Honeywell

EQUIPMENT OPERATORS' MANUAL

SERIES 16

HARDWARE

Honeywell EQUIPMENT OPERATORS' MANUAL

SERIES 16

SUBJECT:

Operator and Programmer Information Required to Operate the Model 316, and 516 Computers from the Control Panel.

SPECIAL INSTRUCTIONS:

This manual supersedes the Series 16 Equipment Operators' Manual, Order Number BX48, Rev. 1 (Document Number 70130072165D) dated February 1972; Addendum A, July 1972; Addendum B, October 1972; and Addendum C, April 1973. Technical changes have been made in accordance with ECO 31953. Change bars in the margins indicate revisions; asterisks denote deletions.

This edition incorporates technical changes made in accordance with ECO 9182, February 9, 1971; ECO 9689, July 15, 1971; ECO 20324, July 1972; ECO 20464, October 1972; and ECO 30014, April 1973.

DATE:

November 1973

ORDER NUMBER:

BX48, Rev. 2

DOCUMENT NUMBER:,

70130072165F

PREFACE

This Manual is intended for use by computer operators and programmers when operating the Model 316, or 516 computer directly from the control panel. It contains general operating descriptions and references to more detailed information.

Console use, generation of self-loading tapes, loading and operating procedures, verification and test routines, error messages, and paper tape formats are described herein.

The information in this manual should be supplemented with a Run Book which contains specific instructions for a particular installation (e.g., disk file names and starting addresses of standard programs).

Recommended general supplementary reading includes the various language manuals and the programming manuals for Series 16 computers and I/O devices.

The following specific manuals directly supplement the information contained in this manual.

Name	Order Number	Document Number
Models 316 and 516 Programmers' Reference Manual	BX47	70130072156
DAP-16 and DAP-16 Mod 2 Assembly Language	BY09	70130072442
Series 16 ASR Teletypewriter/ High-Speed Paper Tape Reader/ High-Speed Paper Tape Punch Programmers' Reference Manual	BY17	70130072443
Series 16 FORTRAN IV Manual	BX32	70130071364
Series 16 Software Documentation Guide	BY18	70130072468
Series 16 016-XREF, SSUP, and MAC Source Language Processors	AC94	70130072582

File No. 1A03

CONTENTS

ILLUSTRATIONS

TABLES

SECTION I
CONTROL PANEL

Basic Honeywell 316/516 computers include a main frame, a control panel, and an optional ASR teletypewriter. In addition to these standard devices, the installation may include peripheral devices such as the following:

High-speed paper tape reader	Line printer
High-speed paper tape punch	Magnetic tape unit
Card reader or reader/punch	Disk storage unit
Storage display unit	Drum storage unit

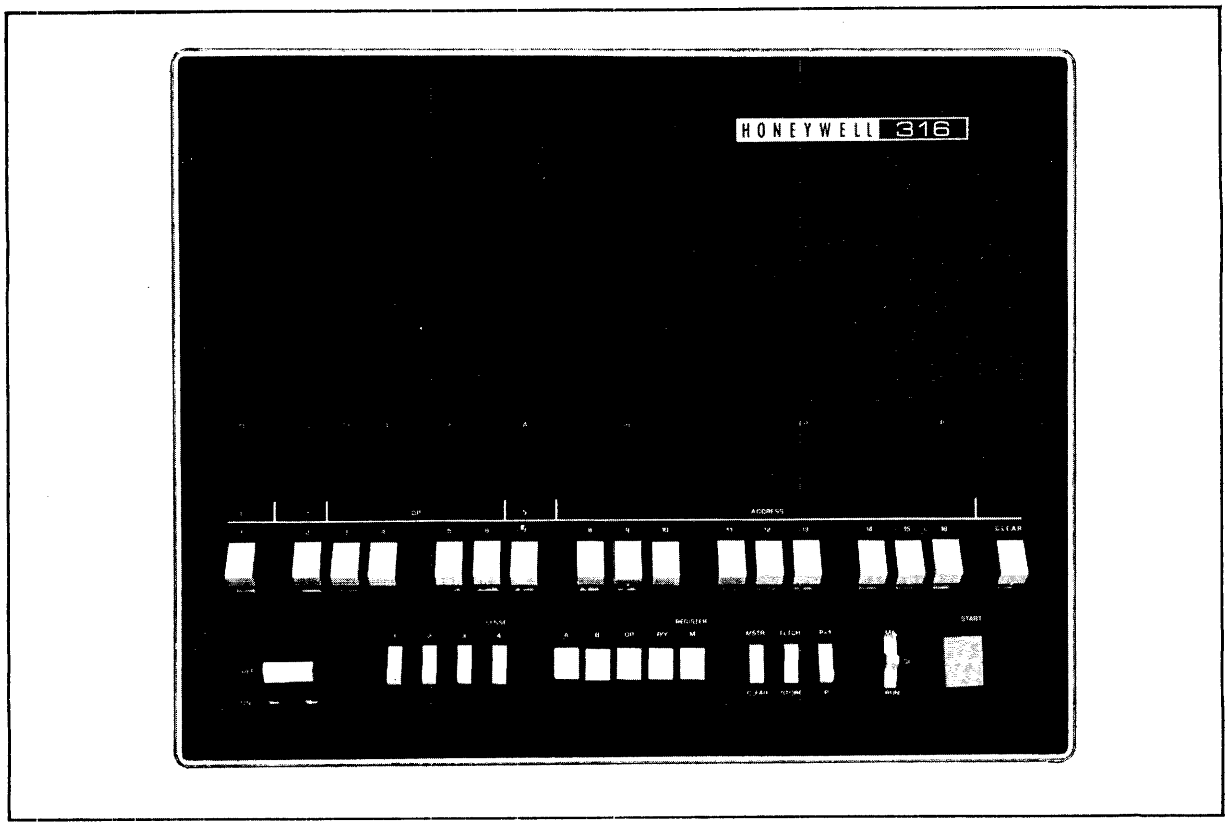
CONTROLS AND INDICATORS

The control panel (Figures 1-1 and 1-2) has mounted on it the controls and indicators for normal system operation. These controls and indicators are listed and described in Tables 1-1 through 1-3. The two panel versions shown in Figure 1-1 are identified with equipment types 316-01, 316-0100 and 316-0110. These types differ primarily in mechanical design and configuration. The Type 316-0110 contains the improved memory module.

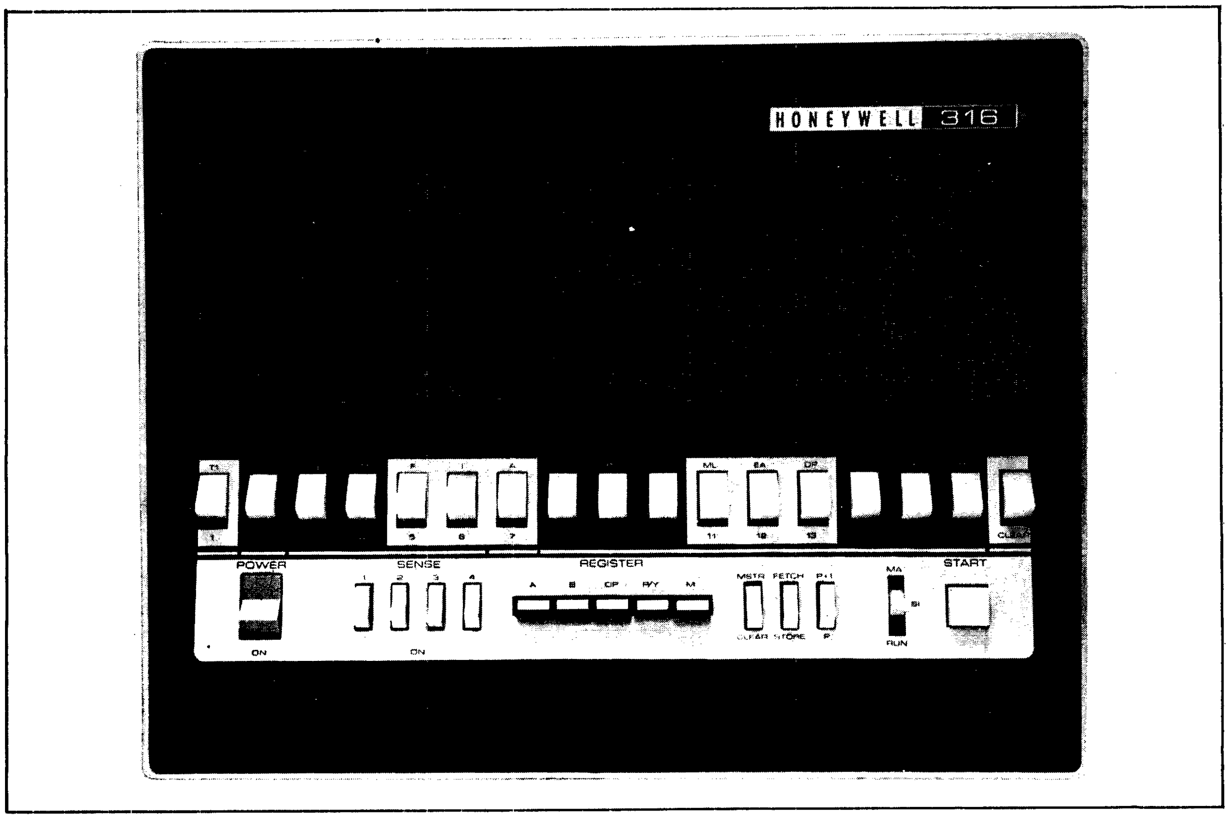
Table 1-1. Registers Displayed on Honeywell 316/516 Control Panel

Register	Bits	Display
X	1-16	Index register (516 only)
A	1-16	Register A (primary arithmetic and logic register)
B	1-16	Register B (secondary arithmetic and logic register)
OP	1-16	State of key flip-flops in system (refer to Table 1-4).
P/Y	1-16	Register Y (memory address register). When computer is operating in memory access (MA) mode, registers P and Y contain the same address. When in single instruction (SI) or run mode, register P contains an address which is one greater than the address contained in register Y.
M	1-16	Register M (memory buffer register). Contains contents of memory location specified by register Y.

NOTE: While there is no X-register indicator for the 316, it is nevertheless present internally.



Type 316-01



Types 316-0100 and 316-0110

Figure 1-1. 316 Control Panel

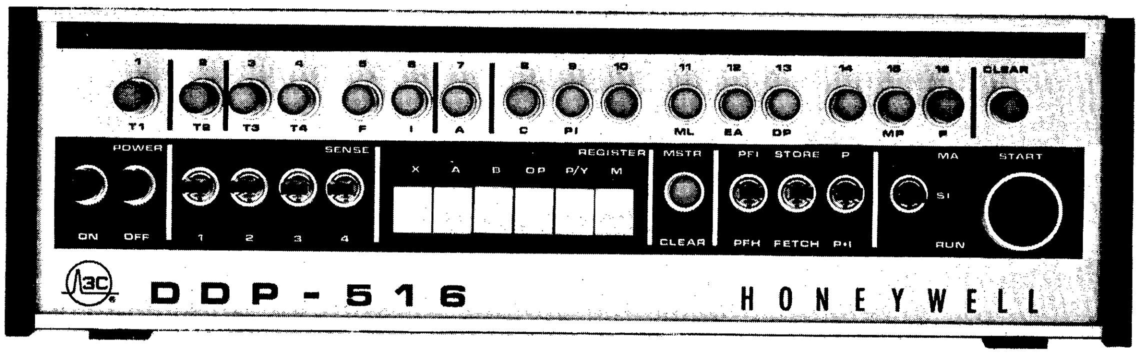


Figure 1-2. 516 Control Panel

Table 1-3. Function of Control Panel Selector Switches

Switch	Position		Function
SENSE 1, 2, 3, 4			Enables altering the course of a program via switch interrogation during program operation.
	516	316	
	Up	Down	Switches are set.
	Down	Up	Switches are reset.
PFI/PFH (516 only, power fail interrupt/ power fail halt)	PFI		Allows machine to cause a program interrupt when power fails.
	PFH		Allows machine to halt when power fails.
STORE/FETCH	STORE		Enables data to be written into memory when computer is in MA mode.
	FETCH		Enables data to be read out of memory when computer is in MA mode.
P/P+1	P		Enables accessing a specific memory location when computer is in MA mode.
	P+1		Enables accessing consecutive memory locations when computer is in MA mode.
MA/SI/RUN [a] (memory access/ single instruction/ run)	MA (memory access mode)		Enables data to be written into or read from a location in memory and disables protection for location $1\text{-}17_8$.
	SI (single instruction mode)		Enables step-by-step execution of a program and enables control panel switches.
	RUN (run mode)		Enables normal operating mode.

[a] Switch must be in SI or MA to enable master clear, data clear, or manual insertion of data into registers.

Table 1-3. Function of Control Panel Selector Switches

Switch	Position		Function
SENSE 1, 2, 3, 4			Enables altering the course of a program via switch interrogation during program operation.
	516	316	
	Up	Down	Switches are set.
	Down	Up	Switches are reset.
PFI/PFH (516 only, power fail interrupt/ power fail halt)	PFI		Allows machine to cause a program interrupt when power fails.
	PFH		Allows machine to halt when power fails.
STORE/FETCH	STORE		Enables data to be written into memory when computer is in MA mode.
	FETCH		Enables data to be read out of memory when computer is in MA mode.
P/P+1	P		Enables accessing a specific memory location when computer is in MA mode.
	P+1		Enables accessing consecutive memory locations when computer is in MA mode.
MA/SI/RUN[a] (memory access/ single instruction/ run)	MA (memory access mode)		Enables data to be written into or read from a location in memory and disables protection for location 1-17_8.
	SI (single instruction mode)		Enables step-by-step execution of a program and enables control panel switches.
	RUN (run mode)		Enables normal operating mode.

[a]Switch must be in SI or MA to enable master clear, data clear, or manual insertion of data into registers.

OPERATING PROCEDURES

Data within memory and several of the mainframe registers can be monitored on the control panel. Data can also be manually entered into memory and several registers from the control panel. Procedures for initializing the system and reading out and entering data into memory and the mainframe registers are provided in the following paragraphs.

System Initialization

The system can be initialized by depressing the MSTR CLEAR pushbutton. This operation has no effect on the contents of memory or register X. The master clear function is also performed when power is applied to the system. Individual devices normally are not affected by the master clear function; however, the device controllers are cleared.

Register Display

The contents of register X (516 only), A, B, OP, P/Y or M can be displayed by depressing the appropriate REGISTER pushbutton. The contents of the selected register is displayed by the 16 pushbutton/indicators on the 516 or the indicator lights on the 316. Refer to Table 1-4 for the significance of the control panel display if OP is selected.

Register Load

Data is entered into register A, B, P/Y, or M as follows:

a. Depress appropriate REGISTER pushbutton.
b. Depress MSTR CLEAR or CLEAR pushbutton.
c. Enter desired data by depressing appropriate buttons (1 through 16). (The P-register is limited to the number of memory address bits used.)

Data can be entered into register X by addressing location 000000_8 and entering the data into that location. For the procedure to enter data into memory, refer to the paragraph on Memory Load in this section.

Table 1-4. Control Panel Display When OP Is Selected

Bit	Significance	Bit	Significance
1	T1, timing level 1	9	PI, permit interrupt
2	T2, timing level 2	10	Unassigned
3	T3, timing level 3	11	ML, memory lockout (option)
4	T4, timing level 4	12	EA, extended addressing (bank-switching option)
5	F, fetch cycle	13	DP, double precision (option)
6	I, indirect cycle	14	Unassigned
7	A, execute cycle	15	MP, memory parity error
8	C, C-bit	16	P, I/O parity error

Memory Display

The contents of any memory location is displayed as follows:

a. Set MA/SI/RUN switch to MA.
b. Set FETCH/STORE switch to FETCH.
c. Set P/P+1 switch to P.
d. Depress REGISTER P/Y pushbutton.
e. Depress MSTR CLEAR or CLEAR pushbutton.
f. Depress appropriate buttons (1 through 16) to designate octal address of memory location containing data to be displayed.
g. Depress REGISTER M pushbutton.

h. Depress START pushbutton. The contents of the addressed memory location will be displayed on pushbutton/indicators 1 through 16.

To display the contents of successive memory locations, proceed as follows:

i. Set P/P+1 switch to P+1.

j. Depress START pushbutton. Each time the START pushbutton is depressed, the contents of the next memory location is displayed.

Memory Load

Data is entered into any memory location as follows:

a. Set MA/SI/RUN switch to MA.

b. Set FETCH/STORE switch to STORE.

c. Set P/P+1 switch to P.

d. Depress REGISTER P/Y pushbutton.

e. Depress MSTR CLEAR pushbutton.

f. Depress appropriate buttons to designate octal address of memory location to be loaded.

g. Depress REGISTER M pushbutton.

h. Depress DATA CLEAR pushbutton.

i. Depress appropriate buttons to enter desired data into addressed location.

j. Depress START pushbutton.

The desired data is now in the addressed location. To load successive memory locations, proceed as follows:

k. Set P/P+1 switch to P+1.

l. Repeat steps h. through j. above for each successive memory location to be loaded.

Single-Instruction Operation

A program is executed in the single-instruction mode as follows:

a. Set MA/SI/RUN switch to SI. (When this switch is not in the MA position, the FETCH/STORE and P/P+1 are disabled.)

b. Depress MSTR CLEAR pushbutton.

c. Enter initial parameters into register A, B, or P/Y as required. (Refer to the procedure for Register Load described in this section.) Any instruction placed in M is executed in step d.

d. Depress START pushbutton.

The instruction in M is executed. HLT has no effect. The next instruction is fetched from memory and placed in register M, and may be examined by depressing the REGISTER M pushbutton. Thereafter, each time the START pushbutton is depressed, the previously fetched instruction is executed, the next instruction is fetched, and the computer halts. If the P/Y pushbutton is depressed, the address from which the new instruction was fetched is displayed on pushbutton/indicators 1 through 16. During execution (run operation), the SI position can be used to aid in program debugging if no real-time requirement exists. I/O operations must be carefully considered before using SI.

Run Operation

A program is executed in the run mode as follows:

a. Set MA/SI/RUN switch to SI.

b. Depress MSTR CLEAR button (unless clearly not desired).

c. Enter initial parameters into register A, B, or P/Y as required. (Refer to the procedure for Register Load described in this section.)

d. Set MA/SI/RUN switch to RUN.

e. Depress START pushbutton.

The computer runs until a halt is executed or until the MA/SI/RUN switch is set to SI.

Key-In-Loaders

The instructions listed below in Table 1-5 normally occupy memory locations 1 through 17_8 and enable the loading of self-loading paper tapes or cards by means of the teletypewriter, high-speed paper-tape reader, card reader/punch (316/516-5140), or either card reader (316/516-61 or -5100).

The hardware protects memory locations 1 through 17_8 from modification by a stored program; therefore, under normal conditions, the key-in loader should remain intact in these locations. However, when operating in the MA mode, locations 1 through 17_8 are unprotected; therefore, care must be exerted to avoid inadvertently destroying the key-in loader while loading memory.

Table 1-5. Key-In Loader Instructions

PAPER TAPE INPUT: LOC.	HIGH-SPEED READER	ASR READER	COMMENTS
1 STA '57	010057	010057	SAVE A REGISTER
*			FOR START LOCATION
2 OCP 1/4	030001	030004	START READER
3 INA '1001/'1004	131001	131004	INPUT FIRST FRAME
4 JMP *-1	002003	002003	DELAY UNTIL READY
5 SNZ	101040	101040	IGNORE LEADER
6 JMP *-3	002003	002003	LOOP IN LEADER
7 STA 0	010000	010000	STORE IN LOCATION 0
10 INA '1001/'1004	131001	131004	CLEAR AND INPUT FRAME
11 JMP *-1	002010	002010	DELAY UNTIL READY
12 LGL 8	041470	041470	SHIFT TO PACK
13 INA 1/4	130001	130004	INPUT FRAME
14 JMP *-1	002013	002013	DELAY UNTIL READY
15 STA* 0	110000	110000	INDIRECTLY STORE WORD
16 IRS 0	024000	024000	POINT AT NEXT WORD
17 SZE	100040	100040	TEST INPUT WORD

Table 1-5 (cont). Key-In Loader Instructions

CARD INPUT: LOC.	CARD READER	CARD READER/ PUNCH	COMMENTS
1 STA '57	010057	010057	SAVE A REGISTER
2 OCP '105/'106	030105	030106	READ FIRST CARD
3 INA '1005/'1006	131005	131006	INPUT A '20
4 JMP *-1	002003	002003	
5 STA 0	010000	010000	STORE IN INDEX
6 SKP	100000	100000	SKIP START ADDR.
7 DAC 'NN000	0NN000	0NN000	START OF PALC
10 INA '1005/'1006	131005	131006	CLEAR AND INPUT COLUMN
11 JMP *-1	002010	002010	DELAY UNTIL READY
12 LGL 8	041470	041470	SHIFT TO PACK
13 INA 5/6	130005	130006	INPUT COLUMN
14 JMP *-1	002013	002013	DELAY UNTIL READY
15 STA* 0	110000	110000	INDIRECTLY STORE WORD
16 IRS 0	024000	024000	POINT AT NEXT WORD
17 SZE	100040	100040	TEST INPUT WORD

Normal Memory Load

The key-in loader is entered into memory as follows:

a. Depress MSTR CLEAR pushbutton. The program counter (register P) is set to zero.

b. Set MA/SI/RUN switch to MA. This selects the memory access mode and unlocks the protected area, addresses 1-17_8.

c. Set STORE/FETCH switch to STORE.

d. Set P/P+1 switch to P+1.

e. Enter first instruction into register M.

f. Depress START button. Register P is incremented by one, and the first instruction is entered into location 000001.

g. Repeat step f for each of the remaining instructions to be loaded.

Each time the START pushbutton is depressed, register P is incremented by one, and the instruction in register M is loaded into memory.

Normal Memory Display

The following steps are performed to e[illegible] that the key-in loader has been loaded correctly into the designated memory locations.

a. Set MA/SI/RUN switch to MA.

b. Depress MSTR CLEAR pushbutton.

c. Set STORE/FETCH switch to FETCH.

d. Set P/P+1 switch to P+1.

e. Depress register M pushbutton.

f. Depress START button. Register P is incremented by one, and the contents of memory location 000001 is displayed on pushbutton/ indicators 1 through 16.

g. Repeat step f. for each of the remaining memory locations to be monitored.

Each time the START pushbutton is depressed, register P is incremented by one, and the contents of the addressed memory is displayed.

Use of the key-in loaders is described in Section III.

SECTION II
GENERATION OF SELF-LOADING TAPES

In addition to an extensive library of object tapes, the Honeywell 316/516 software package contains several tapes that are self-loading. Two of them are a loader program (LDR-APM) and a punch and load program (PAL-AP). In addition, all of the test and verification routines are supplied as self-loading tapes. These categories are discussed throughout this section.

Library routines required for card and magnetic tape options are generic by name. Identification of the exact document number required may be obtained from the appropriate Programmers Reference Manual.

Computer systems with 4K of memory are supplied with a DAP-16 self-loading tape. Systems with 8K of memory are supplied with a FORTRAN IV self-loading tape and all object programs necessary to make a DAP-16 self-loading tape. Systems with over 8K of memory are supplied with all object programs necessary to make DAP-16 or FORTRAN IV self-loading tapes.

DEFINITION OF SELF-LOADING TAPES

Program tapes that are called self-loading tapes can be loaded by using the key-in loader and a two-part self-contained loader. These programs are loaded by entering P=1 and pressing the START pushbutton. *

All of these self-loading tapes were punched out of memory by the punch and load (PAL-AP) program. After PAL-AP has been directed to an output device and the limits of memory to be output have been set, it outputs the two-part self-contained loader that is used to supplement the key-in bootstrap at program load time. The first part of the two-part loader is 32_{10} instructions punched in binary 8-8 format. At load time the key-in loader loads these 32_{10} instructions into memory locations 20_8 through 57_8. The second part of the loader is 176_8 locations long, and is also in binary 8-8 format. This portion of the loader is relocatable at program load time and can be loaded into $XX600_8$ (where XX is any sector desired except 2_8 and 42_8). To exercise this option, enter A-register=$XX600_8$ at load time. If the A-register is left with zeros at load time, the 176_8-word loader will go to the sector specified by the default start location (575_8 relocatable in PAL-AP), which is normally the sector PAL-AP is in when the loader is punched.

Following the two-part loader, PAL-AP punches out about 12 inches of leader and then punches the contents of memory between the specified limits. This output is punched in invisible (4-6-6) format and is in blocks of 50_8 words. Each block is preceded by a start-of-message code 201_8 and a starting location. The data block is followed by a checksum, X-OFF (code 223_8), RUB OUT (377_8), and eight frames of leader. Eight combinations of data cause special action in the ASR-33/35. The presence of these characters is tested, and conversions are made to prevent the special action. At load time, the substituted characters are checked for, and the data is reconverted where necessary.

