Battelle

Pacific Northwest Laboratories
Richland, Washington 99352

AEC Research and Development Report

PDP-9 SOFTWARE SYSTEM WITH
A DDC FIXED-HEAD-DISC MEMORY

November 24, 1971

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately-owned rights.

PACIFIC NORTHWEST LABORATORY
operated by
BATTELLE
for the
U.S. ATOMIC ENERGY COMMISSION
Under Contract AT(45-1)-1830

3 3679 00062 1112

BNWL-1632

~~Special Distribution~~

PDP-9 SOFTWARE SYSTEM
WITH A DDC FIXED-HEAD-DISC MEMORY

By

L. H. Gerhardstein
R. A. Burnett
D. R. Friedrichs

Systems Engineering Department
Computers and Control Section

November 24, 1971

BATTELLE
PACIFIC NORTHWEST LABORATORIES
RICHLAND, WASHINGTON 99352

TABLE OF CONTENTS

	<u>Page</u>
1.0 INTRODUCTION	1
2.0 SUMMARY SYSTEM DESCRIPTION	2
2.1 Standard System Programs	3
2.2 New System Programs in DOS.	3
3.0 DISC DATA FORMATS	4
3.1 Disc File Format	6
3.2 Disc Directory Format	6
4.0 DOS KEYBOARD MONITOR	8
4.1 ASSIGN Command	8
4.2 REQUEST Command	10
4.3 List Directory Command	11
4.4 New Directory Command	11
4.5 Teletype, Lineprinter, and Display Commands	11
4.6 HALT Command	12
4.7 NO BOOT Command	12
4.8 Get Core Image Command	13
4.9 Enter DMP File Command	13
4.10 Octal Disc Map Command	14
4.11 Load System Program Commands	14
5.0 I/O MONITOR	15
5.1 CAL Handler.	15
5.2 CAL* Handler	16
5.3 Program Interrupt Skip Chain	17
5.4 IOPS Error Diagnostic Routine (.MED).	17
5.5 API Trap Instructions	18
5.6 System Communication Table (.SCOM)	18
5.7 Device Assignment Table (.DAT).	18
5.8 Disc Unit Communication Table (.DKCOM)	19
5.9 Interrupt Initialization Routine (.SETUP)	19
5.10 Teleprinter Device Handler (TTA.).	21
5.11 Save Core Routine (↑Q)	21

TABLE OF CONTENTS (continued)

	<u>Page</u>
6.0 KEYBOARD MONITOR INFORMATION FILES.	22
6.1 Skip Chain & I/O Configuration File (.SKIO)	23
6.2 System Program Information File (.COMM)	24
7.0 DISC I/O DEVICE HANDLERS	28
7.1 Disc Device Handler Characteristics.	28
7.2 Read Directory Function (.RDIR)	31
8.0 DOS UTILITY ROUTINES	32
8.1 Disc/Core Transfer Routine (DKCORE).	33
8.2 DOS Library Copier (LIBCPY)	35
8.3 Disc Compressor (DKCOMP)	36
8.4 Disc Save/Restore Routine (DKSAVE)	36
9.0 SYSTEM GENERATION	37
9.1 New System Creation	38
9.2 System Generation by Copying with PIP	40
10.0 DOS MODIFICATION.	41
10.1 Modify Core Memory Size.	41
10.2 Modify Teletype Model	42
10.3 Addition of Hardware Interrupts	42
10.4 Addition of I/O Device Handlers	43
10.5 Updating a Disc Library.	43
10.6 Addition of a System Program	43
10.7 System Program Modification	44

TABLE OF CONTENTS (continued)

APPENDIX A - SYSTEM DIRECTORY LISTING AND OCTAL LISTING
APPENDIX B - STANDARD .DAT ASSIGNMENTS
APPENDIX C - SYSTEM COMMUNICATION TABLE (.SCOM)
APPENDIX D - DKBOOT AND DKCORE SOURCE PROGRAMS
APPENDIX E - IOPS AND LOAD ERRORS
APPENDIX F - DEVICE HANDLER FUNCTIONS
APPENDIX G - I/O HANDLER CHARACTERISTICS
APPENDIX H - DOS SYSTEM MEMORY MAPS
APPENDIX I - CORE MEMORY TO DECTAPE TRANSFERS USING THE DECTAPE BOOTSTRAP

FOREWORD

This paper describes work performed at Battelle-Northwest to produce an efficient disc operating system (DOS) software package. The DOS operates on a PDP-9 digital computer with a DDC disc memory unit.

This work was performed for the Water and Land Resources Department of Battelle-Northwest for application to the Radionuclides in Soils Program sponsored by the Atlantic Richfield Hanford Company (ARHCO). This software package, developed for use with the Computer Controlled Data Display System, also has application to Fast Flux Test Facility (FFTF) programs involving the FFTF Hybrid Computer System.

PDP-9 SOFTWARE SYSTEM
WITH A DDC FIXED-HEAD-DISC MEMORY

1.0 INTRODUCTION

A Computer Controlled Data Display System is under development for engineering use in the Battelle-Northwest, Water and Land Resources Department, Radionuclides in Soils Program funded by the Atlantic Richfield Hanford Company.⁽¹¹⁾ ⁽²⁾ The disc operating system (DOS) described in this report has been developed as an integral part of the Computer Controlled Data Display System and is essential in providing operational flexibility and efficient hardware utilization.

The DOS is also being used on the Fast Flux Test Facility (FFTF) Hybrid Computer System⁽²⁾ which consists of a PDP-9 digital and an AD-4 analog computer. This hybrid system is being used for modeling, simulation⁽⁴⁾ and display of the nuclear reactor dynamics and control. The Data Display System and the FFTF Hybrid, at the time of this writing, have the following basic characteristics:

	<u>Data Display System</u>	<u>FFTF Hybrid</u>
Digital Computer	DEC PDP-9 ⁽⁷⁾	DEC PDP-9
Core Memory	16K	24K
Automatic Priority Interrupts	Yes	Yes
Hardware Multiply-Divide	Yes	Yes
Direct Memory Access	Yes	Yes
High Speed Paper Tape	Yes	Yes
DEC Tapes	2	3
Disc	DDC Model 14120 ⁽¹⁾	DDC Model 14120
Line Printer	Info-max 57	Gould 4800 ⁽⁶⁾ Potter 3502 ⁽³⁾
CRT Display	UNIVAC 1558 ⁽¹⁾	Philco-Ford D30 ⁽⁵⁾
Analog System	Edwin PF-40 Digitizer	Applied Dynamics AD-4 Analog Computer ⁽²⁾

The Digital Development Corporation (DDC) Model 14120 disc⁽⁸⁾ was chosen for use with these PDP-9 digital computers since:

- The DDC disc employs fixed-heads which do not touch the disc surface thereby producing a high incidence of reliability
- The DDC disc has 458,752 total data words storage with an average access time of 8.5 millisec and transfer rate of 103 kHz.
- The disc has automatic checksums and manually selected write protection.

The use of this particular disc memory required extensive modification of the standard PDP-9 software. These modifications, plus the addition of some new software, have resulted in an entire operating system which resides on and operates from the DDC disc memory.

2.0 SUMMARY SYSTEM DESCRIPTION

The DOS is tailored after the Digital Equipment Corporation (DEC) PDP-9/15 keyboard monitor software system.^(7,9) The DEC system operates with either a DECTape (magnetic tape) transport or a disc as the system device. All system programs, input-output drivers, and math library routines are loaded automatically from the system device. The DOS, likewise, performs automatic loading of the PDP-9 system routines from the DDC disc. However, there are several unique features in DOS; the major areas where DOS and the DEC systems differ are discussed in the following sections. System directory listings are shown in Appendix A.

The PDP-9 system programs that have been incorporated in DOS, with few exceptions, are unmodified images of their counterparts in the PDP-9 DECTape system. The major changes were made in

1. The system I/O monitor and keyboard monitor programs
2. The system program loader
3. The system bootstrap
4. The disc I/O device handlers
5. The disc resident data and directory formats.

2.1 Standard Systems Programs

The DOS contains the following standard PDP-9 system programs:

PIP	Peripheral Interchange Program
EDIT	Symbolic text editor
MACRO	Macro Assembler
F4	FORTTRAN IV compiler
UPDATE	Library update routine relocatable
.LOAD	Linking loader
DDT	On-line debugging routine
CHAIN*	FORTTRAN chain segment builder
EXECUTE*	FORTTRAN chain executer
FOCAL**	Formula calculator
DTCOPY***	Dectape copier
SRCCOM***	Source comparator
CREF***	Cross-reference program
.LIBR	FORTTRAN IV math library & FORTTRAN operate-time-system (OTS)

There are four standard PDP-9 system programs that are not included in DOS:

SGEN	System generator
PATCH	System program patches
DUMP	Core dump routine
CONV	PDP-7 to 9 converter

The first three programs were incompatible with DOS due to the dissimilarity of the DOS disc data formats and those in the DEC system. The 7-to-9 convertor was deleted because of its lack of use on the systems where DOS has been installed.

2.2 New System Programs in DOS

Several programs in DOS are either new developments or modified versions of standard PDP-9 routines:

**The V5A version of DOS does not presently contain these routines*
***Copyright 1969 by Digital Equipment Corporation, Maynard, MA*
****Version V5A only*

KM9	DOS keyboard monitor program
.SKIO	Skip chain and I/O configuration information file
.COMM	System program information file
.SYSLD	System program loader
DKBOOT	DOS bootstrap
DKA.	Disc device handler, Input or Output
DKB.	Disc device handler, Input or Output
DKC.	Disc device handler, Input only
DKD.	Disc device handler, Input or Output
DKCORE	Core-disc transfer utility program
LIBCPY	Disc Library Copier & Formatter
DKCOMP	Disc file/directory compressor
DKSAVE	Disc-DEctape block copier.

3.0 DISC DATA FORMATS

The format of disc resident data files was developed specially for the DDC 14120 disc memory. The important characteristics of the disc data format are:

1. Line buffer headers are not stored as a part of the disc files. Thus, disc files contain only the file data.
2. A single line buffer may extend from one sector into the next, thus fully utilizing file space.
3. All files are contiguous; i.e., they are stored in continuous blocks of disc words and sectors.
4. For reasons 1, 2, and 3 above, all disc files may be treated as core dumps and vice versa.
5. The disc resident directories are contiguous and expandable (by a manual user command) to any number of entries.

The disc can be divided into as many as eight units of arbitrary length, each having a separate directory file which starts at the first sector of the unit. Files within a unit may be segmented. If a file is segmented, then the data within each segment are contiguous. The disc map, with normal unit assignments, is shown in Figure 3.1.

Disc Unit No.	Typical Use
0	Operating System
1	Secondary systems and user core dumps
2	Source and Binary Programs
3	Scratch

Figure 3.1 Disc Map

3.1 Disc File Format

Let a "file" be defined as an "ordered group of data words,"
i.e.:

file $X \triangleq \{x_0, x_1, \dots, x_{n-1}\}$ in that order, where n is the number of words in the file.

Disc files or file segments generated by DOS are contiguous; i.e., if s is the starting sector address of file X , and w is the sector length in words, then words $x_0 \dots x_{w-1}$ reside on sector s , $x_w \dots x_{2w-1}$ reside on sector $s+1$, $x_{kw} \dots x_{(k+1)w-1}$ reside on sector $s+k$, etc.

New files are created starting at the sector immediately following the last file recorded. When a file is deleted, the files that follow are not automatically shifted to fill in the deleted area. Thus, after a number of files have been deleted, it is likely that several disc sectors will contain obsolete data. The disc compressor program (COMPRES) should be used periodically to overlay the deleted files (see Section 8.3).

3.2 Disc Directory Format

The format of the disc directory is an ordered set of directory entries, each having eight data words. The entry order is the same as the order of the files on the disc. The format of a directory entry is shown in Figure 3.2. The first three words (0-2) contain file identification (name and extension) in six bit ASCII trim code. Word #3 contains four file status bits and a 14-bit file starting disc address. The status bit functions are as follows:

- P = Permanent file bit. Permanent files (P-bit+1) cannot be deleted by any software programs in DOS (i.e.: PIP, device handlers).
- D = Deleted file bit. A file whose directory entry is marked with a 1 in the D-bit is considered deleted and cannot be accessed for input. The file and its directory entry will be overlayed by the disc file compressor program (COMPRES).

