape Control (TC01) and at least one DECtape Transport (TU55)

LOADING

The TC01 Bootstrap Loader may be toggled into core using the console switches or it may be read into core using the RIM Loader. The locations and corresponding instructions are listed below.

Location	Instruction
7600	6224
7601	6774
7602	1221
7603	4213
7604	1222
7605	3355
7606	1223
7607	4213
7610	0000
7611	0000
7612	0000
7613	0000
7614	6766
7615	3354
7616	6771
7617	5216
7620	5613
7621	0600
7622	7577
7623	0220

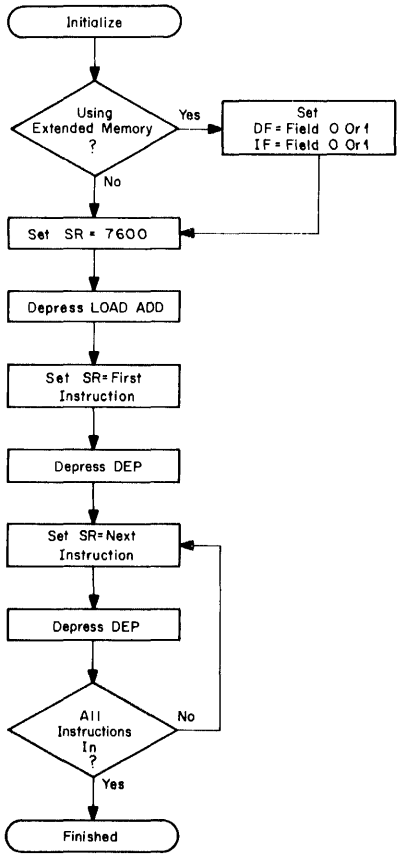


Figure DECTAPE-1 Toggling in TC01 Bootstrap Loader

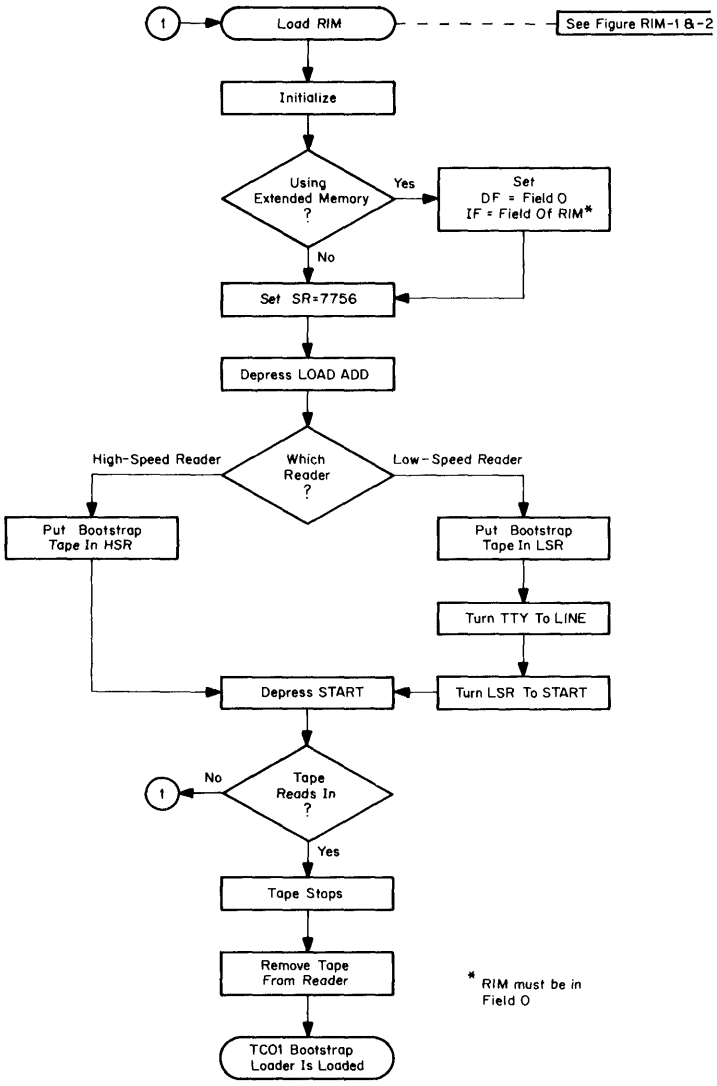


Figure DECTAPE-2 Loading the TC01 Bootstrap Loader Using RIM.

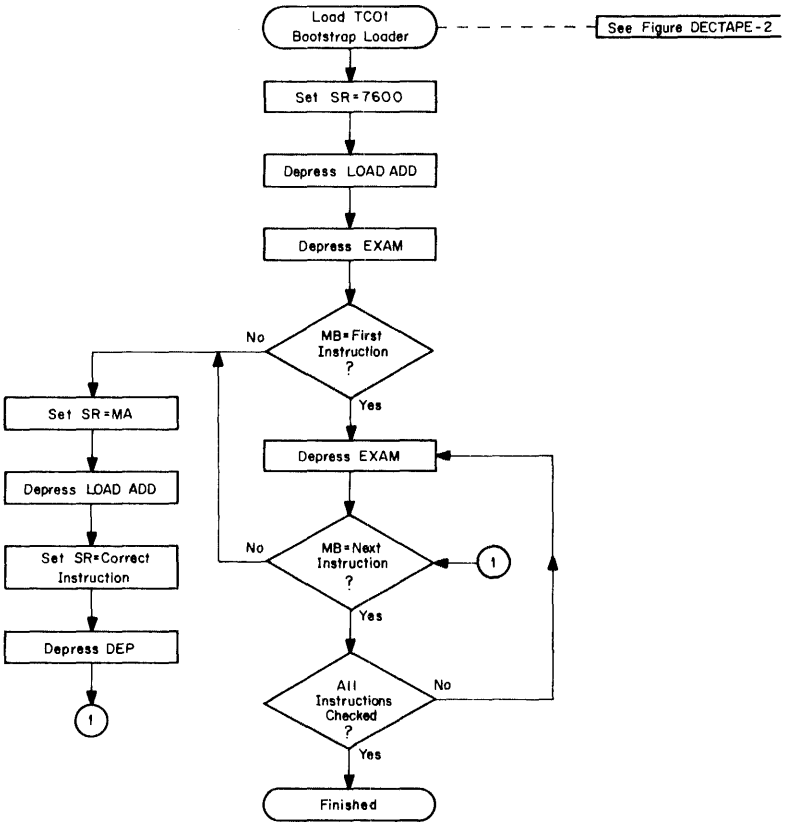


Figure DECTAPE-3 Checking TC01 Bootstrap Loader

PURPOSE

The DECtape Library System is a collection of five programs (INDEX, ESCAPE, UPDATE, DELETE, and GETSYS) stored on DECtape. They are used to load named files into core memory, define new named files, delete named files, and to create a new Library System. See DEC-08-SUB0-D for details.

STORAGE REQUIREMENT

The five library programs will occupy the first 40_8 blocks of a certified DECtape.

EQUIPMENT REQUIREMENT

The DECtape Library System requires a DECtape control (TC01) and at least one DECtape transport (TU55).

LOADING

The TC01 Bootstrap Loader is used to load the DECtape Library System from DECtape into core memory.