NOTE: To minimize the output from the PAL-AP program, it is recommended that memory be cleared prior to loading the object program to be dumped, because PAL-AP does not punch blocks of zeros. Instead, it punches a control word to indicate the number of zeros in a block. Thus, by clearing core, unnecessary data is not punched on the system tapes.

The following routine may be keyed in manually and used to clear all of core above memory location 00022.

Memory Location	Octal Contents	DAP Mnemonics
00000	000022	
00020	000013	EXA
00021	002015	JMP 15

Press MSTR CLEAR and start at location 00020.

With a computer system having more than 4096 memory locations, the user will find it desirable and sometimes necessary to have a self-loading version of PAL-AP that loads into the highest sector of memory. He also will find it desirable and necessary to make a self-loading version(s) of the relocating loader program(s) (i.e., LDR-APM, SLDR-A, or SLDR-P) that also loads into the highest sectors of memory. Having these routines located at the top of memory makes all of the lower core available for program loading and allows the loader(s) to make checks for memory overflow and set aside a common storage area.

The following procedures for creating DAP and FORTRAN self-loading tapes assume that a loader self-loading tape has been constructed. As illustrated in Figure 2-1, the procedure for generating a self-loading tape (SLST) involves six basic operations. These operations are:

a. Entering the loader into high sectors of memory.

b. Loading the main program (DAP-16, FRTN, etc.) into memory with specified starting locations at which loading and cross-sector linkage are to begin.

c. Loading the I/O routines (IOS, I/O library, etc.) to satisfy the main program calls.

d. Obtaining a memory map to determine the area of memory that contains data to be punched out on the self-loading tape.

e. Loading the punch and load (PAL-AP) program into the high sector of memory.

f. Punching out the SLST from limits determined by the memory map.

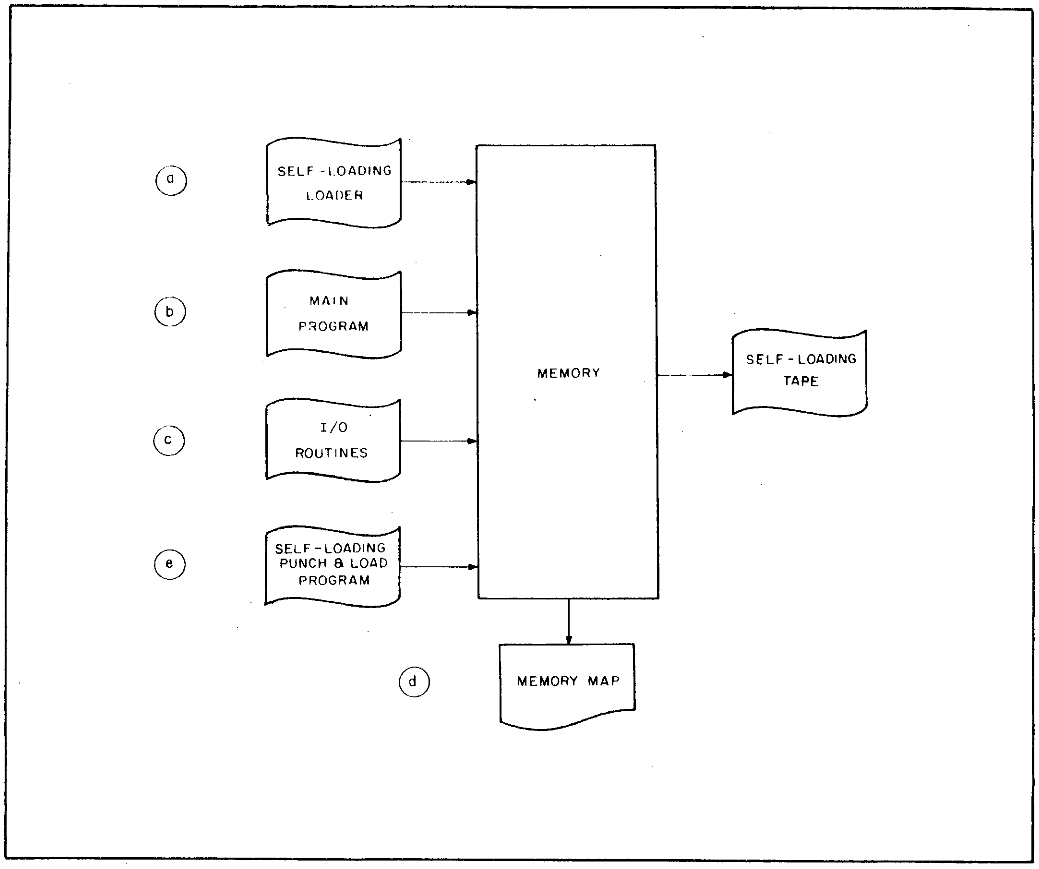


Figure 2-1. Self-Loading Tape Generation

GENERATION OF PAL-AP SYSTEM

Routines Required

a. LDR-APM 4K self-loading tape, No. 180005100[1]

b. PAL-AP object tape, No. 180311000

Simplified Procedure (See Section III, Operating Procedures)

a. Load LDR-APM 4K self-loading tape (set P to 00001_8).

b. Use LDR-APM (start at 7000_8) to load PAL-AP object tape at XX000, where XX is the highest sector in memory (refer to Table 2-1).

c. Obtain memory map (for reference only).

NOTE: Establish the future starting address of the 176_8-word loader (refer to Section IV for a definition of PAL-AP generated tapes). Location $XX575_8$ now specifies sector XX, the highest sector. This, however, can be changed to any sector desired by manually inserting the value of $NN600_8$ in memory location $XX575_8$ (NN is the sector desired and XX is the sector PAL-AP is in).... This paragraph is intended for information only, as XX is the sector desired for step d.

d. Punch out PAL-AP program just loaded in self-loading tape format. Set limits of PAL-AP from $XX000_8$ to $XX577_8$. Do not be concerned that the upper limit given to be punched is less than the upper limit of the memory map. The area from $XX600_8$ to $XX776_8$ is part of the PAL-AP loader and is punched automatically. (Refer to Table 2-1 for values of XX.)

Table 2-1. Memory Sector Substitution Chart

Memory Size	UU	WW	ZZ	VV	YY	XX
4K	02	03	04	05	06	07
8K	12	13	14	15	16	17
12K	22	23	24	25	26	27
16K	32	33	34	35	36	37
24K	52	53	54	55	56	57
32K	72	73	74	75	76	77

where: UU is the highest available sector -5
WW is the highest available sector -4
ZZ is the highest available sector -3
VV is the highest available sector -2
YY is the highest available sector -1
XX is the highest available sector in memory

[1] To order, contact your Honeywell Sales Representative.

GENERATION OF LDR-APM SYSTEM

LDR-APM loads all object programs from the DAP-116 assembler, DAP-16 assembler, and X-16 FORTRAN IV compiler. It loads object programs by means of the ASR, high-speed reader, magnetic tape unit, cards, or disk. With this variety of input devices, provisions have been made to configure the loader to the specific device(s) and device unit numbers to be used. Coding for paper-tape inputs is within the LDR-APM object program. Coding for magnetic tape input is by means of the I/O library routines I$MA-U and M$UNIT-U. Coding for card input is by means of the I/O library routine I$CB. Requests for program loading by means of the disk are transferred to the disk operating system (DOP) which returns control to the loader after loading the program.

The simplest configuration for an LDR-APM self-loading tape would be for paper tape input only. This configuration requires four sectors of memory, and placement would normally be in the four highest sectors of memory. For every feature added, LDR-APM must be placed lower in core and the features added placed in the next sector above it. Next, in order of complexity, would be to include magnetic tape or card load capabilities. This configuration would best be accomplished by placing the magnetic tape or card input routines in the highest sector of memory and the LDR-APM program in the four sectors immediately preceding the high sector. This method, while not the only one possible, is recommended because the names table used by the loader starts at the base of the loader and proceeds toward lower core. When the disk file is to be used with LDR-APM, the disk operating program and its associated I/O routines (M$10 or D$10) would be loaded above LDR-APM. To load above LDR-APM, the user must change the location defined in the common base (XX000 - 1000, where XX000 is the loader starting point) to a value above the highest location into which he wishes to load. Linkage between LDR-APM and DOP is established through dedicated memory locations 770_8 through 777_8, thus eliminating the requirement for both routines to be on one self-loading tape.

The following procedures describe the generation of LDR-APM self-loading tape with and without magnetic tape capabilities.

Generation Without Magnetic Tape or Card Reader

ROUTINES REQUIRED

a. LDR-APM self-loading tape, No. 180005100

b. LDR-APM object tape, No. 180005000

c. PAL-AP object tape, No. 180311000

d. LDR-DUMY object tape, No. 180835000

SIMPLIFIED PROCEDURE (See Section III, Operating Procedures)

a. Load LDR-APM 4K self-loading tape.

b. Load LDR-APM object tape at $ZZ000_8$, where ZZ is the highest sector minus 3; i.e., ZZ = 14 for 8K (refer to Table 2-1 for ZZ). TTY prints "MR."

c. Load LDR-DUMY. This call could remain unsatisfied with no effect on loading capabilities. TTY prints "LC."

d. Take a memory map for future reference.

e. Insert desired input device code in location $VV000_8$, where VV is 1000_8 REL; i.e., VV = 15 for 8K. The octal code occupies bits 14-16 and is preset to 1, where:

1	= high-speed paper-tape reader
2	= ASR paper-tape reader
4	= disk drive

f. If loading is to be always from the same device, the halt for allowing device selection is omitted. This is accomplished by inserting an NOP (101000_8) into location 3443_8 (relocatable to $ZZ000_8$; i.e., for 8K, ZZ = 14). Thus, 3443_8 (relocatable) would be 17443_8. (The location is renamed HHHT.)

NOTE: This location applies to LDR-APM, Rev B and later.

g. Load PAL-AP object tape into sector NN, where NN is any convenient sector except 02 or 42.

h. Change control of PAL-AP default location $NN575_8$ from $NN600_8$ to $XX600_8$, where XX is the highest sector occupied by the loader (refer to Table 2-1).

i. Execute PAL-AP at $NN000_8$ to dump between $ZZ000_8$ and $XX577_8$. The output tape is the new self-loading loader.

Generation With Magnetic Tape and Without Card Reader

ROUTINES REQUIRED

a. LDR-APM 4K self-loading tape, No. 180005100

b. LDR-APM object tape, No. 180005000

c. I$MA object tape

d. M$UNIT object tape

e. PAL-AP object tape, No. 180311000

f. LDR-DUMY object tape, No. 180835000

SIMPLIFIED PROCEDURE (See Section III, Operating Procedures)

a. Load LDR-APM 4K self-loading tape.

b. Load LDR-APM object tape at $WW000_8$, where WW is the highest sector of memory minus 4; i.e., for 16K, WW = 33. Get MR on ASR.

c. Satisfy loader call for I$MC by loading I$MA object tape at $XX000_8$, where XX is the highest available sector; i.e., for 16K, XX = 37.

NOTE: To continue loading with a new origin, set A-register to new origin, set P-register to 7003_8, and depress the START pushbutton. Get MR ASR.

d. Satisfy loader call for SUBR's by loading M$UNIT. Get MR or TTY.

e. Satisfy loader call for SUBR's by loading LDR-DUMY. Get LC on TTY.

f. Take a memory map for future reference.

g. Insert changes in M$UNIT for proper magnetic tape I/O channel and magnetic-tape drive numbers. Insert proper codes into bits 14-16 of the following addresses specified by the memory map.

M$TY	Channel type: I/O bus = 0, DMC = 1, DMA = 2
M$T0	Physical number assigned to logical tape 0
M$T1	Physical number assigned to logical tape 1
M$T2	Physical number assigned to logical tape 2
M$T3	Physical number assigned to logical tape 3
M$T4	Physical number assigned to logical tape 4
M$T5	Physical number assigned to logical tape 5
M$T6	Physical number assigned to logical tape 6
M$T7	Physical number assigned to logical tape 7
M$C0	DMA or DMC channel number assigned to logical tape 0
M$C1	DMA or DMC channel number assigned to logical tape 1
M$C2	DMA or DMC channel number assigned to logical tape 2
M$C3	DMA or DMC channel number assigned to logical tape 3
M$C4	DMA or DMC channel number assigned to logical tape 4
M$C5	DMA or DMC channel number assigned to logical tape 5
M$C6	DMA or DMC channel number assigned to logical tape 6
M$C7	DMA or DMC channel number assigned to logical tape 7

h. Insert desired device code in location VV000$_8$, where VV000$_8$ is (1000$_8$ relocatable) to the beginning of the loader. The octal code occupies bits 14-16 and is preset to 1, where:

1 = high-speed paper-tape reader

2 = ASR paper-tape reader

3 = magnetic tape drive

4 = disk drive

If magnetic tape is selected, bits 11-13 should be set up to physical tape number from which loading takes place.

i. If loading is to be always from the same device, the halt for allowing device selection is omitted. This is accomplished by changing the first halt to a NOP.

NOTE: This applies to LDR-APM, Rev B and later.

j. Load PAL-AP object into sector NN, where NN is any convenient sector except sector 02 or 42.

k. Change contents of PAL-AP default location NN575$_8$ from NN600$_8$ to XX600$_8$, where XX is the highest available sector in memory.

l. Execute PAL-AP at NN000$_8$ to dump between low and high locations obtained from memory map taken in step f. Tape obtained is the new LDR-APM self-loading tape.

Generation With Card Reader

ROUTINES REQUIRED

a. LDR-APM 4K self-loading tape, No. 180005100

b. LDR-APM object tape, No. 180005000

c. I$CB object tape

d. LDR-DUMY object tape, No. 180835000

e. PAL-AP object tape, No. 180311000

SIMPLIFIED PROCEDURE (Refer to Section III for detailed operating procedures)

a. Load LDR-APM 4K self-loading tape.

b. Load LDR-APM object tape at $WW000_8$, where WW is the highest sector of memory minus 4; i.e., for 16K, WW = 33. Get MR on ASR.

c. Satisfy loader call for MR by loading I$CB object tape at $XX000_8$, where XX is the highest available sector; i.e., for 16K, XX = 37.

NOTE: To continue loading with a new origin, set A-register to new origin, set P-register to 7003_8, and press START pushbutton. Get MR on ASR.

d. Satisfy loader call for MR by loading LDR-DUMY. Get LC on ASR.

e. Take a memory map.

f. Insert desired device code in location $VV000_8$, where $VV000_8$ is (1000_8 relocatable) to the address WW in step b. The octal code occupies bits 14-16 and is preset to 1, where:

1 =	high-speed paper-tape reader
2 =	ASR paper-tape reader
4 =	disk
5 =	card reader

g. If loading is to be always from the same device, the halt for allowing device selection is omitted. This is accomplished by changing the first halt to a NOP.

NOTE: This applies to LDR-APM, Rev B and later.

h. Load PAL-AP object into sector NN, where NN is any convenient sector except 02 or 42.

i. Change contents of PAL-AP default location $NN575_8$ from $NN600_8$ to $XX600_8$, where XX is highest available sector in memory.

j. Execute PAL-AP at $NN000_8$ to dump between $WW000_8$ and high location obtained from memory map taken in step e. The tape obtained is the new LDR-APM self-loading tape.

Generation With Magnetic Tape and Card Reader

ROUTINES REQUIRED

a. LDR-APM 4K self-loading tape, No. 180005100

b. LDR-APM object tape, No. 180005000

c. I$CB object tape

d. I$MA object tape

e. M$UNIT object tape, No. 180602000

f. PAL-AP object tape, No. 180311000

SIMPLIFIED PROCEDURE (Refer to Section III for detailed operating procedures)

a. Load LDR-APM 4K self-loading tape.

b. Load LDR-APM object tape at WW000$_8$, where WW is the highest sector of memory minus 4; i.e., for 16K, WW = 33. Get MR on ASR.

c. Satisfy loader call for MR by loading I$MA object tape at XX000$_8$, where XX is the highest available sector; i.e., for 16K, XX = 37.

NOTE: To continue loading with a new origin, set A-register to new origin, set P-register to 7003$_8$, and press START pushbutton. Get MR on ASR.

d. Satisfy loader call for MR by loading M$UNIT.

e. Get MR on ASR.

f. Satisfy loader call for MR by loading I$CB. Get LC on ASR.

g. Take a memory map.

h. Insert changes in M$UNIT for proper magnetic-tape I/O channel and magnetic-tape drive numbers. Insert proper codes into bits 14-16 of the following addresses specified by the memory map.

M$TY	Channel type: I/O bus = 0, DMC = 1, DMA = 2
M$T1	Physical number assigned to logical tape 1
M$T2	Physical number assigned to logical tape 2
M$T3	Physical number assigned to logical tape 3
M$T4	Physical number assigned to logical tape 4
M$T5	Physical number assigned to logical tape 5
M$T6	Physical number assigned to logical tape 6
M$T7	Physical number assigned to logical tape 7
M$C1	DMA or DMC channel number assigned to logical tape 1
M$C2	DMA or DMC channel number assigned to logical tape 2
M$C3	DMA or DMC channel number assigned to logical tape 3
M$C4	DMA or DMC channel number assigned to logical tape 4
M$C5	DMA or DMC channel number assigned to logical tape 5
M$C6	DMA or DMC channel number assigned to logical tape 6
M$C7	DMA or DMC channel number assigned to logical tape 7

i. Load PAL-AP object into sector NN, where NN is any convenient sector except 02 or 42.

j. Change contents of PAL-AP default location NN575$_8$ from NN600$_8$ to XX600$_8$, where XX is highest available sector in memory.

k. Execute PAL-AP at NN000$_8$ to dump between WW000$_8$ and high location obtained from memory map taken in step g. The tape obtained is the new LDR-APM self-loading tape.

GENERATION OF SLDR-A OR SLDR-P SYSTEM

Routines Required

a. LDR-APM 4K self-loading tape, No. 180005100

b. PAL-AP object tape, No. 180311000

c. SLDR-A object tape, No. 180341000

d. SLDR-P object tape, No. 180342000

Simplified Procedure (See Section III, Operating Procedures)

a. Load 4K loader self-loading tape.

b. Use 4K loader to load SLDR-A, or SLDR-P at VV000$_8$, where VV is the highest sector minus 2; i.e., 8K = 15000$_8$ (refer to Table 2-1).

c. Load PAL-AP object tape into sector ZZ000$_8$, where ZZ is the highest sector minus 4 (refer to Table 2-1).

d. Change PAL-AP default instruction at memory location ZZ575$_8$ from ZZ600$_8$ to XX600$_8$, where XX is the highest sector in memory. This enables the 176$_8$-word portion of the PAL-AP loader to be loaded automatically into sector XX at program load time.

e. Use PAL-AP at ZZ000$_8$ to dump new SLDR-A or SLDR-P self-loading tape. Set limits of PAL-AP at VV000$_8$ to XX577$_8$.

GENERATION OF MINILOAD SYSTEM

Routines Required

a. LDR-APM 4K self-loading tape, No. 180005100

b. PAL-AP object tape, No. 180311000

c. MINILOAD object tape, No. 180580000

Simplified Procedure (See Section III, Operating Procedures)

a. Load 4K LDR-APM self-loading tape.

b. Use 4K loader to load MINILOAD object tape at XX000, where XX is the highest sector in memory (refer to Table 2-1 for values of XX).

c. Take a memory map for future reference.

d. Use 4K loader again to load PAL-AP object tape into YY000, where YY is the next to the highest sector (refer to Table 2-1).

e. Use PAL-AP at YY000$_8$ to dump out MINILOAD self-loading tape. Give PAL-AP a lower limit of XX000$_8$ and upper limit of XX677$_8$.

NOTE: Placing PAL-AP at sector YY automatically establishes sector YY for the loading of the 176$_8$-word relocatable portion of the loader. Loading the MINILOAD self-loading tape destroys the last 200$_8$ words of sector YY.

GENERATION OF SSUP SYSTEM

This information is contained in the Series 16 manual O16-XREF, SSUP, and MAC Source Language Processors, Order Number AC94, (Document Number 70130072582A).

*

GENERATION OF DAP-16 SYSTEM

This information is contained in Section VII of the Series 16 DAP-16 and DAP-16 Mod 2 Assembly Language manual.

GENERATION OF FORTRAN IV SYSTEM

FORTRAN IV, like the DAP-16 assembler, makes use of the computer's peripheral devices. It does not call an I/O routine directly. Instead, FORTRAN IV leaves the selection of the desired I/O device up to the FORTRAN IV Input/Output Supervisor. FORTRAN IV calls for five services from the I/O Supervisor selected. They are:

a.	F4$INT:	Initialization
b.	F4$IN:	Input
c.	F4$OUT:	Binary output
d.	F4$SYM:	Symbolic output
e.	F4$END:	Compilation complete

The FORTRAN IV I/O Supervisor is F4-IOS-0, Document Number 70181568000, Order Number M-1894. It makes calls to the standard drivers that are contained in the I/O libraries for the following devices.

Teletype
High-speed paper tape reader
High-speed paper tape punch
Line printer
Card reader
Card punch
Magnetic tape drive
Disk drive

NOTE: The compiler is satisfied with any driver written for the called generic device, but the program will not run unless the driver written for the specific device is loaded. For example, a Honeywell Model 5120 card reader can run only with the I$CA card reader driver written for that model. An attempt to load the I$CA for the Model 5140 will satisfy the load, but will not operate the Model 5120 card reader. Therefore, the user must use the appropriate I/O library.

This generation can be performed on systems with 12K and over. An 8K user receives a FORTRAN system with his software package. A 4K user cannot execute the FORTRAN IV compiler.

Paper tape input and output is assumed, although the process is essentially the same regardless of the media used. Details on loading and using LDR-APM and PAL-AP are given in Section III.

Routines Required

a. LDR-APM

b. FORTRAN IV compiler object tape, Document Number 70180463000 (Rev J or later)

c. OPDUM object tape, Document Number 70181981000 (Rev A or later)

NOTE: OPDUM pertains to FORTRAN users not making calls on OP-16. FORTRAN users may choose to substitute OPMOD, Document Number 7018198000, Rev A or later) for OPDUM to provide for use of OP-16 FORTRAN statements and in-line assembly code. This feature requires one and one-half additional sectors of memory. Instructions on using OPMOD are included in the OP-16 User's Guide.

d. F4-IOS-0 object tape, Document Number 70181568000 (Rev B or later)

e. I/O library tapes for all devices used

f. F4-DUM object tape, Document Number 70180588000 (Rev F or later)

g. PAL-AP

Simplified Procedure (See Section III, Operating Procedures)

a. Load LDR-APM into highest convenient sectors of memory.

b. Load FORTRAN IV compiler object tape. MR is printed.

c. Load OPDUM object tape. MR is printed.

d. Load F4-IOS-0 object tape at the start of the next sector. MR is printed.

e. Satisfy loader requests for more routines (MR) by loading the I/O libraries for the desired devices.

f. Load F4-DUM object tape. LC (loading complete) is printed. F4-DUM is used to satisfy all undesired calls made by IOS and to set a default data pool size (see step h).

g. Take a complete memory map (see step c. under Options Following Loader Halt in Section III). The BASE entry must not exceed 624_8 or it will interfere with the compiler. (If BASE is higher, this procedure must be restarted at step b.) Suggestions to help keep the final BASE value within limits follow.

   1. Minimize cross-sector references by preventing any routines from crossing a sector boundary when they load. (This action is taken at the expense of available core.)

   2. Do not load routines that never will be used; e.g., if a system is equipped with a high-speed paper tape reader and a card reader, and if FORTRAN source statements always are to be prepared on cards, load the input driver for the card reader only.

h. Set data pool size. The size of memory available to the compiler is ascertained by F4-DUM. The default state is that F4-DUM will determine the upper limit of physical memory ('17777, '27777, or '37777). If all of memory cannot be used, the highest address that can be used must be placed in location TOP within F4-DUM prior to the first execution of the compiler (e.g., if DOP is to be used). The default action is taken when TOP equals zero (the state when loaded).

i. If the magnetic tape library has been loaded, alter the M$UNIT locations according to the assignments described below.

M$TY	0 (for I/O bus), 1 (for DMC), or 2 (for DMA)
M$C2, M$C3, M$C4	No assignments are necessary for I/O bus; for DMC, 1 or 2 should be specified for channel 1 or 2; for DMA, 1, 2, 3, or 4 should be specified for channel 1, 2, 3, or 4.
M$T2	0 for source input on transport 0
M$T3	1 for listing output on transport 1
M$T4	2 for object output on transport 2

If a DMC channel other than 1 or 2 is assigned to the magnetic tape, that channel cannot be used, and operation must be accomplished through the I/O bus. Other locations in M$UNIT are disregarded. For more details, see the Programmer's Reference Manuals for the various magnetic tape options.

j. Load PAL-AP into NN000, where NN is a convenient sector above the HIGH address obtained from the complete memory map (see step g).

k. Execute PAL-AP using 100_8 as the low-limit address and the HIGH address obtained from the complete memory map as the high-limit address.

The PAL-AP program punches a self-loading version of the compiler, OPDUM, IOS, I/O library routines, and F4-DUM in PAL format.

FORTRAN Octal Storage Requirements

Table 2-2 gives the approximate storage requirements in octal for the basic components of a FORTRAN system.