Word No.	Contents	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17							
0	File Name	N ₁				N ₂				N ₃																
1																										
2	File Extension	E ₁				E ₂				E ₃																
3	Status + DA	P	D	O	S																					
4	Unused	U																								
5	Core Load Addr*	U				CA																				
6	Word Count	U				WC																				
7	Start Addr*	U				SA																				

<u>Nomenclature</u>	<u>Meaning</u>
N _i	File name character, six-bit-trimmed ASCII
E _i	File extension character, six-bit-trimmed ASCII
U	Unused bit or bits
P	Permanent file bit
D	Deleted file bit
O	Open file bit
S	Segmented file bit
DA	Starting disc address of file
CA	Starting core address for program load
WC	File word count
SA	Program starting address for execution

* Does not apply to data files

Figure 3.2 Disc Directory Entry Format

- O = File open bit. The directory entry for partially completed files are marked with a 1 in the O-bit. When the file-open bit is set, the file may not be used as input. Open files are not overlayed by the disc file compressor program.
- S = Segmented file bit. A file whose length is greater than $77700_8 = 32704_{10}$ words or $444_8 = 292_{10}$ sectors must be segmented. If the S-bit is set to 1, the system software will search the directory for another file with the same name and extension. Segmenting is recursive in that any number of segments may be chained by setting the S-bit of the entries to a 1. The last segment entry will have the S-bit set to zero.

4.0 DOS KEYBOARD MONITOR

The basic purpose of the keyboard monitor (KM9) is to act as a communication link between the operator and system or user programs. The keyboard monitor program is loaded from disc into core whenever the system bootstrap is started or the bootstrap is read into memory. The system bootstrap is a $60_8 = 48_{10}$ word hardware-read-in tape and should be read into memory at location 17720 of the highest memory bank. (i.e.: 17720 for 8K, 37720 for 16K memory, etc.) See Appendix D.

When the keyboard monitor is loaded into the core memory it types a dollar sign (\$) indicating that it is ready to accept and act upon any operator commands in its repertoire. The keyboard monitor is the only DOS program which types \$ as a ready signal. KM9 commands, excluding those which load system programs, are summarized in Table 4.1.

4.1 Assign Command

The I/O monitor contains a standard set of initial .DAT slot assignments. (See Appendix B.) Each time the keyboard monitor is loaded, the assignments are reinitialized. The assign command allows the user to specify the I/O configuration to be used by a system or user program. The command format is:

ASSIGN (A) $DV_1 n_1, n_2 \dots / DV_2 m_1, m_2 \dots / \dots$

<u>Command</u>	<u>Action by Keyboard Monitor</u>
ASSIGN A	Assign I/O device(s) to DAT slot(s).
REQUEST R	Print current I/O device assignments on specified listing device.
DIRECTORY D	Print disc directory on specified listing device.
NEWDIRECT	Writes a blank directory on specified disc unit.
TELETYPE TTY	Selects the console teletype as the listing device.
LP	Selects the line printer as the listing device.
DISPLAY DIS S	Selects the CRT display as the listing device.
HALT	Causes the PDP-9 to halt after illegal IOPS error messages (does not apply to IOPS4)
NOBOOT	Causes the system bootstrap to be overlayed by user programs
GET G	Restores core from a specified area or file on disc which was written by †Q.
ENTER	Causes a core image created by †Q to be entered in specified disc directory as a standard binary disc file.
DISCMAP DKMAP DM	Lists octal contents of all words in specified disc sector on specified listing device.

Table 4.1 Keyboard Monitor Commands

Where DV denotes a device handler name and unit number and n,m denote dat slot numbers.

The latter command causes DV₁ to be assigned to DAT slots n₁, n₂, etc; DV₂ to DAT slots m₁, m₂, etc.; etc.

The device handler names can have any of the following example formats:

PRA	3 letter name, assumes unit 0
DTB2	3 letter name, unit 2
TT	2 letter name, assumes TTA0
DK2	2 letter name, assumes DKA2

In the Data Display System, the available device handler names are:

TTA	DTA	DKA	DSA
PRA	DTB	DKB	DZA
PRB	DTC	DKC	LPA
PPA	DTD	DKD	LPB
PPB	DTE		
PPC	DTF		

The device handler names in the FFTF Hybrid System are:

TTA	DTA	DKA	LPA
PRA	DTB	DKB	CLB
PRB	DTC	DKC	
PPA	DTD	DKD	
PPB	DTE		
PPC	DTF		

4.2 Request Command

The request command causes a listing of .DAT slot assignments to be made. If REQUEST(R) is used, all .DAT slots are listed (i.e.: -15 to -1 and +1 to +10). If REQUEST(R) Name is used, the assignments for system program "Name" are listed. In the latter case, the DOS keyboard monitor searches the system program information file (.COMM) for the program's .DAT slot usage table. A sample listing for REQUEST is shown in Appendix B.

4.3 List Directory Command

The command "DIRECTORY n" (DIRECT or D) causes the directory of disc unit n to be listed. The null assumption for n is the system disc unit (unit no. on which DOS is stored). A directory listing for the system disc unit is given in Appendix A. The directory listing consists of:

- File name
- File extension
- Octal disc starting address of the file
- Decimal sector count of the file
- Decimal number of files (segments)
- Decimal number of deleted files
- Decimal number of vacant sectors

4.4 New Directory Command

The NEWDIRECT command is used to initialize the directory for a disc unit. The directory length assigned is four (4) sectors (56₁₀ entries maximum). The initial directory contains all zeros except for the first 8 words which contain:

```

560411          .SIXBT /.DIR@XXX/
2200000
303030
4000000 + disc unit addr  Directory file S.A.
0000000
0000000
000700          Directory file W.C.
0000000

```

4.5 Teletype, Lineprinter, and Display Commands

The commands TELETYPE (TTY), LP, and DISPLAY (S) cause the DOS keyboard monitor listings to be output on the console teleprinter, lineprinter, or CRT display respectively. The normal mode is LP if the system has a lineprinter; otherwise the normal mode is TTY.

4.6 Halt Command

If the HALT command is given to the DOS keyboard monitor, all subsequent IOPS errors printout will be followed by a program halt. The normal condition is that control is transferred to a routine that waits for the operator to type ↑P, ↑C, ↑S, or ↑Q. The command is ineffective if DDT is loaded.

4.7 No-Bootstrap Command

The NOBOOT command causes .SCOM+Ø (location 1ØØØ -see Appendix C) to be set to the highest address in memory (17777 for 8K, 37777 for 16K memory, etc.), ↑C to be disabled, and the ↑Q feature to be deleted. A total of $165_8 = 117_{10}$ locations are retrieved, 48_{10} from the bootstrap and 69_{10} from the ↑Q routine. After using NOBOOT, the bootstrap must be reread into memory in order to reactivate the system. The effect of using NOBOOT is:

Non-DDT user programs	User programs are loaded from the top of memory down (17777_8 , etc.) rather than the location below the bootstrap ($1772Ø_8$, etc.). The total open memory is increased by $165_8 = 117_{10}$ locations.
DDT user programs	DDT is loaded from the top of memory down. User programs load below DDT. The total open memory is increased by 117_{10} locations.
Relocatable system programs (e.g.: CHAIN, COMPRES)	The relocation factor is reduced by $1Ø5_8$. The total open memory is increased by $165_8 = 117_{10}$.
Absolute system programs (e.g.: EDIT, MACRO, F4)	The total open memory (e.g.: MACRO's symbol table buffer) is increased by $1Ø5_8 = 69_{10}$ locations. The program's relocation factor does change.

4.8 Get Core Image Command

The GET command is for reloading of a core image which was saved on disc memory by the monitor's ↑Q feature. The possible formats for the GET command are:

Get m Restores core starting at disc address
m (octal)

$0 \leq m \leq 7667_8$ (8K system)*

$0 \leq m \leq 7556_8$ (16K system)*

$0 \leq m \leq 7445_8$ (24K system)*

$0 \leq m \leq 7333_8$ (32K system)*

Get n NAME EXT Restores core from the core image file
(on disc unit n) with the specified name;
extension. The null assumption for EXT
is DMP.

The automatic starting address of a DMP file is

1. DDT's starting address if DDT is present
- or 2. Users ↑P address if a .INIT was issued to the teleprinter
device handler
- or 3. The ↑S address if neither 1 or 2 are present.

4.9 Enter DMP file command

The ENTER command's function is to create a directory entry for a core image on disc that was created by the monitor ↑Q feature. The format for the ENTER command is:

ENTER m n NAME EXT

where:

m = starting disc address of the core image dump

n = disc unit number

NAME = file name

EXT = file extension, null assumption is DMP.

**These figures are based on a maximum disc address of 7777_8*

Example:

The user loads a program into core memory. When all program initialization has been completed, the user strikes ↑Q on the console teletype keyboard and then types 7000 (a disc address), and a carriage return. The core has now been saved starting at disc sector 7000₈. The operator can now reload the keyboard monitor and type:

\$ ENTER 7000 2 PROG↓

which causes the core image starting at disc address 7000₈ to be transferred to disc unit 2 and be entered in the disc unit 2 directory as file PROG DMP. The file PROG DMP can now be transferred, by PIP, to dec tape for permanent storage of the core image.

4.10 Octal Disc Map Command

The DISCMAP (DKMAP or DM) instruction generates an octal listing of one to three disc sectors on the listing device. The format for this command is:

DISCMAP S₁ S₂ S₃

or DKMAP S₁ S₂ S₃

where the S_i are octal disc sector addresses. A sample listing for DKMAP is shown in Appendix A.

4.11 Load System Progress Commands

All commands to the keyboard monitor which are not listed in Table 4.1 are interpreted by the keyboard monitor as commands that load a system program. The keyboard monitor loads and searches the .COMM file for the load program command (e.g.: EDIT, PIP, etc.). If the command is found in .COMM, loading of the specified system program then takes place. If the command is not found in the .COMM file, the keyboard monitor responds by typing:

WHAT?

\$

5.0 INPUT/OUTPUT MONITOR

The input/output monitor resides in the lower part of core memory ($1734_8 \cong 990_{10}$ locations), and is loaded and initialized whenever the keyboard monitor is loaded into core. The input/output monitor performs the following functions:

- CAL Handler - IOPS functions dispatcher
- CAL* Handler - IOPS ASCII packing and unpacking routines
- Program Interrupt Skip Chain - PI service routines dispatcher
- Error diagnostic printout routine
- API trap instructions and transfer vectors for service routines
- System Communications Table (.SCOM) - Contains system information
- Device Assignment Table (.DAT) - Contains I/O device assignments
- Disc Unit Communications Table - Contains disc unit starting addresses
- Interrupt (PI or API) Initialization Routine (.SETUP) - Sets interrupt transfer vectors
- Teleprinter Device Handler, TTA. - All ASCII modes, all functions
- Save Core Routine ($\uparrow Q$) - Saves core on a specified part of disc

5.1 CAL Handler

The I/O monitor CAL handler accepts Input/Output requests from system or user programs and dispatches control to the appropriate device handler program. The functions 1 to 13_8 are handled by the device handlers, while functions No. 15_8 and 16_8 (.EXIT and .SETUP) are handled by the I/O monitor. The function .TIMER (14_8) is illegal in the DOS I/O monitor. A discussion of the CAL format and IOPS functions can be found in reference 9. A summary of the IOPS functions is given in Appendix F.

5.2 CAL* Handler

The CAL indirect instruction is used by the DOS I/O monitor to call either an IOPS ASCII packing or unpacking routine. These routines save and restore the MQ and calculate the appropriate word pair count.

The legal CAL* instructions are:

Ø376ØØ	Pack ASCII with debreak
Ø377ØØ	Pack without debreak
Ø376Ø1	Unpack ASCII with debreak
Ø377Ø1	Unpack without debreak

An example program to pack IOPS ASCII is:

```

PACK = 376ØØ
      DZM  CNT
      LAC  (BUFFER+2)
      DAC  LB
      .
      .
      .
      /AC = ASCII character
      PACK
CNT    Ø           /character counter
LB     Ø           /data pointer
      /AC = word pair count (not incl. header)
      .
      .
      .