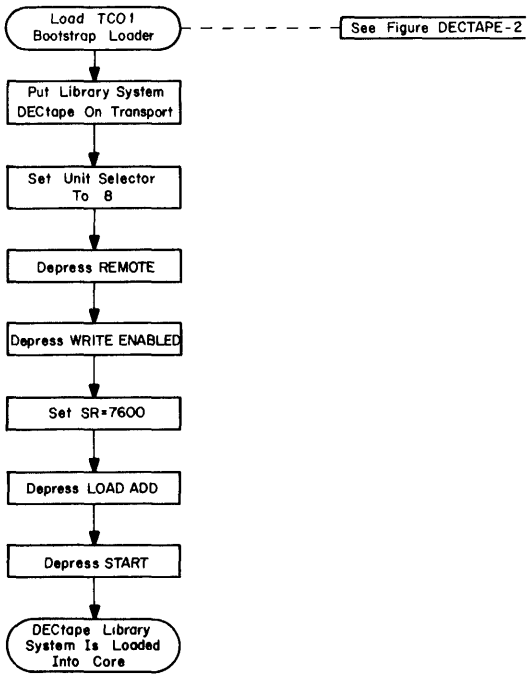


Figure DECTAPE-4 Loading DECtape Library System Using TC01 Bootstrap Loader

LIBRARY SYSTEM

The Library System tape contains the five permanent programs explained below.

INDEX — causes the names of all files on the library tape to be typed.

ESCAPE — causes the Library System to exit core, and restores the RIM, BIN, and DECtape Bootstrap Loaders in core.

UPDATE — allows the user to add files to the library tape. When called, UPDATE types questions to be answered by the user (questions are underlined):

NAME OF PROGRAM: FRTRAN

user types a program name consisting of from one to six characters delimited by a carriage return.

SA (OCTAL) : 0200

user types an octal address delimited by a carriage return.

PAGE LOCATIONS: <0,2200><2400><4600,7577>;

user types the locations required by his program. A typing error causes UPDATE to retype the question.

DELETE — removes specified user program from the library tape. DELETE types a question to be answered by the user:

NAME OF FILE TO BE DELETED: FRTRAN

user types name of program to be deleted.

GETSYS — creates, on a specified tape unit, a new Library System tape consisting of the loaders and the system programs. When called, GETSYS types a question to be answered by the user:

SKELETON TAPE WILL BE CREATED ON UNIT# 5

user types a single digit from 1 to 8 terminated by a carriage return. Unit number 8 should always be the first unit assigned.

SECTION 4

DISK MONITOR SYSTEM

PURPOSE

The Disk System Builder program is an easy-to-use dialogue technique used to build the customized Monitor suited to the particular machine configuration and to store the created Monitor on the system device. The Monitor is then used to create and save the System Program Library on the system device. See DEC-D8-SDAB-D for details.

STORAGE REQUIREMENTS

See DEC-D8-SDAB-D for specific core requirements.

EQUIPMENT REQUIREMENTS

A 4K PDP-8/I computer with 3-cycle data break, an ASR33 Teletype, a high-speed reader/punch, and a DF32 Disk. A TC01 DECtape Control with at least one TU55 DECtape Transport unit may also be used.

LOADING

BIN is used to load the Disk System Builder program into core.

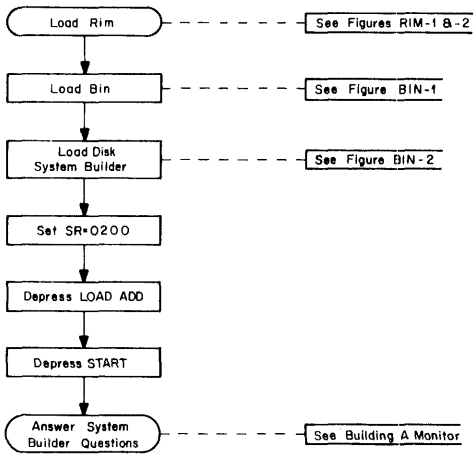


Figure DISK-1 Loading the Disk System Builder
Using the BIN Loader

BUILDING A MONITOR When the System Builder is first loaded into core it will ask the following questions, which the user answers according to his machine configuration. (User response is underlined, and each response is terminated by depressing the RETURN key.)

```
*TYPE SIZE OF CORE (IN K)
*8                                  User enters core size of his computer
*HIGH SPEED PAPER TAPE?
*YES                                User answers YES or NO
*PDP-8/S?
*NO                                 User answers YES or NO
*DISK?
*YES                                User answers YES or NO
*TYPE NUMBER OF DISK UNITS?
*1                                  User types number of disk units in his machine
                                    configuration.
*TAPE?
*YES                                User types YES if he has DECtape, NO if he does
                                    not
.
```

NOTE: If specified as present, the disk is automatically selected as the system device; if not, DECtape unit 8 is selected.

SYSTEM MODES

The system is always running in either Monitor mode or user mode.

Monitor Mode	is entered whenever the Monitor is started or when CTRL/C is typed; Monitor responds with a dot (.) typeout.
User Mode	is entered whenever the system is executing a system or user program; user mode is indicated by an asterisk (*) typeout.

DISK SYSTEM BOOTSTRAP LOADER

PURPOSE

The Disk System Bootstrap loaders are used to load the Disk Monitor into core. The bootstrapping of Monitor into core is necessary only when the resident Monitor area (locations 7600-7777) has been cleared or its contents have been otherwise destroyed. System Builder leaves the resident portion of Monitor in core after building. There are two bootstrap routines, depending upon the type of system device. See DEC-D8-SDAB-D for details.

STORAGE REQUIREMENTS

Disk requires locations 200-204 and 7750-7751 (7 locations). Starting Address=0200.

DECtape requires location 200-222 (23_8 locations). Starting Address=0200.

REQUIRED EQUIPMENT

A 4K PDP-8/I computer with 3-cycle data break; and ASR33 Teletype; a high-speed reader/punch; a DF32 Disk and/or a TC01 DECtape Control with at least one TU55 DECtape Transport.

LOADING

Disk and DECtape Bootstrap loaders are loaded (toggled) into core memory using the console switches. The locations and corresponding instructions for both system devices are listed below.

Disk Bootstrap Loader

Location	Instruction		Symbolic
0200	6603		DMAR
0201	6622		DFSC
0202	5201		JMP .-1
0203	5604		JMP I .+1
0204	7600		7600
7750	7576		
7751	7576		

DECtape Bootstrap Loader

			*200
0200	7600	BEG,	7600
0201	1216		TAD MVB
0202	4210		JMS DO
0203	1217		TAD M201
0204	3620		DCA I CA
0205	1222		TAD RF
0206	4210		JMS DO
0207	5600		JMP I BEG
0210	0000	DO,	0000

Location	Instruction		Symbolic
0211	6766		DTXA DTCA
0212	3621		DCA I WC
0213	6771		DTSF
0214	5213		JMP .-1
0215	5610		JMP I DO
0216	0600	MVB,	0600
0217	7577	M201,	-201
0220	7755	CA,	7755
0221	7754	WC,	7754
0222	0220	RF,	0220

After toggling in one of the above bootstrap routines, set the SR to 0200 and depress LOAD ADDress and START. Monitor should respond with a dot (.) after it has been brought into core.