Table 2-2. FORTRAN Octal Storage Requirements

Program	Size in Octal
FRTN	14700
OPDUM	50
(or OPMOD)	(1300)
F4-IOS-0	600
F4-DUM	35

The I/O library storage requirements also must be considered. These can be found in the Programmer's Reference Manuals for the various I/O options.

SECTION III
LOADING AND OPERATING PROCEDURES

This section presents detailed procedures for loading, executing, assembling, and compiling programs, as well as generating self-loading paper tapes. The assembly compilation, and updating procedures describe the use of the self-loading tapes which are generated according to the instructions in Section II. Some of the procedures described for specific programs in Section II have been generalized to cover all programs on paper tape.

LOADING PROCEDURES

Key-In Loader

Refer to Section I for a description of the key-in loader.

Loading Self-Loading Tapes

Self-loading tapes make use of the key-in loader. This procedure assumes that the proper key-in loader for the input device used is in core. A definition of self-loading tapes appears in Section II. For computers with extended addressing, refer to Appendix B.

a. Set MA/SI/RUN switch to SI position.

b. Depress MSTR CLEAR pushbutton.

c. Load 000001_8 into register P.

d. Insert self-loading tape into appropriate input device (ASR or high-speed reader).

e. Enter one of the following into register A:

   1. All zeros. The 176_8-word relocatable portion of the loader goes to $NN600_8$, where NN is the sector specified by PAL-AP when the tape is being punched. Normally, NN is the highest sector of memory.

   2. $XX600_8$, where XX is the sector in which the 176_8-word self-contained portion of the loader is to be loaded.

f. Set MA/SI/RUN switch to RUN position.

g. Execute one of the following sets of instructions for the input device used.

   1. If the ASR-33 is used as the input device:

      (a) Depress START pushbutton.

      (b) Momentarily set ASR-33 reader control switch to START position.

2. If the ASR-35 is used as the input device:

   (a) Position manual control switch to OFF position.

   (b) Position ASR reader control switch to STOP position.

   (c) Position ASR mode switch to K position.

   (d) Depress control key (CTRL) and Q key simultaneously.

   (e) Reposition ASR mode switch to T.

   (f) Depress START pushbutton.

   (g) Set ASR-35 reader control switch to RUN position.

3. If the photo reader is used as the input device, depress the START pushbutton.

   NOTE: Observe register X (516-only), which should indicate counting if the tape is being loaded properly. The X-register displays the locations into which the tape is being loaded. The self-loading program is loaded into the locations it occupied when it was punched out. Loading any self-loading tape destroys the contents of locations 20_8 through 57_8.

Load Indications

Self-loading tapes use a checksum test to verify correct loading. If the computer and tape stop after the last punched frame is read, then the load is satisfactory.

Loading Object Tapes

Object tapes are the direct output of the DAP-116, DAP-16 assembler, or the FORTRAN IV compiler. Object tapes may be in the absolute (ABS) or relocatable (REL) mode. As loading procedures for the modes vary somewhat, it is recommended that the operator check the program listing to determine the mode in which the tape was punched. Generally, DAP-16, FRTN, and SSUP are supplied as absolute programs. Library subroutines are examples of relocatable format. All Honeywell-supplied DAP-16 object tapes are prepared in the two-pass mode.

Object tapes produced by the DAP-16 assembler or FORTRAN IV compiler consist of a 24-bit word for each 16-bit computer word. The extra bits allow the assembler or compiler to assign addresses without consideration of crossing memory sectors. Actually, the assembler assigns a 15-bit address for each instruction (a capability of 32K direct access) and a 3-bit code that indicates if the address is absolute, relocatable, etc. (See Section VI, Paper Tape Formats.)

The relocating loader program reduces each 24-bit word to an appropriate 16-bit word, and stores it in memory. (Refer to Figure 3-1, Desectorized Program Loading.)

The software package for the Honeywell 316/516 contains several relocating loaders to lead object programs. Loaders vary in size of program, input devices used, and types of programs they load. For time and space consideration, the operator usually selects the shortest loader that meets the loading requirements. (See Table 3-2 for loader selection.)

Operating Instructions for Relocating Loaders LDR-RPM, SLDR-A, SLDR-P, LDR-C, and SLDR-C.

The loaders discussed in the following paragraphs have several options; each option is selected by the starting location entered.

LOADING OPTIONS

In the following, XX represents the highest sector occupied by the loader.

a. P = XX000 = Normal object program loading.

b. P = XX004 = Force-load a subroutine. This entry loads a subroutine even if it has not been called by a main program.

c. P = XX006 = (LDR-APM only). Loads object programs in the extended desectorizing mode; otherwise, identical to P = XX000.

LOADING INSTRUCTIONS

a. Set MA/SI/RUN switch to SI position.

b. Enter desired loading option into register P (reference Loading Options above).

c. If program is relocatable (REL), enter octal address of location at which loading is to begin into register A. If register A is 0, starting location of 1000_8 is assumed. To start loading at 00000_8, set sign bit of register A. Note that locations 1-17 are hardware-protected.

d. If program is absolute (ABS), A-register setting is unnecessary.

e. Enter octal address of location at which intersector indirect-address word table is to begin into register B. If register B is zero, starting location of 100_8 is assumed.

   NOTE: With loaders SLDR-A, SLDR-P, or SLDR-C, the intersector indirect word table must be in sector 0 for Honeywell 316/516 computers not equipped with the memory lockout option. Loaders LDR-APM and LDR-C support the SETB pseudo-operations. Thus, the intersector indirect word table may be program controlled and may be in sector 0 or the same sector as the instruction. If a computer is equipped with the memory lockout option, the table may be in any sector.

f. Insert object program into appropriate input device. (If card reader is used, turn validity OFF and make it READY.)

g. Set MA/SI/RUN switch to RUN position.

h. (LDR-APM only): Depress START pushbutton. Computer will halt for input device selection. (See note.)

1. Position MA/SI/RUN switch to SI. Enter into bits 14-16 of of register A one of the following octal codes:

1 =	High-speed paper-tape reader
2 =	ASR paper-tape reader
3 =	Magnetic tape
4 =	Disk/drum
5 =	Card reader

If magnetic tape is selected, set bits 11-13 of register A to physical tape number from which loading takes place.

2. If register A is set to zeros, the preselected device is used.

NOTE: Step h. may be eliminated by preselecting I/O devices and placing an NOP in location 3443_8 (REL). (Refer to instructions for generating an LDR-APM system tape, Section II.)

i. Depress START pushbutton. Object program is loaded into memory until loader message is produced on teletypewriter and halt occurs. (See Table 3-1 for significance of loader messages.)

NOTE: When starting LDR-APM after a loader message typeout, the device selection code must be entered in register A, if loading is to continue from a device other than the one specified in the device default location (1000_8 REL).

Table 3-1. Loader Messages

Message	Meaning	Action Required[a]
LC	Loading complete	Depress START pushbutton to begin program execution.
MR	More subroutines required	Insert required subroutine tapes into appropriate input device and depress START pushbutton to continue loading.
CK	Checksum error in last block read	Depress START pushbutton to ignore block and continue loading. Valid load is not assured.
BL	Block too large or improperly formatted	Depress START pushbutton to ignore block and continue loading. Valid load is not assured.
MO	Memory overflow due to program attempting to overwrite loader[b]	Depress START pushbutton to obtain memory map. Refer to Appendix A for memory map format. No recovery is possible from this error.

[a]When a loader halt occurs and a message is produced, several options can be performed in addition to those specified in this table. These options are described in the paragraphs following this table.
[b]When loading the loader or any program higher in core than the loader, change loader location FIL7 from NN700 to XX700, where NN is the last sector occupied by the loader, and XX is the highest sector in memory. The 4K system loaders supplied have been modified. The loader program listing indicates the location of FIL7; otherwise, an MO error will result.

The loader cannot detect a program overlaying the indirect word tables or the tables overlaying the program. A memory map must be obtained to determine if overlay exists.

OPTIONS FOLLOWING LOADER HALT

a. Reload object tape by repeating procedure for loading object programs described previously.

b. Recover from a missing end-of-tape block. The procedure is:

   1. Set MA/SI/RUN switch to SI, which causes computer to halt.
   2. Load $XX001_8$ into register P.
   3. Set MA/SI/RUN switch to RUN.
   4. Depress START pushbutton.

c. Print a memory map. The procedure is:

   1. Set MA/SI/RUN switch to SI.
   2. Load $XX002_8$ into register P.
   3. Set MA/SI/RUN switch to RUN.
   4. Depress START pushbutton. A memory map is produced on the teletype. (See Appendix A, Memory Map.)

d. Set a program break. Continue loading with a new origin. The procedure is:

   1. Set MA/SI/RUN switch to SI.
   2. Load $XX003_8$ into register P.
   3. Load register A with address of location where loading is to continue. If register A is cleared, origin for loading remains unchanged.
   4. Set MA/SI/RUN switch to RUN.
   5. Depress START pushbutton to continue loading.

e. Force-load a subprogram. The procedure is:

   1. Set MA/SI/RUN switch to SI.
   2. Load $XX004_8$ into register P.
   3. Repeat step 3. above.
   4. Insert tape into appropriate input device if required.
   5. Depress START pushbutton to load tape into memory.

f. Begin executing the object program. The procedure is:

   1. Set MA/SI/RUN switch to SI.
   2. Load $XX005_8$ into register P.
   3. Set MA/SI/RUN switch to RUN.
   4. Depress START pushbutton to execute program.

Table 3-2. Loader Selection

Loader	Number of Sectors Used	Device Used	Restrictions[a]
MINILOAD[b]	1	ASR, high-speed reader, or card reader	Loads only DAP-16 two-pass object programs; does not load subroutines, external calls, or in extended memory mode. Psuedo-operations SETC and SETB are not supported. Mixed modes of assembly are not handled. Example: REL mode to ABS mode / ABS mode to REL mode
SLDR-A	3	ASR-33/35	Does not load DAP-16 one-pass object tapes, or extended memory mode programs. Pseudo-operations SETC and SETB are not supported.
SLDR-P	3	High-speed reader	
LDR-APM[c]	4 or 5	ASR, high-speed reader, card reader, magnetic tape, or disk	This is the expanded loader which loads both one- and two-pass DAP-16 object tapes as well as FORTRAN IV object tapes. When using disk, DOP must be in core and initialized.
LDR-C	4	200-cpm card reader	SETC pseudo-operations are not supported.
SLDR-C	3	200-cpm card reader	Does not load DAP-16 one-pass object programs or in extended memory mode. SETC pseudo-operations are not supported.

[a]Restrictions listed are the most salient restrictions. The individual listings contain the complete lists of restrictions.

[b]MINILOAD at execution time requires only one sector, but the 177_8-word portion of the PAL-AP loader must go to a sector other than the last sector when MINILOAD is being loaded.

[c]LDR-APM uses four sectors without magnetic tape/cards or five sectors with magnetic tape/cards. For the disk features, it requires the disk operating program (DOP) and the appropriate disk I/O routine to be in memory.

LDR-APM ADVANTAGES

a. Allows loading of programs in normal mode in the upper 16K.

b. Provides a default sector zero in the same 16K.

c. Provides a map that shows START as the load point if not otherwise specified.

d. Removes redundant cross-sectorizing in the case of a SETB and sector zero reference.

e. Merges flag and tag bits in DACs rather than adding them.

f. Ignores tag on a DAC when loading with extended desectorizing. (Cannot index address constants in extended mode.)

g. Provides 14-bit address for symbolic address in normal mode. Provides 15-bit addresses for symbolic address in extended mode. Provides 16-bit addresses for absolute address in either mode.

h. Negative DACs (symbolic) are modulo 64K.

i. Provides start after LC by depressing the START pushbutton. Mode of execution is mode of loading (extended or normal).

j. Location of COMMON is found in XX000, where XX is the next to the last sector (i.e., location 6000_8 for 4K loader).

k. Basic storage of variable data overlays the bootstrap, so patching space (about 100_8 words) is available at the end of the last sector.

l. Can be entered in normal mode (no manual EXA required). Sets the index register to zero on entry (and must keep it zero in order to run properly for other entries).

m. Allows loading in extended desectorizing by an alternate entry even though the program did not have EXD (entry 'XX006, where XX is last sector of loader).

n. Provided as 4K SLT which can load above itself and as an object tape. PAL-AP dumps of this loader that are to reside above first 16K should be dumped with Revision D or higher of PAL-AP.

OPERATING INSTRUCTIONS FOR MINILOAD

Restrictions

Miniload does not operate on:

a. Object tapes produced by FORTRAN IV.

b. Object tapes produced in one-pass mode or with external references (including subroutines).

c. Extended memory option or the relocatable sector zero option.

d. Mixed modes of assembly. Examples: Relocatable mode to absolute mode.
Absolute mode to relocatable mode.

Operating Procedures

a. Set MA/SI/RUN switch to SI position.

b. Depress MSTR CLEAR pushbutton.

c. Set P = $XX000_8$, where XX is sector in which miniload was stored. Set register A equal to starting location where loading begins (if A is zero, a 1000_8 starting location is assumed). Set register B to starting location (in sector zero) for linkage table (if B is zero, 100_8 is assumed).

d. Set SSW1 if loading from ASR. Set SSW2 if loading from card reader. Reset all sense switches if loading from high-speed reader.

e. Press START pushbutton. Loading continues until an error is encountered or until loading has been completed. In either case, a halt occurs, and registers A and B contain a code which specifies the reason for the halt.

Table 3-3. Miniload Halts

A(1)	A(2)	A(13-16)	B(1-16)	Reason
0	0	Next available location	Next available location	Normal end-of-load halt; clear A and begin execution of program just loaded. Leave A and B unchanged or modify (A cannot be zero) to load next program.
0	1	Undefined	Undefined	Block error or illegal address type (forward ref., etc.); press START pushbutton to load new program.
1	0	Next available location	Undefined	Checksum error; press START push-button to ignore and continue loading.
1	1	Undefined	Undefined	Sector zero full; press START push-button to load.
NOTE: If errors were encountered, the program may not be executed.				

*

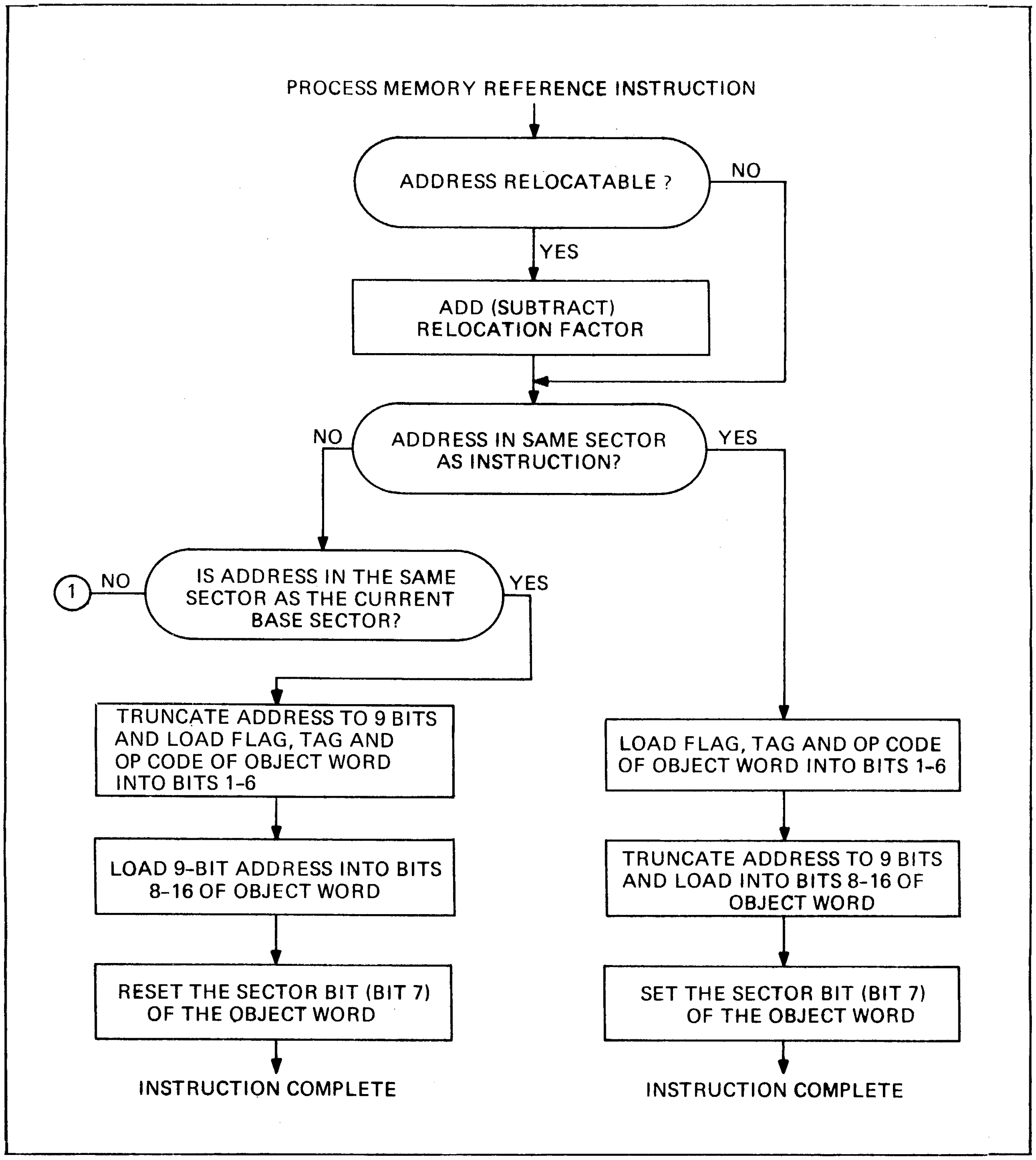

PROCESS MEMORY REFERENCE INSTRUCTION
ADDRESS RELOCATABLE ?
NO
YES
ADD (SUBTRACT) RELOCATION FACTOR
ADDRESS IN SAME SECTOR AS INSTRUCTION?
NO
YES
IS ADDRESS IN THE SAME SECTOR AS THE CURRENT BASE SECTOR?
1
NO
YES
TRUNCATE ADDRESS TO 9 BITS AND LOAD FLAG, TAG AND OP CODE OF OBJECT WORD INTO BITS 1-6
LOAD 9-BIT ADDRESS INTO BITS 8-16 OF OBJECT WORD
RESET THE SECTOR BIT (BIT 7) OF THE OBJECT WORD
INSTRUCTION COMPLETE
LOAD FLAG, TAG AND OP CODE OF OBJECT WORD INTO BITS 1-6
TRUNCATE ADDRESS TO 9 BITS AND LOAD INTO BITS 8-16 OF OBJECT WORD
SET THE SECTOR BIT (BIT 7) OF THE OBJECT WORD
INSTRUCTION COMPLETE

Figure 3-1. Desectorized Program Loading (Part 1 of 3)

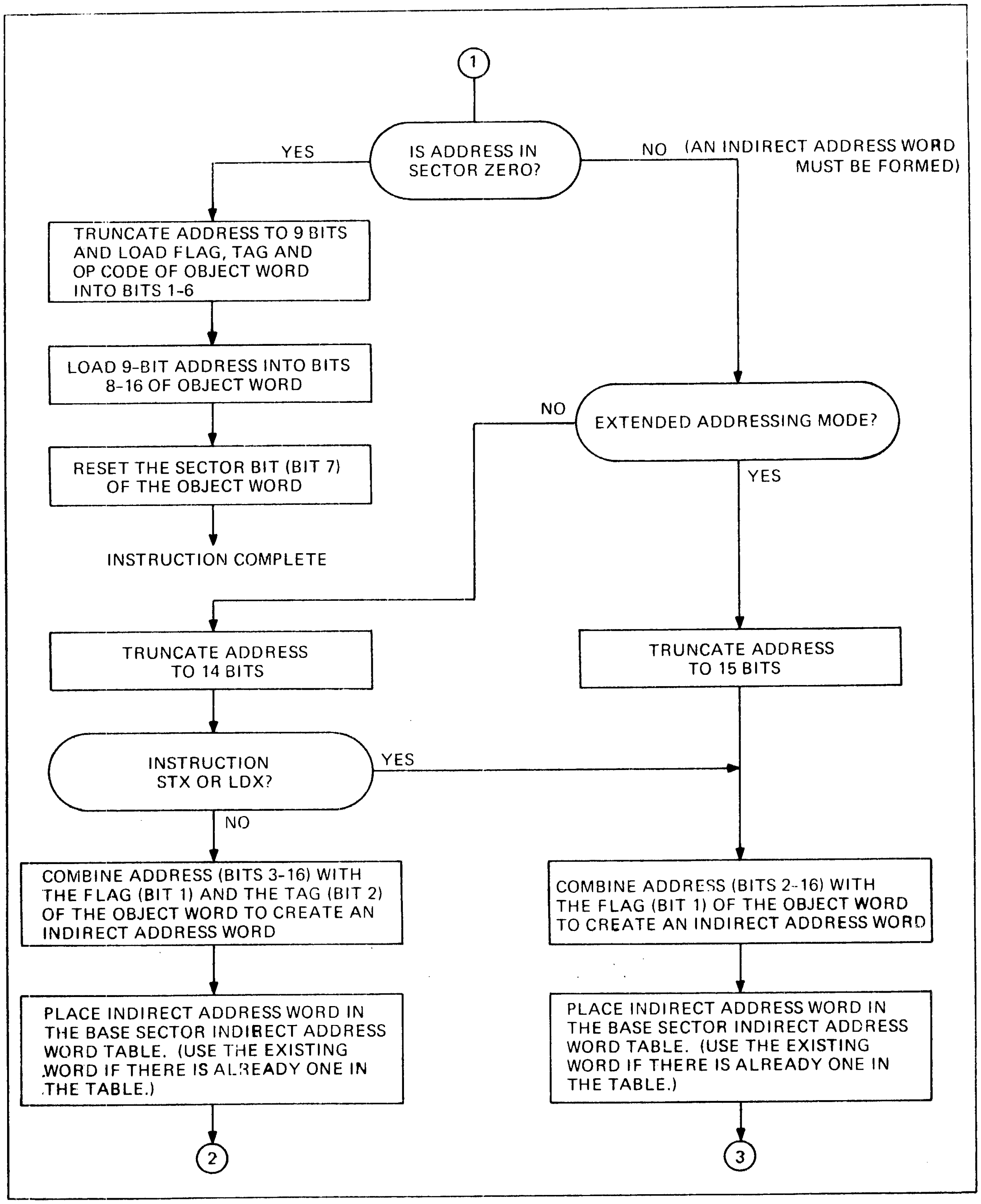

1
IS ADDRESS IN SECTOR ZERO?
YES
NO (AN INDIRECT ADDRESS WORD MUST BE FORMED)
TRUNCATE ADDRESS TO 9 BITS AND LOAD FLAG, TAG AND OP CODE OF OBJECT WORD INTO BITS 1-6
LOAD 9-BIT ADDRESS INTO BITS 8-16 OF OBJECT WORD
RESET THE SECTOR BIT (BIT 7) OF THE OBJECT WORD
INSTRUCTION COMPLETE
EXTENDED ADDRESSING MODE?
NO
YES
TRUNCATE ADDRESS TO 14 BITS
TRUNCATE ADDRESS TO 15 BITS
INSTRUCTION STX OR LDX?
YES
NO
COMBINE ADDRESS (BITS 3-16) WITH THE FLAG (BIT 1) AND THE TAG (BIT 2) OF THE OBJECT WORD TO CREATE AN INDIRECT ADDRESS WORD
COMBINE ADDRESS (BITS 2-16) WITH THE FLAG (BIT 1) OF THE OBJECT WORD TO CREATE AN INDIRECT ADDRESS WORD
PLACE INDIRECT ADDRESS WORD IN THE BASE SECTOR INDIRECT ADDRESS WORD TABLE. (USE THE EXISTING WORD IF THERE IS ALREADY ONE IN THE TABLE.)
PLACE INDIRECT ADDRESS WORD IN THE BASE SECTOR INDIRECT ADDRESS WORD TABLE. (USE THE EXISTING WORD IF THERE IS ALREADY ONE IN THE TABLE.)
2
3

Figure 3-1. Desectorized Program Loading (Part 2 of 3)