```

An example program to unpack IOPS ASCII is:

```

UNPACK = 376Ø1
      DZM  CNT
      LAC  (BUFFER+2)
      DAC  LB
      .
      .
      .
      UNPACK
CNT    Ø           /character counter
LB     Ø           /data pointer
      /AC = unpacked character
      .
      .
      .
BUFFER .BLOCK 42      /IOPS ASCII line buffer

```


5.3 Program Interrupt Skip Chain

The resident I/O monitor contains a routine that "catches" program interrupts, polls the I/O device flags to determine which device caused the interrupt, and dispatches control to the appropriate interrupt service routine. The device flag polling routine is called the "skip chain". The skip chain also contains the service routine transfer vectors (TV) for both program interrupt (PI) devices and automatic priority interrupt (API) I/O devices. The skip chain is located at the end of the resident I/O monitor. The PI skip chain format is:

```

      .LOC      Ø
      Ø          /stores status + PC
      EEM        /extended mode
      JMP      .IOSK    /go to skip chain
      .
      .
      .
/PI SKIP CHAIN
.IOSK PSF          /punch flag
      SKP
      JMP*      INT1
      TSF        /teletype flag
      SKP
      JMP*      INT2
      KSF        /keyboard flag
      SKP
      JMP*      INT3
.ERØ3 IORS        /illegal interrupt
      DAC      .MED    /I/O status
      LAW      3
      JMP      .MED+1
INT1  _____ } PI service routines
INT2  _____ } transfer vectors
INT3  _____ }
INT4  _____ } API service routines
INT5  _____ } transfer vectors
/END OF RESIDENT MONITOR

```

5.4 IOPS Error Diagnostic Routine (.MED)

The error diagnostic routine is entered by the CAL handler and device handlers in order to report error conditions to the user via the teleprinter. A list of the errors is given in Appendix E.

5.5 API Trap Instructions

The API trap locations (40 to 77) are a part of the DOS I/O monitor. All API trap instructions are JMS* (jump to subroutine, indirect). The transfer vectors are established as follows:

<u>Channel use</u>	<u>Transfer Vector Location</u>
Software API (locations 40-43)	112 - 115
Hardware Interrupt	T.V. follows skip chain
Unused channel	Error diagnostic routine (IOPS 30)

5.6 System Communication Table (.SCOM)

The system communication table is described in reference 9, p. 2-13 and in Appendix C. The only variations in the DOS I/O monitor .SCOM table are:

- If NOBOOT is used, .SCOM+0 is set to the highest address of core memory
- .SCOM+4 is as follows:

Bit 0	1 = API
Bit 1	1 = EAE
Bit 2	1 = TTY is 35/37
Bits 3,4	memory extension
	i.e.: 8K 00
	16K 01
	24K 10
	32K 11
Bit 5	1 = DOS; 0 = Dectape

Bits 0 and 1 of .SCOM+4 will always be set to one's in the DOS I/O monitor since the DOS monitor *must* have EAE and API in order to operate.

5.7 Device Assignment Table (.DAT)

The CAL handler (IOPS functions dispatcher) dispatches control to I/O device handlers via the Device Assignment Table (.DAT). The .DAT acts as a dispatch table for the CAL handler. The .DAT is centered at location 135₈ (141₈ in System V5A) in the DOS I/O monitor and extends 15₈ locations down and 10₈ locations up from that point

(i.e.; 120_8 to 145_8 or 124_8 to 151_8 in V5A). The format of a .DAT entry is:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17		
u	Device handler No.	Initialized condition. (Before device handlers have been loaded)
u	Device handler address	Condition after device handler is loaded

where u denotes unit number from 0 to 7.

5.8 Disc Unit Communication Table (.DKCOM)

The DOS I/O monitor contains a table, .DKCOM, which has as its entries the starting disc addresses of all disc units. The first entry of this table is at location 160_8 . Illegal disc units have a negative number stored in their corresponding table entry (location .DKCOM + u). The locations following .DKCOM + 7 (167_8) also contain system information. The entire .DKCOM table is shown in figure 5.1. (note: The corresponding disc map is shown in Figure 3.1)

5.9 Interrupt Initialization Routine (.SETUP)

Device handlers in the PDP-9 advanced software system are loaded relocatable. Therefore, there must be a means to initialize interrupt transfer vectors with the handler's service routine address. This function is performed by the .SETUP command, both for PI and API service routines. The calling sequence to set up a transfer vector in the P.I . skip chain is:

```

CAL
16
IOTSKP          /eg: PSF
SRVC            /service routine address

```

The calling sequence for the setup of an API transfer vector is:

```

CAL + CHANNEL NO.
16
SRVC            /service routine address

```

Core Location		Contents	Use
Symbolic	Octal		
.DKCOM+0	160	0	Disc Unit 0 SA
+1	161	2000	Disc Unit 1 SA
+2	162	4000	Disc Unit 2 SA
+3	163	7400	Disc Unit 3 SA
+4	164	410000	Units 4-7 illegal
+5	165	760000	
+6	166	760000	
+7	167	760000	
.DKADH	170	10000	Disc High Addr + 1
.SYSU	171	0	System Unit Number
.FIRST	172	JMP FIRST	Program Interrupt restore routine
.STRT	173	JMP .START-1	Program Start (AC = SA)
.TABSZ	174	-12 ₈	TTY 33 Tab Size
.FRMSZ	175	-20 ₈	TTY 33 form feed size

Figure 5.1 .DKCOM, Disc Unit
Communication Table

The PI skip IOT's and the API trap addresses are specified by the skip chain I/O configuration file .SKIO. If a skip IOT or API channel number is used in a .SETUP that was not specified in the .SKIO file, an IOPS error will occur (see Appendix E).

5.10 Teleprinter Device Handler - TTA

The teleprinter device handler is a part of the resident DOS I/O monitor. The operating characteristics are nearly identical to that described in Reference 9. The main exception is that in addition to $\uparrow P$ and $\uparrow T$, $\uparrow O$ may be used for manual program restart. The calling sequence for the initialization of the teleprinter device handler establishes the restart addresses. The .INTT format is:

$$\begin{array}{l} \text{CAL} + a_{9-17} \\ 1 \\ b_{0-1} + \text{Restart Address} \\ 0 \end{array}$$

where: a = dat slot number
and b = 00 for $\uparrow P$
 01 for $\uparrow C$
 10 for $\uparrow T$
 11 for $\uparrow O$

5.11 Save Core Routine - $\uparrow Q$

The DOS I/O monitor has the built-in capability to save the contents of core on any part of the disc memory. The procedure is:

1. Determine a portion of disc where core can be saved. The number of sectors required to save core memory is:

<u>Memory Size</u>	<u>No. Words</u>	<u>Disc Sectors</u>
8K	17720 ₈	111 ₈
16K	37720 ₈	222 ₈
24K	57720 ₈	333 ₈
32K	77720 ₈	444 ₈

2. Strike $\uparrow Q$ on the teletype keyboard. The computer will echo " $\uparrow Q$ " and "DKAD ".

3. Type the octal starting disc address and a carriage return.

The contents of core can be restored by using the keyboard monitor GET command (Section 4.8). The starting address for the file loaded from disc is:

	DDT's starting address (†T)
or	User's starting address (†P)
or	The program start address (†S).

6.0 KEYBOARD MONITOR INFORMATION FILES

The DOS Keyboard Monitor and system loader are written in such a way that system program changes and additions can be made without rewriting the latter programs. Two files on the system disc unit account for this flexibility:

1. .SKIO Skip chain and I/O configuration file
2. .COMM System program information file.

These files completely identify the system hardware I/O, device handlers, and all system programs. Whenever system information is required by the keyboard monitor, it accesses these files and performs a search to locate the necessary information.

Whenever the system programmer wishes to incorporate a system change, he can do so by simple modification of the .SKIO and/or .COMM files on the disc. The changes, or specifications, that can be incorporated in these files are:

1. Addition of a hardware flag to the PI skip chain
2. Addition of an API trap instruction and transfer vector
3. Addition of an I/O device handler name to the system
4. Addition of a system program name to the keyboard monitor command table
5. Modification of an existing system program .DAT usage table, loading instructions, etc.

6.1 Skip Chain and I/O Configuration File - .SKIO

The SKIO file specifies the interrupt flags and API channels used by the system's I/O hardware, and the names of all device handlers in the DOS library. The DOS keyboard monitor reads the .SKIO file during its initialization and builds a system characterization table.

The following example demonstrates the .SKIO file format: Suppose the following hardware and device handlers are to be built into the file.

Program Interrupt Flags:	TSF (Teleprinter)
	PSF (Punch)
	KSF (Keyboard)
API Channels:	44
	47
	50
	51
Device Handlers:	NONE (must be #0)
	TTA (Teletype)
	PPA
	PRA
	DKC (systems device handler)
	DKA
	DTA
	LPA

The .SKIO listing for these characteristics would be:

```

      .TITLE .SKIO FILE
/ SKIP CHAIN AND I/O CONFIGURATION INFORMATION FILE
/ PI FLAG DEFINITIONS:
TSF=700401
PSF=700201
KSF=700301
      .FULL
      .LOC 0
/ I/O HARDWARE TABLE
SKIRL  SKEND-SKIRL-1  / TABLE ENTRY COUNT
      TSF&77777  / PI FLAGS (SKIP CHAIN ORDER)
      PSF&77777
      KSF&77777
      JMS+44  / API CHANNELS
      JMS+47
      JMS+50
      JMS+51
SKEND=.
```

```

/ I/O DEVICE HANDLER TABLE
IOTBL      IOEND-IOTBL-3      / TABLE ENTRY COUNT
          SYS-IOTBL-3        / SYSTEM DEVICE HANDLER NO.
          TTY-IOTBL-3        / TELETYPE DEVICE HANDLER NO.
          .SIXBT /NON/       / DH #0 = NONE
TTY         .SIXBT /TTA/
          .SIXBT /PPA/
          .SIXBT /PPA/
SYS         .SIXBT /DVC/
          .SIXBT /DKA/
          .SIXBT /DTA/
          .SIXBT /LPA/
IOEND=.
      .END

```

An I/O hardware table entry has the following format:

<u>Bits (0-2)</u>	<u>Bits (3-17)</u>
0	Normal Skip IOT & 77777*
4	Reverse skip IOT & 77777*
1	API Channel No.

The device handler table entries are six-bit-trimmed ASCII words corresponding to the device handler names.

There are two requirements of the keyboard monitor that must be followed in the .SKIO file:

1. Device handler no. 0 must be "NONE"
2. The .SKIO file must immediately precede the keyboard monitor file on the disc. (i.e.: .SKIO disc address = 4, KM9 disc address = 5).

6.2 System Program Information File - .COMM

The .COMM file contains a symbol table of the legal keyboard monitor commands to load systems programs. An entry in the .COMM symbol table contains:

1. Command characters - six-bit-ASCII trim. One to nine character commands are allowable.

**A normal skip IOT skips when the flag is set. The reverse skip IOT skips when the flag is not set.*

2. System Program negative dat slot usage
3. System Program positive dat slot usage
4. ASCII messages for the keyboard monitor REQUEST command
5. Program-type status bits and miscellaneous loading instructions
6. System program file-name and extension.

The exact format of a .COMM symbol-table entry is shown in figure 6.1. The following example listing shows a typical .COMM file for system programs EDIT, PIP, and COMPRES.

```

.TITLE .COMM FILE
/ KMS SYSTEM PROGRAM LOAD COMMANDS SYMBOL TABLE
.FULL
.LOC 0
END-3/14          / +TABLE ENTRY COUNT
3                 / WORDS PER SYMBOL
11                / WORDS PER VALUE PART

.SIXBT /EDIT@@@@/      / COMMAND CHARACTERS (9)
610300            / NEGATIVE DAT SLOT USAGE
000000            / POSITIVE DAT SLOT USAGE
EDITDAT           / DAT SLOT USAGE TABLE
0                 / UNUSED
0                 / UNUSED
400000            / ABSOLUTE SYSTEM PROGRAM SWITCH
.SIXBT /EDIT@SYS/      / FILE NAME:EXTENSION

.SIXBT /PIP@@@@@/
000300
776000
PIPDAT
0
0
400000
.SIXBT /PIP@SYS/

.SIXBT /COMPRES@/
000300
000000
COMDAT
0
0
200000            / RELOCATABLE SYSTEM PROGRAM SWITCH
.SIXBT /DKCOMPRI/

```

END=.