PURPOSE

The Disk System Program Library is a collection of programs in an open ended Monitor Librarian which allows for easy additions and deletions. The standard package of programs includes an Editor, DDT, Assembler, FORTRAN, Peripheral Interchange Program (PIP), and a Loader. The above programs are device independent and may use the disk for source files, binary output, symbol table space, and overlay. The system is file structured and hardware independent via the System Builder program.

STORAGE REQUIREMENTS

See DEC-D8-SDAB-D for specific storage requirements.

LOADING

The Disk System Programs are loaded using the Monitor. See DEC-D8-SDAB-D for details.

DISK LIBRARY

The library system includes the following programs. See DEC-D8-SDAB-D for complete details.

DDT-D — is the standard DDT-8 but is overlayed to appear to be only two pages long, and it has three breakpoints.

EDITOR-D — is a device independent version of the Symbolic Editor, allowing sources to be edited using disk, high-speed reader, or Teletype in any combination.

PAL-D — is a device independent version of the MACRO-8 Assembler (without macros and floating-point pseudo-ops) which allows storage for over 1000 symbols.

FORTRAN-D — is a device independent version of 4K FORTRAN, and is capable of load-and-go operation.

PIP — is a general utility program; it relieves the programmer of tape copying tasks. It includes directory list and delete functions.

LOADER — is a transparent, device independent version of the BIN Loader.

SAVE/CALL — are Monitor commands which allow the user to save and retrieve, on a page basis, segments of core.

SECTION 5

SYSTEM DEMONSTRATIONS

The demonstration programs convert the decimal numbers 20 through 30 into octal and type the octal numbers on the Teletype printer. These runnable programs demonstrate the ease with which DEC's system, utility, and service programs can be used. Each major step involved in writing, loading, assembling, compiling, and executing the programs is explained; for most operations, the reader is referred to the appropriate flowchart in Section 2.

The machine configuration being used is a 4K PDP-8 computer, an ASR33 Teletype, and a high-speed paper tape reader/punch.

A 4K FORTRAN PROGRAM CALLING A PAL III SUBPROGRAM

This 4K FORTRAN program calls a PAL III subprogram to convert the decimal numbers 20 through 30 into octal and to type the octal numbers on the Teletype printer.

All tape input is through the high-speed reader, and all tape output is from the high-speed punch. The Teletype keyboard is used to issue on-line commands and write the programs using the Editor program. The Teletype printer provides hard copy of all typed input, symbol tables, diagnostics, memory map, program listings, and program results.

To Use the System

Initialize (Section 1, Initializing the System)
Load RIM Loader (using high-speed reader version) (Figure RIM-1)
Load BIN Loader (Figure BIN-1)

To Generate Source Program Tapes

Load Editor (Figure BIN-2)
Start Editor at location 0200
Command Editor to append incoming text to text buffer; type source program.

```
A
C       * TYPE THE DECIMAL NUMBERS FROM 20 TO 30 IN OCTAL *
C
        DIMENSION TOCT(9),NUM(1)
        DO 50 I=20,30
        NUM=I
        PAUSE 3940
        TYPE 60
50;     CONTINUE
        STOP
60;     FORMAT(/)
        END
```

Reserve 1 location for number.

Reserve 27 locations (3 x 9) for the subprogram TOCT

Pause number = $3967_{10}-27_{10}=3940_{10}$ ($3967_{10}=7577_8$)

```
                                    Output is on the high-speed punch
                                      (Figure EDIT-2)
T                                   Generate leader tape

P                                   Command Editor to punch entire text buffer;
                                      depress CONT
F                                   Insert FORM FEED character onto tape

T                                   Generate trailer tape

                                    Remove punched tape from HSP and write some
                                      identification on its leader tape, e.g.,
                                      FORTRAN Source, 6/19/68.

                                    Restart Editor at location 0176 to clear
                                      text buffer

                                    Command Editor to append incoming text
                                      to text buffer; type source program

A
/ASSEMBLY PROGRAM TO TYPE AN OCTAL NUMBER
        *7543                       Compute this address by subtracting the number
NUM,    0                             of words reserved in the DIMENSION state-
TOCT,   0                             ment above from 7577(8).
        TAD NUM
        CLL RAL
        DCA TEM
        TAD M4
        DCA INDX                    27(10) + 1 = 28(10) = 34(8)
LOOP,   TAD TEM
        RTL                         7577(8) - 34(8) = 7543(8)
        RAL
        DCA TEM
        TAD TEM
        AND C7
        TAD C260
        TSF
        JMP .-1
        TLS
        CLA
        ISZ INDX
        JMP LOOP
        JMP I TOCT
/
TEM,    0
INDX,   0
M4,     -4
C7,     7
C260,   260
$
                                    Type CTRL/FORM to return to command mode
                                    Output is on the high-speed punch (Figure EDIT-2)
T                                   Generate leader tape
P                                   Command Editor to punch entire text buffer;
                                      depress CONT
F                                   Insert FORM FEED character onto tape
T                                   Generate trailer tape
                                    Remove punched tape from HSP and write identifica-
                                      tion on leader tape, e.g., PAL III Source, 6/19/68.
```

To Generate PAL III Object Program Tape

Load PAL III (Figure BIN-2)

Perform Pass 1 of assembly (Figure PAL-2)

Note error diagnostics, if any.

```
C260    7574
C7      7573
INDX    7571
LOOP    7552
M4      7572
NUM     7543
TEM     7570
TOCT    7544
```

Symbol table is typed on printer.

Perform Pass 2 of assembly (Figure PAL-2)

PAL III object tape is punched on the high-speed punch

Perform Pass 3 of assembly to get the program listing. (Figure PAL-2)

```
                /ASSEMBLY PROGRAM TO TYPE AN OCTAL NUMBER
                        *7543
7543    0000    NUM,    0
7544    0000    TOCT,   0
7545    1343            TAD NUM
7546    7104            CLL RAL
7547    3370            DCA TEM
7550    1372            TAD M4
7551    3371            DCA INDX
7552    1370    LOOP,   TAD TEM
7553    7006            RTL
7554    7004            RAL
7555    3370            DCA TEM
7556    1370            TAD TEM
7557    0373            AND C7
7560    1374            TAD C260
7561    6041            TSF
7562    5361            JMP .-1
7563    6046            TLS
7564    7200            CLA
7565    2371            ISZ INDX
7566    5352            JMP LOOP
7567    5744            JMP I TOCT
                /
7570    0000    TEM,    0
7571    0000    INDX,   0
7572    7774    M4,     -4
7573    0007    C7,     7
7574    0260    C260,   260

C260    7574
C7      7573
INDX    7571
LOOP    7552
M4      7572
NUM     7543
TEM     7570
TOCT    7544
```

The program listing is typed on the printer.

The symbol table concludes the program listing.

SYSTEM

To Compile the FORTRAN Object Tape

Load 4K FORTRAN Compiler (Figure BIN-2)

Compile the 4K FORTRAN object program using the high-speed reader/punch (Figure 4K FORTRAN-1)

Load Symbolprint (Figure BIN-2)

```
I      7576
MUN    7575
  5206  7575
```

Execute Symbolprint to get memory map (Figure 4K FORTRAN-2)

Lower and upper limits of the program

To Execute the Program

Load PAL III subprogram (Figure BIN-2)

Load 4K FORTRAN Operating System (Figure BIN-2)

Load compiled 4K FORTRAN program (Figure 4K FORTRAN-3)

```
0024
0025
0026
0027
0030
0031
0032
0033
0034
0035
0036
!
```

Execute the stored 4K FORTRAN and PAL III programs (Figure 4K FORTRAN-4)

The program results are typed on the printer.