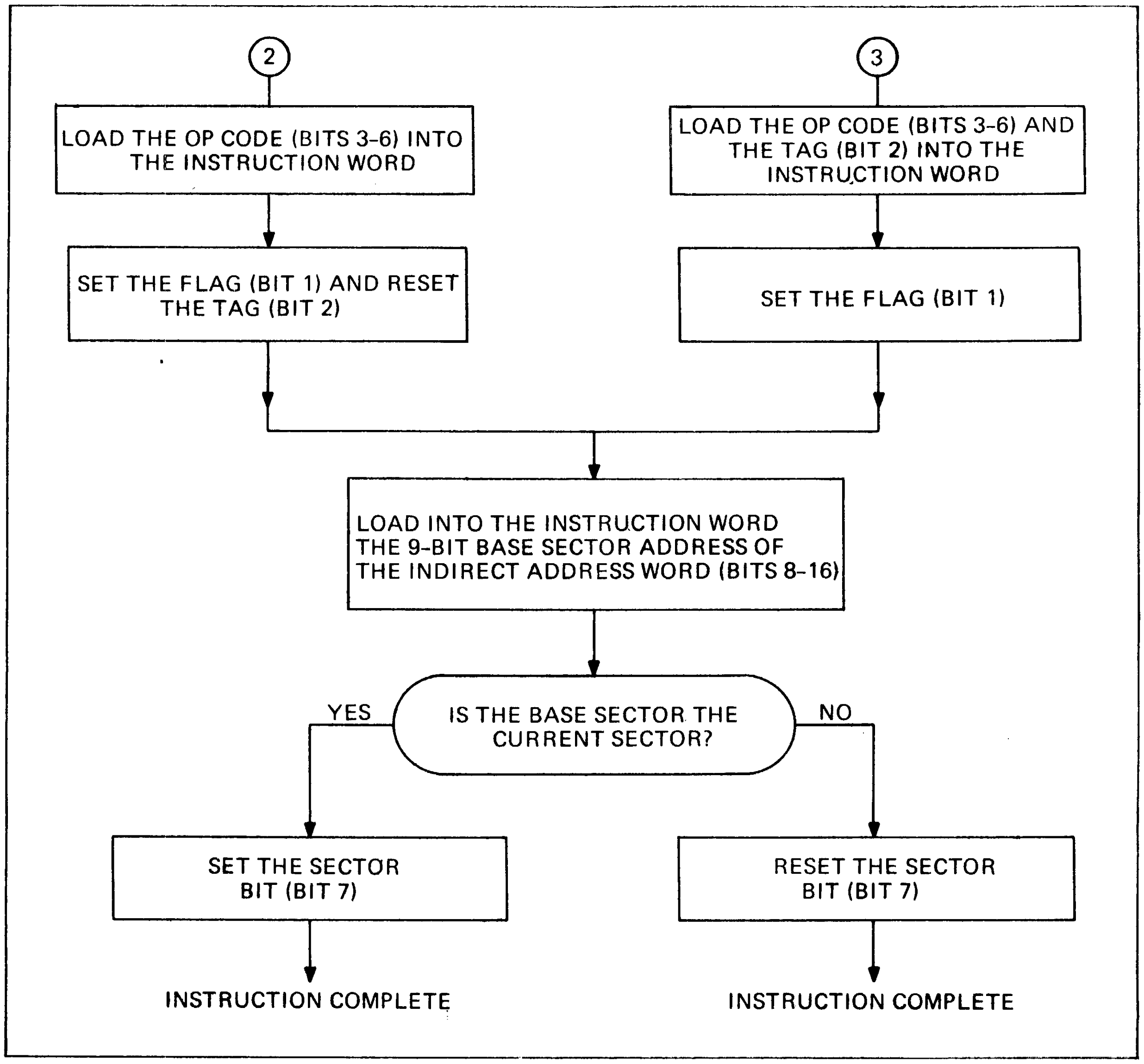

2
3
LOAD THE OP CODE (BITS 3-6) INTO THE INSTRUCTION WORD
LOAD THE OP CODE (BITS 3-6) AND THE TAG (BIT 2) INTO THE INSTRUCTION WORD
SET THE FLAG (BIT 1) AND RESET THE TAG (BIT 2)
SET THE FLAG (BIT 1)
LOAD INTO THE INSTRUCTION WORD THE 9-BIT BASE SECTOR ADDRESS OF THE INDIRECT ADDRESS WORD (BITS 8-16)
IS THE BASE SECTOR THE CURRENT SECTOR?
YES
NO
SET THE SECTOR BIT (BIT 7)
RESET THE SECTOR BIT (BIT 7)
INSTRUCTION COMPLETE
INSTRUCTION COMPLETE

Figure 3-1. Desectorized Program Loading (Part 3 of 3)

OPERATING INSTRUCTIONS FOR SSUP

This information is contained in the Series 16 O16-XREF, SSUP, and MAC Source Language Processors manual.

OPERATING PROCEDURES FOR DAP-16

This information is contained in the Series 16 DAP-16 and DAP-16 Mod 2 Assembly Language manual, Sections V and VI.

COMPILATION OF FORTRAN IV SOURCE TAPES

Without Disk or Drum

The following step-by-step procedures are required to compile FORTRAN IV without disk for systems with 8K and above. A FORTRAN IV compiler system tape is necessary and is supplied to 8K users in the standard software package. Over 8K users must generate a system tape.

a. Load FORTRAN IV compiler systems tape.

b. Select a bit pattern from Table 3-4 to designate I/O devices to be used during compilation. Load this into register A.

c. Insert FORTRAN IV source tape into appropriate input device.

d. Set P = 1000_8.

e. Depress START pushbutton. When the compilation is completed, an end-of-job message is printed.

The FORTRAN IV compiler indicates coding, typing, or printing errors by printing error flags on the listing following the statement containing the error. Some errors are recoverable and do not interfere with the compilation. Others, however, halt the computer and must be corrected before continuing. (See Section V, Error Messages.)

Table 3-4 indicates the required device assignments. Sense switch 3 can be set at any time during compilation to halt the computer and allow new device assignments tp be entered.

Table 3-4. FORTRAN IV Compilation Time Options

Bit Selection	Meaning
1	If set, certain error checks are not made. This option is used when compiling those parts of the library written in FORTRAN; it permits the library to use the compiler logic to generate special coding.
2	If set, the symbolic listing is expanded to include side-by-side octal information.
3	If set, the symbolic listing is inhibited (as long as bit 2 is not set).
4	Unused.
5	If set, the computer will halt before and after transmitting each block of object text. This pause allows for turning the ASR-33 punch on and off to avoid punching of extraneous data on object tape.

Table 3-4 (cont). FORTRAN IV Compilation Time Options

Bit Selection	Meaning
6	If set, the object coding being generated will include trace coupling information regardless of any TRACE statements within the program. (Operator override).
7	Unused.
8 - 10	Input device selection: 0 = Reprocess last record. 1 = ASR-33/35 keyboard (or its paper tape reader) 2 = High-speed paper tape 3 = Card reader 4 = Line printer (not used for input) 5 = Magnetic tape unit 2 (logical) 6 = Disk (via DOP) 7 = Spare
11 - 13	Binary output selection: 0 = Suppress binary output. 1 = ASR-33/35 binary output 2 = High-speed paper tape 3 = Card punch 4 = Line printer (not used for output) 5 = Magnetic tape unit 4 (logical) 6 = Disk (via DOP) 7 = Spare
14 - 16	Symbolic listing selection: 0 = Suppress all symbolic listings. 1 = ASR-33/35 keyboard 2 = High-speed paper tape (not used for listing) 3 = Card reader/punch (not used for listing) 4 = Line printer 5 = Magnetic tape unit 3 (logical) 6 = Disk (via DOP) 7 = Spare

With Disk or Drum

The following describes the step-by-step procedures required to compile FORTRAN IV programs with systems having 12K or more memory and disk or drum.

A FORTRAN IV compiler system tape is necessary and must be generated for systems with 12K and above. The system must contain F4-IOS-0 (Rev. B and later) for disk and DOP (disk operating program). The disk must have been previously formatted by M$FT if moving-head disk or by DOP if fixed-head disk. Additionally, the file name(s) must have been previously entered in the disk directory, and DOP must be initialized.

a. Load FORTRAN IV compiler systems tape.

b. Load selected I/O bit pattern into A-register. (See Table 3-4.)

c. Set sense switches to desired settings.

d. Set P-register = 1000_8. Depress START pushbutton. If any disk I/O is requested, DOP requests by means of the keyboard the source, object, and/or list name. The operator types the appropriate names on the keyboard and, if the names are present in the DOP directory, compilation begins. When compilation is completed, an end-of-job message is printed.

DISK OPERATING PROGRAM (DOP) OPERATING INSTRUCTIONS

DOP provides the user with a disk operating system containing files of information of varying length on the disks. DOP maintains a directory of the information and performs the commands of initialization, update directory, print directory, and load. DOP will operate with the following configurations:

Type 4400 Disk Control Unit Type 4420, 4421, 4422, 4423 Disk Drive Type 4406 DMC Subchannel Type 4408 DMA Subchannel	Fixed-Head Disk Subsystem
Type 4600 Disk Control Unit Type 4606 DMC Subchannel for 4600 Type 4608 DMA Subchannel for 4600 Type 4620 100 Tracks/Surface Disk Drive Type 4621 200 Tracks/Surface Disk Drive	Removable Disk Subsystem (Moving-Head Disk)

DOP must be resident in the same 16K of memory as the user's program for normal mode operation. DOP uses about six sectors of storage. Also, DOP uses sector zero location 770_8-773_8 for its software interface.

DOP requires the use of either D$IO (fixed-head input/output driver, 516) or M$IO (moving-head input/output driver, 316). DOP contains references to three external parameters, which are defined below:

PART	0 - Moving-Head Disk File 1 - Fixed-Head Disk File
PAR 1	0 - DMA 1 - DMC
PAR 2	Subchannel DMA 1-4 Subchannel DMC 1-16

These parameters are selected by the user, depending on the configuration. The parameters are satisfied by creation of a subroutine which is loaded after the disk driver. The subroutine format is:

```
SUBR PART
SUB PAR 1
SUB PAR 2
REL
PART OCT X      *  Type of Disk — 0 = Moving Head; 1 = Fixed head
PAR 1 OCT X     *  Type of channel — 0 = DMA; 1 = DMC
PAR 2 OCT X     *  Subchannel number — DMA, 1-4; DMC, 1-16
         END
```

To satisfy the calls to the alternate driver entries, the user executes one of the two dummy routines described below:

Using Moving-Head Disk (Removable Disk Subsystem)

```
     SUBR  D$IO, DUM
     SUBR  C$DI, DUM
     SUBR  C$FD, DUM
     REL
DUM  DAC   **
     JMP*  DUM
     END
```

Using Fixed-Head Disk

```
     SUBR  M$IO, DUM
     SUBR  M$IN, DUM
     REL
DUM  DAC   **
     JMP*  DUM
     END
```

DOP utilizes four different file types, plus a directory. The characteristics of each file are fixed within the DOP system. The user has control over only the size of each file and its location on the disk. For the fixed-head disk within the DOP system:

a. Records are contiguous within each file.

b. Record numbers are contiguous within each track, starting with record number 1 and increasing by 1, until the maximum number of records per track for that record size.

c. The directory occupies tracks 3, 4, and 5.

For the moving-head disk, DOP assumes:

a. Records and associated record numbers are contiguous within each file.

b. Record numbers within each file start with 1 and increase by 1, until the end of the file.

c. The directory occupies track 0, heads 3, 4, and 5.

d. The disk has been preformatted using M$FT.

REMOVABLE DISK SUBSYSTEM (MOVING-HEAD DISK FILES) OPERATING INSTRUCTIONS

Operation of the moving-head disk file on a Model 316/516 requires that the disk file be configured and formatted. M$FT (180666000) is the stand alone off-line program used to accomplish the configuring and formatting necessary. M$FT is designed for use on a Model 316/516 with the following equipment:

Type 4600 — disk control unit for up to 4 disk drives

Type 4604 — DMC subchannel for 4600

Type 4608 — DMA subchannel for 4600

Type 4620 — 100 track/surface disk drive (9433A)

Type 4621 — 200 track/surface disk drive (9433)

M$FT cannot be used for disk drives connected to I/O bus. The procedure for using M$FT is explained in the Type 4600 Moving-Head Disk File Programming Manual, Order Number BY05 (Document Number 70130072398).

Moving-Head Disk Files with DOP

DOP requires five different file formats:

a.	Directory	— Thirty-six 246-word records
b.	File 1	— Core image
c.	File 2	— Source
d.	File 3	— Object
e.	File 4	— Listing

DIRECTORY

The DOP Directory must be formatted with 246-word records (6 records per track) on cylinder 0, heads 0 to 5. The following command should be used:

```
210003602460000000000000100000100
```

FILE 1, CORE IMAGE

The core image file must immediately follow the directory. It must also be formatted in 246-word records. The user chooses the number of core image records to be formatted. The following command should be used, with XXXXX replaced with the decimal number of records desired:

```
21XXXXX02460000060000000100000100
```

FILE 2, SOURCE

The source image file is formatted in 40-word records (30 records per track). The user chooses the number of records to be formatted and the starting track. The following command should be used, with XXXXX replaced with the decimal number of records desired, YYY replaced with the starting cylinder decimal number, and X replaced with the starting head number:

```
21XXXXX00400YYYZ000000000100000100
```

FILES 3 AND 4, OBJECT AND LISTING

These two files are both formatted in 60-word records (22 records per track). The user chooses the number of records to be formatted and the starting track for each. The following command should be used once for File 3 and once for File 4, with XXXXX replaced with the decimal number of records desired, YYY replaced with the starting cylinder decimal number, and Z replaced with the starting head number:

21XXXXX00600YYYZ000000000100000100

DOP Operating Instructions

DOP consists of four main commands:

a. I, Initialization

b. U, Update directory

c. P, Print directory

d. L, Load

INITIALIZATION OF FIXED-HEAD DISK

When DOP is executed, the following message is printed on the ASR:

SELECT I, U, P, L

The user types in I (CR). DOP enters into the initialization routine which allows for formatting the directory. Before formatting the directory, DOP allows an option to clear the directory. This option is signalled when the following message is printed on the ASR:

CDIC

The user may bypass clearing the directory by typing zero (0) (CR) on the ASR. Any nonzero character and a (CR) clears the directory. After the option is exercised, DOP formats the directory with records of 246 words. The initialization routine then proceeds to the section, allowing the user to format the disk according to one of four types:

a.	File 1	Core image, record length of 246 words
b.	File 2	Source, record length of 40 words
c.	File 3	Object, record length of 60 words
d.	File 4	Listing, record length of 60 words

At the beginning of this formatting section, the following message is printed on the ASR:

FILE X
DISC ADDRESS T (CR) N (CR)

where X is 1, 2, 3, or 4. The option to bypass formatting of the file is again present. The user may bypass formatting the file by typing in zero (0) (CR) at this point.

The starting track number is typed in decimal. The program does an ASR read and returns control to the ASR. The user types in the number of tracks to be formatted in decimal. The file is formatted beginning at the starting track and continues for the number of tracks selected. The preceding is repeated for each file, and the program returns to print the following message upon completion:

SELECT I, U, P, L

INITIALIZATION OF MOVING-HEAD DISK

DOP assumes that the disk has been preformatted, which basically accomplishes much of the initialization portion of the program. However, the initialization command should be utilized after loading DOP to configure the disk and optionally clear the directory.

UPDATE DIRECTORY COMMAND

The program prints the message:

SELECT I, U, P, L

The user types U (CR). The update command is selected and the following message is printed on the ASR:

C, D, A (CR)

where: C = Change an entry

D = Delete an entry

A = Add an entry

If the user types in an A, the program searches for the first blank name in the directory and proceeds. The following message is printed on ASR:

PROGRAM NAME =

The user enters the name of the program (up to six ASCII characters) followed by a carriage return. The program searches for the name entered. If program is not found, the program returns to Select I, U, P, L, printout. If the name is found and a D (delete) was requested, the name is filled with blanks and returned to the directory. If A or C was requested, the program types the message:

DATE = XX/XX/XX

The user enters the present date, and the following message is printed on the ASR:

FILE 1, 2, 3, 4 (CR)

The user types 1, 2, 3, or 4 followed by a carriage return. The program checks the type or file. For a file 1 input, the following message is printed on the ASR:

BMA =

The user enters the beginning memory address in octal followed by a carriage return. The program prints the following message:

EMA =

The user enters the ending memory address in octal followed by a carriage return. The program prints the following message on the ASR:

BTRK =

The user enters the beginning track number in decimal followed by a carriage return. If moving-head disk, the program prints the following message:

SHED =

The user enters the starting head number in decimal followed by a carriage return. The program prints the following message:

BREC =

The user enters the beginning record number in decimal followed by a carriage return. The program prints the following message:

JADD =

The user enters the proper JUMP address in octal followed by a carriage return. The record is then restored in the directory and the program returns to Select I, U, P, L routine.

LOAD COMMAND

The program prints the message:

SELECT I, U, P, L

To load a program from disk, an L followed by a carriage return is typed by the operator. The program prints the following:

SELECT R, W, OTHER (CR)

The name of the program is typed followed by a carriage return. The program searches the directory, finds the name, and if it is a file 1 type, loads the program from disk to memory and prints the following:

READY

The program halts. When the START pushbutton is depressed, the program executes an indirect JUMP to the address furnished by the user. If the name was not found, the program returns to the Select I, U, P, L routine.

WRITING IN FILE 1

The program prints:

SELECT I, U, P, L

An L (CR) is typed by the operator. The load command is selected and the program prints:

SELECT R, W, OTHER (CR)

A W is typed followed by a carriage return. The program prints:

PROGRAM NAME =

The name of the program to be transferred to the disk is typed. The program searches the directory. When the program is found, DOP checks that it is file 1 and writes the program on the disk beginning at the starting track and record numbers as they appear in the directory. When all records are transferred, the ending track and ending record numbers are entered in the directory. The program prints:

OK

and returns to the Select I, U, P, L routine.

TRANSFERRING SOURCE OR OBJECT TAPES TO DISK (ASR OR HIGH-SPEED READER)

The program prints:

SELECT I, U, P, L

An L followed by a carriage return is typed by the operator. The program selects the load command and prints:

SELECT R, W, OTHER (CR)

An R is typed followed by a carriage return. The program selects the read command and prints:

SOURCE S, OBJECT O (CR)

An S or an O is typed. The program sets up for a source or object read from paper tape and prints:

SOURCE NAME =

or

OBJECT NAME =

The program name is typed. The program searches the directory for the program name. When the name is found, the program prints:

ASR-0, PT 1 CR)

Depending on the input device, the operator should type an A, O, or 1. The program reads the paper tape a record at a time and transfers the record to file 2 or file 3. When an end-of-tape indicator is encountered (a $ in column 1 of source tape, a 203 end-of-tape indicator for object tapes), the last record is written on the disk and the directory updated with the last track and record numbers. The object input is checked for checksum errors and/or block errors. DOP has the capability of evaluating the object tape. If it is bad, it gives the following error messages:

CK
BL

The program returns to the Select I, U, P, L routine.

PRINTING DIRECTORY INFORMATION

The DOP print command lists information for all programs in the directory. The information, when printed, is preceded by a header. The sequence of operations is:

SELECT I, U, P, L

A P is entered followed by a carriage return. The program prints the header in the following format:

```
NAME DATE BMA EMA BTRK BRA ETRK ERA FT JADD
```

where:	NAME	= Program name
	DATE	= Date last changed
	BMA	= Beginning memory address
	EMA	= Ending memory address
	BTRK	= Beginning track address, also includes head number if moving-head disk
	BRA	= Beginning record address
	ETRK	= Ending track address, also includes head number if moving-head disk
	ERA	= Ending record address
	FT	= File type 1, 2, 3, or 4
	JADD	= Jump address

ERRORS

An error halt at relative location 3702 (octal) occurs for disk transmission errors. The A-register contains the error types. Bit 1 set indicates a setup error; bit 2 set indicates a data error. While inputting information through the keyboard, if an error occurs, a left arrow may be keyed in and the correct information entered.

DOP-4700 OPERATING INSTRUCTIONS

DOP-4700 is a disk operating program that recognizes four forms of user-allocated files, provides keyboard interface, and permits user-program control. To function with its software interface (which supports FORTRAN, DAP, LDR-APM, and user programs), DOP-4700 must reside in the same 16K of memory as these programs.

Since DOP-4700 and its I/O drivers require about five sectors of memory, compilation with disk can be performed on a configuration of 12K core, and assembly with disk can be performed with 8K core. Although multiunit access is available, DOP-4700 controls only one disk drive at a time.

Formatting the Disk with M$FT

Before DOP-4700 can be used, the user must format a disk pack using M$FT, Document Number 70181570000. The user formats the directory and files 1 - 4 using the type 1 format command (see M$FT). Instructions for using M$FT are contained in the Moving-Head Disk Options 4623, 4651, and 4720, Programmers' Reference Manual, Order Number CA38 (Document Number 70130072522), which also includes detailed instructions for the standard library driver M$IO. In the formats outlined below, the first record address, the algebraic address increment, and the interface factor all equal one. The size of each file type area (except the directory) must be determined by user analysis.

Directory:	Format 6 records/track and 246 words/record.
	(Cylinder 0, heads 0-2 for 10- and 20-surface M.H.D.)
	(Cylinder 0, heads 0-1, cylinder 1, head 0 for 2-surface M.H.D.)
File 1: (Core image)	Format 6 records/track and 246 words/record.
	(Start at cylinder 0, head 3 for 10- and 20-surface M.H.D.)
	(Start at cylinder 1, head 1 for 2-surface M.H.D.)

For the 20-surface M.H.D., the following record address sequence is generated.

```
Track 3 (cylinder 0, head 3)   1 - 6
Track 4 (cylinder 0, head 4)   7 - 12
  *
  *
Track 19 (cylinder 0, head 19) 97 - 102
Track 20 (cylinder 1, head 0) 103 - 108
  *
  *
Track 80 (cylinder 4, head 0) 463 - 468 (if used)
  *
  *
Track 86 (cylinder 4, head 6) 499 - 504 (if used)
 etc.
```

NOTE: For DOP-4700, record numbers are in decimal (1 to N).

For M$FT-4700, record numbers are in octal (1 to N).

File 2: (Source)	Format 30 records/track and 40 words/record.
File 3: (Object)	Format 22 records/track and 60 words/record.
File 4: (Listing)	Format 22 records/track and 60 words/record.

The user must maintain a record of the pack format in order to make continuing use of DOP-4700.

Disk Allocation

DOP-4700 uses four file types, plus a directory. The characteristics of each file are fixed within the DOP system. The user has control only over the size of each file and its physical position on disk. (Files need not be contiguous.) DOP-4700 positions the heads to cylinder zero at the outset, then operates on the following premises.

a. Records and their associated record numbers are contiguous within each file.

b. Record numbers within each file type start with 1 and increase by 1 until the end of the file type.

c. The directory occupies cylinder 0, heads 0, 1, and 2 for 10- and 20-surface-disks, and cylinder 0, heads 0 and 1, and cylinder 1, head 0 for 2-surface disks.

d. The disk has been formatted using M$FT.

Loading DOP-4700

DOP-4700 is a relocatable object program, loadable by the standard 316/516 linking loaders. DOP-4700 forces the sector above it to preserve the indirect links generated and to avoid driver-generated indirect links. DOP-4700 uses sector zero locations '770 to '773 for its software interface. DOP-4700 also requires the standard disk driver M$IO and the I/O drivers for those devices from which the user plans to input his source and object tapes (or cards). The loading sequence is outlined below.

a. Load the DOP-4700 object tape or deck, starting at sector 'XX000.

b. Load M$IO.

c. Load any of the following drivers that are necessary:

   1. I$AA and I$AB (for source and object input through the ASR keyboard)

   2. I$PA and I$PB (for source and object input through the high-speed reader)

   3. I$CA and I$CB (for source and object input through cards). An I$-- routine that would cross a sector boundary in back-to-back loading should be loaded at the next sector boundary to avoid generating indirect links.

   NOTE: DOP-4700 is programmed to avoid sector 0 references.

d. Transfer to location 'XX000 to execute DOP-4700.

DOP-4700 Commands

DOP-4700 has four main commands, each of which has subcommands associated with it. The main commands are listed below and described in the following paragraphs.

I = Initialize

P = Print directory

U = Update directory

L = Load

INITIALIZE COMMAND

Prior to initialization, DOP-4700 configures the disk according to user replies to the following four messages.

```
INITIALIZE DISK - DMC (0) OR IMA (1)   ? :
```

Type 0 for DMC or 1 for DMA.

```
CHANNEL (1-16)   ? :
```

Type the number of the channel to be used to transfer the data.

```
SURFACES/DISK (2, 10 OR 20)   ? :
```

Type 2, 10, or 20 to specify the number of surfaces per disk.

```
NO. OF SPINDLES (1-8)   ? :
```

Type a number from 1 through 8 to specify the number of spindles.

When the disk has been configured, this message prints out.

```
I,U,L OR P   ?:
```

To initialize (clear) the directory, type Ic/r. DOP-4700 will type the following message to ascertain if the directory is to be cleared.

```
CLEAR DIRECTORY (C)   ?:
```

Type C to clear (set to ASCII blanks) the directory. The directory will be saved if any other character is typed.

Whenever the I,U,L OR P ?: message is printed, the software interface locations '770 - '773 are reestablished. The Initialize command need not be used again except to seek zero again or to reestablish the directory.