EDTDAI	OUTSCR	/ -15
	INOUT	/ -14
	SECIN	/ -13
	TTOERR	/ -3
	COMSTR	/ -2
PIPDAT	CTLERR	/ -3
	COMSTR	/ -2
	INOUT	/ 1
	INOUT	/ 2
	INOUT	/ 3
	INOUT	/ 4
	INOUT	/ 5
	INOUT	/ 6
	INOUT	/ 7
	INOUT	/ 10
COMDAT	CTLERR	/ -3
	COMSTR	/ -2
INOUT	.ASCII #INPUT/OUTPUT#<177>	
SECIN	.ASCII /SECONDARY INPUT/<177>	
TTOERR	.ASCII /TELETYPE OUTPUT AND ERRORS/<177>	
COMSTR	.ASCII /COMMAND STRING/<177>	
CTLERR	.ASCII /CONTROL AND ERROR MESSAGES/<177>	
OUTSCR	.ASCII #OUTPUT/SCRATCH#<177>	
	.END

Switch	Meaning
A	Absolute System Program
R	Relocatable System Program
IO	Generate I/O configuration above .SCOM +1
F	Store file name; extension in .SCOM + 7, + 10, + 11
L	.LOAD used (Linking loader)
D	DDT type .LOAD
G	GO type .LOAD
N	DDT and no symbols (DDTNS) type .LOAD

Figure 6.1. .COMM Symbol Table Entry Format

7.0 DISC I/O DEVICE HANDLERS

There are four disc I/O device handlers in the DOS I/O system. The basic functions of these routines are approximately the same as for the equivalent decktape I/O handlers. The DOS disc device handler's basic characteristics are:

<u>Device Handler Name</u>	<u>Number of Files</u>	<u>Input</u>	<u>Output</u>	<u>Functions</u>	<u>Words</u>
DKA.	3	X	X	All	2211 ₈
DKB.	2	X	X	Limited	1526 ₈
DKC.	1	X		Limited	567 ₈
DKD.	1	X	X	All	1431 ₈

The meanings of "all" and "limited" functions with respect to these handlers are:

<u>Function</u>	<u>All</u>	<u>Limited (I/O)</u>	<u>Limited (Input Only)</u>
.INIT	X	X	X
.DELETE	X		
.RENAM	X		
.FSTAT	X		
.RDIR	X		
.SEEK	X	X	X
.ENTER	X	X	
.CLEAR			
.CLOSE	X	X	X
.MTAPE			
.READ	X	X	X
.WRITE	X	X	
.WAIT	X	X	X
.WAITR	X	X	X
.TRAN	X		

7.1 Disc Device Handler Characteristics

The operation of the DDC Disc I/O device handlers is much the same as for Decktape device handlers. The basic differences are:

- The Disc directories and therefore the disc device handlers do not require bit-file-maps
- The internal data buffers are $112_{10} = 160_8$ words long, one for each file. (e.g.: DKB. has two 112_{10} word data buffers.)

- - Line buffer headers are not stored as a part of the data on the disc
- - Line buffers may extend from one sector into the next sector
- - The disc device handlers do not have a .CLEAR function
- - The handlers DKA. and DKD. have a "Read Directory Entry" function which can be used to read entries selectively or non-selectively from a disc directory
- - The functions .DELETE, .RENAM, .FSTAT, and .RDIR store the number 2_8 in bits 0-2 of word 3 (file specification pointer) of the CAL sequence for those functions
- - The standard line buffer size reported by .INIT is 377_8 but the maximum allowable line buffer length is 162_8 words including the header word pair. The reason that 377 is reported by .INIT is to inform PIP, MACRO and other system programs that the DK handlers are file-oriented
- - No attempt is made by the disc handlers to fill in vacant spots on disc left by deleted files. All new files are written following the last undeleted file on the disc unit. The "disc file compressor" program must be used periodically to fill in these vacant areas on disc
- - Disc files are deleted by one of two means:
 - 1) If the directory entry is the last entry in the directory, the entry is cleared to zero. All entries immediately preceding a cleared entry will also be cleared if the delete bit is set in the directory entry.
 - 2) If the directory entry is not the last entry in the directory, the delete bit is set in the directory entry (Bit 1 of Word 3)
- - A disc unit directory may be any number of sectors in length; however, once that area is filled, the handlers will not automatically expand the directory to a greater number of sectors.

Directory expansion is provided for in the "Disc File Compressor" program

- At the time of this writing, the handlers do not use the disc API interrupt, and control of the CPU is retained by the handlers until all disc transfers are completed. Thus, a .WAITR function will never return control to the "Busy-Return-Address." The maximum time that the disc handler will retain control for the various functions is approximately:

<u>Function</u>	<u>Dead Time (Max.)*</u>
.WRITE	1/60 sec.
.READ	1/60 sec.
.WAIT	negligible
.WAITR	negligible
.CLOSE (input)	negligible
.CLOSE (output)	1/20 sec.
.SEEK	1/30 sec.
.ENTER	1/60 sec.
.TRAN	1/60 sec.
.DELETE	1/30 sec.
.RENAM	1/30 sec.
.FSTAT	1/60 sec.
.RDIR	1/60 sec.

- The only legal data modes for the disc device handlers are IOPS ASCII and IOPS Binary. IOPS Binary data are read from the disc 30_8 words per line buffer (15_8 word pairs including header) unless a lesser number of words reaches the end of the file segment. IOPS ASCII lines are marked with a "1" in bit 17 of the last word of the line, thus variable length ASCII lines can be read and rewritten.
- The maximum file segment length is $77700_8 = 32704_{10}$ words, or $444_8 = 292_{10}$ sectors. Any data file that is longer than the latter word count is segmented by the disc handlers; multiple directory entries are made, one for each segment.

**These figures are based on a disc revolution rate of 3600 rpm and a maximum access time of 1/60 second.*

7.2 Read Directory Entry Function - .RDIR

The read-directory-entry function can be used to read selectively or non-selectively any of the entries in a disc unit directory. The expansion for .RDIR is:

```

                                .RDIR a,D
LOC 0      CAL+4000+a9-17
LOC 1      2      *
LOC 2      D

```

where a = dat slot No.

D = file specification pointer.

The format of the file specification block is:

```

D      Key Word
      .BLOCK 10

```

The device handler transfers the directory entry into locations D+1 to D+10. There are two ways to use .RDIR, selective entry reading and non-selective entry reading:

- 1) To read entries selectively, initialize location D with zero and locations D+1 to D+3 with the name and extension of the desired file. Multiple calls to .RDIR will transfer all directory entries with the specified name; ext, one at a time. When no entry with the specified name; ext is found, control will be returned from .RDIR with AC = 0. (Otherwise, AC = starting sector of the file segment.) When a file segment is found, its entry is transferred into locations D+1 to D+10.
- 2) To read entries unselectively, initialize D with 4000000 and execute repeated calls to .RDIR. The .RDIR function will transfer, one at a time, all undeleted and closed directory entries to locations D+1 to D+10. The contents of the AC on

*The CAL handler places the unit No. in bits 0-2.

**The device handler places 2₀ in bits 0-2 indicating a file-oriented device.

return is the file starting disc address. When the end of the directory is reached, the AC upon return is zero.

.RDIR Example:

Read the directory entry for file named JOE, extension XYZ from .DAT#4.

DZM	FILE	
CAL+4004		/.RDIR 4, FILE
2		
FILE		
SNA		
JMP	NFD	/NOT FOUND
		/FOUND
FILE	Ø	
NAME	.SIXBT /JOE/	/FILE,NAME WD1
	Ø	/FILE NAME WD2
EXT	.SIXBT /XYZ/	/FILE EXTENSION
DA	Ø	
	Ø	
	Ø	
WC	Ø	/FILE SEG WORD COUNT
	Ø	

8.0 DOS UTILITY ROUTINES

There are three routines in the DOS that are intended to aid in the generation of a disc system, and the formatting of disc data:

DKCORE - Disc/core transfer routine

LIBCPY - Library copier

DKCOMP - Disc compressor

DKSAVE - Disc to DEC Tape Save/Restore Routine

The purpose of DKCORE is to perform manually controlled disc-to-core or core-to-disc transfers. LIBCPY copies a library and performs formatting according to DOS requirements. The disc compressor, DKCOMP, allows wasted disc area (as a result of file deletion) to be retrieved by over-laying those areas with other files. DKSAVE is a disc-DECTape block copier for saving entire disc units.

8.1 Disc/Core Transfer Routine (DKCORE)

Initial DOS generation and periodic system maintenance is aided by the DKCORE routine. DKCORE is a hardware-read-in paper tape program which loads in the upper 48_{10} words of any 8192 word memory bank. The load address for DKCORE is therefore the same as for the .ABS paper tape loader:

17720₈ - 8K
 37720₈ - 16K
 57720₈ - 24K
 or 77720₈ - 32K

DKCORE receives its parameters from the switch register (SWR or accumulator switches). The DKCORE operation instructions are:

1. I/O reset. Put the DKCORE hardware-read-in tape in the paper tape reader. Set the address switches to 17720₈ of the memory bank selected for loading of DKCORE and press hardware-read-in (HRM).
2. If DKCORE is already core resident, START 17720₈, etc.
3. Place the starting core address for the transfer in the SWR and press CONTINUE.
4. Place the positive word count for the transfer in the SWR and press CONTINUE.
5. If "read" (disc-to-core) put AC switch 0 down, or if "write" (core-to-disc) place switch 0 up. Set the remaining switches to the starting disc address for the transfer and press CONTINUE.
6. The computer will halt with information in the accumulator:
 0: Transfer Complete - no errors
 non 0: Errors

AC Bit	Meaning, if Bit=1
0	Error (or of 1-3)
1	Parity error on READ
2	Illegal address on WRITE
3	Disc not ready
4	Disc interrupt enables
5	0=READ 1=WRITE

7. To perform another transfer using DKCORE, to to step 3 of these instructions.

Within the discussions in this report, a shorthand notation for DKCORE transfers is used. This nomenclature is shown in the following examples:

- (1) DKCORE MA0, WC700, DA0, W
- (2) DKCORE MA300, WC160, DA6000, R
- (3) DKCORE MA, WC, DA, T
- (4) DKCORE MA, 17720-MA, DA, W

The latter examples mean:

- (1) Core memory address = 0

Word count = 700₈

Disc address = 0

WRITE

- (2) Core memory address = 300₈

Word count = 160₈

Disc address = 6000₈

READ

- (3) This notation indicates a general transfer where the parameters are to be determined by the particular procedure being performed.
- (4) This notation is for a "write" operation with general disc address and core memory address. The word count of 17720-MA indicates all core memory from MA to 17717 inclusive.

8.2 DOS Library Copier (LIBCPY)

The LIBCPY routine performs two formatting functions on library files:

1. All internal (DDT) symbol definitions⁽¹⁰⁾ are deleted from the library file.
2. All data line buffers are re-formatted to a uniform size (user specified).

A uniform line buffer size is required in order for system programs such as UPDATE and .LOAD to function properly with the DOS disc device handlers.

The LIBCPY operation instructions are:

1. Load the keyboard monitor and assign:
 DKØ to -1 and -4
 Library input to +6
 Library output to +7
2. Type "DDT", which calls for linking loader with DDT.
3. When the linking loader has been loaded, enter a command to load LIBCPY.
4. When DDT responds, open and modify the following locations:

LBSIZE/	15
FILE+Ø/	file name .SIXBT C ₁ , C ₂ , C ₃
FILE+1/	file name .SIXBT C ₄ , C ₅ , C ₆
FILE+2/	file extension .SIXBT E ₁ , E ₂ , E ₃
5. Start LIBCPY by typing an apostrophe (" \$G" for V5A DDT)
6. When DDT is restarted, another library can be copied by returning to step 4 of these instructions.

8.3 Disc Compressor (DKCOMP)

Compression of disc files and directories in the DOS is a necessary function since directory entry and file overlaying is not performed by the disc I/O device handlers. A listing of a disc unit directory will indicate when disc compression is necessary. In addition to performing disc compression, DKCOMP will also set the directory file size. By using this procedure, additional space for directory entries can be obtained (the directory entry density is $112/8 = 14_{10}$ entries per disc sector).

DKCOMP is a standard DOS relocatable system program and can be loaded by a keyboard monitor command - COMPRES. When the compressor has loaded, it will ask the operator for two parameters:

- (1) Disc unit no. to be compressed
- (2) The directory length for that unit in number of sectors.

Example:

Compress disc unit no. 2 files and directory, and set the directory size to six (6) sectors (84 entries maximum). The following underlined user commands, entered at the teletype keyboard, caused the latter operation to be performed:

```
$COMPRES↓
DISC COMPRESSOR
TYPE UNIT NO.   2↓
TYPE NO. SECTORS 6↓
```

8.4 Disc Save/Restore Routine (DKSAVE)

The purpose of the DKSAVE system program is to transfer (block mode) the entire data portion of a disc unit to dectape. The program can also be used to transfer (block mode) from dectape to disc.

The keyboard monitor command to load the save routine is:

```
$ SAVE↓
```

The SAVE input-output assignments are:

<u>.DAT</u>	<u>Use</u>
-3	Teletype output (TTA)
-2	Teletype input (TTA)
6	Dectape (DTA or DTD)
10	Disc. (DKA or DKD)

When loaded, SAVE asks the operator for an input and an output device name. These names should be in the form of DTN or DKM. (e.g.: DT2, DK0, DK2, etc.)

The dectapes which are written by SAVE contain 1076_8 blocks of data (equivalent to 2440_8 disc sectors). If the data on a disc unit contain greater than 2440_8 sectors, SAVE will ask that a second dectape be mounted after the first tape has finished. After mounting the new tape, the proper operator response is to type "↑P" on the console teletype.