The ! (exclamation point) indicates that the Operating System has come to the END of the 4K FORTRAN program.

A FOCAL PROGRAM CALLING A PAL III SUBPROGRAM

This is a demonstration of a FOCAL program calling an assembled PAL III subprogram to covvert the decimal numbers 20 through 30 into octal and to type the octal numbers on the Teletype printer. A thorough knowledge of PAL III and FOCAL are required for this program.

All tape input is through the low-speed reader, and all tape output is from the low-speed punch. The Teletype keyboard is used to write the PAL III subprogram, to issue on-line commands to the Editor program, and to communicate with FOCAL. As in the previous demonstration program, the Teletype printer provides hard copy of all input and output.

To Use the System

Initialize (Section 1, Initializing the System)

Load RIM Loader (using low-speed reader version) (Figure RIM-1)

Load BIN Loader (Figure BIN-1)

To Generate PAL III Source Tape

Load Editor (Figure BIN-2)

Start Editor at location 0200

Command Editor to append incoming text to text buffer.

Type the source program.

```
          *410                        /FOR FOCAL, 1969 ONLY
          XFNEW
          *4550
XFNEW,              JMS I INTEGER     /GET INTEGER PART OF FLOATING AC
          TAD FLAC +1                 /AND BRING INTO AC
          CLL RAL
          DCA TEM
          TAD M4                      /INITIALIZE COUNTER
          DCA INDX
LOOP,     TAD TEM
          RTL
          RAL
          DCA TEM
          TAD TEM
          AND C7
          TAD C260
          JMS I OUTDEV                /CALL FOCAL'S TYPE ROUTINE
          ISZ INDX
          JMP LOOP
          JMP I EFUN 3I               /RETURN TO MAIN PROGRAM
/
TEM,      0
C7,       7
INDX,     0
M4,       -4
INTEGER=53
OUTDEV=63
EFUN3 I=136
FLAC=44
C260=113
          *35
BOTTOM,   4550-1                      /SHORTEN TEXT BUFFER FOR FNEW
$
```

SYSTEM

```
                                        Line 4 is listed
4L
XFNEW,          JMS  I  INTEGER         /GET INTEGER PART OF FLOATING AC
                                        The extra tab character is removed
4C
XFNEW,          JMS  I  INTEGER         /GET INTEGER PART OF FLOATING AC
                                        Type CTRL/FORM to return to Command
                                          Mode
                                        Punch source tape on low-speed punch
                                          (Figure EDIT-2)
T                                       Generate leader tape
P                                       Command Editor to punch entire text buffer;

          *41Ø                          /FOR FOCAL, 1969 ONLY
          XFNEW
          *455Ø
XFNEW,    JMS  I  INTEGER               /GET INTEGER PART OF FLOATING AC
          TAD  FLAC +1                  /AND BRING INTO AC
          CLL  RAL
          DCA  TEM
          TAD  M4                        /INITIALIZE COUNTER
          DCA  INDX
LOOP,     TAD  TEM
          RTL
          RAL
          DCA  TEM
          TAD  TEM
          AND  C7
          TAD  C26Ø
          JMS  I  OUTDEV                /CALL FOCAL'S TYPE ROUTINE
          ISZ  INDX
          JMP  LOOP
          JMP  I  EFUN 3I               /RETURN TO MAIN PROGRAM
/
TEM,      Ø
C7,       7
INDX,     Ø
M4,       -4
INTEGER=53
OUTDEV=63
EFUN3I=136
FLAC=44
C26Ø=113
          *35
BOTTOM,   455Ø-1                        /SHORTEN TEXT BUFFER FOR FNEW
$
T                                       Generate trailer tape
                                        Remove punched tape from LSP and write some
                                          identification on its leader tape, e.g., PAL III
                                          Source Tape, 6/19/68.
```

To Assemble and Generate PAL III Object Tape

Load PAL III Assembler (Figure BIN-1)

Perform Pass 1 of assembly (Figure PAL-1)

Note error diagnostics, if any.

```
BOTTOM    0035
C260      0113
C7        4572
EFUN3I    0136
FLAC      0044
INDX      4573
INTEGE    0053
LOOP      4556
M4        4574
OUTDEV    0063
TEM       4571
XFNEW     4550
```

Symbol Table is typed on printer

Perform Pass 2 of assembly (Figure PAL-1)

```
BD%(E($+%99<;98899:$3;+.-?<@%'
```

- Disregard meaningless characters typed

Note error diagnostics, if any.

Perform Pass 3 of assembly (Figure PAL-1)

```
0410  4550                *410              /FOR FOCAL, 1969 ONLY
                          XFNEW
                          *4550
4550  4453  XFNEW,        JMS I INTEGER     /GET INTEGER PART OF FLOATING AC
4551  1045                TAD FLAC +1       /AND BRING INTO AC
4552  7104                CLL RAL
4553  3371                DCA TEM
4554  1374                TAD M4            /INITIALIZE COUNTER
4555  3373                DCA INDX
4556  1371  LOOP,         TAD TEM
4557  7006                RTL
4560  7004                RAL
4561  3371                DCA TEM
4562  1371                TAD TEM
4563  0372                AND C7
4564  1113                TAD C260
4565  4463                JMS I OUTDEV      /CALL FOCAL'S TYPE ROUTINE
4566  2373                ISZ INDX
4567  5356                JMP LOOP
4570  5536                JMP I EFUN3I      /RETURN TO MAIN PROGRAM
                  /
4571  0000  TEM,           0
4572  0007  C7,            7
4573  0000  INDX,          0
4574  7774  M4,            -4
            INTEGER=53
            OUTDEV=63
            EFUN3I=136
            FLAC=44
            C260=113
                          *35
0035  4547  BOTTOM,       4550-1            /SHORTEN TEXT BUFFER FOR FNEW
```

```
BOTTOM     0035
C260       0113
C7         4572
EFUN3I     0136
FLAC       0044
INDX       4573
INTEGE     0053
LOOP       4556
M4         4574
OUTDEV     0063
TEM        4571
XFNEW      4550
```

Symbol Table concludes the program listing

To Execute the Programs

Load FOCAL (Figure BIN-2)

Start FOCAL at location 0200 and reply to Initial Dialogue

```
CONGRATULATIONS!!
YOU HAVE SUCCESSFULLY LOADED 'FOCAL, 1969' ON A PDP-8 COMPUTER.

SHALL I RETAIN LOG, EXP, ATN?:  YES

PROCEED.
```

Load PAL III Object Program tape (Figure BIN-2)

```
*
```

Restart FOCAL at location 0200 and type the FOCAL program (?00.00 denotes a manual restart)

```
 ?00.00
*
*1.10 C TYPE THE DECIMAL NUMBERS 20 THRU 30 IN OCTAL
*1.20 C
*1.30 FOR I=20, 30; DO 2.0
*1.40 QUIT
*
*2.10 SET A=FNEW (I)
*2.20 TYPE !
*
*GO
0024
0025
0026
0027
0030
0031
0032
0033
0034
0035
0036
*
```

Execute the FOCAL program and it will call the PAL III subprogram to type the results on the printer

A FOCAL PROGRAM

This program demonstrates the ease and convenience of FOCAL. In three lines of programming, FOCAL alone converts the decimal numbers 20 through 30 into octal and types them on the Teletype printer. The other line, line 1.1, is used to title the columns of typed numbers.