PRINT DIRECTORY COMMAND

The operator types Pc/r in response to the I,U,L OR P ?: message to select the Print Directory command. In response, the full directory is printed out, giving the name and other information about each file on the disk. A sample printout is given below, followed by an explanation of its contents.

```
NAME     DATE      BMA     EMA     BTRK    BRA     ETRK    ERA     FT   JADD
DOP-47   10/29/71  32000   37204   00003   00001   00004   00011   1    32000
M$FT     10/21/71  00100   10550   00004   00012   00007   00029   1    01000
SHORTE   11/2/71                   00834   00760   00838   00843   4    00000
OBJ      11/5/71                   00603   00073   00603   00087   3    00000
```

NAME	=	Program name
DATE	=	Date last changed
BMA	=	Beginning memory address
EMA	=	Ending memory address
BTRK	=	Beginning track address (decimal)
BRA	=	Beginning record address (decimal)
ETRK	=	Ending track address (decimal)
ERA	=	Ending record address (decimal; last record used for file type 1; next record available for file types 2, 3, and 4)
FT	=	File type (1, 2, 3, or 4)
JADD	=	Jump address

UPDATE DIRECTORY COMMAND

It is the user's responsibility to update information in the directory (including space allocation), as the directory is used to locate the position on disk where information is to be stored or read. The operator types Uc/r in response to the I, U, L OR P ? : message to select the Update Directory command. DOP-4700 responds with this message.

```
C, D OR A  ?:
```

Type C (change an entry), D (delete an entry), or A (add an entry), followed by a carriage return (c/r) to specify the update function. The program determines which entry has been made and sets the proper conditions to perform the function. If an A was entered, the program locates the first blank name in the directory, then types the following message.

```
PROGRAM NAME  ?:
```

Enter the program name (up to six ASCII characters) and type c/r. The program searches the directory for the specified name. If the name is not found (except for the ADD function, the I, U, L OR P ? : message prints out again. If the name is found and the Delete function was specified, the name in the directory is filled with ASCII blanks. If the name is found and the Change function was specified, or if the Add function was specified, the following message prints out on the ASR.

```
DATE  ?:
```

Enter the date as an eight-character message, which is placed in the directory before the next query prints out.

```
FILE 1, 2, 3, OR 4  ?:
```

Type 1, 2, 3, or 4, followed by c/r. For file types 2, 3, and 4, the program bypasses the next two queries, which do print out for type 1 files.

```
BMA  ?:
```

Enter the beginning memory address in octal, followed by c/r. The address must be higher than any DMC dedicated location in use. (Normally the DMC high is '20 or '21.) DOP-4700 places the BMA in the directory and requests the ending memory address.

```
EMA  ?:
```

Enter the ending memory address in octal, followed by c/r. The BMA and EMA (which are inclusive for type 1 (files) must not enclose any part of DOP-4700, except when DOP-4700 writes itself on the disk. (To fetch DOP-4700, a bootstrap routine is needed.)

The following messages print out for all file types.

```
BTRK  ?:
```

Enter the beginning track number in decimal, followed by c/r. Allocation of space on the disk is determined by comparing the available disk space and the amount of data to be stored.

BREC ?:

Enter the beginning record number in decimal, followed by c/r.

JADD ?:

JADD is requested by type 1 files only. Enter the jump address (if any) in octal, followed by c/r.

LOAD COMMAND

The Load command, in conjunction with the Read and Write subcommands, can perform the following functions.

a. Transfer to core any program (core image) stored as a type 1 file, including DAP, FORTRAN, and LDR-APM. Transfers to or from type 1 files can be of variable length (not restricted to 246-word blocks).

b. Transfer any program in core to a type 1 file. (See Write command below.

c. Transfer source or object data from tape or cards to disk (source to type 2 file, or object to type 3 file). (See Read command below.)

The user types Lc/r in response to the I, U, L OR ?: message to select the Load command. DOP-4700 responds with this message.

R, W OR PGM NAME ?:

To load a program from disk (file type 1), type the name of the program, followed by c/r. The program searches the directory for the name, and if it is not found, the I, U, L OR ?: message prints out again. If the name is not found, and is a type 1 file, the file is loaded from disk into memory. At completion, this message prints out.

READY

When the START button is depressed, the program jumps to the address furnished by the user.

Write Command

To write into an established type 1 file, respond to the I, U, L OR P ?: message with Lc/r and to the R, W OR PGM NAME ?: message with Wc/r. DOP-4700 responds with this query.

PROGRAM NAME ?:

Type the name of the program to be transferred from memory to disk. The program searches the directory for the name, and if it is not found, the I, U, L OR P ?: message prints out again.

If the name is found in the directory and is a type 1 file, the program is written on disk beginning at the starting track and record number as they appear in the directory. When all records from the starting to the ending memory addresses have been transferred, the ending track and ending record numbers are entered into the directory. The program types "OK" and the I, U, L OR P ?: message prints out.

Read Command

The Read command allows the user to transfer source (file type 2) and object (file type 3) data from tape or cards to the disk file using the ASR, high-speed reader, card reader, or card reader/punch. The user types Lc/r in response to the I, U, L OR P ?: message to select the Load command. This message prints out.

```
R, W OR PGM NAME ?:
```

Type Rc/r to select the Read command. This message prints out.

```
SOURCE (S) OR OBJECT (0) ?:
```

Type Sc/r or Oc/r. The program sets up for the specified type of read and prints one of the following messages.

```
SOURCE NAME ?:
     or
OBJECT NAME ?:
```

Type the name of the file to be transferred. The program searches the directory for the name, and if it is not found, the I, U, L OR P ?: message prints out again. If the name is found, DOP-4700 prints this message.

```
ASR (1), H S RDR (2) OR CARD (3) ?:
```

Type 1, 2, or 3 to specify the input device. The program reads one record at a time and transfers the records to the file. After an end-of-message (end-of-file) indicator is reached (see File Termination below), the last record is written on disk and the program updates the directory with the last track and record numbers. When the function is complete, the I, U, L OR P ?: message prints out again.

An object file can consist of several object subroutines, e.g., a FORTRAN library paper tape, which usually is produced by a batch assembly or compilation. Since object input is checked for checksum errors, a CK error message may print out. To ignore the message, press START.

File Termination

Termination procedures for files 2, 3, and 4 are given below. The following notes should be kept in mind.

NOTES:
1. Directory updating never occurs during reading of disk files.
2. The ending record address (ERA) in the directory for file types 2, 3, and 4 indicates the next available record; the ERA for file type 1 is the last record used.

TERMINATING SOURCE (TYPE 2 FILES)

When a source file is being written, certain conditions cause DOP-4700 to do the following:

a. Write on the disk a record whose first character is $.

b. Update the directory (ETRK and ERA).

c. Print the message: I, U, L OR P ? :

The above results are produced under one of the following conditions.

a. During a write using entry '771 or the Load and Read commands, DOP-4700 reads a record whose first character is $. (The $-record read is the $-record written.)

b. During a write using Load and Read commands and paper tape source, DOP-4700 reads an EOM record (ETX ('203 = CTRL, C) X-OFF Δ RUB OUT). A $0 is written.

c. During a write using Load and Read commands and card source, DOP-4700 reads a card that contains an EOF punch (end-of-file, 11-8-6). A $0 is written.

TERMINATING OBJECT (TYPE 3 FILES)

When an object file is being written, certain conditions cause DOP-4700 to do the following:

a. Write on the disk a record whose first two words each equal '300 (EOJ).

b. Update the directory (ETRK and ERA).

c. Print the message: I, U, L OR P ? : (for Load and Read commands), or return to the calling program (for the '772 entry).

The above results are produced under one of the following conditions.

a. During a write using entry '772 or the Load and Read commands, DOP-4700 reads a record whose first word equals '300 (EOJ).

b. During a write using Load and Read commands and object paper tape, DOP-4700 reads an EOM record (ETX ('203 = CTRL, C) X-OFF Δ RUB OUT).

TERMINATING LISTING (TYPE 4 FILES)

For assembler- or compiler-generated listings, the following actions occur after DAP or FRTN writes the last listing record.

a. The last object record(s) is (are) written.

b. The object file entry in the directory is updated.

c. The listing file entry in the directory is updated.

d. DOP returns to DAP or FRTN.

Since a listing file update is dependent upon an object file being written on the disk, the user must write the object file even if he wishes only to list. (The object file can be deleted later.)

For user-program-generated listings, if the user wishes to write a listing file and have the directory updated, he must do the following:

a. Set bit 2 of the A register (in addition to bit 16, described below under Software Interface with DOP-4700) before going to the DOPI entry.

b. Have $0 as the first two characters of his last print record.

Software Interface with DOP-4700

To take advantage of DOP, FRTN and DAP have been provided with special IOS routines and LDR-APM with an internal IOS structure, all of which are linked to DOP-4700 through absolute locations '770 through '773 of sector 0 to provide disk I/O capability. User programs also can use this software interface. The four entries are described below.

DOPI entry	('770)	is used to initialize access to files.
DOPS entry	('771)	is used to read or write a source record to or from the disk.
DOPO entry	('772)	is used to read or write an object record to or from the disk.
DOPL entry	('773)	is used to read or write a list record to or from the disk.

The DOPI entry is used to initialize (OPEN) access to file types 2 (source), 3 (object), and 4 (listing). Not more than one of each of these types of file can be open at one time. Once a source, object, or listing file has been opened, it is available for record-by-record access.

The DOPI entry is available for use by the Read command in addition to DAP, FRTN, LDR-APM, and user programs. The calling sequence follows.

```
LDA   < bit configuration>
JST*    '770
< return>
```

If the first bit position of the A register is not set (equals zero), DOPI determines the direction of each file by the bit(s) that is (are) set, with the following meanings.

Bit 14 set — Write source on disk (file type 2)

Bit 15 set — Read object from disk (file type 3)

Bit 16 set — Read listing from disk (file type 4)

If the first bit position of the A register is set (equals one), DOPI determines the direction of each file by the bit(s) that is (are) set, with the following meanings.

Bit 14 set — Write source on disk (file type 2)

Bit 15 set — Read object from disk (file type 3)

Bit 16 set — Read listing from disk (file type 4)

After the proper initialization call to DOPI, the user can initiate the calling sequence to have a record transferred.

```
JST*   '771 (or '772 or '773)
DAC    < buffer address>
< return >
```

DOP-4700 Errors

ERROR HALTS

An error halt occurs if there has been a disk transmission error. At relative locations '154 and '1631, the A register contains the error type; press START to ignore the error. At relative location '3405, the A register equals '52525 and the B register contains the error type; press START to print the I, U, L OR P   ?: message. See M$IO (Document Number 70181509000) for error types.

KEYBOARD ERRORS

If a user error occurs while DOP information is being input through the keyboard, the user can type a left arrow to replace the incorrect character with the correct one, or he can type a commercial at-sign @ to replace an incorrect line of data.

Sample DOP-4700 Programs

Listings of several sample user programs using DOP-4700 appear on the following pages.

```
                     * ROOTSTRAP (FOR DOP-4700, 20-SURFACE M,H,D,)
                     * NOT RE-EXECUTABLE (MUST RELOAD)
                     *
                     *  THIS PROGRAM ASSUMES THAT DOP AND ITS DRIVERS HAVE BEEN WRITTEN (VIA
                     *  DOP AND DMC CHANNEL 1) ON UNIT 0, CYLINDER 0, HEADS 3 AND 4 (RECORDS
                     *  1-11).  UPON SUCCESSFUL COMPLETION, CONTROL TRANSFERS TO DOP,
                     *  SIGNIFIED BY THE PRINTING OF THE 'I,U,L OR P' MESSAGE.
                     *
                          REL
                     *
00000    14 0025          OCP   '0025           SEEK TRACK ZERO
00001    140040           CRA                   SETUP WORD FOR UNIT 0 =0
00002    74 0025          OTA   '0025           OUTPUT SETUP WORD
00003    0 01 00002       JMP   *-1
00004    34 0425          SKS   '0425           SKIP IF M,H,D, NOT BUSY
00005    0 01 00004       JMP   *-1
00006    34 0325          SKS   '0325           SKIP IF NO ERROR
00007    000000           HLT                   ERROR HALT
                     *
00010    0 02 00061 NEXT  LDA   SADR
00011    140500           SSM                   SET BIT 1 FOR DMC INPUT
00012   *0 04 00055       STA*  SDMC
00013    0 02 00062       LDA   EADR
00014   *0 04 00056       STA*  EDMC
00015    14 0625          OCP   '0625           READ A RECORD
00016    0 02 00057       LDA   SUP1
00017    74 0025          OTA   '0025           OUTPUT FIRST SETUP WORD
00020    0 01 00017       JMP   *-1
00021    0 02 00060       LDA   SUP2            OUTPUT SECOND SETUP WORD
00022    74 0025          OTA   '0025
00023    0 01 00022       JMP   *-1
00024    0 02 00061       LDA   SADR
00025    0 06 00066       ADD   =246
00026    0 04 00061       STA   SADR
00027    0 02 00062       LDA   EADR
00030    0 06 00066       ADD   =246
00031    0 04 00062       STA   EADR
00032    0 12 00060       IRS   SUP2            BUMP TO READ NEXT RECORD
00033    0 12 00053       IRS   M6              HAVE 6 RECORDS BEEN READ
00034    100000           SKP                   NO
00035    0 01 00045       JMP   A               YES
                     *
00036    34 0425    BSCK  SKS   '0425           SKIP IF NOT BUSY
00037    0 01 00036       JMP   *-1
00040    34 0325          SKS   '0325           SKIP IF NO ERROR
00041    000000           HLT                   ERROR HALT
00042    0 12 00054       IRS   RCTR            HAVE ALL 11 RECORDS BEEN READ
00043    0 01 00010       JMP   NEXT            NO - READ NEXT RECORD
00044   *0 01 00063       JMP*  DOP             YES-ALL DONE - START UP DOP
                     *
00045    0 02 00065 A     LDA   =-6             RESET
00046    0 04 00053       STA   M6              COUNTER
                     *
00047    0 02 00057       LDA   SUP1            BUMP TO READ UNIT 0,
00050    0 06 00064       ADD   ='100           HEAD 4,
00051    0 04 00057       STA   SUP1
00052    0 01 00036       JMP   BSCK            GO TO BUSY CHECK
                     *
                     *
                     *
00053    177772     M6    DEC   -6              6 RECORDS/TRACK
00054    177765     RCTR  DEC   -11             11 RECORDS TO BE READ
00055    000020     SDMC  OCT   20              DMC START ADDRESS
00056    000021     EDMC  OCT   21              DMC END ADDRESS
00057    000300     SUP1  OCT   300             READ, UNIT 0, HEAD 3
00060    000001     SUP2  OCT   1               START READING AT RECORD 1
00061    032000     SADR  OCT   32000           DOP-4700 START ADDRESS
00062    032365     EADR  OCT   32365           245(DEC)= '365
00063    032000     DOP   OCT   32000
                     *
00064    000100           END
00065    177772
00066    000366
```

Figure 3-2. Program Listing for DOP-4700 Bootstrap (20-Surface M. H. D.)

```
                                * PACK FILE 2
                                *
                                * THE FOLLOWING PROGRAM MAY BE USED TO PACK A SOURCE FILE.  THE USER
                                * MUST FIRST DEFINE IN THE DIRECTORY A TEMPORARY LISTING FILE.  WHEN
                                * THIS PROGRAM IS EXECUTED, THE USER WILL BE ASKED FOR SOURCE AND
                                * LISTING FILE NAMES.  THEN THIS PROGRAM WILL HALT AT 'XXX12.  THE USER
                                * SHOULD NOW CHANGE THE SPECIFICATIONS OF HIS SOURCE FILE IN THE
                                * DIRECTORY,  THEN CONTINUE EXECUTION OF THIS PROGRAM AT 'XXX13 AND
                                * SPECIFY THE SOURCE AND LISTING NAMES AGAIN.  WHEN TRANSFER IS
                                * COMPLETE, DOP WILL UPDATE THE SOURCE FILE ENTRY IN THE DIRECTORY
                                * (ETRK AND ERA) AND PRINTS THE 'I,O,L OR P' MESSAGE.  (CONTROL DOES
                                * NOT RETURN TO THIS PROGRAM).
                                *
                                    REL
                                *
00000    0 02 00120                 LDA    =5                    TELL DOP WE'RE READING SOURCE AND
                                *                                WRITING LISTING
00001   *0 10 00770                 JST*   '770                  DOPI ENTRY
                                *                                DOP WILL ASK FOR SOURCE AND LISTING NAMES
                                *
00002   *0 10 00771 RSRC JST*   '771                  READ A SOURCE RECORD
00003    0 000022                   DAC    BUFF
00004   *0 10 00773                 JST*   '773                  AND WRITE IN LISTING FILE
00005    0 000022                   DAC    BUFF
00006    0 02 00022                 LDA    BUFF                  WAS IT END OF FILE
00007    0 07 00117                 SUB    =A$0                  $0
00010    100040                     SZE
00011    0 01 00002                 JMP    RSRC                  NO-READ ANOTHER SOURCE RECORD
00012    000000                     HLT                          CHANGE SOURCE FILE ENTRY IN DIRECTORY
                                *
00013    0 02 00116                 LDA    ='100005              TELL DOP WE'RE WRITING SOURCE AND READING
                                *                                LISTING
00014   *0 10 00770                 JST*   '770                  DOPI ENTRY
                                *                                DOP WILL ASK FOR SOURCE AND LISTING NAMES
                                *
00015   *0 10 00773 RLST JST*   '773                  READ A LISTING RECORD
00016    0 000022                   DAC    BUFF
00017   *0 10 00771                 JST*   '771                  AND WRITE IN SOURCE FILE
00020    0 000022                   DAC    BUFF
00021    0 01 00015                 JMP    RLST                  READ ANOTHER LISTING RECORD
                                *
00022    000000         BUFF BSZ    60
                                *
00116    100005                     END
00117    122260
00120    000005
```

Figure 3-3. Program Listing for DOP-4700 Pack File 2 (Source)

```
                                * PACK FILE 3
                                *
                                * THE FOLLOWING PROGRAM MAY BE USED TO PACK AN OBJECT FILE.  THE USER
                                * MUST FIRST DEFINE IN THE DIRECTORY A TEMPORARY SOURCE FILE.  WHEN THIS
                                * PROGRAM IS EXECUTED, THE USER WILL BE ASKED FOR SOURCE AND OBJECT
                                * NAMES.  THEN THIS PROGRAM WI L HALT AT EITHER 'XXX11 OR 'XXX21.  THE
                                * USER SHOULD NOW CHANGE SPECIFICATIONS OF HIS OBJECT FILE IN THE
                                * DIRECTORY,  THEN CONTINUE EXECUTION OF THIS PROGRAM AT 'XXX25 AND
                                * SPECIFY THE SOURCE AND OBJECT NAMES AGAIN.  WHEN TRANSFER IS COMPLETE,
                                * DOP UPDATES THE OBJECT FILE ENTRY IN THE DIRECTORY (ETRK AND ERA) AND
                                * RETURNS TO THIS PROGRAM WHICH WILL HALT AT 'XXX40 OR 'XXX51.
                                *
                                * (A SIMILAR PROCEDURE MAY BE USED TO PACK A LISTING FILE).
                                *
                                    REL
                                *
00000    0 02 00244                 LDA    ='100004              TELL DOP WE'RE WRITING SOURCE AND READING
                                *                                OBJECT
00001   *0 10 00770                 JST*   '770                  DOPI ENTRY
                                *                                DOP WILL ASK FOR SOURCE AND OBJECT NAMES
                                *
00002   *0 10 00772 ROBJ JST*   '772                  READ AN OBJECT RECORD (60 WORDS)
00003    0 000052                   DAC    BUFF
00004   *0 10 00771                 JST*   '771                  WRITE THE FIRST 40
00005    0 000052                   DAC    BUFF
00006    0 02 00052                 LDA    BUFF
00007    0 07 00243                 SUB    ='300                 WAS IT END OF FILE
00010    101040                     SNZ
00011    000000                     HLT                          CHANGE OBJECT FILE ENTRY IN DIRECTORY
                                *
00012   *0 10 00772                 JST*   '772                  NO - READ 60 MORE WORDS
00013    0 000146                   DAC    BUFF+60
00014   *0 10 00771                 JST*   '771                  WRITE 41 - 80
00015    0 000122                   DAC    BUFF+40
00016    0 02 00146                 LDA    BUFF+60
00017    0 07 00243                 SUB    ='300                 WAS IT END OF FILE
00020    101040                     SNZ
00021    000000                     HLT                          CHANGE OBJECT FILE ENTRY IN DIRECTORY
                                *
00022   *0 10 00771                 JST*   '771                  NO - WRITE THE LAST 40 (81 - 120)
00023    0 000172                   DAC    BUFF+80
00024    0 01 00002                 JMP    ROBJ                  READ ANOTHER OBJECT RECORD
                                *
                                *
                                *
                                *
00025    0 02 00242                 LDA    =6                    * * RESTART HERE ***      TELL DOP WE'RE
                                *                                READING SOURCE AND WRITING OBJECT
00026   *0 10 00770                 JST*   '770                  DOPI ENTRY
                                *                                DOP WILL ASK FOR SOURCE AND OBJECT NAMES
```

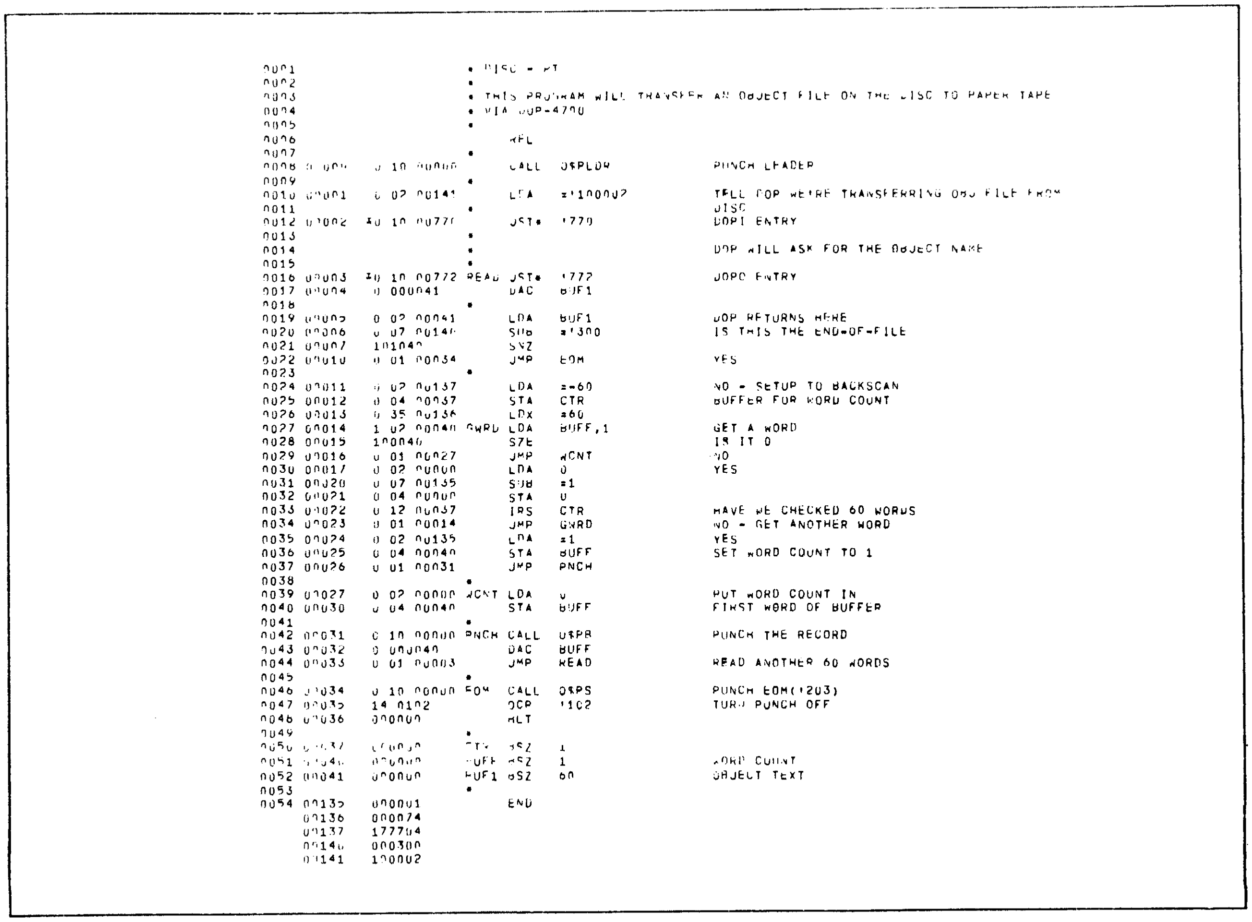
Figure 3-4. Program Listing for DOP-4700 Pack File 3 (Object)

```
                       *
00027  *0 10 00771 RSRC JST*  '771          READ 1ST 40
00030   0 000052        DAC   BUFF
00031  *0 10 00771      JST*  '771          READ 2ND 40
00032   0 000122        DAC   BUFF+40
00033  *0 10 00772      JST*  '772          WRITE 1ST 60
00034   0 000052        DAC   BUFF
00035   0 02 00052      LDA   BUFF
00036   0 07 00243      SUB   ='300         WAS IT END OF FILE
00037   101040          SNZ
00040   000000          HLT                 YES - DONE
                       *
00041  *0 10 00771      JST*  '771          NO - READ LAST 40
00042   0 000172        DAC   BUFF+80
00043  *0 10 00772      JST*  '772          WRITE LAST 60
00044   0 000146        DAC   BUFF+60
00045   0 02 00146      LDA   BUFF+60
00046   0 07 00243      SUB   ='300         WAS IT END OF FILE
00047   100040          SZE
00050   0 01 00027      JMP   RSRC          NO - READ ANOTHER SOURCE RECORD
00051   000000          HLT                 YES - DONE
                       *
00052   000000     BUFF BSZ   120
                       *
00242   000006          END
00243   000300
00244   100006
```