The SAVE routine uses the entire PDP-9 available memory as a buffer for the disc-dectape transfer. All transfers are performed in multiples of $3400_8 = 1792_{10}$ words (7 dectape blocks or 16_{10} disc sectors). A maximum of 1976 dectape blocks are written starting at block 0. The approximate buffer size is:

<u>Total Memory</u>	<u>Buffer Size</u>	<u>No. Disc Sectors</u>	<u>No. Dectape Blocks</u>
8K	4K	32	14
16K	12K	96	42
24K	20K	176	77
32K	28K	256	112

9.0 SYSTEM GENERATION

There are two methods for generation of a DOS. The first method should be used when new system software (i.e., a new operating system version) is to be incorporated and no copy is available. The second method is for re-generation of the DOS from a DOS copy which can be paper tape, magnetic tape, etc.

9.1 New System Creation

The original form for all absolute system programs in DOS is either an absolute paper tape or dectape core image of the program. Absolute paper tapes are obtained when these programs are assembled. These system programs also reside on a system dectape in the form of an absolute core dump. In either case, the system programmer must determine:

- (1) Program Load Address - MA
- (2) Program Word Count - WC
- (3) Program Starting Address - SA

The memory addresses MA and SA are for an 8K computer. For the case when the program is an absolute core image on dectape, the following must also be obtained:

- (4) Dectape starting block address of the core image - BA

In addition, a starting disc address for the disc program file must be determined:

- (5) Starting disc sector address - DA

When the program in question is in .ABS paper tape form, the following operations should be performed:

- (1) HRM 1772Ø
- (2) DKCORE MA, WC, DA, W

It is also required that a directory entry be constructed for absolute system routines. (See Section 3.2) The following parameters should be entered in the directory for absolute routines:

(1) .SIXBT /NAME/	program file name
(2) .SIXBT /SYS/	file extension
(3) DA	Disc address
(4) MA	Load address
(5) WC	Word count
(6) SA	Starting address

The next available disc address above the file just generated can be calculated by:

$$DA_{next} = DA + \left(\frac{WC + 157_8}{160_8} \right)$$

In this manner, a disc map of the absolute system program files can be constructed.

When the original version of the program to be entered on the disc is in Dectape form, the system programmer should use the dec-tape bootstrap to load the core image into core memory (see Appendix G) and then use DKCORE to write the disc file (see Section 8.1):

DTBOOT U, BA, MA, WC, R

DKCORE MA, WC, DA, W

A directory entry must also be constructed for these files.

The absolute program files in DOS should be ordered in the following manner:

<u>Disc Addr</u>	<u>File</u>	<u>Word Count</u>
0	.DIR XXX	700 ₈
4	.SKIO KM9	160 ₈
5	KM9 SYS	14000 ₈
74	.SYSLD SYS	
.	all absolute System	
.	Programs (e.g.: EDIT, PIP)	
.	and .COMM in any order.	

A Source directory file should be prepared (.DIR SRC), containing all the absolute program directory entries. The assembler output from this assembly should be a hardware-read-in paper tape (.FULL). This file, and other .FULL data files, can be transferred to disc by:

(1) HRM 0

(2) DKCORE 0, WC, DA, W

Once all absolute program files have been entered on the disc, the system programmer should:

- (1) Use UPDATE and LIBCPY to format a DOS library.
- (2) Use PIP (Dectape System with DOS disc I/O handlers) to transfer all the remaining files to the disc. (all files with extension BIN)
- (3) Use PIP to generate a copy (Dectape) of all files on the disc system; or, use SAVE to create a disc image on Dectape.

9.2 System Generation by Copying with PIP

When a DOS copy is available, the system generation procedure is somewhat simpler than for a new system. The following steps should be followed when copying the DOS from another device (Dectape).

- (1) Clear the system disc directory:

- Zero memory locations 0-677:
- Deposit

<u>Location</u>	<u>Contents</u>
0/	560411
1/	2200000
2/	303030
3/	4000000
4/	0000000
5/	0000000
6/	000700
7/	0000000

- (2) Transfer all absolute program files:

- Load the Dectape system with DOS disc I/O device handlers
- Call PIP and transfer all absolute program files in the exact order that they appear in .DIR SRC.

- (3) Re-write the system disc directory:

- Obtain the .DIR HRM paper tape (assembler output from .DIR SRC)
- HRM 0
- DKCORE 0, 700, 0, W

(4) Transfer all BIN-files:

- Load the Dectape system with DOS disc I/O device handlers
- Call PIP and transfer all BIN-files on the DOS copy to the disc.

10.0 DOS MODIFICATIONS

Several system changes can be made in the DOS. The procedures for the various modifications are listed in the following sections. The types of modifications possible are:

- Memory size change
- Teletype model (auto-tab)
- Hardware interrupt facilities
- I/O device handler additions
- Disc resident library updating
- Addition of a system program.

10.1 Modify Core Memory Size

The PDP-9 software is written so that the system can operate in and utilize 8K, 16K, 24K, or 32K words. To change the system core size, bits 3 and 4 of .SCOM+4 (in KM9) should be set to the highest bank in the memory, i.e.:

<u>Core Size</u>	<u>.SCOM+4 bits 3, 4</u>
8K	00
16K	01
24K	10
32K	11

To perform core size modification:

- (1) DKCORE 0,140000,5*,R
- (2) Set location .SCOM+4 (104)

*KM9 SYS disc address

(3) DKCORE Ø,14ØØØ,5*,W

(4) Transfer KM9 to DOS Dectape Copy.

10.2 Modify Teletype Model

The DOS Keyboard monitor can be used with either a model 33 teleprinter or model 35/37 with automatic tabs. When in model 33 mode, the monitor replaces tab characters (Ø11 ASCII) with multiple spaces. Bit 2 of .SCOM+4 indicates the teletype model:

<u>Teletype Model</u>	<u>.SCOM+4 Bit 2</u>
Model 33	Ø
Model 35/37	1

To permanently modify the system tabbing feature:

- (1) DKCORE Ø,14ØØØ,5*,R
- (2) Set location .SCOM+4 (1Ø4)
- (3) DKCORE Ø,14ØØØ,5*,W
- (4) Transfer KM9 to DOS Dectape copy.

10.3 Addition of Hardware Interrupts

The addition of hardware flags to the program interrupt skip chain or API trap instructions and their corresponding transfer vectors is accomplished by modification of the .SKIO Keyboard Monitor information file. (See Section No. 6.1) The steps for addition (or deletion) of hardware interrupt facilities are:

- (1) Edit and Assemble .SKIO SRC
- (2) HRM Ø
- (3) DKCORE Ø,16Ø,4**,W
- (4) Transfer .SKIO to DOS Dectape Copy

*KM9 SYS disc address
 **.SKIO KM9 disc address

10.4 Addition of I/O Device Handlers

An I/O device handler can be added to the system by

- (1) Add the relocatable binary device handler to the system library (.LIBR BIN) (see Section 10.5).
- (2) Add the device handler name to the .SKIO information file.

The system programs UPDATE, LIBCPY, and PIP are used to perform modification of the system library. Modification of the .SKIO file is the same as in Section 10.3.

10.5 Updating a Disc Resident Library

Updating of a disc resident library (e.g.: inserting a new library routine) is identical to the Dectape procedure except that disc formatting must also be performed. The LIBCPY (Section 8.3) routine should be used for disc library formatting so that the binary file will have uniform line buffer sizes. The necessary steps for disc library updating are:

- (1) Use UPDATE to update with dectape assigned to the output
- (2) Use LIBCPY to copy and format the library with LBSIZE=15₈.

10.6 Addition of a System Program

System programs in the DOS are of two types:

Absolute - The system program disc file is an absolute core image of the routine.

Relocatable - The system program disc file is in relocatable linking loader binary format.

The procedure for adding a system program file to the disc differs for each type of program. When the system routine to be added is relocatable, PIP can be used to transfer the file in question to the system disc unit. When the program is absolute, system creation procedures should be used for that program only (see Section 9.1) so that the necessary directory entry will be made in the system unit disc directory.

When a system program of either type is added, the .COMM file must be modified to enable the keyboard monitor to recognize commands to load the new program:

- (1) Edit the .COMM source file to add a keyboard monitor command
- (2) Assemble .COMM to obtain a .FULL binary tape
- (3) Use DKCORE to re-write the .COMM file on disc
- (4) Transfer .COMM to the DOS Dectape copy.

10.7 System Program Modification

When an existing system program is to be modified, it is not necessary to modify the .COMM file unless the I/O assignments for the system program are to be changed. If the system program file is relocatable, PIP should be used to re-write the file. If the program is absolute, it may also be necessary to modify the directory entry MA, WC, and SA for that program, for which Section 9.1 procedures should be used.

BIBLIOGRAPHY

1. *"An Interactive Graphics Display System for Hydrologic Analysis"*, K. M. Busness, Dec. 1970, BNWL-1525.
2. *"The FFTF Hybrid Computer System"*, K. M. Busness, Sept. 1970, BNWL-1498.
3. *"Line Printer Interface to the PNL Hybrid Computer Facility"*, K. M. Busness, August 1970, BNWL-B-3.
4. *"Analog-Hybrid Dynamics Simulation of the FFTF Reactor and Heat Transport System"*, C. D. Flowers and L. H. Gerhardstein, April 1968, BNWL-707.
5. *"FFTF Process Data Display Prototype"*, B.E. Dozer, April 1970, BNWL-1330.
6. *"PDP-9/Gould 4800 Device Handler and Graphics Library"*, L. H. Gerhardstein, March 1971, BNWL-B-58.
7. *"PDP-9 User Handbook"*, Digital Equipment Corp., Maynard, MA, 1968.
8. *Service Manual for Magnetic Disc Memory System Model No. 14120*, Digital Development Corporation, San Diego, CA, 1969.
9. *"PDP-9 Advanced Software System Monitors"*, Digital Equipment Corp., Maynard, MA, 1968.
10. *"Utility Programs - Advanced Software System"*, Digital Equipment Corp., Maynard, MA, 1967.
11. *"A Systems Approach to Management of The Hanford Ground Water Basin"*, D. B. Cearlock, August 1971, BNWL-SA-3860.

APPENDIX A

SYSTEM DIRECTORY LISTING AND OCTAL LISTING

APPENDIX ASYSTEM DIRECTORY LISTING AND OCTAL LISTING

Directory listings are created by the keyboard monitor DIRECT command (refer to Section 4.3). Octal listings of disc sectors are generated by use of the DISMAP instruction (refer to Section 4.10). These listings, for the system directory of the DOS, are shown in Figures A.1 and A.2.

The directory listing contains:

- File name
- File extension
- Octal disc address of file
- Decimal no. of sectors in file
- Decimal no. of un-deleted files on unit
- Decimal no. of deleted files
- Decimal no. of vacant sectors on unit

Figure A.2 shows the octal values of directory entries (for entry format, see Section 3.2). Each entry contains eight (8) words; thus, each 8 word line from Figure A.2 is a directory entry for some file, e.g., Line 040 of Sector 0 is the directory entry for the system routine "PIP". The parameters for that file are:

File name	=	PIP
File extension	=	SYS
P	=	1 (file protected)
D	=	0 (un-deleted)
O	=	0 (file not-open)
S	=	0 (un-segmented)
Disc Addr.	=	105
Load Addr.	=	10051
Word Count	=	7566
Start. Addr.	=	10051

The entry at line 140 of Sector 1 is a blank indicating "end-of-directory" and the location where the next entry will be written. The starting disc address for the next file would be determined from the previous file parameters:

$$\begin{aligned}
 DA_{K+1} &= DA_K + \left(\frac{WC_K + 157_8}{160_8} \right) \\
 &= 1624_8 + \frac{1000_8 + 157_8}{160_8} \\
 &= 1624_8 + 5 \\
 &= 1631_8
 \end{aligned}$$

The entry at line 120 of Sector 1 (•LIBR BIN) has the S-bit set to a 1(0400000) indicating a segmented file; thus, the next file entry has the same name and extension. (Also see Figure A.1.)

