


**digital**  
**software**

**OS/8**

**Handbook Update**

Order No. DEC-S8-OSHBA-A-DN4



**PDP8**  
more than 30,000 installed worldwide

# OS/8

## Handbook Update

Order No. DEC-S8-OSHBA-A-DN4

### ABSTRACT

This document includes all previous corrections to the *OS/8 Handbook*, and contains the documentation to support V3D of OS/8.

**SUPERSESSION/UPDATE INFORMATION:** This manual updates the *OS/8 Handbook* for V3D of OS/8.

**OPERATING SYSTEM AND VERSION:** OS/8 V3D

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

First Printing, October 1975  
Revised: December 1975  
September 1977

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by DIGITAL.

Copyright © 1975, 1977 by Digital Equipment Corporation

The postage prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-10
DECCOMM	DECSYSTEM-20	TYPESET-11

## INTRODUCTION

This update is a collection of changes and additions to the *OS/8 Handbook* for use with Version 3D of OS/8. These changes and additions are documented herein by Chapter as they appear in the handbook. This update also includes information on several new or revised OS/8 utility programs. These utility programs, included as Appendixes to the handbook, are:

RXCOPY	—	Appendix I
SET	—	Appendix J
FUTIL	—	Appendix K
DUMP	—	Appendix L
RKLFMT	—	Appendix M



# FRONT MATTER AND CHAPTER 1

## ADDITIONS AND CHANGES

Page	Addition/Correction
ii	In the second line from the bottom, change “Communications Services, Parker Street” to “Software Distribution Center”.
viii	In the first line of the second paragraph, change “impossible” to “possible”. At the end of the second paragraph, add “OS/8 also runs on the VT-78.”
1-2	Step 1, change DEC-S8-OSYSB-A-UC1 to AL-4711C-BA
1-4	Step 1, change DEC-S8-OSYSB-A-UC2 to AL-4712C-BA
1-5	Step 6, First line of example, change “form” to “from” Step 6, change DEC-S8-OSYSB-A-UC1 to AL-4711C-8A
1-9	Step 1, change DEC-12-OSYSB-A-AC1 to AL-3580C-BM
1-10	Step 1, change DEC-S8-OSYSB-A-TC1 to AR-4585C-BA
1-15	Step 1, change DEC-S8-OSYSB-A-TC2 to AR-4586C-BA Step 2, change DEC-S8-OSYSB-A-TC3 to AR-4587C-BA
1-16	Step 4, In the MCPIP: program line 5 is changed, lines 12 and 13 are deleted, and three lines are appended. The complete program reads as follows:  *SYS:CCL.SV<CSA0:CCL.SV *SYS:DIRECT.SV<CSA0:DIRECT.SV *SYS:FOTP.SV<CSA0:FOTP.SV *SYS:PIP.SV<CSA0:PIP.SV *SYS:LIB8.RL<CSA0:LIB8.RL *SYS:EDIT.SV<CSA0:EDIT.SV *SYS:PAL8.SV<CSA0:PAL8.SV

1-16 (cont.)

```

*SYS:CREF.SV<CSA0:CREF.SV
*SYS:BITMAP.SV<CSA0:BITMAP.SV
*SYS:BOOT.SV<CSA0:BOOT.SV
*SYS:CAMP.SV<CSA0:CAMP.SV
*SYS:FORT.SV<CSA1:FORT.SV
*SYS:SABR.SV<CSA1:SABR.SV
*SYS:LOADER.SV<CSA1:LOADER.SV
*SYS:SRCCOM.SV<CSA1:SRCCOM.SV
*SYS:EPIC.SV<CSA1:EPIC.SV
*SYS:PIP10.SV<CSA1:PIP10.SV
*SYS:RESORC.SV<CSA1:RESORC.SV
*SYS:DTCOPY.SV<CSA1:DTCOPY.SV
*SYS:TDCOPY.SV<CSA1:TDCOPY.SV
*SYS:TDFRMT.SV<CSA1:TDFRMT.SV
*SYS:DTFRMT.SV<CSA1:DTFRMT.SV
*SYS:LIBSET.SV<CSA1:LIBSET.SV
*SYS:RXCOPY.SV<CSA1:LIBSET.SV
*SYS:HELP.SV<CSA1:HELP.SV

```

Step 5, change the current Step 5 to Step 6 and insert the following as the new Step 5.