FOCAL is loaded into core memory using the high-speed reader. Communication with FOCAL is through the Teletype keyboard. The Teletype printer, as before, provides hard copy of all typed input and output.

To Use the System

Initialize (Section 1, Initializing the System)

Load HELP Loader (Figures HELP-1 and 2)

Load FOCAL (Figure BIN-2)

To Activate FOCAL for Use

Start FOCAL at location 0200 and reply to Initial Dialogue

```
CONGRATULATIONS!!
YOU HAVE SUCCESSFULLY LOADED 'FOCAL, 1969' ON A PDP-8 COMPUTER.

SHALL I RETAIN LOG, EXP, ATN?:  YES

PROCEED.
```

Using FOCAL

Type the FOCAL program

```
*
*01.08 TYPE "NUMBER,DECIMAL          NUMBER,OCTAL",!
*01.10 FOR N=20,30; DO 2 ; TYPE %3,N,"                  ",P,!
*
*02.10 S P=0; S M=N
*02.20 F J=0,4; S A=8↑(4-J); S D=FITR(M/A;S M=M-D*A;S P=P+D*10↑(4-J)
*
*GO
```

Execute the FOCAL program

```
NUMBER,DECIMAL          NUMBER,OCTAL
=+ 20                   =+ 24
=+ 21                   =+ 25
=+ 22                   =+ 26
=+ 23                   =+ 27
=+ 24                   =+ 30
=+ 25                   =+ 31
=+ 26                   =+ 32
=+ 27                   =+ 33
=+ 28                   =+ 34
=+ 29                   =+ 35
=+ 30                   =+ 36
*
```

APPENDICES

APPENDIX A

USASCII CHARACTER SET*

Character	8-Bit Octal	6-Bit Octal	Character	8-Bit Octal	6-Bit Octal
A	301	01	!	241	41
B	302	02	"	242	42
C	303	03	#	243	43
D	304	04	$	244	44
E	305	05	%	245	45
F	306	06	&	246	46
G	307	07	'	247	47
H	310	10	(	250	50
I	311	11	)	251	51
J	312	12	*	252	52
K	313	13	+	253	53
L	314	14	,	254	54
M	315	15	-	255	55
N	316	16	.	256	56
O	317	17	/	257	57
P	320	20	:	272	72
Q	321	21	;	273	73
R	322	22	<	274	74
S	323	23	=	275	75
T	324	24	>	276	76
U	325	25	?	277	77
V	326	26	@	300	
W	327	27	[	333	33
X	330	30	\	334	34
Y	331	31	]	335	35
Z	332	32	↑	336	36
0	260	60	←	337	37
1	261	61	Leader/Trailer	200	
2	262	62	Line Feed	212	
3	263	63	RETURN	215	
4	264	64	Space	240	40
5	265	65	RUBOUT	377	
6	266	66	Blank	000	
7	267	67	ALT MODE (ESC)	375	
8	270	70	CTRL	2nn**	
9	271	71			

* An abbreviation for USA Standard Code for Information Interchange.

**The CTRL (Control) key is used in conjunction with certain alphabetic characters. When used, the 8-bit octal representation is 100 less than the normal octal representation for the alphabetic character, e.g., CTRL/L=214, where L is normally 314.

APPENDIX B

GLOSSARY OF TERMS

The following list of computer/programming terms is by no means complete. However, it does include many of the terms used in data processing.

Words underlined are defined elsewhere in this glossary.

Absolute Address	(1) An address that is permanently assigned by the machine designer to a storage location. (2) A pattern of characters that identifies a unique storage location without further modification.
Accumulator	A register in which the result of an operation is formed; Abbreviation: AC
Acronym	A word formed from the first letter or letters of the successive words of a multiple word term.
Accuracy	The degree of freedom from error, i.e., the degree of conformity to truth or to a rule.
Address	A label, name, or number which designates a register or a location where information is stored. That part of an instruction which specifies the location of an operand.
Address Register	A register in which an address is stored.
Algorithm	A prescribed set of well-defined rules or processes for the solution of a problem in a finite number of steps.
Alphabet	An ordered set of unique representations called characters, e.g., the 26 letters of the Roman alphabet.
Alphanumeric	Pertaining to a character set that contains both letters and numerals, and usually other characters.
Arithmetic Unit	The component of a computer where arithmetic and logical operations are performed.
ASCII	An abbreviation for USA Standard Code for Information Interchange.
Assemble	To translate from a symbolic (source) program to a machine language (object) program by substituting binary operation codes for symbolic operation codes and absolute or relocatable addresses for symbolic addresses.
Assembler	A program that assembles.
Auto-Indexing	When an absolute location 0010 through 0017 is addressed indirectly, the content of that location is incremented by one, rewritten in that same location, and then read as the effective address of the next instruction.
Auxiliary Operation	An operation performed by equipment not under direct control of the computer. Off-line operation.
Auxiliary Storage	Storage that supplements the primary storage.

Binary	(1) Pertaining to a characteristic or property involving a selection, choice, or condition in which there are two possibilities. (2) Pertaining to the numeration system with a radix of two.
Binary Digit	One of the symbols 1 or 0. A digit in the binary scale of notation; called a bit.
Bit	A binary digit.
Blank Character	A character used to produce a space on an output device.
Block	A set of things, such as words, characters, or digits, handled as a unit.
Bootstrap	A technique or device designed to bring itself into a desired state by means of its own action, e.g., a routine whose first few instructions are sufficient to bring the rest of itself into the computer from an input device.
Branch	A point in a routine where one of two or more choices is made under control of the routine, i.e., a conditional transfer (jump).
Buffer Storage	A part of core memory where information is stored temporarily during transfer; it may attempt to match the speeds of internal computation and the I/O device, thus permitting simultaneous computation and input/output.
Byte	A group of binary digits usually operated upon as a unit, e.g., 8-bit or 6-bit byte.
Call	To transfer control to a specified routine.
Calling Sequence	A specified set of instructions and data necessary to set up and call a given routine.
Carriage Return	The Teletype operation that causes the next character to be printed at the left margin.
Central Processing Unit	The unit of a computing system that includes the circuits controlling the interpretation and execution of instructions; the computer proper, excluding I/O and other peripheral devices.
Character	A single letter, numeral, or space mark used to represent information.
Clear	To erase the contents of a storage location by replacing the contents with blanks or zeros.
Closed Subroutine	A subroutine not stored in the main part of a program. Such a subroutine is entered by a jump operation and provision is made to return control to the main routine at the end of the subroutine.
Coding	To write instructions for a computer using symbols meaningful to the computer.
Command	A control signal, usually written as a character or group of characters, to direct action by a system program.
Compile	To produce a machine language routine from a routine written in source language by selecting appropriate subroutines from a subroutine library, as directed by the instructions or other symbols of the original routine, supplying the linkage which combines the subroutines into a workable routine and translating the subroutines and linkage into machine language.
Compiler	A program that compiles.
Complement	To form the negative of a binary word by replacing all 0 bits with 1 bits and vice versa.