DIRECTORY LISTING
DISC UNIT 0

FILE NAME	FILE EXT	DISC ADDR	TOTAL SECT
.DIR	XXX	00000	4
.SKIO	KMO	00004	1
KMO	SYS	00005	55
.SYSLD	SYS	00074	9
PIP	SYS	00105	36
UPDATE	SYS	00151	19
EDIT	SYS	00174	22
EXECUT	SYS	00222	4
HVMAC	SYS	00226	69
MACRO	SYS	00333	51
F4	SYS	00416	56
.COMM	KMO	00506	4
DKCOMP	BIN	00512	7
MXEDIT	BIN	00521	11
LIQCPV	BIN	00534	5
TTBIT	BIN	00541	45
.LOAD	BIN	00616	14
DDTQ	BIN	00634	20
CHAIN	BIN	00660	19
FOCAL	BIN	00703	35
FNEW	BIN	00746	2
LRIN	BIN	00750	10
DKSAVE	BIN	00762	7
.LIBIO	BIN	00771	119
.LIBR	BIN	01160	292
.LIBR	BIN	01624	5

26 FILES

0 DELETED FILES

103 VACANT SECTORS

Figure A.1 - System directory
listing for DOS

DISC OCTAL MAP

SECTOR 00000

000/	560411	220000	303030	400000	000000	000000	000700	000000
010/	562313	111700	131571	400004	000000	000000	000160	000000
020/	131571	000000	233123	400005	000000	000000	014000	000000
030/	562331	231404	233123	400074	000000	015750	001750	016540

040/	201120	000000	233123	400105	000000	010051	007566	010051
050/	252004	012405	233123	400151	000000	013670	003750	013671
060/	050411	240000	233123	400174	000000	013000	004636	013000
070/	053005	032524	233123	400222	000000	017100	000536	017100

100/	103115	010300	233123	400226	000000	001000	016720	004233
110/	150103	221700	233123	400333	000000	004550	013076	004550
120/	066400	000000	233123	400416	000000	003465	014152	004143
130/	560317	151500	131571	400506	000000	000000	000700	000000

140/	041303	171520	021116	000512	000000	000000	001364	000000
150/	153005	041124	021116	000521	000000	000000	002244	000000

SECTOR 00001

000/	141102	032031	021116	000534	000000	000000	000714	000000
010/	242422	112400	021116	000541	000000	000000	011600	000000
020/	561417	010400	021116	000616	000000	000000	003010	000000
030/	040424	710000	021116	000634	000000	000000	004124	000000

040/	031001	111600	021116	000660	000000	000000	004050	000000
050/	061703	011400	021116	000703	000000	000000	007504	000000
060/	061605	270000	021116	000746	000000	000000	000330	000000
070/	140211	160000	021116	000750	000000	000000	002020	000000

100/	041323	012605	021116	000762	000000	000000	001330	000000
110/	561411	021117	021116	000771	000000	000000	031700	000000
120/	561411	022200	021116	041160	000000	000000	077700	000000
130/	561411	022200	021116	001624	000000	000000	001000	000000

140/	000000	000000	000000	000000	000000	000000	000000	000000
150/	000000	000000	000000	000000	000000	000000	000000	000000

Figure A.2 - System Directory
Octal listing for DOS

APPENDIX B
STANDARD .DAT ASSIGNMENTS

APPENDIX BSTANDARD .DAT ASSIGNMENTS

The keyboard monitor REQUEST command (refer to Section 4.2) generates I/O assignment listings as shown in Figures B.1 and B.2. Figure B.1 shows the entire Device Assignment Table (DAT) standard assignments (null assumptions). Figure B.2 shows REQUEST-listings for system programs EDIT, F4, and DDT. Device assignments can be altered by the keyboard monitor "ASSIGN" command (Section 4.1).

ALL INPUT/OUTPUT ASSIGNMENTS

.DAT	DEVICE	USE
-15	DKA3	OUTPUT
-14	DKA2	INPUT
-13	DKB2	OUTPUT
-12	TTA0	LISTING
-11	DKB2	INPUT
-10	PRA0	INPUT
-7	DKC0	SYSTEM INPUT FOR KM9 & .SYSLD
-6	PPA0	OUTPUT
-5	NONE	EXTERNAL LIBRARY
-4	DKC2	SYSTEM INPUT
-3	TTA0	TELEPRINTER OUTPUT
-2	TTA0	KEYBOARD INPUT
-1	DKC0	SYSTEM INPUT FOR .LOAD
1	TTA0	USER
2	TTA0	USER
3	PRA0	USER
4	PPA0	USER
5	DTA1	USER
6	DTA2	USER
7	DKA2	USER
10	DKA3	USER

Figure B.1 Standard .DAT assignments

\$ REQUEST EDIT

EDIT INPUT/OUTPUT ASSIGNMENTS

.DAT	DEVICE	USE
-15	DKA3	OUTPUT/SCRATCH
-14	DKA2	INPUT/OUTPUT
-10	PRA0	SECONDARY INPUT
-3	TTA0	TELEPRINTER OUTPUT AND ERRORS
-2	TTA0	COMMAND STRING

\$ R F4

F4 INPUT/OUTPUT ASSIGNMENTS

.DAT	DEVICE	USE
-13	DKB2	OUTPUT
-12	TTA0	LISTING
-11	DKB2	INPUT
-3	TTA0	CONTROL AND ERROR MESSAGES
-2	TTA0	COMMAND STRING

\$ REQUEST DDT

DDT INPUT/OUTPUT ASSIGNMENTS

.DAT	DEVICE	USE
-5	NONE	USER LIBRARY
-4	DKC2	USER PROGRAM(S)
-3	TTA0	CONTROL AND ERROR MESSAGES
-2	TTA0	COMMAND STRING
-1	DKC0	SYSTEM LIBRARY

\$

Figure B.2 - EDIT, F4, and DDT .DAT assignments

APPENDIX C

SYSTEM COMMUNICATION TABLE (,SCOM)

APPENDIX CSYSTEM COMMUNICATION TABLE (.SCOM)

The System Communication Table (.SCOM) provides a list of registers that can be referenced by the monitor, IOPS, and system programs. A complete list of .SCOM entries, and the purpose of each, is given in Table C.1. The System Communication Table begins at location 100_g.

TABLE C.1
SYSTEM COMMUNICATION TABLE (•SCOM) ENTRIES

Word	Purpose
•SCOM	First free register below resident portion of System Bootstrap
•SCOM + 1	First free register above resident KM-9
•SCOM + 2	First free register in unused memory
•SCOM + 3	Last free register in unused memory
•SCOM + 4	Hardware options available: Bit 0 1 = API Bit 1 1 = EAE Bit 2 1 = TTY is 35/37 Bit 3,4 Memory Bank bits 00 8K 01 16K 10 24K 11 32K Bit 5 1 = DOS; 0 = Dectape
•SCOM + 5	System program starting location
•SCOM + 6	User starting location (bits 3 through 17), and: Bit 0 1 = DDT Load Bit 1 1 = G Load Bit 2 1 = No-symbol-table Load
•SCOM + 7-11 ₈	Device numbers of Linking Loader's devices. These are used to avoid loading user handlers already in core for the Loader itself.
•SCOM + 12-15 ₈	API software-level transfer vectors
•SCOM + 16	Contains PC on keyboard interrupts
•SCOM + 17	Contains AC on keyboard interrupts
•SCOM + 20*	Bit 0 1 = 12K, 20K, or 28K system Bits 3-17 First free register in extra 4K memory
•SCOM + 21*	Reserved for mag. tape handler
•SCOM + 22*	Reserved for mag. tape handler
•SCOM + 23*	Pointer to •DAT

*Version V5A only

APPENDIX D

DKBOOT AND DKCORE SOURCE PROGRAMS

D.1 DOS Bootstrap

.TITLE DKBOOT

/ PDP-9 DDC DISC SYSTEM BOOTSTRAP
/ OCT 14, 1969
/ L H GERHARDSTEIN
/ BATTTELLE NORTHWEST
/ P O BOX 999
/ RICHLAND, WASHINGTON
/ 99352

DKMA=707002	/ LOAD MEMORY ADDRESS FROM ACCUMULATOR
DKWC=707022	/ LOAD TWO'S COMPLEMENT WORD COUNT FROM ACCUMULATOR
DKRD=707042	/ LOAD DISC ADDRESS FROM ACCUMULATOR AND READ
DKWR=707062	/ LOAD DISC ADDRESS FROM ACCUMULATOR AND WRITE
	/ DISC ADDRESS FORMAT:
	/ BITS FUNCTION
	/ 0 ENABLE DISC API IF A 1
	/ 6 DISC NO.
	/ 7 SURFACE NO.
	/ 8-13 TRACK NO.
	/ 14-17 SECTOR NO.
DKCF=707004	/ CLEAR DISC FLAGS
DKRF=707001	/ SKIP IF DISC READY FLAG SET
DKCF=707021	/ SKIP IF DISC CHECK-SUM ERROR FLAG NOT SET
DKIA=707041	/ SKIP IF ILLEGAL DISC ADDRESS (API)
DKDF=707061	/ SKIP IF DISC DONE FLAG SET (API)
DKRS=707132	/ READ DISC STATUS REGISTER INTO ACCUMULATOR
	/ BIT FUNCTION
	/ 0 ERROR (OR OF BITS 1-3)
	/ 1 PARITY ERROR
	/ 2 ILLEGAL ADDRESS
	/ 3 DISC NOT READY
	/ 4 DISC INTERRUPT ENABLED
	/ 5 0=READ 1=WRITE
	/ DISC API TRAP ADDRESS = 47 - LEVEL 1

/ KM-9 BOOTSTRAP

	.FULL	
	.LOC 17720	
.BOOT	IOF+10	/ PI OFF
	ISA	/ API OFF
	EEM	/ ENTER EXTENDED MEMORY MODE
	D7M .DKCA	/ KM-9 BEGIN ADDRESS
.KMSZ	LAW 00060	/ (-17720) KM-9 SIZE
	DAC .DKWC	/ DISC TRANSFER WORD COUNT
	LAC .KMSDA	/ KM-9 DISC ADDRESS
	DAC .DKAD	/ DISC ADDRESS OF TRANSFER
	D7M .DKRWS	/ SET READ/WRITE SWITCH TO READ
.DKREG	JMS .DKTRN	/ TRANSFER DISC TO CORE/CORE TO DISC
	LAC* .SCOM5	/ PROGRAM START (RE-START) ADDRESS
	DAC .DKGO	/ START ADDR T.V.
	JMP* .DKGO	/ GO TO PROG S.A.
	.EJECT	

```

/ BOOTSTRAP PARAMETERS
.SCOM5      105          / PROG STARTING ADDRESS T.V.
.DKWP       DKWR         / DISC WRITE INST
.KMODA      00005        / KM-9 DISC ADDRESS
.DKCA       0            / CORE ADDRESS FOR TRANSFER
.DKWC       0            / TWO'S COMPLEMENT WORD COUNT FOR TRANSFER
.DKAD       0            / DISC ADDRESS FOR TRANSFER
.DKRWS      0            / READ/WRITE SWITCH

/ TRANSFER DISC TO CORE OR CORE TO DISC SUBROUTINE
/ ARGS TO SET PRIOR TO CALLING TRANSFER SUBROUTINE:
/ .DKCA     CORE ADDRESS
/ .DKWC     TWO'S COMPLEMENT WORD COUNT
/ .DKAD     DISC ADDRESS
/ .DKRWS    DISC BOOTSTRAP READ/WRITE SWITCH (0/NON-0)
.DKTRN      0
      DKRF              / DISC READY ?
      JMP .DKHLT        / NO
      LAC .DKRWS         / READ/WRITE SWITCH
      SNAICLA           / 0 IF READ - NON0 IF WRITE
      DKRD-DKWR
      TAD .DKWR          / (DKWR) - DKRD OR DKWR
      DAC .DKGO
      LAC .DKCA          / BEGIN TRANSFER CORE ADDRESS
      DKMA
      LAC .DKWC          / TRANSFER TWO'S WORD COUNT
      DKWC
      LAC .DKAD          / DISC ADDRESS
      AND C37777        / (37777) MASK DISC ADDRESS FOR NON-API
.DKGO       XX          / DKRD OR DKWR
.DKDF       DKDF        / DONE?
      JMP .DKIA          / NO
      DKEF              / YES - ERROR?
      JMP .DKHLT        / YES
      JMP* .DKTRN       / NO - RETURN
.DKIA       DKIA        / INVALID ADDRESS?
      JMP .DKDF         / NO

/ .DKHLT AND .DKHLT+1 GET MODIFIED BY KM-9
/ SO THAT MONITOR CAN HANDLE .IOPS 4 ERROR
.DKHLT      HLTICLAICLL  / ERROR HALT - NOT READY
      JMP .DKTRN+1      / TRY AGAIN

/ BOOTSTRAP PARAMETERS
C37777      37777        / DISC ADDRESS MASK

      .END .BOOT