Figure 3-4 (cont). Program Listing for DOP-4700 Pack File 3 (Object)

```
                        * DISC - ASR PRINTER
                        *
                        * THIS PROGRAM WILL READ A LISTING FROM DISC AND PRINT IT ON THE ASR
                        * VIA DOP-4700.  IT ASSUMES THAT EITHER THE FIRST TWO CHARACTERS OF THE
                        * LAST LINE ARE $0 (E.G. FORTRAN) OR THE 9TH CHARACTER OF THE LAST LINE
                        * IS A NON-BLANK CHARACTER (E.G. DAP-16).  WHEN THIS PROGRAM IS
                        * EXECUTED, THE OPERATOR WILL BE ASKED FOR THE LISTING FILE NAME.
                        * AFTER THE PRINTOUT, IT WILL HALT AT EITHER 'XXX13 OR 'XXX21.
                        *
                             REL
                        *
00000   0 02 00122           LDA   ='100001             TELL DOP WE'RE READING A LISTING FILE
00001  *0 10 00770           JST*  '770                 DOPI ENTRY
                        *                               DOP WILL ASK FOR LISTING NAME
                        *
00002   0 10 00000           CALL  O$AF                 LINE FEED
                        *
00003  *0 10 00773 GLIN JST*  '773                 DOPL ENTRY - GET A LINE
00004   0 000023             DAC   BUFF
                        *
00005   0 10 00000           CALL  O$AP                 PRINT THE LINE
00006   0 000023             DAC   BUFF
                        *
00007   0 10 00000           CALL  O$AC                 CARRIAGE RETURN
00010   0 10 00000           CALL  O$AF                 LINE FEED
                        *
00011   0 02 00023           LDA   BUFF                 WAS THERE A
00012   0 07 00121           SUB   ='122260             $0
00013   101040               SNZ
00014   000000               HLT                        YES - ALL DONE
00015   0 02 00027           LDA   BUFF+4               NO - WAS 9TH CHAR.
00016   0 03 00120           ANA   ='177400             NON-BLANK
00017   0 07 00117           SUB   ='120000
00020   100040               SZE
00021   0 01 00003           JMP   GLIN                 NO - GET ANOTHER LINE
00022   000000               HLT                        YES - ALL DONE
                        *
00023   000000          BUFF BSZ   60                   LINE OF TEXT
                        *
00117   120000               END
00120   177400
00121   122260
00122   100001
```

Figure 3-5. Program Listing for DOP-4700 Transfer from Disk to ASR Printer



```
                             * DISC - PT
                             *
                             * THIS PROGRAM WILL TRANSFER AN OBJECT FILE ON THE DISC TO PAPER TAPE
                             * VIA DOP-4700
                             *
                                  REL
                             *
00000    0 10 00000               CALL   O$PLDR             PUNCH LEADER
                             *
00001    0 02 00141               LDA    ='100002           TELL DOP WE'RE TRANSFERRING OBJ FILE FROM
                             *                              DISC
00002   *0 10 00770               JST*   '770               DOPI ENTRY
                             *
                             *                              DOP WILL ASK FOR THE OBJECT NAME
                             *
00003   *0 10 00772 READ JST*    '772               DOPO ENTRY
00004    0 000041            DAC    BUF1
                             *
00005    0 02 00041               LDA    BUF1               DOP RETURNS HERE
00006    0 07 00140               SUB    ='300              IS THIS THE END-OF-FILE
00007    101040                   SNZ
00010    0 01 00034               JMP    EOM                YES
                             *
00011    0 02 00137               LDA    =-60               NO - SETUP TO BACKSCAN
00012    0 04 00037               STA    CTR                BUFFER FOR WORD COUNT
00013    0 35 00136               LDX    =60
00014    1 02 00040 GWRD LDA    BUFF,1             GET A WORD
00015    100040                   SZE                       IS IT 0
00016    0 01 00027               JMP    WCNT               NO
00017    0 02 00000               LDA    0                  YES
00020    0 07 00135               SUB    =1
00021    0 04 00000               STA    0
00022    0 12 00037               IRS    CTR                HAVE WE CHECKED 60 WORDS
00023    0 01 00014               JMP    GWRD               NO - GET ANOTHER WORD
00024    0 02 00135               LDA    =1                 YES
00025    0 04 00040               STA    BUFF               SET WORD COUNT TO 1
00026    0 01 00031               JMP    PNCH
                             *
00027    0 02 00000 WCNT LDA    0                  PUT WORD COUNT IN
00030    0 04 00040               STA    BUFF               FIRST WORD OF BUFFER
                             *
00031    0 10 00000 PNCH CALL   O$PB               PUNCH THE RECORD
00032    0 000040                 DAC    BUFF
00033    0 01 00003               JMP    READ               READ ANOTHER 60 WORDS
                             *
00034    0 10 00000 EOM  CALL   O$PS               PUNCH EOM('203)
00035    14 0102                  OCP    '102               TURN PUNCH OFF
00036    000000                   HLT
                             *
00037    000000     CTR  BSZ    1
00040    000000     BUFF BSZ    1                  WORD COUNT
00041    000000     BUF1 BSZ    60                 OBJECT TEXT
                             *
00135    000001                   END
00136    000074
00137    177704
00140    000300
00141    100002
```

Figure 3-6. Program Listing for DOP-4700 Transfer from Disk to Paper Tape

```
                     * DISC - MT
                     *
                     * THE FOLLOWING PROGRAM MAY BE USED TO TRANSFER A SOURCE FILE FROM
                     * DISC TO MAG TAPE.  THE PROGRAM ASSUMES THE MAG TAPE UNIT IS 0 (LOGICAL
                     * AND PHYSICAL), AND IS ON DMC CHANNEL 2.  THE PROGRAM WILL FIRST
                     * REWIND THE TAPE AND THEN THE OPERATOR WILL BE ASKED FOR THE NAME OF
                     * THE SOURCE FILE.  WHEN TRANSFER IS COMPLETE, THE PROGRAM WILL HALT AT
                     * 'XXX25.
                     *
                          EXT   M$TY
                          EXT   M$CO
                     *
                          REL
                     *
00000   0 02 00101        LDA   =1                 SET CHANNEL TYPE
00001   0 04 00000        STA   M$TY               TO 1 (DMC)
00002   0 06 00101        ADD   =1
00003   0 04 00000        STA   M$CO               DMC CHANNEL NO. = 2
                     *
                     *                             (LOGICAL UNIT NO.=0=PHYSICAL UNIT NO. BY
                     *                             DEFAULT)
                     *
00004   0 10 00000        CALL  C$MR               REWIND
00005   000000            DEC   0                  UNIT 0
                     *
00006   0 02 00100        LDA   =4                 TELL DOP WE'RE READING SOURCE
00007  *0 10 00770        JST*  '770               DOPI ENTRY - DOP WILL ASK FOR NAME OF
                     *                             SOURCE FILE
                     *
00010  *0 10 00771   RSRC JST*  '771               READ A SOURCE RECORD FROM DISC
00011   0 000026          DAC   BUFF
00012   0 10 00000        CALL  O$MC               AND WRITE ON MAG TAPE (16-BIT BINARY MODE)
00013   0 000026          DAC   BUFF
00014   000050            DEC   40                 WORD COUNT
00015   000000            DEC   0                  UNIT 0
00016   0 000076          DAC   EOT                (SHOULDN'T HAPPEN)
                     *
00017   0 02 00026        LDA   BUFF
00020   0 07 00077        SUB   ='122260           WAS IT END OF FILE  ($0)
00021   100040            SZE
00022   0 01 00010        JMP   RSRC               NO-READ ANOTHER SOURCE RECORD
00023   0 10 00000        CALL  O$ME               YES - WRITE END OF FILE MARK
00024   000000            DEC   0                  UNIT 0
00025   000000            HLT                      A L DONE
                     *
00026   000000       BUFF BSZ   40
00076   000000       EOT  HLT                      END OF TAPE (SHOULDN'T HAPPEN)
                     *
00077   122260            END
00100   000004
00101   000001
```

Figure 3-7. Program Listing for DOP-4700 Transfer from Disk to Magnetic Tape

```
                     * MT - DISC
                     *
                     * THE FOLLOWING PROGRAM MAY BE USED TO TRANSFER A SOURCE FILE FROM
                     * MAG TAPE TO DISC.  THE PROGRAM ASSUMES THE MAG TAPE UNIT IS 0 (LOGICAL
                     * AND PHYSICAL), AND IS ON DMC CHANNEL 2.  THE PROGRAM WILL FIRST
                     * REWIND THE TAPE AND THEN THE OPERATOR WILL BE ASKED FOR THE NAME OF
                     * THE SOURCE FILE.  WHEN TRANSFER IS COMPLETE, THE SOURCE FILE ENTRY IN
                     * THE DIRECTORY WILL BE UPDATED (ETRK AND EMA) AND DOP WILL PRINT THE
                     * 'I,U,L OR P' MESSAGE.  CONTROL DOES NOT RETURN TO THIS PROGRAM AFTER
                     * THE '$0' IS READ.
                     *
                     * INITIALLY THE OPERATOR MUST DEFINE A SOURCE FILE IN THE DIRECTORY.
                     *
                          EXT   M$TY
                          EXT   M$CO
                     *
                          REL
                     *
00000   0 02 00076        LDA   =1                 SET CHANNEL TYPE
00001   0 04 00000        STA   M$TY               TO 1 (DMC)
00002   0 06 00076        ADD   =1
00003   0 04 00000        STA   M$CO               DMC CHANNEL NO. = 2
                     *
                     *                             (LOGICAL UNIT NO.=0=PHYSICAL UNIT NO. BY
                     *                             DEFAULT)
                     *
00004   0 10 00000        CALL  C$MR               REWIND
00005   000000            DEC   0                  UNIT 0
                     *
00006   0 02 00075        LDA   ='100004           TELL DOP WE'RE WRITING SOURCE
00007  *0 10 00770        JST*  '770               DOPI ENTRY - DOP WILL ASK FOR NAME OF
                     *                             SOURCE FILE
                     *
00010   0 10 00000   RSRC CALL  I$MC               READ A SOURCE RECORD FROM DISC - 16-BIT
                     *                             BINARY MODE
00011   0 000022          DAC   BUFF
00012   000050            DEC   40                 WORD COUNT
00013   000000            DEC   0                  UNIT 0
00014   0 000073          DAC   ERR1               E ROR RETURN
00015   0 000072          DAC   EOT                END OF TAPE RETURN (SHOULDN'T HAPPEN)
00016   0 000074          DAC   ERR2               END OF FILE RETURN (SHOULDN'T HAPPEN - DOP
                     *                             WILL TAKE CONTROL BEFORE EOF MARK IS READ)
                     *
00017  *0 10 00771        JST*  '771               WRITE THE SOURCE RECORD ON DISC
00020   0 000022          DAC   BUFF
                     *
00021   0 01 00010        JMP   RSRC               READ ANOTHER SOURCE RECORD FROM TAPE
                     *
00022   000000       BUFF BSZ   40
00072   000000       EOT  HLT
```

Figure 3-8. Program Listing for DOP-4700 Transfer from Magnetic Tape to Disk

```
                       * [illegible]

00073   000000         ERR1 HLT
00074   000000         ERR2 HLT
                       *
00075   100004              END
00076   000001
```

Figure 3-8 (cont). Program Listing for DOP-4700 Transfer from Magnetic Tape to Disk

```
                             * TEST DOPL ENTRY
                             *
                             *    THE FOLLOWING PROGRAM DEMONSTRATES THE USAGE OF THE DOPL ENTRY
                             *    ('773) AND THE GENERATING OF A LISTING FILE UPDATE IN THE
                             *    DIRECTORY.  THREE LISTING RECORDS WILL BE WRITTEN.  AFTER THE 3RD
                             *    RECORD, WHOSE FIRST TWO CHARACTERS ARE '$0', IS WRITTEN, THE
                             *    DIRECTORY IS UPDATED (ETRK AND ERA) AND UPON RETURNING, THIS
                             *    PROGRAM WILL HALT.
                             *
                             *    THE USER MUST FIRST DEFINE A LISTING FILE IN THE DIRECTORY.
                             *
                                  REL
                             *
00000    0 02 00014               LDA    PTR1                INITIALIZE
00001    0 04 00005               STA    PTR2
                             *
00002    0 02 00114               LDA    ='40001             TELL DOP WE'RE WRITING A LISTING FILE (BIT
                             *                               16 SET) AND WE WANT TO CHECK FOR A
                             *                               $0-RECORD (BIT 2 SET FOR DIRECTORY UPDATE).
00003   *0 10 00770               JST*   '770                DOPI ENTRY - DOP WILL ASK FOR LISTING NAME
                             *
00004   *0 10 00773      WRIT JST*   '773                DOPL ENTRY - WRITE A LISTING RECORD
00005     000000         PTR2 BSZ    1
                             *
00006   *0 02 00005               LDA*   PTR2                DID WE JUST WRITE A $0-RECORD
00007    0 07 00113               SUB    =A$0
00010    101040                   SNZ
00011    000000                   HLT                        YES - ALL DONE
00012    0 12 00005               IRS    PTR2                NO
00013    0 01 00004               JMP    WRIT                WRITE ANOTHER RECORD
                             *
                             *
00014    0 000015        PTR1 DAC    BUFF
00015    140701          BUFF BCI    29,AABB$0CCDDEEFFGG HIIJJKKLLMMNNOOPPQQRRSSTTUUVVWWXXYYZZ0011
00016    141302
00017    122260
00020    141703
00021    142304
00022    142705
00023    143306
00024    143707
00025    144310
00026    144711
00027    145312
00030    145713
00031    146314
00032    146715
00033    147316
00034    147717
00035    150320
```

Figure 3-9. Program Listing for DOP-4700 DOPL Entry Test

```
          00036   150721
          00037   151322
          00040   151723
          00041   152324
          00042   152725
          00043   153326
          00044   153727
          00045   154330
          00046   154731
          00047   155332
          00050   130260
          00051   130661
     0035 00052   131262             BCI   29,2233445566778899 ABBCCDDEEFFGGHHIIJJKKLLMMNNOOPPQQRRSSTTUU
          00053   131663
          00054   132264
          00055   132665
          00056   133266
          00057   133667
          00060   134270
          00061   134671
          00062   140701
          00063   141302
          00064   141703
          00065   142304
          00066   142705
          00067   143306
          00070   143707
          00071   144310
          00072   144711
          00073   145312
          00074   145713
          00075   146314
          00076   146715
          00077   147316
          00100   147717
          00101   150320
          00102   150721
          00103   151322
          00104   151723
          00105   152324
          00106   152725
     0036 00107   153326             BCI   4,VVWWXXYY
          00110   153727
          00111   154330
          00112   154731
     0037                      *
     0038 00113   122260             END
          00114   040001
```

Figure 3-9 (cont). Program Listing for DOP-4700 DOPL Entry Test

DOP-VRC OPERATING INSTRUCTIONS

The operating procedures for DOP-VRC (Document Number 70181651000) are similar to those for DOP-4700. Each track on the drum contains 16 records (0-15); each record contains eight segments (1 segment = 128_{10} words = 200_8 words). No formatting is required. All physical transfers to and from the drum are one record in length. Type 1 file transfers from the drum to the user's core area are buffered by DOP-VRC if they are not one record in length, thus allowing variable record transfers. Blocking and unblocking for file types 2, 3, and 4 is performed by DOP-VRC. The directory is fixed in length and occupies all of track 0.

The Initialize command (I) is used only to clear the directory. To verify the clear instruction, the user is asked to type I a second time. Any other typein saves the directory directory.

DOP-VRC uses subroutines IAB, IPB, N$IO-VRC, and a configuration subprogram (see the Type 9310 Drum Storage Unit Programming Manual, Order Number BY07 (Document Number 70130072417).

DOP-VRC does not support source input through cards. Also, the directory is not updated when user programs write source or listing files.

An error halt occurs at relative location 2714 if there has been a drum transmission error. The A register contains the error type. (See the drum driver N$IO-VRC for error types.) Press START to ignore the error.

PAL-AP OPERATING INSTRUCTIONS

PAL-AP generates self-loading system tapes. These tapes are made by dumping the specified limits of memory in a format that is readable by a self-contained loader. (See Section II, Generation of Self-Loading System Tapes.) Self-loading tapes are always loaded back into the same locations from which they were punched.

PAL-AP requires one sector of memory. Loading must start at the beginning of a sector (i.e., NN000, where NN is the sector).

No error checks are made when PAL-AP punches a self-loading tape. A checksum is punched on the tape for error checking when the PAL-AP punched tape is being loaded back into the computer.

PAL-AP punches out a two-part loader program that is used to load in the actual contents of memory that PAL-AP is to punch. The second portion of the loader is relocatable and automatically goes to the last 176_8 locations of the sector specified by the PAL-AP default location, $XX575_8$. To exercise this option, the operator has a choice of:

a. Entering the A-register = $XX600_8$ at program load time, or

b. Inserting the value $XX600_8$ into PAL-AP location $NN575_8$ prior to punching out the self-loading tape. Procedures used in these sections establish and use the highest available memory sector for this loader.

The operating procedure is as follows:

a. Depress MSTR CLEAR pushbutton.

b. Set register P to NN000, where NN is the sector into which PAL-AP has been loaded.

c. Select output device. Enter register A with one of the following configurations:

   1. All bits reset = high-speed paper-tape punch.

   2. Bit 1 set = ASR-33 (turn on punch).

   3. Bit 2 set = ASR-35.

d. Set RUN and depress START pushbutton; the program halts at XX003.

e. Enter into register A the first location of information to be dumped from memory.

f. Depress START pushbutton; the program halts at XX006.

g. Enter into register A the last address of information to be dumped from memory.

h. Depress START pushbutton; the preselected device begins punching a self-loading system tape.

i. Upon completion of the punch, if another tape is to be punched at the same device, continue from step e.

PALC OPERATING INSTRUCTIONS

PALC is a program which produces loaders and punchers for system card decks. Its output is self-loading, using the key-in loader in locations 1 to '17. PALC itself requires a card punch (normally Type 5140) with a binary card punch driver (O$CB) and elements from V$LIB. These are loaded in the usual fashion.

PALC is not normally supplied to customers; however information on it is germane to the following two sections. It may also be useful if a customer wishes to develop specialized bootstraps or loaders.

Figure 3-10 illustrates the outputs available from PALC. If the pre-configured PPALC output is desired, the appropriate binary drivers I$CB and O$CB must be force loaded to locations '3460 and '3600 respectively. Equipment supported by PALC revision C is:

-61 Burroughs Card Reader
Honeywell Type 5120 400/800 CPM Card Reader
Honeywell Type 5140 400/100 CPM Card Reader/Punch

To force load the drivers to the required locations, make the input device "not ready" and start loader LDR-APM at 'XX000 with the address ('3460 or '3600) in the A register. After the loader has hung up waiting for the non-ready device, halt it, ready the device and restart the loader at 'XX004. The driver will load. A map to verify the result may be obtained by restarting the loader at 'XX002.

The starting (or restarting) location is '1000. Once the object equipment answer is in the question "L, P, T OR U ?:" is asked. The typing of one letter will cause the appropriate output and the question will be repeated. To change the equipment selection, restart at '1000.

The output of PALC should be carefully marked since it is difficult to identify the various combinations once they are produced.

LOADING A SYSTEM DECK

To load a system deck, the key-in loader for either paper tape or cards (see Table 1-5) must be in place. Installations with a card reader or reader/punch are normally supplied with a copy of the "L" output of PALC configured for their equipment.

If the input is entirely cards, ready the card reader with:

a. one card bootstrap
b. two card loader
c. system deck
d. at least two blank cards

Either clear the A register, which will cause the loader to execute in the sector indicated by location 7 of the key-in, or set it to 0XX000, which will cause it to execute in locations 'XX600 to 'XX717. Start the computer at location 1 and the decks will load. Extended mode will be used if available.

When either a blank card or a transfer card is read, the computer will halt at location zero or will jump to the indicated address. If a checksum error is detected the computer will halt in the loader. Pushing start will ignore the error.

Locations '20 to '57 are used by the bootstrap, however they may be loaded as required afterwards.

If the paper tape bootstrap is to be used, place it in either the paper tape reader or the ASR reader and ready the card reader with the two card loader, the system deck and at least two blank cards. Except that the A register setting must be used, the remainder of the process is identical.

PPALC OPERATING INSTRUCTIONS

PPALC is a preconfigured self-loading deck produced by PALC. It is normally supplied to customers with a Type 5140 Card Reader/ Punch (in addition to the two card loader) since its primary use is in punching system decks. It may also be used in reading (loading) system decks or in verifying them by calculating the checksums. The latter may make it valuable to some installations with the Burroughs -61 or Honeywell Type 5120 Card Readers.

The loading process is similar to the preceding section. PPALC is contained on 10 (reader only) or 12 cards (reader and punch). As before, either the one card bootstrap or the paper tape bootstrap may be used.

PPALC occupies the entire sector to which it is directed. When bootstrapped in, execution will start immediately. If brought in by other loading techniques, it may be started (or restarted) at 'XX000. Initial execution will type "PPALC" on the Teletype. Extended mode will be set if available.

PPALC solicits a reply from the operator by typing "?: ". The operator must ready the card equipment and respond with one of four replies:

a.	(cr)	If sense switch one is reset: read, verify and load a system deck from the reader. If switch one is set: read and verify, but do not load a system deck.
b.	AAA(cr)	Transfer control (i.e., jump) to AAA.
c.	FFF, LLL(cr)	Punch a system deck from location FFF through and including LLL.
d.	FFF, LLL, AAA(cr)	Punch a system deck plus a transfer card that will pass control (jump) to AAA when loaded.

When either a blank card or a transfer card is read the computer will return to the PPALC question mark or a jump mode to the indicated address. If a checksum error is detected the message "CK" and a question mark are typed. The response I causes the error to be ignored and the data loaded (if sense switch one is reset). A carriage return (cr) causes the data to be discarded.

Each system deck card contains an address, a checksum, a data word count and from zero to 57 data (core image) words. The ordering of the cards is not material and decks may be combined or shuffled. A card with zero data words is interpreted as a transfer card. A blank card is interpreted as the end of the deck.

Typing errors may be corrected by typing any non-octal character except a comma. A commercial-at @ is recommended.

X16-DUPE OPERATING INSTRUCTIONS

X16-DUPE duplicates and verifies punched paper tapes. The computer must have a high-speed reader and a punch.

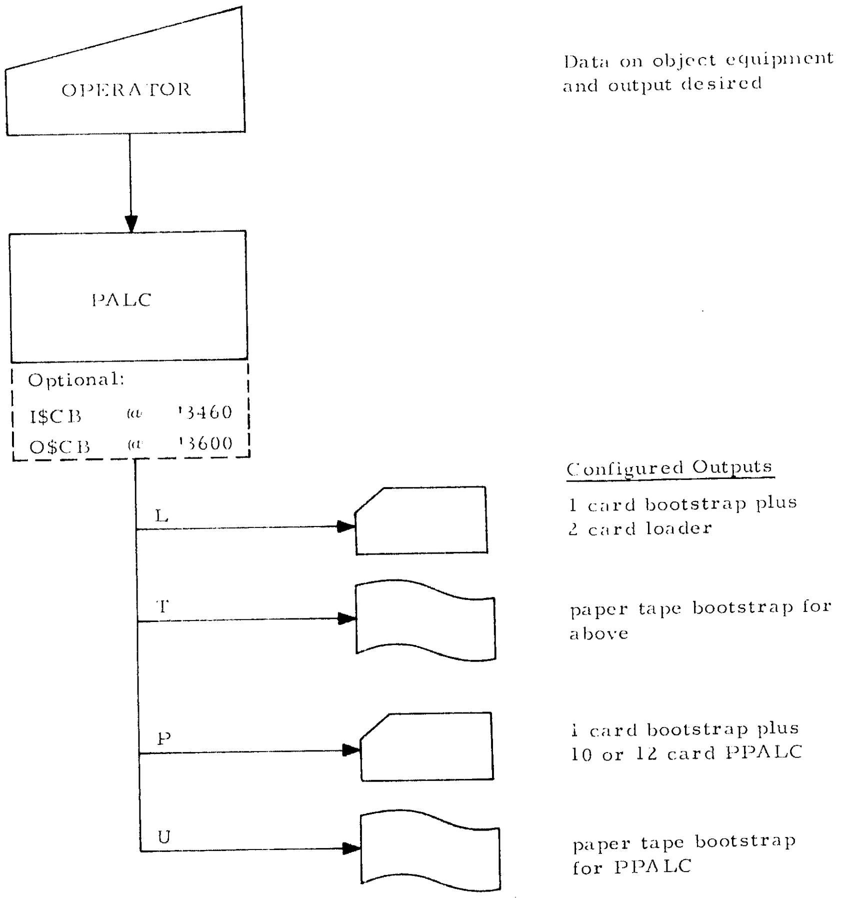
Procedures

a. Load X16-DUPE tape. Place tape to be reproduced in high-speed reader. Depress MSTR CLEAR pushbutton. Set P to OCT 100 and depress START pushbutton. The master tape will be read in. Memory is too small if MO (memory overflow) message is typed.

b. After LC message, place master back in reader. Set sense switch 1. Press START pushbutton. Message VC indicates that master tape was loaded correctly.

c. If no error, reset sense switch 1 and press START pushbutton. This procedure will cause one copy to be punched. Press START pushbutton for each additional copy. Message PC means "punch complete."

d. Place tape to be verified in high-speed reader. Set sense switch 1 and press START pushbutton. Message VE means "verify error." Message VC means "verify complete." Repeat this step for all tapes to be verified.