Computer	A device capable of accepting information, processing it, and providing the results in a usable form.
Computer Program	A plan or routine for solving a problem on a computer.
Computer Word	A sequence of 12 bits treated as a unit and capable of being stored in one computer location.
Console	Usually the external front side of a device where controls and indicators are available for manual operation of the device.
Control Character	A character whose occurrence in a particular context initiates, modifies, or stops a control operation, e.g., a character to control carriage return.
Control Panel	The part of a device console that contains manual controls.
Convert	To change the representation of data from one form to another.
Copy	To reproduce data, leaving the original data unchanged.
Core Memory	The main storage device in the PDP-8 in which binary data is represented by the direction of magnetization in each unit of an array of magnetic material.
Cycle	To repeat a set of operations until a stated condition is met.
Cycle Time	An interval of time in which one set of events is completed.
Data	A general term used to denote any or all facts, numbers, letters, and symbols. It connotes basic elements of information which can be processed or produced by a computer.
Data Break	A facility which permits I/O transfers to occur simultaneously with program execution on a cycle-stealing basis.
Debug	To detect, locate, and correct mistakes in a program.
Decision	A determination of future action.
Delay	The amount of time by which an event is retarded.
Delimiter	A character that separates and organizes items of data.
Diagnostic	Pertaining to the detection and isolation of a malfunction or mistake.
Digit	A character used to represent one of the non-negative integers smaller than the radix, e.g., in binary notation, either 0 or 1.
Digital Computer	A device that operates on discrete data, performing sequences of arithmetic and logical operations on this data.
Direct Address	An address that specifies the location of an operand.
Display	A visual presentation of data.
Document	A medium on which information is recorded for human or machine use.
Double Precision	Pertaining to the use of two computer words to represent a number.
Downtime	The time interval during which a device is inoperative.
Dummy	An artificial address, instruction, or record of information inserted solely to fulfill prescribed conditions.
Dump	To copy the contents of all or part of core memory, usually onto an external storage medium.

Dynamic Dump	A dump that is performed during the execution of a program.
Edit	To rearrange information for machine input or output.
Effective Address	The address actually used in a particular execution of a computer instruction.
End-Around Carry	The action of adding the most significant bit of a binary number to the least significant bit.
Execute	To carry out an instruction or run a program on the computer.
Executive Routine	A routine that controls or monitors the execution of other routines.
External Storage	A facility or device, not an integral part of the computer, on which data usable by the computer is stored, such as paper tape, DECtape, or DECdisk.
File	A collection of related records treated as a unit.
Fixed Point	In a numeration system the position of the radix point is fixed with respect to one end of the numerals, according to some convention.
Flip-Flop	A basic computer circuit or device capable of assuming either one of two stable states at a given time.
Floating Point	A numeration system in which the position of the radix point is indicated by one part (the exponent part), the other part represents the significant digits (the fractional part).
Flowchart	A graphical representation of the sequence of instructions required to carry out a data processing operation.
Format	The arrangement of data.
Function	A specific purpose of an entity or its characteristic action.
Hardware	Physical equipment, e.g., mechanical, electrical, or electronic devices.
Head	A device that reads, records, or erases data on a storage device.
Heuristic	Pertaining to exploratory methods of problem solving.
I/O	Input or output or both.
Identifier	A symbol whose purpose is to identify, indicate, or name a body of data.
Indirect Address	An address in a computer instruction which indicates a location where the address of the referenced operand is to be found.
Initialize	To set counters, switches, and addresses to zero or other starting values at the beginning of, or at prescribed points in, a computer routine.
Input	The transferring of data from auxiliary or external storage into the internal storage of the computer.
Instruction	A set of bits (in an object program) or characters (in a source program) which as a unit cause the computer to perform an operation.
Internal Storage	The storage facilities forming an integral physical part of the computer and directly controlled by the computer. Also called main memory and core memory.
Interrupt	To stop a process in such a way that it can be resumed.
Jump	A departure from the normal sequence of executing instructions in a computer.

Label	An identifier.
Language	A set of representations, conventions, and rules used to convey information.
Leader	The blank section of tape at the beginning of the tape.
Least Significant Digit	The rightmost digit of a binary number.
Library	An organized collection of standard and proven routines and subroutines which can be incorporated in larger routines.
Library Routine	A proven routine that is maintained in a program library.
Load	To place data into internal storage.
Location	A place in storage or memory where a unit of data or an instruction may be stored.
Loop	A sequence of instructions that is executed repeatedly until a terminal condition prevails.
Machine Instruction	An instruction written in machine language.
Machine Language	A language designed for interpretation and use by the machine without translation.
Macro Instruction	An instruction in a source language that is equivalent to a specified sequence of machine instructions.
Manual Input	The entry of data by hand into a device at the time of processing.
Manual Operation	The processing of data in a system by direct manual techniques.
Memory	(1) The erasable storage in the computer. (2) Pertaining to a device in which data can be stored and from which it can be retrieved.
No Op	An instruction that specifically instructs the computer to do nothing, except to proceed to the next instruction in sequence.
Object Program	The machine language program which is the output after translation from the source language. The binary program which runs on the computer.
Octal	(1) Pertaining to a characteristic or property involving a selection, choice, or condition in which there are eight possibilities. (2) Pertaining to the numeration system with a radix of eight.
Off Line	Pertaining to equipment or devices not under direct control of the computer.
On Line	Pertaining to equipment or devices under direct control of the computer; also to programs operating directly and immediately to user commands, e.g., FOCAL and DDT.
Open Subroutine	A subroutine that must be relocated and inserted into a routine at each place it is used.
Operand	That which is effected, manipulated, or operated upon.
Origin	The absolute address of the beginning of a program.
Output	Information transferred from the internal storage of a computer to output devices or external storage.

Overflow	The generation of a quantity beyond the capacity of a register.
Page	In the PDP-8/I, a unit of 200 (octal) locations which may be addressed directly.
Patch	To modify a routine in a rough or expedient way.
Predefined Process	A named process consisting of one or more operations or program steps that are specified elsewhere in a routine.
Procedure	The course of action taken for the solution of a problem.
Processor	A computer program that includes the compiling, assembling, translating, and related functions for a specific programming language.
Program	The complete sequence of instructions and routines necessary to solve a problem.
Program Library	A collection of available computer programs and routines.
Programming Language	A language used to prepare computer programs.
Protected Location	A storage location reserved for special purposes in which data cannot be stored without undergoing a screening procedure to establish suitability for storage therein.
Punched Paper Tape	A paper tape on which a pattern of holes is used to represent data.
Pushdown List	A list that is constructed and maintained so that the next item to be retrieved is the most recently stored item in the list, i.e., last in, first out.
Radix	The quantity of characters for use in each of the digital positions of a numbering system.
Read	To transfer information from an input device to internal storage; also refers to the internal acquisition of data from memory.
Real Time	Pertaining to computation performed while the related physical process is taking place so that results of the computation can be used in guiding the physical process.
Record	A collection of related items of data, treated as a unit.
Register	A device capable of storing a specified amount of data, such as one word.
Reset	To restore a storage device to a prescribed state.
Restart	To reestablish the execution of a program.
Routine	A set of instructions arranged in proper sequence to cause the computer to perform a desired task.
Run	A single, continuous performance of a program.
Scan	To examine sequentially part by part.
Search	To examine a set of items for those that have a desired property.
Set	To place a storage device into a specified state.
Single Step	Operation of the computer in which each instruction is performed in response to a single manual operation.