```

D.2 Disc/Core transfer routine

.TITLE DKCORE

```

/ PDP-9 DDC DISC TO CORE
/ OR CORE TO DISC PROGRAM
/ OPERATION:
/ WRI AT 17720, OR 37720 ETC.
/ PUT BEGIN CORE ADDRESS IN ACS AND CONTINUE
/ PUT +WORD COUNT IN ACS AND CONTINUE
/ PUT DISC ADDRESS IN ACS
/ SET ACS0 UP FOR CORE TO DISC
/ DOWN FOR DISC TO CORE AND CONTINUE
/ PROGRAM HALTS:
/ AC=0 NORMAL HALT AFTER TRANSFER COMPLETE
/ AC=DISC STATUS REGISTER WHEN ERROR

```

```

/ OCT 14, 1969
/ L H GERHARDSTEIN
/ BATTELLE NORTHWEST
/ P O BOX 999
/ RICHLAND, WASHINGTON
/ 99352

```

```

DKMA=707002      / LOAD MEMORY ADDRESS FROM ACCUMULATOR
DKWC=707022      / LOAD TWO'S COMPLEMENT WORD COUNT FROM ACCUMULATOR
DKRD=707042      / LOAD DISC ADDRESS FROM ACCUMULATOR AND READ
DKWR=707062      / LOAD DISC ADDRESS FROM ACCUMULATOR AND WRITE
                  / DISC ADDRESS FORMAT:
                  / BITS      FUNCTION
                  / 0         ENABLE DISC API IF A 1
                  / 6         DISC NO.
                  / 7         SURFACE NO.
                  / 8-13      TRACK NO.
                  / 14-17     SECTOR NO.
DKCF=707004      / CLEAR DISC FLAGS
DKRF=707001      / SKIP IF DISC READY FLAG SET
DKEF=707021      / SKIP IF DISC CHECK-SUM ERROR FLAG NOT SET
DKIA=707041      / SKIP IF ILLEGAL DISC ADDRESS (API)
DKDF=707061      / SKIP IF DISC DONE FLAG SET (API)
DKRS=707132      / READ DISC STATUS REGISTER INTO ACCUMULATOR
                  / BIT      FUNCTION
                  / 0         ERROR (OR OF BITS 1-3)
                  / 1         PARITY ERROR
                  / 2         ILLEGAL ADDRESS
                  / 3         DISC NOT READY
                  / 4         DISC INTERRUPT ENABLED
                  / 5         0=READ 1=WRITE
                  / DISC API TRAP ADDRESS = 47 - LEVEL 1

```

```

.FULL
.LOC 17720
.EJECT

```

/ GET PARAMETERS FROM AC SWITCHES

```

BEGIN      CLA
BGN        HLT
           IOF+10      / PI OFF
           ISA         / API OFF
           CAF         / CLEAR FLAGS
           EEM         / ENTER EXTENDED MEMORY MODE
           LAS         / GET BEGIN CORE ADDR
           DAC .DKCA
           HLT
           LAS         / GET +WORD COUNT
           DAC .DKWC
           HLT
           LAS         / GET DISC ADDRESS
           DAC .DKAD
           SMAICLA     / ACQ=1 IF WRITE TO DISC
           LAW -20     / DKRD-DKWR
           TAD .DKWR   / (DKWR
           DAC .DKGO   / DKRD OR DKWR
           DKRF        / DISC READY ?
           JMP ERROR   / NO
           LAC .DKCA   / BEGIN CORE ADDRESS
           DKMA
           LAC .DKWC   / +WORD COUNT
           CMA
           TAD ONE     / TWO'S WORD COUNT
           DKWC
           LAC .DKAD   / DISC ADDRESS
.DKGO      XY         / DKRD OR DKWR
           DKIA       / ILLEGAL DISC ADDR ?
           JMP .DKDF   / NO
ERROR      DKRS       / ERROR - READ DISC STATUS REGISTER
           JMP BGN
.DKDF      DKDF       / DISC DONE ?
           JMP .DKGO+1 / NO
           DKRF       / CSE ?
           JMP ERROR   / YES
           JMP BEGIN   / TRANSFER COMPLETE - NO ERRORS

/ DISC PARAMETERS
.DKCA      0          / BEGIN CORE ADDRESS
.DKWC      0          / WORD COUNT
.DKAD      0          / DISC ADDRESS
.DKWR      DKWR       / DISC WRITE INSTRUCTION
ONE        1

.END BEGIN

```


BNWL-1632

APPENDIX E

IOPS AND LOAD ERRORS

APPENDIX E
IOPS AND LOAD ERRORS

E.1 IOPS Errors

<u>Error Code</u>	<u>Error</u>	<u>Error Data</u>	<u>Comments</u>
0	Illegal function CAL	CAL address	The address points to a CAL which did not have a legal function code (1 to 16) in bits 3 to 17 of the word after the CAL.
1	CAL* illegal	CAL address	The instruction CAL* (Indirect) is an illegal Monitor CAL.
2	•DAT slot error	CAL address	<ol style="list-style-type: none"> 1. The •DAT slot number in Bits 9 to 17 of the CAL was 0, greater than 10, or less than -15. 2. The •DAT slot did not contain a handler address (no •IODEV was given for this •DAT slot).
3	Illegal interrupt	I/O status	An interrupt occurred which did not have an active device handler associated with it. The contents of the IORS word at the time of the interrupt is printed out.
4	Device not ready (type control R when ready)		<p>This error can occur whenever any not ready condition occurs.</p> <ol style="list-style-type: none"> 1. DECTape or MAGtape - unit not selected or not write enabled. 2. Punch-out paper tape 3. Line printer - off line 4. Card reader - off line, out of cards, stacks full, or card jam 5. Disk - massive failure 6. Drum - attempt to write on locked out area or non-existent tract, timing errors.
5	Illegal •SETUP CAL	CAL address	Use of SETUP when appropriate skip not placed in skip chain at system generation time.

Error Code	Error	Error Data	Comments
6	Illegal handler function	CAL address	A function (•READ, •WRITE, etc.) was issued to a handler which is incapable of performing that function (•READ to paper tape punch, •WRITE to C version of handler (Read only)).
7	Illegal data mode	CAL address	1. Illegal data mode for this version of the handler used. 2. Use of input commands after device has been INITed for output.
10	File still active	CAL address	Failure to close a file before another seek or enter on the same •DAT slot.
11	SEEK/ENTER not executed	CAL address	A read or write was issued without a prior SEEK, ENTER, or MTAP command.
12	Unrecoverable DECTape error	DECTape status register B and Unit Number	DECTape error with status register B in bits 0 to 11 and the unit # in bits 15 to 17.
13	File not found	CAL address	The file name specified by the directory entry section (pointer to entry is in CAL address plus 2) was not found.
14	Directory full	CAL address	The directory entry section of the current device in use is full.
15	DECTape full	CAL address	All blocks available for file storage are currently full.
16	Output buffer overflow	CAL address	The word pair count on the current •WRITE is greater than 177 ₈ .
17	Too many files for handler	CAL address	Too many files are currently open on the handler referenced by this CAL (e.g., 4 files on DTA will cause error while 2 files on DTD would cause same error).

Error Code	Error	Error Data	Comments
20	Disk failure	Disk status register	Disk failure with status register printed out.
21	Illegal disk address	CAL address	The CAL pointed to caused the disk control to reference an illegal or write protected address.
22	Two output files on one unit	CAL address	Two concurrent output files have been opened on one unit.
23	Illegal Word Pair Count	Sector address	The word pair count on the current input or output line equals zero or greater than 177_8 .
24	339 Handler called without push down list set up	CAL address	An attempt was made to use the 339 handler without reserving an area for the push down list.
25	Illegal sector address	CAL address	The current read or write CAL caused an illegal drum sector address to be used.
26	Illegal Drum unit	CAL address	The unit number in bits 0 to 2 of the word following the CAL is illegal for the drum size specified in bits 15 to 77 of $\bullet\text{SCOM} + 4$.
27	Illegal disk unit	CAL address	The unit number in bits 0 to 2 of the word after the CAL is 3 or 7 which are both non-existent on the Disk.
30	API error	API status register	An API break occurred which did not have the appropriate transfer vector set.
31	Non-existent memory reference	Program counter	Non-existent memory reference with protect mode on without a user defined violation routine.
32	Memory protect violation	Program counter	Reference to a location below the memory protect boundary without a user defined violation routine.

<u>Error Code</u>	<u>Error</u>	<u>Error Data</u>	<u>Comments</u>
33	Memory parity error	Program counter	Memory parity error without a user defined parity error routine.
34	Power fail with no skip setup	Program counter	Power low flag came up but a user defined routine to save appropriate registers not in core.
35	Illegal drum size	CAL address of first •INIT	Drum size in •SCOM + 4 (bits 15 to 17) is not 1 through 5.
40	Header label errors	CAL address	The internal header label for the currently opened file is incorrect.
42	Accessability map overflow	CAL address	Too many files recorded on the current MAGtape. Copy the tape to retrieve storage occupied by unwanted files.
43	Directory recording error	CAL address	Directory cannot be recorded at the beginning of the tape - use the utility program to re-make the tape.
77	Permanent disc file	CAL address	The disc device handler was requested to delete a permanent disc file.

E.2 Linking Loader and System Loader Errors

The following error codes are output by the Linking Loader and by the System Loader. When output by the Linking Loader, the errors are identified as shown below. When output by the System Loader, the errors are identified as "SYSLD n" instead of "LOAD n".

<u>Error</u>	<u>Meaning</u>
LOAD 1	Memory overflow - the Loader's symbol table and the user's program have overlapped. At this point, the Loader memory map will show the addresses of all programs loaded successfully before the overflow. Increased use of COMMON storage may allow the program to be loaded as COMMON can overlay the Loader and its symbol table, since it is not loaded into until run time.
LOAD 2	Input data error - parity error, checksum error, illegal data code, or buffer overflow (input line bigger than Loader's buffer).
LOAD 3	Unresolved Globals - any programs or subroutines required but not found, whether called explicitly or implicitly, are indicated in the memory map with an address of 00000. If any of the entries in the memory map have a 00000 address, loading was not successful; the cause of trouble should be remedied and the procedure repeated.
LOAD 4	Illegal •DAT slot request - the •DAT slot requested was: <ol style="list-style-type: none"> Out of range of legal •DAT slot numbers, Zero, Unassigned, that is, was not set up at System Generation Time or (in the case of the Keyboard and Background/Foreground Monitors) was not set up by an ASSIGN command. Or, in the Background/Foreground system, the BACKGROUND program requested a •DAT slot which called for an I/O handler and device number which conflicted with the FOREGROUND job's I/O.

APPENDIX F

DEVICE HANDLER FUNCTIONS

APPENDIX FDEVICE HANDLER FUNCTIONSF.1 Nomenclature

CAL = Memory reference instruction - call subroutine (Op Code = 00)

a = Dat slot number

m = Data mode

0	Iops Binary
1	Image
2	Iops ASCII
3	Image
4	Dump

F = File type

0	Input file
1	Output file

R = Restart address plus character code in bits 0-1

00	↑P
01	↑C
10	↑T
11	none

L = Line buffer address or core location for •TRAN

W = Word count for line buffer or •TRAN transfer

D = File specification pointer, i.e.: Points to •SIXBT/Name Ext/

BA = Busy return address for •WAITR

B = Device address, e.g.: Dectape block no.

n = Standard line buffer size

F.2 I/O Handler Functions Summary

Function No.	Function Name	Macro Call	Macro Expansion	Action Taken by Device Handler
1	•INIT a,F,R		CAL+F ₇₋₈ +a ₉₋₁₇ 1* R n**	Causes device and device handler to be initialized
2	•FSTAT a,D		CAL+3000+a ₉₋₁₇ 2* D***	Check file status - AC on return = device address or zero when file not found
2	•RENAM a,D		CAL+2000+a ₉₋₁₇ 2* D	Rename file - D = Starting address of two three word blocks containing the old name; ext and the new name; ext in that order
2	•DELETE a,D		CAL+1000+a ₉₋₁₇ 2* D	Delete file from Directory
2	•RDIR a,D		CAL+4000+a ₉₋₁₇ 2* D	Read directory entry of file. D = Starting address of 9 word block; one keyword and eight word directory entry
3	•SEEK a,D		CAL+a ₉₋₁₇ 3* D	Search for directory entry and setup to read an input file. IOPS 13 will occur if the file is not found
4	•ENTER a,D		CAL+a ₉₋₁₇ 4* D	Examine device directory and setup to write an output file which will be entered in the directory when closed
5	•CLEAR a		CAL+a ₉₋₁₇ 5*	Clear the directory to contain no entries
6	•CLOSE a		CAL+a ₉₋₁₇ 6*	Close file Input - frees device handler to open another file

Function No.	Function Name	Macro Call	Macro Expansion	Action Taken by Device Handler
				Output - finishes writing of file data, makes directory entry, frees device handler to open another file
10	•READ a,M,L,W	CAL+M 10* L DEC -W	$6-8+a_{9-17}$	Transfer the next line of data from device to user's program
11	•WRITE a,M,L,W	CAL+M 11* L DEC -W	$6-8+a_{9-17}$	Transfer one line of data from user's program to device
12	•WAIT a	CAL+a 12*	a_{9-17}	Causes user's program to be suspended until such time as •READ is complete, •WRITE is complete, or •TRAN is complete
12	•WAITR a,BA	CAL+1000+a 12* BA	a_{9-17}	Causes return at BA if I/O is in progress or at the normal return address if I/O is complete
13	•TRAN a,d,B,L,W	CAL+d 13* B L DEC -W	$7-8+a_{9-17}$	Data is transferred as a block from core to device or from device to core according to the parameter d
				Input forward = 0 Output forward = 1 ****Input reverse = 2 ****Output reverse = 3

*The CAL handler places the unit number in bits 0-2 of this word.