OPERATOR
Data on object equipment and output desired
PALC
Optional:
I$CB @ '3460
O$CB @ '3600
Configured Outputs
L
1 card bootstrap plus 2 card loader
T
paper tape bootstrap for above
P
1 card bootstrap plus 10 or 12 card PPALC
U
paper tape bootstrap for PPALC

Figure 3-10. PALC (Punch and Load Cards) Use and PPALC (Preconfigured PALC) Generation

Method

The master is read in and packed one frame per word for the first word and two frames per word for the rest of the tape. If an INA 1 does not skip, a counter is incremented. When this counter overflows, it is assumed that the reader has run out of tape. Five is subtracted from the last address to accommodate the tear. The tape is punched out from the first address to the last address minus the loader. The tape is compared from the first address to the last address minus the leader.

Settings

Function	Recommended	Alternate
Load Master	P to 100_8	P to 100_8
Verify Tape	SS1 up	P to 101_8
Punch Tape	SS1 down	P to 102_8

ASR Messages

LC	=	Load complete
MO	=	Memory overflow
VC	=	Verify complete
VE	=	Verify error
PC	=	Punch complete

UPSB16 OPERATING INSTRUCTIONS

UPSB16 duplicates, adds to, or deletes from a paper tape of binary library subroutines. It can make up to three insertions or deletions in a single pass over the library tape. A new tape can be created by merging two or more tapes, each containing one or more subroutines, with a single pass over each tape.

This program operates on a standard 516 computer and requires a high-speed or ASR-33/35 paper-tape reader and a high-speed or ASR-33/35 punch on I/O bus.

Procedures

a. Load self-loading UPSB-16 program.

b. Set A-register for input/output device:

   Bits 11-13 set input. Bits 14-16 set output.

   Set A 3 for ASR-33.

   Set A 5 for ASR-35.

   Zero indicates high speed. For example:

   A = 000000 indicates high-speed I/O.

   A = 000033 indicates ASR-33 I/O.

   A = 000005 indicates high-speed input and ASR-35 output.[1]

   A = 000030 indicates ASR-33 input and high-speed output

c. Set P-register to 001000, place computer in run mode, and depress START pushbutton. The ASR then types out:

   ENTER INSTRUCTION

   The operator may then type an instruction in one of the following formats:

   1. DUPLICATE (executes a block-for-block reproduction).
   2. OMIT\AAAA (sets a name table in memory for reference by the library instruction).
   3. INSERT\AAAA>BBBB (stores new programs in memory for reference by the library instruction).
   4. LIBRARY (the actual update run).

[1] If bit 1, the leftmost bit, is set, it signifies that the output tape contains STOP characters after each subroutine. If the bit is not set, no STOP characters are present.

When updating, the instructions must be typed in the order shown above (i.e., omit lines must be typed before the insert line, and the insert line must be typed before the library line). If a typing error occurs (e.g., a name is misspelled or an incorrect control key is pressed), the RUBOUT key may be pressed. This causes ENTER INSTRUCTION to be typed. The ASR is now ready to accept the correct instruction. If a typing error occurs and a complete reinitialization is desired, the bell (upper case G) may be pressed. This completely reinitializes UPSB16 to prepare for a new update and causes ENTER INSTRUCTION to be typed.

AAAA and BBBB are subroutine names. The full name (1 to 6 characters) of the subroutine must be given, and it must be the name in the location field of the first subroutine operation for that subroutine (primary name). Additional information on each format follows.

a. DUPLICATE. — After this instruction has been typed in, followed by a carriage return, the computer halts so that P/T may be mounted. Depress START pushbutton to cause the duplication to execute.

   NOTE: If an ASR-33 or 35 is to be used, it is recommended that duplication be executed off-line.

b. OMIT\AAAA. — After this instruction has been typed in, such that omit is followed by a form character (upper case L used in a field separator) and the name is followed by a carriage return, the ASR will again type ENTER INSTRUCTION. A maximum of three omit lines may be typed for one update pass.

c. INSERT\AAAA>BBBB, CCCC>DDDD, EEEE>FFFF. — After this instruction has been typed with the \ and > and , characters as shown (\being the field separator, > an upper case period used as a symbol meaning after, and the comma used to separate the subcommands), followed by a carriage return, the computer halts so that tape containing the new subroutines may be mounted. For the preceding example, this tape contains AAAA, CCCC, EEEE, and possibly other subroutines. Depressing the START pushbutton causes the tape to be read and the selected subroutines to be stored. The reader stops when all the required subroutines have been extracted, and the ASR types the octal address of the next available memory cell. It is possible to insert a new subroutine as the first subroutine of the new tape by entering the instruction as ---INSERT AAAA>00--- using zeros as the second name of the instruction.

   NOTES: A maximum of three subroutines to be inserted may be typed for one update pass.

   If ASR-33 is to be used, turn punch switch off. If ASR-35 is to be used, set mode SW to T.

d. LIBRARY. — After this command has been typed, followed by a carriage return, the computer halts so that the old subroutine library tape may be mounted. Depress START pushbutton to cause the old tape to be read and an update tape to be punched. When all the modifications have been made, this instruction behaves like the duplicate instruction.

   NOTE: UPSB16 is coded to accept DAP-16, or DAP-116, or FORTRAN IV format on both the high-speed and ASR-33/35 readers. However, output from UPSB16 follows the standard ASR-compatible form, i.e., block format consisting of a SOM(201), followed by data, and terminated by an X-OFF(223) and RUBOUT character (377). Subroutines punched from memory are separated by 60 frames of leader (blank tape). If more than three omit instructions are attempted, the computer will print a message and halt. Depress the START pushbutton to enable the ENTER INSTRUCTION message to be retyped for continuation of the update. If an ASR-33/35 is to be used, the program will halt after the reading or punching of each block (this halt is for operator intervention). The program will print a message and halt. This is to allow the user to enable the ASR punch. The machine will also halt at memory location 776 to allow the user to disable the ASR punch.

ASR-33 enable and disable with the punch switch.
ASR-35-enable punching with mode switch to KT.
ASR-35-disable punching with mode switch to T.

Method

The program scans the command and takes appropriate action. Subroutines read by means of paper tape have their names scanned and compared to the names that were typed into the ASR. Subroutines to be inserted are read and packed two frames per word in successive memory locations. The end location is flagged by the program for punching at a later time. Subroutines to be omitted are bypassed by the program during the library run.

DEBUG

To aid in program checkout.

Function Codes

A.....Access memory word(s)

B.....Breakpoint set

C.....Copy memory block in memory

D.....Dump memory block to typewriter

E.....Execute subroutine

F.....Fill memory block with constant

H.....High-speed (BRPE) punch in self-loading format

J.....Jump-trace object program (dynamic)

L.....Low-speed (ASR) punch in self-loading format[1]

M ....Monitor object program for effective address (dynamic)

P.....Patch object program

R.....Run object program

S.....Search memory block under mask

T.....Trace object program (dynamic)

V.....Verify memory block against copy in memory

[1] The L function requires that the answer-back drum not respond to a WRU code. The teletype test specifically tests to determine whether the function is disabled.

Restrictions

DEBUG will load and execute in any sector except zero.

DEBUG is relocatable but must be loaded at the beginning of any sector.

Storage

Octal (1000)

Decimal (512) plus location '60 of sector zero

Format

Each command consists of a single-letter function code, followed by a colon and one or more octal values. Values are separated by commas, and the last value used must be followed by a carriage return.

Values are right-justified octal integers: if no digit follows a comma, the value is made zero.

The first value is a starting address.

The second value, if used, is usually the end address of the block started by the first value (functions C, D, F, H, L, S, V) or an initial content for register A (functions E, J, M, R, T).

The third value, if used, is either the start address of another block (functions C, V), or an initial content for register B (functions E, J, M, R, T), or an object pattern (functions F, S).

Other values are interpreted variously (see specific functions).

If values V4 and/or V5 are unspecified, they are set to all one's (177777).

Errors

A slash may be used to abort any input and return to start.

Prolonged outputs (e.g., DUMP, TRACE) may be aborted by activating the ASR BREAK key.

The function code is defined by the last character before the colon. If a wrong character is keyed, follow it with the proper letter code for desired function.

Octal value fields ignore all characters except 0, 1, 2, 3, 4, 5, 6, 7, /, *, comma, and carriage return. To cancel an incorrect octal value, key in (*).

NOTE: If more than 5 digits are keyed, the last 16 bits are used.

Use

A:V1(CR)

Access word(s) in memory (starting) at location V1. The DEBUG program types out the address, V1, its content, and then waits for keyboard input.

To change the content, key in the new octal value, followed by a carriage return. The program then types out the next higher address and its content.

To process to the next higher address without changing the contents of the current location, key in a comma (characters keyed in before this comma are ignored).

The look/change cycle continues until the operator keys in a '/'.

B:V1(CR)

Insert breakpoint link in object program at location V1.

If object program is executed later, and if control reaches location V1, an indirect jump through location '00060 returns control to the DEBUG program, which prints the register contents, then awaits further commands.

Print format is given under functions E and R.

Only one breakpoint can be inserted in a program.

The actual breakpoint jump is placed in the object program only at execution time and is removed after each use; however, the breakpoint address is retained for reuse and requires user action only to change it.

If object program does not reach breakpoint, stop computer manually, and restart in DEBUG at 'XX007.

To remove breakpoint completely, key in B:1(CR).

C:V1, V2, V3(CR)

Copy memory block at locations V1 through V2 into block at locations V3 through V3 + V2-V1. If V2 does not exceed V1, only the word at location V1 will be copied (into V3). If V3 lies between V1 and V2, the block between V1 and V3-1 will be repeated cyclically until location V3 + V2-V1 has been written into.

D:V1, V2(CR)

Dump memory block at locations V1 through V2 to ASR typewriter. The basic typing format is eight octal words per line, preceded by the octal address of the first word printed on the line. (If this address does not end with a zero digit, appropriate spaces are inserted to maintain column integrity.) Repetitious words are suppressed as follows:

a. If the remainder of the current line is identical to the word last printed, the line is terminated.

b. If one or more subsequent lines are identical to the word last printed, the typewriter skips one line.

E:V1, V2, V3(CR)

Execute subroutine by performing JST to location V1.

Prior to subroutine entry, V2 is loaded into register A, and (except on the 416) V3 is loaded into register B.

The subroutine return should be by an indirect jump through its entry point, incremented by 0, 1, or 2.

Upon return from the subroutine, the DEBUG program prints the register contents as noted under function R, except that one or two meaningless words may precede the specified format to indicate that the subroutine has incremented its return link by 1 or 2.

F:V1, V2, V3(CR)

Fill memory block at locations V1 through V2 with V3. If V2 does not exceed V1, only location V1 will be filled.

H:V1, V2(CR)

High-speed punch, PAL2 format, of memory block at locations V1 through V2.

J:V1, V2, V3(CR)

Dynamically trace object program starting at location V1, with registers A and B initially set to V2 and V3, respectively. A diagnostic printout is produced prior to the interpretive execution of any object JMP or JST or HLT. (See function T for further details and restrictions.)

L:V1, V2(CR)

Low-speed punch, PAL2 format, of memory block at location V1 through V2. Operator should turn on ASR punch approximately two seconds after keying CR.

M:V1, V2, V3, V4(CR)

Dynamically monitor object program starting at location V1, with registers A and B initially set to V2 and V3, respectively. A diagnostic printput is produced prior to the interpretive execution of any object memory-reference instruction whose effective address equals V4. (See function T for further details and restrictions.)

P:V1, V2(CR)

Insert patch to V1 in object program at location V2 by replacing instruction at V2 with jump to location V1, storing the displaced instruction at V1, and entering function A with value V1.

Operator must key in desired patch, with suitable return.

NOTE: V1 and V2 must lie in the same sector.

R:V1, V2, V3(CR)

Run object program by performing JMP to location V1. Prior to program entry, V2 is loaded into register A, and (except on the 416) V3 is loaded to register B.

Control does not return to the DEBUG program unless a breakpoint is encountered or the operator takes over manual control via the computer console.

If a breakpoint is encountered, the print format is:

```
      116: INSTR  (A)  (B)  |0|  (C)
      416: INSTR  ...  (A)  |0|  (A)
  316/516: INSTR  (A)  (B)  (X)  (keys)
```

where keys represent values of C, DP, PMI, and SC.

S:V1, V2, V3, V4(CR)

Search memory block at locations V1 through V2 for words equal to V3 under the mask V4. (If no mask is specified, the entire word is tested.) When a match is found, the address and its content are typed out, and the search continues until location V2 has been tested.

T:V1, V2, V3(CR)

Dynamically trace object program starting at location V1, with registers A and B initially set to V2 and V3, respectively. A diagnostic printout is produced prior to the interpretive execution of each object instruction.

Printout is formatted as eight octal words, representing:

```
      116: (P)  INSTR  EA  EA  (A)  (B)  [0]  (C)
      416: (P)  INSTR  EA  EA  ...  (A)  [0]  (A)
  316/516: (P)  INSTR  EA  EA  (A)  (B)  (X)  (keys)
```

NOTE: For nonmemory reference instructions, the third word = '077777, and the fourth repeats the instruction word.

T:V1, V2, V3, V4(CR)

Same as preceding, except a printout is produced only when P = V4.

T:V1, V2, V3, 777777, V5(CR)

Same as above, except a printout is produced for every V5 instruction.

NOTE: If V5 is negative, its absolute value is used.
If V5 is zero, it is treated as 65536.

T:V1, V2, V3, V4, 0(CR)

Same as above, except a printout is produced the first time P = V4 and every instruction thereafter.

Restrictions on all T functions (including J and M):

a. HLT instructions always cause printout followed by a stop. Tracing resumes when the RUN button is actuated.

b. Interrupts are executed in real time, not in interpretive mode. Tracing is resumed when the interrupt routine exits.

c. Tracing is invalid for object programs using extended-addressing mode.

d. Tracing of input-output routines is possible only if no timing constraints are violated by the fact that internal speeds are reduced by a factor of about 60 (avg) to 80 (max) when no printout is involved.

e. Tracing is not possible with ASR I/O.

V:V1, V2, V3(CR)

Verify memory block at locations V1 through V2 against a copy in locations V3 through V3 + V2-V1. The program types the address and content of each location in the V1 block that does not match the corresponding word in the V3 block.

LINE PRINTER DUMP OPERATING INSTRUCTIONS

The Line Printer Dump Program LP-DMP-5 prints selected areas of core memory on the line printer. This program must be loaded starting on a sector boundary. It calls the line printer driver O$LA. The appropriate version of O$LA must be linked with LP-DMP-5. The dumpprogram occupies most of one sector and the beginning of the following sector. Driver O$LA should be loaded starting at the end of LP-DMP-5. In this way no cross-sector references need be generated by the loader. Proceed as follows.

a. Load program, O$LA, and whatever routines O$LA calls.

b. Depress MSTR CLEAR pushbutton.

c. Enter lower limit to be dumped in register A.

d. Enter upper limit to be dumped in register B.

e. Enter start location of LP-DMP-5 in register P.

f. Depress START pushbutton.

The program dumps the contents of the desired locations in groups of eight, such that the first location is always 0, modulo 8, and the last location is always 7, modulo 8. The format for a line is:

a. Location of first word — five octal digits.

b. Two spaces.

c. Three-character mnemonic operation code if one is applicable, or three spaces if one is not.

d. Space or asterisk (asterisk indicates indirect addressing).

e. Six octal digits of the word.

f. Format of items c. through e. repeated seven more times.

After outputting the first line of the dump, the program continually tests to determine whether consecutive memory cells contain identical information. Whenever one or more lines of repeats is found, the repetition is indicated. For example:

```
01000  LDA 005060  STA 011432  CRA 140040  STA 011450  STA 011451  STA 011072  LDA 005450  SZE 100040
01010  JST*120101  LDA 005057  STA 011450  JMP 003031  HLT 000000  HLT 000000  HLT 000000  HLT 000000
                              01020...01027  HLT 000000
01030  HLT 000000  LDA 005451  STA 011455  CRA 140040  STA 051031  IRS 025455  LDA 005072  LGL 041475
01040  STA 011071  OCP 030001  INA 130101  JMP 003042  STA 011447  IMA 027450  STA 011443  JST*120102
01050      000003  NOP 101000  JMP 003073  ERA*152305  STA*151724  JST*120240  JST*120240      000203
01060      000202  HLT 000000  HLT 000000  HLT 000000  HLT 000000  HLT 000000  HLT 000000  HLT 000000
```

ASR-DUMP OPERATING INSTRUCTIONS

The ASR Dump Program, ASR-DUMP, prints selected areas of core memory on an ASR-33 or ASR-35. The program must be loaded starting on a sector boundary and requires about 350 locations (no subroutines are required). Proceed as follows.

a. Load ASR-DUMP.

b. Depress MSTR CLEAR pushbutton.

c. Enter lower limit to be dumped in Register A.

d. Enter upper limit to be dumped in Register B.

e. Enter start location of ASR-DUMP in Register P.

f. Depress START pushbutton.

The program dumps the contents of the desired locations in groups of four, such that the first location is always 0, modulo 4, and the last location is always 3, modulo 4. The format for a line is:

a. Location of first word – five octal digits.

b. Two spaces.

c. Three-character mnemonic operation code if one is applicable, or three spaces if one is not.

d. Space or asterisk (asterisk indicates indirect addressing).

e. Six octal digits of the word.

f. Format of items c. through e. repeated three more times.

Example:

```
00010  INA 131001  JMP 002010  LGL 041470  INA 130001
0014   JMP 002013  STA*110000  IRS 024000  SZE 100040
```

$TRANS OPERATING INSTRUCTIONS

$TRANS is a utility program for transferring source information from one media to another. Source information can be transferred from:

a. Magnetic tape to the line printer, card punch, paper tape punch or to another magnetic tape.

b. Card reader to paper tape punch, magnetic tape or the line printer, and

c. Paper tape to magnetic tape, the card punch or the line printer.

Input Commands

ML......transfer from mag tape (Unit 2) to the line printer.
MC......transfer from mag tape (Unit 2) to the card punch.
MP......transfer from mag tape (Unit 2) to the paper tape punch.
MM......transfer from mag tape (Unit 2) to mag tape (Unit 3).
CP......transfer from card reader to the paper tape punch.
CM......transfer from card reader to mag tape (Unit 3).
CL......transfer from card reader to the line printer.
PM......transfer from paper tape to mag tape (Unit 3).
PC......transfer from paper tape to the card punch.
PL......transfer from paper tape to the line printer.

Sense Switch Settings

Sense Switch 1 Up	creates a break when going mag tape to paper tape and punches a (MOR)
Sense Switch 1 Down	nothing (No MOR)
Sense Switch 2 Up	input a 60 word image
Sense Switch 2 Down	input a 40 word image
Sense Switch 4 Up	output a 60 word image
Sense Switch 4 Down	output a 40 word image

External Programs Used

I$MA	Mag Tape Input Driver
O$MA	Mag Tape Output Driver
O$ME	Write End of File Mag Tape Driver
EXIT	Any Exit Routine to Operating System
O$CH	Card Punch Driver
I$CA	Card Reader Driver
I$PA	Paper Tape Reader Driver
I$PI	Paper Tape Reader Tab Initializer
O$PI	Paper Tape Punch Tab Initializer
O$PS	Paper Tape Punch End Block
O$CS	Card Punch End Block
O$PA	Paper Tape Punch Driver
O$PLDR	Paper Tape Punch Leader Driver
O$LA	Line Printer Driver
O$LH	Line Printer Heading Driver
O$LE	Line Printer Eject a Page Driver
C$6TO8	BCD to ASCII (Mag Tape)
C$8TO6	ASCII to BCD (Mag Tape)
C$MR	Rewind Mag Tape Unit Driver
C$FF	Forward Space File (Mag Tape) Driver
DUMY-X16	Dummy for any of the above

Procedures

a. Load the object of $TRANS to the beginning location (Octal 1000).

b. Load the desired Input/Output routines listed under External Programs used.

c. Place the source media on the proper device (source mag tape on MTU2).

d. Place the object media on the proper device (output mag tape on MTU3).

e. Set the sense switches as explained under Sense Switch Settings.

f. Enter the starting location ('1000) in the P Register and press START.

g. $TRANS will print "516 COPY PROGRAM" on the ASR.

h. $TRANS will print "NAME OF PROCESSOR" and HALT.

i. Type the two character input command desired (see Input Commands) followed by a carriage return.

j. If the source is mag tape, $TRANS will print out "INPUT FILE NUMBER" and HALT. Type the number of the file (decimal value) to be transferred followed by a carriage return and the source tape will be positioned to the proper file. The program will print out "NUMBER OF FILES" and HALT. Type the total number of files to be transferred followed by a carriage return.

k. If the output media is mag tape, $TRANS will print out "OUTPUT FILE NUMBER" and HALT. Type the output file number (decimal value) followed by a carriage return and the object tape will move forward to the proper file.

l. If the output media is paper tape, $TRANS will print "TABS(T) or FULL IMAGE (F)" and HALT. Type in T or F and a carriage return (Tabs are set at 6, 12 and 30 for DAP Source).

m. $TRANS will now transfer the prescribed source(s) from one media to the other, will HALT when finished, and type "END-OF-JOB" and "PRESS START TO CONTINUE".

Operational Notes

A second End-Of-File (End-Of-Tape) mark sensed in a row with card and paper tape input will cause a second End-Of-File to be written when outputting to mag tape.

The process of MM will automatically put a second End-Of-File on the Output Tape when the total number of desired files have been transferred.

All Card Decks must be followed by a $END in cols. 1-4 for an End Of File.

All Source Paper Tapes must have a $END at the end of the tape.

When punching Source Paper Tape from Mag Tape, 801 blocks will be punched and an MOR will be punched on the paper tape in cols. 6, 7 and 8 giving about a four (4) inch roll of Source Paper Tape with tabs. Without tabs, the tape will be a little longer. To continue, just press the start button and it will continue until 801 more blocks are punched or an End-Of-File is sensed. At an End-Of-File, a length of leader and a $END in cols. 1-4 will be punched with another length of leader and stop. If doing more than one file, press start to continue on the next file. When the total number of files is done, the program comes back to input command.

Error Messages

a. "IMPROPER DEVICE TRY AGAIN", will be typed on the ASR if an illegal instruction is input.

b. "LAST CARD HAS INVALID PUNCH", will be typed on the ASR if the last card does not have $END in columns 1-4.

c. "EOT MT3" will be typed on the ASR if an END-OF-TAPE occurs on writing.

d. "EOT MT2" will be typed on the ASR if an END-OF-TAPE occurs during reading.

IOUPMHD OPERATING INSTRUCTIONS

IOUPMHD provides the user with an input/output debug package for the moving head disc.

Function Codes

General
D. . . . . . Dump Memory Block onto Line Printer
F. . . . . . Fill Memory Block with Constant
J . . . . . . Jump into Program
P. . . . . . Print Memory Block
R. . . . . . Replace Memory word(s), or to print one location
S. . . . . . Search Memory Block under a mask
V. . . . . . Verify, Memory Versus Input Device
I . . . . . . Load Memory from Device
O. . . . . . Output Memory to Device

Input
IP. . . . . . Input from Paper Tape
IK. . . . . . Input from 10 surface moving head disc
IS . . . . . . Input from 2 or 20 surface moving head disc
IM. . . . . . Input from magnetic tape

Output
OP. . . . . . Output to paper tape, checksum format
OK. . . . . . Output memory block to the 10 surface disc
OS . . . . . . Output memory block to the 2 or 20 surface disc
OM. . . . . . Output memory block to the magnetic tape

Verification
VP. . . . . . Verify memory block against paper tape
VC . . . . . . Verify one memory block against another
VM. . . . . . Verify memory block against the magnet tape

Hardware Considerations

This program operates on a DDP-516 or an H-316 and requires 3K of core. The following options are supported:

ASR Keyboard Printer

High-Speed Paper Tape Reader/Punch

Moving Head Disc (on DMC Channel 1) --

316/516-4650 Single Spindle Moving Head Disc Store Unit
316/516-4651 Dual Spindle Moving Head Disc Store Unit
316/516-4720 High-Capacity Moving Head Disc Store Unit
DDP516/4600 Moving Head Disc File Option

316-516-5500 Line Printer Option

Magnetic Tape

316/516-4021, -4022 7-Track Mag Tape Option
DDP-516-4100 7-Track Mag Tape Option

COMPATIBILITY

This program will operate on a DDP-516 or an H-316 with extended mode (up to 32K), and may be located in any part of core memory.