Skip	To ignore one or more instructions in a sequence of instructions.
Software	The collection of programs and routines associated with the computer.
Source Language	A symbolic language that is an input to a given translation process.
Source Program	A program written in a symbolic (source) language.
Statement	A meaningful expression or generalized instruction in a source language.
Step	One operation in a routine.
Storage Allocation	The assignment of blocks of data to specified blocks of storage.
Storage Capacity	The amount of data that can be contained in a storage device.
Storage Device	A device into which data can be entered, in which it can be held, and from which it can be retrieved.
Store	To enter data into a storage device.
String	A connected sequence of entities such as characters in a command string.
Subroutine	A routine that can be part of another routine.
Switch	A device or programming technique for making selections.
Symbolic Address	An address expressed in symbols convenient to the programmer. A label.
Symbolic Coding	Writing instructions using symbolic notation instead of actual machine instruction notation.
System	An assembly of software and hardware united to form an organized whole.
Tape Drive	A device that moves tape past a head.
Temporary Storage	Storage locations reserved for intermediate results.
Terminal	A point in a system at which data can either enter or leave.
Time Sharing	The interleaving of the time of a device.
Toggle	Pertaining to the operation of a flip-flop or switch.
Translate	To convert from one language to another.
Underflow	The condition that arises when a computation yields a result whose magnitude is smaller than the system is capable of representing.
Variable	A quantity that can assume any of a given set of values.
Word	A 12-bit unit of data in the PDP-8/I which may be stored in one addressable location.
Word Length	The number of bits in a word.
Write	To transfer information from internal storage to an output device or to auxiliary storage.

APPENDIX C

OFF-LINE TAPE PREPARATION AND EDITING

In order to run a program on the computer, instructions and data must first be fed into the computer from the input device.

The program and data could be typed into the computer on-line. However, computer time is valuable, and hunt-and-peck typing on-line can be an expensive process. For this reason, it may be desirable to prepare the program and data off-line, that is, to punch the program and data onto paper tape using a separate machine, one not actually connected to the computer.

The ASR33 Teletype can be used off-line to prepare source program tapes, to duplicate tapes, and to edit tapes previously punched in the ASCII format. (Tapes punched from the Teletype keyboard are in ASCII format.)

When the Teletype power control switch is turned to LOCAL, the unit becomes an off-line tape preparation facility. Procedures for using the Teletype off-line are listed below. The Teletype controls are described in Section 1, ASR33 Teletype, and are shown in Figure INTRO-2.

DUPLICATING TAPES

a. Turn TTY to LOCAL.
b. Set LSR to FREE.
c. Put original tape into LSR.
d. Depress LSP ON.
e. Depress HERE IS key to generate leader tape.
f. Set LSR to START. (New tape is punched and data is typed on printer.)
g. After the original tape is read in, depress HERE IS key to generate trailer tape.
h. Remove tapes from LSR and LSP.

PREPARING NEW PROGRAM TAPES

When preparing a program tape off-line, the user should observe the same conventions of his programming language as when preparing a program on-line using Editor. Following are the manual operating procedures for off-line tape preparation.

a. Turn TTY to LOCAL.
b. Depress LSP ON.

c. Depress HERE IS key to generate leader tape.

d. Type the source program, observing the conventions of the programming language being used.

NOTE

The RETURN and LINE FEED keys must be depressed at the end of each line.

Depressing the CTRL/TAB keys perforates the tab character onto the tape, and the typewheel moves only one position to the right. When the computer reads the punched tab character on output, it will cause the typewheel to tab (a tab is usually equal to 10 spaces).

e. After the source program is punched, depress HERE IS to generate trailer tape.

f. Remove the source program tape from LSP.

CORRECTING TYPING ERRORS

Typing errors can be corrected using the B. SP. button and the RUBOUT key. The B. SP. button backspaces the tape one column for each depression of the button, and the RUBOUT key perforates all eight channels of the tape (this perforation is ignored by the computer).

EDITING

Punched tapes can be edited off-line. However, the user must be able to read the perforations on the tape, otherwise, off-line editing is virtually impossible.

a. Turn TTY to LOCAL.

b. Set LSR to FREE.

c. Put tape to be edited into LSR.

d. Depress LSP ON.

e. Depress HERE IS to generate leader tape.

f. Set LSR to START.

g. Observe the printer as the program is being typed, and

h. Set LSR to STOP a few characters ahead of the text to be edited.

i. Advance the tape one character at a time by toggling the LSR control from START to STOP.

For Minor Edit: Advance tape past the text to be edited and use the B. SP. and RUBOUT keys to erase old text, then type and punch new text.

For Major Edit:

(1) Set LSR to STOP one character ahead of the text to be edited.

(2) Type new text.

(3) Depress LSP OFF.

(4) Set LSR to START to advance tape past edited area (printing but not punching tape), then set LSR to STOP.

(5) Depress LSP ON.

(6) Set LSR to START.

j. Repeat from step f until editing is completed.

k. Set LSR to START.

l. After new source program tape is punched, depress HERE IS to generate trailer tape.

m. Remove old tape from LSR and discard.*

n. Remove new tape from LSP and save.

*It's good programming practice to list the new tape before discarding the old, ensuring that the new tape is correct.

APPENDIX D

SUMMARY OF PDP-8/I SUBROUTINES

Name	DEC Number	Calling Sequence*	Memory Locations (Decimal)
		Function Subroutines	
1. Square Root Single Precision	FMAA	- /Square in AC JMS SQRT /Call - /Return with root	23
2. Signed Multiply Single Precision	FMBA	- /Multiplier in AC JMS MULT /Call ADDRESS /Address of multipli-/cand - /Return. High order product in AC; low order in MP1	44
3. Signed Divide Single Precision	FMCA	- /High dividend in AC JMS DIVIDE /Call LOWD /Low dividend DIVSOR /Divisor - /Return quotient in AC; remainder in HDIVND	62
4. Signed Multiply Double Precision	FMDA	- /AC ignored JMS DMUL /Call HORDMD /Address of high order multiplicand HORDMR /Address of high order multiplier - /Return high order product in AC. Remainder of product in B, C, D.	125
5. Signed Divide Double Precision	FMEA	- /Address of high order dividend in AC JMS DUBDIV /Call HORDDR /Address of high order divisor - /Return. High order quotient in AC; low order quotient in DIVND4. High and low remainder in DIVND 1 and DIVND2	105

*All of the calling sequences here assume that the data is in the correct format and that there are no overflow conditions to check upon completion. For details on the data and indicators for overflow conditions, the user is referred to the appropriate program write-up.