**The device handler places the standard line buffer size in this word.

***Bits 0-2 of this location should be cleared prior to execution of the CAL. File-oriented handlers will store a non-zero number there. If the contents of bits 0-2 is still zero upon return, the device is non-file oriented.

****Applies only to DEC tape.

F.3 IOPS Data Formats

Any system or user program that runs in the DOS I/O monitor environment uses line buffers to transmit data to or from I/O devices. The only exception is when Dump mode is used. A line buffer consists of an n word block of core memory (n is always non-zero and even). The first two words of the line buffer is the header word pair and the remaining $n-2$ words contain the data. (See Figures F.1 and F.2.) Since all system device handlers transmit and accept data of the latter format and all are called through the I/O monitor by identical means, the system is I/O device independent in nature.

F.3.1 IOPS ASCII Data Format

All alphanumeric data transmitted by the DOS system device handlers is in the form of 7-bit ASCII characters. The characters appear in an IOPS ASCII line biffer word pair as 5x7 ASCII. In a 5x7 ASCII word pair, five ASCII characters are packed in two words as shown in Figure F.3.

ASCII data sent to an output device must have been packed in this manner by the user's program. The ASCII data received from input devices is of the same form and must be unpacked by the user's program is a character by character interpretation is needed.

F.3.2 IOPS Binary Format

The format of IOPS binary line buffers is simply $n-2$ 18 bit words (n = line buffer size) preceded by the appropriate line biffer header word pair.

Figure F.1 - Line Buffer Structure

Figure F.2 - Format of Header Word Pair

Figure F.3 - 5x7 ASCII Packing Scheme

APPENDIX G

I/O HANDLER CHARACTERISTICS

APPENDIX GI/O HANDLER CHARACTERISTICSG.1 I/O Device Handler Data Modes

<u>Device Handler Name</u>	<u>IOPS Binary</u>	<u>Image Binary</u>	<u>IOPS ASCII</u>	<u>Image ASCII</u>	<u>Dump</u>	<u>No. Locations (Octal)</u>
TIA			X	X		N.A.
PRA	X	X	X	X	X	674
PRB			X			446
PPA	X	X	X	X	X	571
PPB	X	X		X	X	416
PPC	X					322
DTA	X	X	X	X	X	4421
DTB	X		X			3023
DTC	X		X			1250
DTD	X	X	X	X	X	3041
DKA	X		X			2211
DKB	X		X			1526
DKC	X		X			567
DKD	X		X			1431

"X" denotes a legal data mode and a blank denotes an illegal device handler data mode. The TIA device handler is an integral part of the resident I/O monitor and therefore its length is of no consequence. (i.e.: It is always present in memory.)

G.2 I/O Device Handler Functions

Device Handler Name	•INIT	•FSTAT etc.	•SEEK	•ENTER	•CLEAR	•CLOSE	•MTAPE	•READ	•WRITE	•WAIT & •WAITR	•TRAN
TIA	X	I	I	I	I	X	I	X	X	X	
PRA	X	I	I		I	I	I	X		X	
PRB	X	I	I		I	I	I	X		X	
PPA	X			I	I	X	I		X	X	
PPB	X			I	I	X	I		X	X	
PPC	X			I	I	X	I		X	X	
DTA	X	X	X	X	X	X		X	X	X	X
DTB	X		X	X		X		X	X	X	
DTC	X		X			X		X		X	
DTD	X	X	X	X	X	X		X	X	X	X
DKA	X	X*	X	X		X		X	X	X	X
DKB	X		X	X		X		X	X	X	
DKC	X		X			X		X		X	
DKD	X	X*	X	X		X		X	X	X	X

X denotes a legal function

I denotes a legal but ignored function

"blank" denotes an illegal function

* Disc device handlers DKA and DKD also have the Read-directory-entry function •RDIR

APPENDIX H

DOS SYSTEM MEMORY MAPS

DOS System Memory Maps

The System Bootstrap is loaded via the paper tape reader in HRM mode.

The System Bootstrap loads (DUMP mode) the Keyboard Monitor (resident and non-resident) from the system device.

Figure H.1 - Keyboard Monitor System Memory Maps

The Keyboard Monitor loads (DUMP mode) the System Loader and the system device handler from the system device

The System Loader, during loading of a system program from the system device, builds the loader (GLOBAL) symbol table down from .SCOM+3 and the programs up from .SCOM+2.

The System Loader learns which I/O handlers are required by the requested system program from its table of .IODEV info for system programs, loads the handlers relocatably just above the resident KM-9 and then modifies the System Bootstrap to bring in the system program in Dump mode just below the Bootstrap.

.EXIT from the system program takes the process back to Memory Map B where the system bootstrap reinitializes the Keyboard Monitor.

Refer to Section G.1 for the sizes of the device handlers that may be associated with the .DAT slots used by the System program.

Figure H.2 Keyboard Monitor System Memory Maps (Cont)

The System Loader learns which I/O handlers are required by the Linking Loader, loads them relocatably and then loads the Linking Loader relocatably.

If a DDT load, the Linking Loader just prior to giving control to DDT moves the DDT symbol table down in core so that it overlays all of the Linking Loader except for the small routine that makes the block transfer.

The Linking Loader, during loading of user programs down from .SCOM+2, builds the loader (GLOBAL) and DDT (if DDT) symbol tables up from .SCOM+2. DDT symbol table will not be built if a LOAD, GLOAD, or DDTNS load.

.EXIT from user program takes the process back to Memory Map B where the system bootstrap re-initializes the Keyboard Monitor.

Refer to Section 7.4 for sizes of device handlers.

.SCOM+1 and .SCOM+2 both point to one of two places and non-BLOCK DATA COMMON (FOR-TRAM IV or MACRO-9) output may make use of core as low as they point.

- If the user program did not have any device handlers in common with the Linking Loader.
- If the user program did have at least one device handler in common with the Linking Loader.

Figure H.3 Keyboard Monitor System Memory Maps (Cont)

.EXIT from the user program takes the process back to Memory Map B where the system bootstrap re-initializes the Keyboard Monitor.

Refer to Section 7.4 for sizes of device handlers.

Non-BLOCK DATA COMMON (FORTRAN IV) or MACRO-9 output) may make use of core as low as the DDT symbol table*. However, trouble will occur if the user requests DDT to create symbols or make patches that cause overlaying of the COMMON area.

The Linking Loader device handlers would have been used to satisfy user device requests.

*There is no DDT symbol table if a DDTNS load.

Figure H.4 Keyboard Monitor System Memory Maps (Cont)

The System Loader loads the I/O handler needed by EXECUTE above the resident portion of the Monitor. It then modifies the System Bootstrap to bring in EXECUTE in dump mode just below the Bootstrap.

When the user typed in the name of the XCT file to be run, the non-resident portion of the Monitor stored the file name in .SCOM + 7, 10 and 11. .EXECUTE now locates this file and brings in the first chain.

FORTAN programs pass on data in blank common starting at .SCOM + 2. Macro programs pass on data between .SCOM + 2 and .SCOM + 3.

A call from the running chain to bring in another chain is effected by transferring control back to Execute.

.EXIT from the user chain takes the process back to PHASE 2 where the System Bootstrap re-initializes the Keyboard Monitor.

APPENDIX I

CORE MEMORY TO DECTAPE TRANSFERS

USING THE DECTAPE BOOTSTRAP

APPENDIX ICORE MEMORY TO DECTAPE TRANSFERSUSING THE DECTAPE BOOTSTRAP

System generation procedures for the DOS require core images of system programs for transferring to the disc memory. When the object program is an absolute core dump on Dectape, a Dectape to core transfer routine is needed. The Dectape System bootstrap can be used for this purpose.

The following patch applies to the V4B or V4E bootstrap. These steps allow the Dectape bootstrap to be used in a manner similar to DKCORE (Section 8.1).

1. Load the Dectape bootstrap at location B+17637, where
B = 0, 200000, 400000, or 600000.

```
I/O RESET
HRM B+17637
STOP
```

2. Deposit a SPA!HLT instruction in the bootstrap block-transfer subroutine:

```
B+17740/ 741140
```

3. Determine the following parameters for the transfer to be performed:

```
BLK  Starting Dectape Block No.
MA   Starting Memory Address
WC   Word Count for Transfer
U    Dectape Unit No. (0-7)
R/W  Read or Write
```

4. Deposit the latter parameters as follows:

```
B+17637/ BLK
B+17640/ MA-1
B+17641/ -WC (2's complement)
B+17644/ U(0-2)+210000
```


5. If a "Write" operation is to be performed, manually select Write enable on the specified unit. The computer is now conditioned to perform the required transfer. Start the computer as follows:

<u>Function</u>	<u>Direction</u>	<u>Action</u>
READ	Tape to Core	Start B+17663
WRITE	Core to Tape	Start B+17665

6. When the function has been completed, the computer will halt with zero in the AC if no errors were encountered.
7. To read or write another data block, return to Step 3 of these instructions.

The shorthand notation for the latter operation, as used in this report, is:

DTBOOT U,BLK,MA,WC,T

where T = "R" if READ or "W" if WRITE, and U,BLK, MA, and WC are the parameters given in Step 3 above.

Distr-1

Special Distribution
in UC-32DISTRIBUTIONOFFSITE

<u>No. of Copies</u>		<u>No. of Copies</u>	
1	<u>AEC Chicago Patent Group</u> G. H. Lee	1	Digital Equipment Corp. 1521 - 130th N.E. Bellevue, WA 98004
3	<u>USAEC Div. of Tech. Info.</u>		
3	<u>USAEC Div. of Production</u> F. P. Baranowski	1	J. W. Gee MacMillan Bloedel P. O. Box 487 New Westminster, B.C. Canada

ONSITE

1	<u>AEC Chicago Patent Group</u> R. M. Poteat
2	RDT Asst. Dir. for Pacific Northwest Programs
1	O. J. Elgert
1	W. E. Lotz
1	T. A. Nemzek
1	H. E. Parker
1	M. R. Schneller

BNW

1	L. E. Addison
1	E. L. Alpen
1	R. C. Arnett
1	R. G. Baca
1	A. Brandstetter
1	R. A. Burnett
1	K. M. Busness
10	D. B. Cearlock
1	C. R. Cole
1	M. R. Compton
1	R. E. Connally
1	V. L. Crow
1	D. G. Daniels
1	G. W. Dawson
1	P. J. Dionne
1	G. E. Driver

Distr-2

ONSITE (continued)

BNW (continued)

1	J. R. Eliason
1	M. D. Erickson
1	J. E. Fager
1	H. P. Foote
1	J. C. Fox
3	D. R. Friedrichs
5	L. H. Gerhardstein
1	R. T. Jaske
1	K. L. Kipp
1	J. R. Kosorok
1	R. E. Mahan
1	D. E. Olesen
1	A. E. Reisenauer
1	R. C. Routson
1	T. A. Seim
1	R. J. Serne
1	S. E. Wise
5	Technical Information
2	Technical Publications

ARHCO

1	G. E. Benedict
1	D. J. Brown
1	M. H. Campbell
1	R. P. Corlew
1	J. B. Fecht
1	R. E. Isaacson
1	L. M. Knights
1	G. A. Nicholson
1	G. C. Oberg
1	H. C. Rathvon
1	L. M. Richards
1	A. E. Smith
1	P. W. Smith
1	M. J. Szulinski
1	M. D. Veatch

Westinghouse Hanford Company

1	R. J. Ausere
1	B. E. Dozer
10	R. A. Harvey
1	J. A. Hubbard
1	S. A. Hunt
1	W. F. Lenzke
1	J. D. Lodge
1	G. W. Main
1	W. R. Markillie
1	L. D. VanHees
1	G. W. Waldkoetter
1	N. P. Wilburn
1	G. A. Worth