To write SET.SV and HELP.HL files on the system device, mount AR-4688C-BA in drive 0 and AR-4689C-BA in drive 1. Type the following command line after the asterisk is printed on the terminal.

```

*SYS:SET.SV<CSA0:SET.SV
*SYS:HELP.SV<CSA1:HELP.SV

```

By typing the command line:

```

_R CCL

```

you can run your programs by using CCL commands.

Step 6, in the new Step 6 change DEC-S8-OSYSB-A-TC6 to AR4690C-B.

1-17

Step 2, change (DEC-S8-OSYSB-A-PB1) to AK-4678C-BA.

1-19

Change the NOTE to read as follows:

#### NOTE

When building from the low-speed reader (KS33), after entering PTR followed by carriage return the system responds with an up-arrow; the user must respond by typing any character on the terminal and then immediately turn on the reader. If the reader is not turned on promptly, the system hangs. Remember to turn off the reader when it reaches the leader/trailer at the end of the tape.

1-19 (cont.)

Step 6, change (DEC-S8-OSYSB-A-PB4) to AK-4679C-BA.

Step 7, change (DEC-S8-OSYSB-PB5) to AK-4680C-BA.

1-20

Step 9, change ABSLDR to EPIC

Delete the entire section "Loading System Programs from Paper Tape" pages 1-20 through 1-25, and replace with the following:

#### Loading Paper Tape Binary Kit

Paper tape binary kits for OS/8 V3D are punched using EPIC. This use of EPIC simplifies loading these tapes onto SYS: All tapes, except those used to build a Monitor and a System Head and EPIC itself, must be loaded onto SYS: using EPIC. The procedure for loading paper tape binary kits is described below.

#### NOTE

Skip Step 1 if EPIC.SV exists in system directory.

1. Place the EPIC binary tape (AK-4667C-BA) in the reader and type:

```
_R ABSLDR (CR)
_PTR:($)^
```

Turn on reader and type any key on keyboard.

#### NOTE

(CR) is carriage return (press RETURN key). (\$) is escape or altmode, strike ESC key.

EPIC will be read in by this procedure. If necessary, turn off the reader. Save EPIC as a file by typing:

```
_SAVE SYS:EPIC.SV(CR)
```

2. Type \_R EPIC(CR)

3. To load any paper tape onto SYS:, put the paper tape for that file in the reader and type:

```
*SYS:</0/Y($) for the high speed reader or
```

```
*SYS:</0/Y/L($) for the low speed reader. (Turn on the low speed reader, depress CONT on the operator's console after the computer halts to allow loading the tape in the reader. After the tape has read in, the computer halts again. If there are no more tapes to that file to be loaded, turn off the reader and depress CONT. If there are more tapes to the file continue to Step 4.
```



1-20 (cont.)

4. If the file being created requires more than one tape to be input, the message:

END OF TAPE ENTER NEXT

will be displayed on the console terminal and the computer will halt with 7777(8) in the AC. Place the next tape of the file in the reader, turn it on, and depress CONT. Repeat Step 4 until all tapes for the file are loaded.

If the tapes of a multiple tape file are read out of sequence an error message:

NEED nnnn FOUND mmmm

will be output on the console terminal. Check the tapes of the file and place the proper tape in the reader, and depress CONT on the operators console.

5. Repeat Steps 3 and 4 to load each tape or set of tapes into a file on SYS:

More information on EPIC can be found in Chapter 2 of the OS/8 Handbook (DEC-S8-OSHBA-A-D).

6. If desired, you can load CCL.SV, EDIT.SV, and BATCH.SV (if you have the OS/8 Extension Kit) and then create a batch stream to load the desired files onto SYS:

Create a batch file as follows:

<u>_CREATE LOAD.BI(CR)</u>	
<u>#A(CR)</u>	
<u>\$JOB TO LOAD FILES USING EPIC(CR)</u>	
<u>.R EPIC(CR)</u>	
<u>*SYS:&lt;/0/Y\$(CR)</u>	\$=dollar sign key (shift/4) – not
<u>*SYS:&lt;/0/Y\$(CR)</u>	escape or altmode. Add/L before
.	\$ (see example in Step 3 above)
.	if reading from low speed
.	reader.
<u>*SYS:&lt;/0/Y\$(CR)</u>	Put in a few of the load commands;
<u>.SU LOAD.BI/T(CR)</u>	the more you put in, the fewer times
	the job will re-submit itself.
<u>(CTRL/FORM)</u>	<u>(CTRL/FORM)</u> means hold down
<u>#E(CR)</u>	CTRL key, depress “L” (FORM) key

Then run it, using the command:

\_SU LOAD.BI/T(CR)

Every time the computer stops, replace the tape in the reader with a new one (or next in sequence) and depress CONT. Ignore any

L/T ERROR

messages on console terminal due to running off the end of the paper tape.