LIMITATIONS

This program may not be used with interrupts occurring concurrently unless data transfer is via the DMC.

Format

Each command consists of a single or double letter code, followed by a space and one or more octal values. Values are separated by commas, and the last value used must be followed by a carriage return. Values are right justified octal integers.

Use

D V1, V2 (CR)

* ************

Print the contents of the memory block V1 to V2 on the line printer. A "TOP OF FORM" command is issued before and after the printing. Error message "LP" indicates some type of malfunction, out of paper, yoke not closed, etc.

F V1, V2 (CR)

* ************

Fill Memory Block starting with V1 through V2 with value V3. If V2 does not exceed V1, only one location will be filled.

J V1 (CR)

* *******

Jump to the user's program at location V1. The user's program is entered with extended address enabled.

P V1, V2 (CR) or P V1 (CR)

* ************ ** * *******

Print Memory block at locations V1 through V2 on the ASR teletype. The basic typing format is 8 octal words per line, preceded by the octal address of the first word printed on the line. (If this address does not end with a zero digit, appropriate spaces are inserted to maintain column integrity.) Repetitious words are suppressed as follows:

1. If the last remainder of the current line is identical to the last word printed, the line is terminated.
2. If one or more subsequent lines are identical to the last word printed, the teletype skips one line.

R V1 (CR)

* *******

Replace word(s) in memory starting at location V1. Address V1 and its contents are typed out and then the program waits for Keyboard Input.

> To change the contents, key-in the new value, followed by a "Carriage Return". The next higher address and its contents are then printed.
>
> To progress to the next higher address without changing the contents of the current location, key-in a "Comma". Characters keyed-in before this "Comma" are ignored.
>
> The Look/Change Cycle continues until the operator keys-in a "Slash" (/).

S V1, V2, V3, V4 (CR)

* ********************

Search the memory block at locations V1 through V2 for words equal to V3 under the mask V4.

> When a match is found, the address and its contents are typed out, and the search continues until location V2 has been tested.

OP V1, V2 (CR)

** ************

High-speed paper tape punch of locations V1 through V2, in binary. The paper tape is punched in blocks of 57 words with a checksum at the end of each block.

IP (CR)

** ****

The contents of the paper tape are loaded into core, at the end of each block of 57 words. The checksum punched on the tape is compared against the checksum obtained by reading in the tape. If the two checksums are not the same the reader will Halt. To recover from this Halt:

> Move the tape back one block, line up the sprocket hole exactly over the photo cell so that all of the record gap preceding the block will be read, and clear the Halt by pressing the "Start Button". The program will then attempt to read again the same block, before attempting the re-read. The block causing the Halt should be examined for lint, chad, etc. covering a hole in the tape, and then such material should be removed.

VP V1, V2 (CR) or VP (CR)

** ************ ** ** ****

VP without parameters will verify the entire paper tape. VP with parameters will print out the verification errors within the block V1 to V2.

The contents of core are compared (verified) against the contents of the paper tape. A typeout occurs whenever core differs with the information on the tape. Each typeout consists of the core address, the contents of core, and the contents of the corresponding core location as punched on tape. Zeroed (Blank) locations on tape are not typed. After each typeout, the comparison continues until all of the tape is read and verified.

This typeout, whenever core differs from the contents of tape, is a useful offline debugging tool.

OK V1, V2, V3, V4, V5, V6 (CR)

*****

Output data from core memory to the 10 surface M.H.D. V1 starting address of core memory from which data is to be transferred to the moving head disc. V2 starting address of core memory block to be used as a buffer to verify the disc transfer. V3 disc starting address for the transfer. V4 cylinder. V5 head number. V6 size of the data block to be transferred.

"AE: Error in Accessing the Disc

OS V1, V2, V3, V4, V5, V6 (CR)

*****

Output data from core memory to the 2 or 20 surface moving head disc. Parameters and error messages are the same as for the 10 surface disc above.

IK V1, V2, V3, V4, V5, V6 (CR)

*****

Input Data from the 10 surface moving head disc. Parameters and error messages are the same as for the output function above.

IS V1, V2, V3, V4, V5, V6 (CR)

*****

Input data from the 2 or 20 surface moving head disc. Parameters and error messages are the same as for the output function above.

VC V1, V2, V3 (CR)

*****

Verify the data block of length V3 beginning at core location V1 against the data block of length V3 beginning at core location V2. Any differences are printed on the ASR.

OM V1, V2, V3(CR)

** ***************

Output data from core memory to magnetic tape, V1 is the file number. Data from magnetic tape is transferred as one complete file. If the operator wishes to start a new tape the first record must be file one.

V2, V3 are the core memory starting and ending address of the file to be written. The message "CM" is printed and the operator may type in up to 80 characters of comments. The comments field is terminated by a "Bell Function".

If a key-in error is made while typing comments, a "RUNOUT" will return the program to the start of the comments field.

Errors:

"PF: A Bad Record was written.
"VE" The Data on magnetic tape does not compare with core memory.
"TP" Tape is protected.

IM V1(CR)

** ******

Input Data from magnetic tape to core memory. V1 is a previously written file number. Information from this file is transferred to core memory, any comments that were written with that file number are printed out.

Errors:

> "PE" Record Read has bad Parity.
> "RL" Record length incorrect – generally caused by using the wrong tape density.
> "VE" Magnetic Tape does not compare with core.

VM V1(CR)

** ******

Verify magnetic tape, V1 is a previously written record. Verify differences will be printed out.

Description of Disk Transfer Functions

The user can transfer data between core and the moving head disc, by setting sense switch 4. The user can at the same time verify the disc transfer. When verifying, the user must allocate a buffer the size of his data block. Even if the verify option is not elected there must be a dummy variable in V2.

M$IO, the disc driver used by IOUP, uses the two core locations immediately preceding all transfer and verify buffers specified by user. IOUP saves the original contents of these locations and restores them after each transfer.

Description of Core to Core Comparisons

The user may compare two core blocks and print the differences on the ASR. The contents of all locations examined will remain unchanged.

Cautions and Restrictions

1. All entries must be made in octal.
2. When doing a disc transfer, the number of words specified will be transferred regardless of the number of words per record in the disc format. This can destroy everything else on the track in question, with disastrous results. Be sure that the number of words specified in V6 is the same as the number of words per record in the disc format.
3. M$IO for the 10 surfaces disc is designed to interpret a starting disc address of zero as a command to read the current address. In order to prevent this, it is necessary to change line 378 of M$IO Rev. B in the listing from JMP WRDA to NOP.

4. The disc transfer routines are set up to operate on DMC channel 1. The moving head disc is on unit 0. The user is responsible for formatting his own disc before using IOUP to transfer data.

5. Each type of disc requires a different version of M$IO. The standard disc driver, for the 10 surface disc is:

   EO75-001-6801(M$IO) Dwg. No. 70180616000 Rev. B on up.

   This version requires two external parameters, PAR1 and PAR2, each defined as an octal 1 and loader after M$IO.

   For the 2 or 20 surface disc the proper version of M$IO is:

   EO75(M$IO, 4650, 4720) Dwg. No. 70181509000 Rev. B on up.

SECTION IV
VERIFICATION AND TEST ROUTINES

The instructions for each verification and test routine are given in detail at the beginning of the assembly listing for that routine.

SECTION V
ERROR MESSAGES

Error messages produced by the DAP-16 and DAP-16 Mod 2 Assemblers are listed in the DAP-16 and DAP-16 Mod 2 Language Assembly manual. Error messages produced by the FORTRAN IV Compiler are listed in the Series 16 FORTRAN IV Manual.

If an error is detected by a subroutine, the object error diagnostic subroutine (F$ER) is called. Normally, a two-character mnemonic is typed on the ASR-33/35, and the computer halts. If SENSE switch 3 is on, F$ER returns control to the calling subroutine that exists, with meaningless results in most cases. Table 5-1 lists those subroutines that call on F$ER and their associated error messages.

Table 5-1. FORTRAN IV Library Error Messages

Error Message	Condition	Subroutine
AD	Over/Underflow in double precision	A$66, S$66
AO	Array overflow	SUB$
BF	Backspace routine encountered end of file	F$F5-9
DL	Negative or zero argument	DLOG, DLOG2, DLOG10
DT	Second argument zero	DATAN2
DZ	Division by zero	D$22
EQ	Exponential overflow adding integer to double precision exponent	A$81
EX	Exponential overflow during exponentiation	EXP
FE	Format error integer required at this position	F$IO
GO	Incorrect GO to control variable	F$GA
II	First argument zero, second argument negative	E$11
IM	Over/Underflow during integer multiplication	M$11
IN	Input value out of range	F$IO
IZ	Integer division by zero	D$11
LG	Log of negative or zero argument	ALOG, ALOG10
MD	Double precision MULT/DIV overflow	D$66, M$66
MF	Magnetic tape end of file	F$R5-9
MP	Magnetic tape parity	F$R5-9
MT	Magnetic tape end of tape	F$R5-9
PZ	Double precision division by zero	D$66
RI	Integer to large when converted from real to integer	C$21
SA	Arithmetic overflow	A$22
SD	Divisor unnormalized, overflow (M$22)	D$22, M$22
SM	Exponential over/underflow during multiplication	D$22, M$22
SQ	Negative argument	SQRT

SECTION VI

PAPER TAPE FORMATS

Paper tape is often used as the principal input/output medium for Honeywell 316/516 computers. It consists essentially of an indefinite string of 8-bit characters. Figure 6-1 shows the beginning of a source tape, identifying a few characters and their octal values. Paper tape is also used for object text and core dumps. In these cases a code known as the 4/6/6 code, which consists entirely of nonprinting characters, is used. Paper tape codes are discussed in detail in the Series 16 ASR/Paper-Tape Reader/Paper-Tape Punch Programmers' Reference Manual.

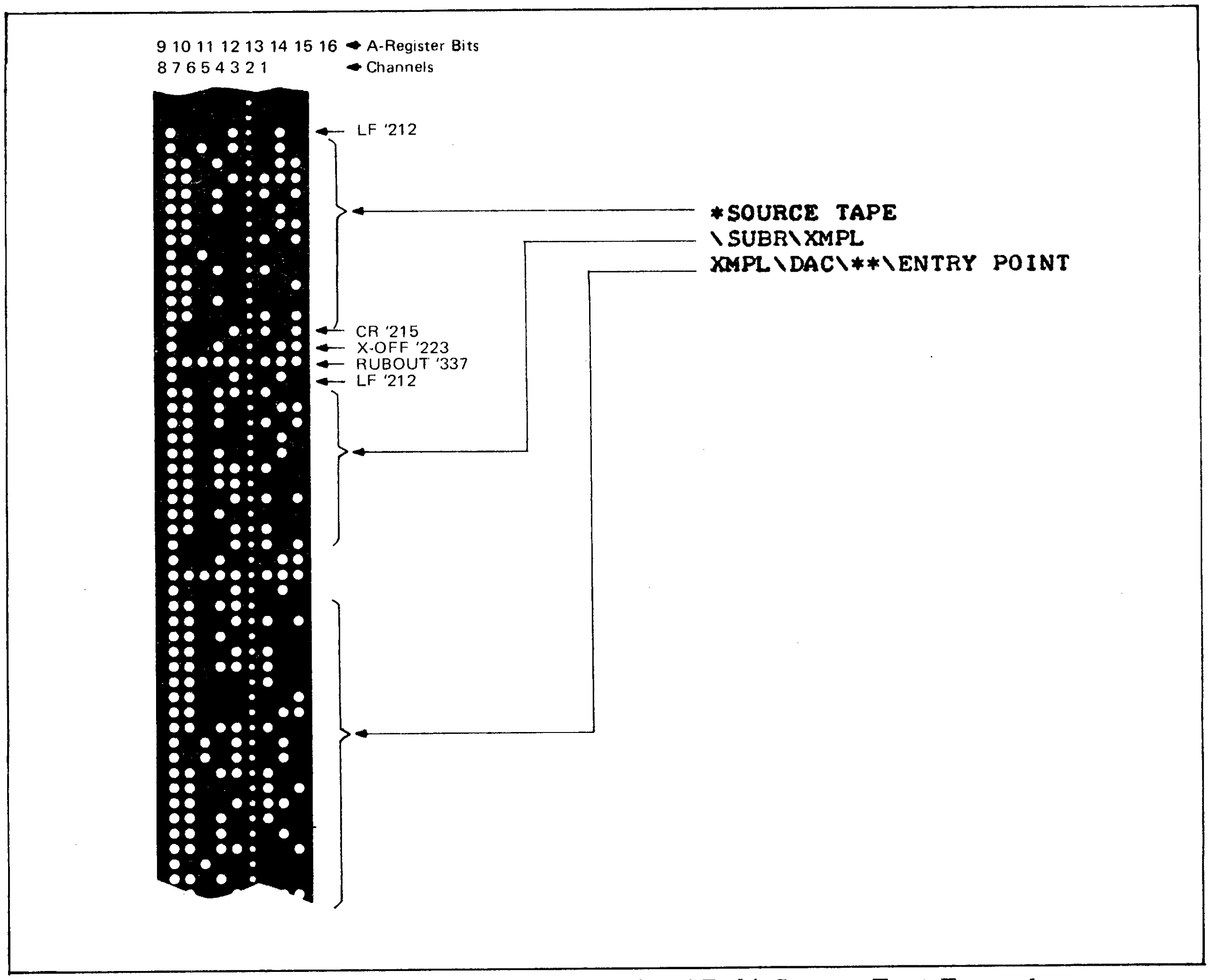


Figure 6-1. Paper-Tape Format with DAP-16 Source Text Example

SOURCE TEXT RECORD FORMAT

Each record in a source text should be formatted as follows:

'212 Line Feed Text
'215 Carriage Return
'223 X-OFF (Control S)
'377 RUBOUT

The X-OFF and RUBOUT need not be used for tapes that will be read by the high-speed reader rather than by a teletypewriter. When the record is read, the line feed, carriage return, X-OFF, and RUBOUT are not stored as part of the record.

USE OF TABS IN SOURCE RECORDS

Tabs may be used to compress the data, as tabs are used on a typewriter. The backslash character (\, '334) is used as a tab code.

The backslash is punched on the ASR-33/35 as an upper case L (also marked FORM on some teletypewriters because of the function control L performs at certain installations). Tabs may be used whenever a string of spaces precedes a tab stop. The tab is punched in place of the spaces.

Another way to describe the backslash is that it is used as a field delimiter. For example, the backslash is ordinarily used in DAP-16 when the location, operation, or variable field is nor present, or in FORTRAN to skip columns 1-6. The ASCII paper-tape read subroutines I$AA and I$PA assume, if not initialized otherwise, that there are three tab positions corresponding to character positions 6, 12, and 30 (DAP-16 source code format). The read routine must be initialized differently for the FORTRAN source.

SOURCE END-OF- MESSAGE RECORDS

An end-of-message record is formatted as follows:

'203 ETX (Control C)
'223 X-OFF (Control S)
'377 RUBOUT

The X-OFF and RUBOUT need not be used for tapes that will be read by the high-speed reader rather than by a teletypewriter.

SOURCE TAPE PREPARATION

ASR-33

PUNCHING LEADER

Blank leader (all NULs) is conveniently punched by depressing HERE IS several times. At least a foot of leader should be punched at the beginning and end of each tape.

CORRECTING ERRORS

Single-character errors may be corrected immediately after they are punched by punching ←(SHIFT O) or by using the B SP key on the punch to backspace the paper tape one frame, and RUBOUT to convert the frame to a RUBOUT. An entire erroneous record may be deleted by punching the sequence @, X-OFF, RUBOUT (i.e., SHIFT P, CONTROL S, RUBOUT). The X-OFF and RUBOUT need not be punched if this source tape is going to be read by a high-speed paper-tape reader. These corrections apply to sources that will be read by I$AA (Rev. F or later), I$PA (Rev. G or later), DAP16-Mod 2 (Rev. C or later), or FORTRAN (Rev. J or later).

ASR-35

PUNCHING LEADER

Blank leader (all NULs) can be punched in two ways — neither as convenient as the HERE IS key on the ASR-33 — by duplicating leader read on the ASR reader or by holding down the BREAK key. If the break key is used, the tape must be backspaced one frame and the last frame changed to a RUBOUT (since it is indeterminate). At least a foot of leader should be punched at the beginning and ending of each tape.

CORRECTING ERRORS

Single-character errors may be corrected immediately after they are punched by using the BACKSPACE and RUBOUT keys to convert the frame to a RUBOUT. An entire erroneous record may be deleted by punching the sequence ←, X-OFF, RUBOUT (i.e., SHIFT O, CONTROL S, RUBOUT). The X-OFF and RUBOUT need not be punched if this source tape is going to be read by a high-speed paper-tape reader.

APPENDIX A
MEMORY MAP

SAMPLE MEMORY MAPS

Figure A-1 shows a map produced after partially loading on OP-16 system. The words preceded by an asterisk are standard loader messages. The other words are entry point names.

```
*LOW    01000
*START  01000
*HIGH   05234
*NAMES  22633
*COMN   26700
*BASE   00126
*BASE   02123
*BASE   01072
AI      01406
IH20    01473
IH40    01544
CLH     01553
CL      01555
XPLT    03000
ER      03004
XH      03011
DSE     03145
XILT    03146
XID1    03153
XID2    03156
XFCT    03176
XCO1    03230
XIV1    03314
XIRT    03372
XLCI    03376
XSPT    03400
XFFF    03401
FP      03405
FL      03512
FIFO    03650
AIO1    04400
AI      04406
FH      04420**
AIO2    04513
FL      0451[illegible]**
NJ      05006
FN      05037**
NJTM    05040

MR
```

Figure A-1. Memory Map, Partially Loaded OP-16 System

*LOW

Address of the lowest location of the program or subroutine (LDR-APM only). This is not to be confused with the lowest location of the Base.

*START

Location as defined by the variable field of the END statement of the main program. If nothing was entered in the variable field, the loader assumes the first location loaded.

*NAMES

The names table begins just below the first instruction of the loader program (i.e., 14122_8 for 8K LDR-APM). *NAMES points to the next available location. As subroutines are called, their names are placed in this table by the loader. Each name takes three locations and is assigned in descending order. At program load time, names of subprograms (from SUBR or ENT pseudo-operation) are read by the loader and are compared with the names table. If a name compares, the subprogram is loaded, and the loader establishes the linkage to the calling program.

*COMM

In FORTRAN IV, the COMMON statement provides a means of sharing memory storage among subprograms or transferring data between subprograms or between segments of a chained program. The location given is provided above the loader and indicates the next available location. Refer to the FORTRAN IV manual for complete use of the COMMON statement.

Figure A-2 shows another map produced after completely loading a program that uses the magnetic tape library and the test and verification services library. The sixteen entry points M$T0 through M$C7 are part of the tape configuration routine M$UNIT. This routine is always loaded with the magnetic tape package. It is used to assign logical tape numbers to physical tape drives and physical DMC or DMA channel numbers. The sixteen entry points are given to enable a user program to make these assignments at run time.

```
*LOW    01000
*START  01000
*HIGH   05616
*NAMES  31457
*COMN   35700
*BASE   00121
PUT     01642
PICK    01667
LCUR    01705
0$LB    04076
0$LI    04155
0$LC    04156
0$FN    04157
0$LW    04160
0$LA    04175
0$LE    04211
0$MA    04524
0$MB    04527
0$MC    04535
0$ME    04736
C$8T06  05010
M$UNIT  05110
M$CHAN  05115
M$TY    05122
M$T0    05125
M$T1    05126
M$T2    05127
M$T3    05130
M$T4    05131
M$T5    05132
M$T6    05133
M$T7    05134
M$C0    05136
M$C1    05137
M$C2    05140
M$C3    05141
M$C4    05142
M$C5    05143
M$C6    05144
M$C7    05145
V$0A    05146
V$0Z    05157
V$0S    05246
V$0D    05324
V$SI    05474
V$SB    05521

LC
```

Figure A-2. Memory Map, Completely Loaded Program Using Magnetic Tape Library and Test and Verification Services Library

*BASE

Indirect references necessary for program linking between sectors are put in the base sector. Normally, this is sector zero. The programmer can control the sector and (except for sector zero) the location within the sector by means of the SETB pseudo-operation. *BASE (or the first *BASE listed if there are more than one) points to the next location into which a cross-sector reference will be loaded. The first base location is specified in register B at the beginning of a load. The default value ((B) = 0) is '100.

Once a base area has been established in a sector (either by a nondefault B-register setting or by a SETB pseudo-operation code), LDR-APM will not establish another in that sector. If this is attempted, the first established base area becomes the current base area (as if no change had been attempted). This automatic restoration is useful when the load requires a number of base areas, some of which may fall in the same sector.

AS-LOADED ENTRY POINTS

The number following a loaded entry point is the address of the entry point as loaded.

UNSATISFIED ENTRY POINTS

Unsatisfied entry points are identified by a number followed by two asterisks. The number is the location of the most recent reference to the entry point.

APPENDIX B
OPERATION WITH EXTENDED ADDRESSING

NORMAL MODE OPERATIONS IN UPPER BANK

Systems tapes may be loaded to any location in core, because there is an EXA instruction included in the relocating loader that is part of a self-loading tape. There is no DXA instruction to return the computer to normal mode, but this is ordinarily accomplished with MASTER CLEAR.

The key-in loader will not operate properly if copied into locations '40001 through '40017.

Register P may be set to any location in core for operations in MA, SI, or RUN mode.

ENTERING EXTENDED ADDRESSING

MASTER CLEAR puts the computer in normal mode addressing. In order to continue operating in extended mode after a MASTER CLEAR the following sequence should be followed:

a. Move MA/SI/RUN switch to SI.

b. Set 000013 in register M.

c. Depress START once.

This sequence should be followed in situations where it is known that the computer has been operating in extended mode.

APPENDIX C
MODIFICATION OF FORTRAN I/O DEVICE ASSIGNMENTS

Device numbers referred to by input or output statements have been tentatively set to the most common devices. Each statement generates a CALL to library subroutine FRn, FWn, or F$Cn, where n is the device number 1 through 9. The devices referred to can be easily changed by writing different subroutines with the same name and substituting the new subroutine for the existing subroutine on the library tape. For example, assume F$R2 is used to read paper tape and no paper tape reader is available, but six magnetic tape units are available. A magnetic tape subroutine (similar to F$R5) can be written, named F$R2, and put on the library tape in place of the original F$R2 subroutine. All references to device two now refer to magnetic tape six instead of the paper tape reader.

If the unit number is referred to symbolically instead of as an integer constant, the value of the symbol n should be loaded into register A before making the call to F$RN or F$WN.

F$RN and F$WN have nine-place tables which interpret n as one of the standard device assignments and transfer to the proper subroutine. If desired, F$RN or F$WN can be modified to rearrange these device assignment tables or to expand them for additional devices. Therefore, by expanding the table in F$RN, device 10 could be magnetic tape 6, device 11 might be typewriter 2, etc. Only symbolic device numbers can exceed 9 in value. Constant device number calls do not use these subroutines and do not, therefore, use the device assignment tables.

Finally, if a symbolic device number reference is made, but the number (such as n = 0 or n = 13) is not in the device assignment table, the computer halts. Illegal device numbers between 0 and 10 (such as attempts to read from the line printer) cause halts. The operator can, at that time, set an acceptable device number into register A and press the computer START button. If the new number is acceptable, input or output is generated using the new device number. However, each time the unacceptable device number is referred to by another READ, WRITE, or CONTROL statement, the halt occurs again.

HONEYWELL INFORMATION SYSTEMS
Technical Publications Remarks Form*

TITLE: SERIES 16 EQUIPMENT OPERATORS' MANUAL

ORDER No.: BX48, REV. 2

DATED: NOVEMBER 1973

ERRORS IN PUBLICATION:

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION:

(Please Print)

FROM: NAME ______________________ DATE: ______________

COMPANY ______________________

TITLE ______________________

____

____

*Your comments will be promptly investigated by appropriate technical personnel, action will be taken as required, and you will receive a written reply. If you do not require a written reply, please check here. ☐

LONG LINE

CUT ALON

FOLD ALONG LINE

FIRST CLASS
PERMIT NO. 39531
WELLESLEY HILLS,
MASS. 02181

Business Reply Mail
Postage Stamp Not Necessary if Mailed in the United States

POSTAGE WILL BE PAID BY:

HONEYWELL INFORMATION SYSTEMS
60 WALNUT STREET
WELLESLEY HILLS, MASS. 02181

ATTN: PUBLICATIONS, MS 050

FOLD ALONG LINE

Honeywell

The Other Computer Company:

Honeywell

HONEYWELL INFORMATION SYSTEMS

14185
11075
Printed in U.S.A.

BX48, Rev. 2