Name	DEC Number	Calling Sequence*	Memory Locations (Decimal)
		Function Subroutines (Cont)	
6. Sine Routine Double Precision	FMFA	- /AC = 0000 JMS DSIN /Call ADDRESS /Address of high order word - /Return. AC = 0 1 = 0 Answer in ADDRESS and ADDRESS + 1	248 (+double precision multiply)
7. Cosine Routine Double Precision	FMGA	- /AC = 0000 JMS DCOS /Call ADDRESS /Address of high order word - /Return. AC = 0 1 = 0 Answer in ADDRESS and ADDRESS + 1	64 (+ double precision sine and double precision multiply)
8. Four-Word Floating Point Package	FMHA	See Floating Point System Programming Manual, Digital-8-5-S	1041
9. Signed Multiply (EAE) Single Precision	8-21-F	/Multiplier in AC JMS MULT /Call ADDRESS /Address of multiplicand - /Return. Most significant product in AC; least significant in MP1	
10. Signed Divide (EAE) Single Precision	8-22-F	- /High dividend in AC JMS SPDIV /Call LOWD /Low dividend DIVSOR /Divisor - /Return. Quotient in AC; remainder in DVD.	45
11. Signed Multiply (EAE) Double Precision	8-23-F	- /AC ignored JMS DMUL /Call HORDMD /Address of high order multiplicand HORDMR /Address of high order multiplier /Return high order product in AC; remainder in B, C, and D.	104

*All of the calling sequences here assume that the data is in the correct format and that there are no overflow conditions to check upon completion. For details on the data and indicators for overflow conditions, the user is referred to the appropriate program write-up.

Name	DEC Number	Calling Sequence*	Memory Locations (Decimal)
		Function Subroutines (Cont)	
12. EAE Floating Point Package	8-25-F	See Floating Point System Programming Manual, Digital-8-5-S	See 8-5-S
		Utility Programs	
Punch Programs			
1. RIM Punch	PMPO	Binary tape (see write-up)	
2. BIN Punch ASR33 75A	8-5-U	Binary tape (see write-up)	
Processor Programs			
1. Logical Subroutines Inclusive OR	FMIA	- JMS INCOR ADDRES - /One argument in AC /Call /Address of second argument /Return	12
Exclusive OR		JMS EXCOR ADDRES /One argument in AC /Call /Address of second argument	14
2. Arithmetic Shift	FMJA	(General Calling Sequence) - JMS** ADDRES - /Negative number of shifts in AC /Call /Address to be shifted /Return with shifted number in AC	
Shift Left, Single Prec.		**SPSL	12
Shift Right, Single Prec.		**SPSR	15
Shift Left, Double Prec.		**DPSL (Least significant part in LSH)	24
Shift Right, Double Prec.		**DPSR (Least significant part in LSH)	
3. Logical Shift	FMKA		
Shift Right, Single Prec.		**LSRSP	12
Shift Right, Double Prec.		**LSRDP (Least significant part in LESTSG)	24
(Left Shift Logical; identical to left shift arithmetic)			

*All of the calling sequences here assume that the data is in the correct format and that there are no overflow conditions to check upon completion. For details on the data and indicators for overflow conditions, the user is referred to the appropriate program write-up.

Name	DEC Number	Calling Sequence*	Memory Locations (Decimal)
		Utility Programs (Cont)	
4. EAE Instruction Set Simulator	8-17-U	(See write-up)	
BCD - Binary Conversion			
1. BCD-to-Binary Conversion	8-10-U	- /BCD number in AC JMS DCDBIN /Call	26
2. BCD-to-Binary Conversion	8-11-U		
3. Binary-to-BCD	8-14-U	- /Binary number in AC JMS BCD /Call - /BCD number in AC	33
4. Binary-to-BCD Conversion (Used primarily for writing mag tape in BCD Format)	8-15-U	- /AC contains binary number JMS BCD /Call - /Return. BCD number in ONE and TWO	53
Teletype Message Sub-Routines			
1. Alphanumeric Message	8-18-U	- /AC ignored JMS MESAGE/Call - /First two characters of message .) .) /Remaining characters .) .) "XX" represents the last character XX00 /End of message code - /Return	51
2. Teletype Output Package	8-19-U	Return is to location following "call" AC = 0000	75
Type One Character		- (/AC 0-5 = 00 (/AC 6-11 = trimmed code JMS TYPE /Call	
Type Two Characters		- /AC 0-5 = 1st character /AC 6-11 = 2nd character JMS TY2 /Call	
Type a Digit		- /AC 8-11 = digit JMS TDIG /Call	

*All of the calling sequences here assume that the data is in the correct format and that there are no overflow conditions to check upon completion. For details on the data and indicators for overflow conditions, the user is referred to the appropriate program write-up.

Name	DEC Number	Calling Sequence*	Memory Locations (Decimal)
		Utility Programs (Cont)	
Type a Space		JMS TYSP /Call	
Type a CR and LF		JMS TYCR /Call	
Type a Tab		JMS TYTB /Call	
3. Character String Typeout	8-20-U	- /AC = initial address JMS TYPSTG /Call of string - /Return. AC clear	64
Decimal Print Subroutines			
1. Unsigned Decimal Print, Single Prec.	8-22-U	- /AC contains numbers JMS DECPRT /Call - /Return. AC clear	38
2. Signed Decimal Print, Single Prec.	8-23-U	- /AC contains number JMS SSPRNT /Call - /Return. AC clear	48
3. Unsigned Decimal Print, Double Prec.	8-24-U	- /AC ignored JMS UDPRNT /Call ADRESS /Address of high order word - /Return. AC clear	73
4. Signed Decimal Print, Double Prec.	8-25-U	- /AC ignored JMS SDPRNT /Call ADRESS /Address of high order word - /Return. AC clear	86
Decimal Input Routines			
1. Decimal to Binary Input. Signed or Unsigned, Single Prec.	8-28-U	- /AC ignored JMS SICONV/Call - /Return. AC contains number	74
2. Decimal to Binary Input, Signed or Unsigned, Double Prec.	8-29-U	- /AC ignored JMS DICONV/Call ADRESS /Address for high order word - /Return. AC clear	110
Miscellaneous			
1. Octal Memory Dump	8-6-U	None	77
2. DECtape Library System Loader	8-3-U	(See write-up)	17

*All of the calling sequences here assume that the data is in the correct format and that there are no overflow conditions to check upon completion. For details on the data and indicators for overflow conditions, the user is referred to the appropriate program write-up.

Name	DEC Number	Calling Sequence*	Memory Locations (Decimal)
		Utility Programs (Cont)	
3. Incremental Plotter Subroutines	8-12-U	(See write-up)	128
4. Symbolic Tape Format Generator	8-21-U	(See write-up)	-
5. DECtape Subroutine	SUB0-D	(See write-up)	256

*All of the calling sequences here assume that the data is in the correct format and that there are no overflow conditions to check upon completion. For details on the data and indicators for overflow conditions, the user is referred to the appropriate program write-up.

READER'S COMMENTS

PAPER TAPE
SYSTEM USER'S GUIDE
DEC-08-NGCC-D

Digital Equipment Corporation maintains a continuous effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback – your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability, and readability.

_____

_____

_____

_____

_____

Did you find errors in this manual? _____

_____

_____

_____

_____

How can this manual be improved? _____

_____

_____

_____

_____

_____

DEC also strives to keep its customers informed of current DEC software and publications. Thus, the following periodically distributed publications are available upon request. Please check the appropriate boxes for a current issue of the publication(s) desired.

☐ Software Manual Update, a quarterly collection of revisions to current software manuals.

☐ User's Bookshelf, a bibliography of current software manuals.

☐ Program Library Price List, a list of currently available software programs and manuals.

Please describe your position. _____

Name _____ Organization _____

Street _____ Department _____

City _____ State _____ Zip or Country _____

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Digital Equipment Corporation
Software Information Services
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754