Insert the following table before the section entitled "Restarting OS/8":

**Table 1-11A RX01 Floppy Disk Bootstrap**

Step	Octal # Values	Switch Register Setting				And Then
		012	345	678	91011	
1	0000	000	000	000	000	press EXTD ADDR LOAD
2	0024	000	000	010	100	press ADDR LOAD
3	7126	111	001	010	110	lift DEP key
4	1060	001	000	110	000	lift DEP key
5	6751	110	111	101	001	lift DEP key
6	7201	111	010	000	001	lift DEP key
7	4053	100	000	101	011	lift DEP key
8	4053	100	000	101	011	lift DEP key
9	7104	111	001	000	100	lift DEP key
10	6755	110	111	101	101	lift DEP key
11	5054	101	000	101	100	lift DEP key
12	6754	110	111	101	100	lift DEP key
13	7450	111	100	101	000	lift DEP key
14	7610	111	110	001	000	lift DEP key
15	5046	101	000	100	110	lift DEP key
16	1060	001	000	110	000	lift DEP key
17	7041	111	000	100	001	lift DEP key
18	1061	001	000	110	001	lift DEP key
19	3060	011	000	110	000	lift DEP key
20	5024	101	000	010	100	lift DEP key
21	6751	110	111	101	001	lift DEP key
22	4053	100	000	101	011	lift DEP key
23	3002	011	000	000	010	lift DEP key
24	2050	010	000	101	000	lift DEP key
25	5047	101	000	100	111	lift DEP key
26	0000	000	000	000	000	lift DEP key
27	6753	110	111	101	011	lift DEP key
28	5033	101	000	011	011	lift DEP key
29	6752	110	111	101	010	lift DEP key
30	5453	101	100	101	011	lift DEP key
31	7024	111	000	010	100	lift DEP key
32	6030	110	000	011	000	lift DEP key
33	0033	000	000	011	011	press ADDR LOAD and press CLEAR and press CONT

- 1-31 Add the following to Table 1-12 Permanent Device Names:
- |      |   |
|------|---|
| RXAn | Diskette n (floppy), where n is an integer in the range of 0-7 inclusive.                 |
| RKBn | DECpack n, where n is an integer in the range 0-1.  |
| NULL | On input this returns an immediate end-of-file; on output this device ignores characters. |
| DUMP | Prints contents of device blocks on LPT.  |

- 1-32 Add the following to Table 1-13 Assumed Extensions:

**NOTE**

Refer to Appendix F for a complete list of commonly used extensions.

- 1-37 Under the GET Command, add the following insert before the last sentence in that paragraph.
- In addition, location 7747 in field 0 is loaded with the block number of the first block of the core image file (.SV) specified.
- Under the "JOB STATUS WORD," delete "Bits 4-9 Unused and reserved for future expansion" and add the following:
- Bit 4=1 A core image file that was generated by LINK containing overlays.
  - Bit 5=1 This program cannot be run by the R, RUN, or GET commands under OS/78.
  - Bit 6-9 Unused and reserved for future expansion.

- 1-41 Replace this page with:
- RUN Command*  
The RUN command is of the form:
- .RUN dev file.ex
- or
- .RU dev file.ex

1-41 (cont.)

The RUN command, like the SAVE command, handles *only* core-image files. The file indicated (file.ex) on the device specified (dev) is loaded into core and its core control block is moved to the system scratch area. The program is started at its starting address. Location 7747 in field 0 is loaded with the block member of the first block of the core image file (.SV).

The RUN command is equivalent to a GET and a START command.

If an extension to the file is not specified, the extension .SV is automatically added to the file name. For example:

```
._RU DTA1PROG
```

causes the file PROG.SV on DECTape 1 to be loaded and started.

#### *R Command*

The R Command is of the form:

```
.R file.ex
```

and is similar to

```
.RUN SYS file.ex
```

This command handles only core image files from the system device. The file is loaded and started and location 7747 is loaded with the block member of the first block of the core image file (.SV). If the file name extension is not specified, the extension .SV is automatically added.

The R command differs from the RUN command in that a core control block is *not* written to the system device. In order to save a program which does not have its core control block in the usual location on the system device, all the optional arguments of the SAVE command must be explicitly stated. System programs are most often called using the R command, since they need not be resaved.

To call a program which is to be later updated and saved, use of the RUN or GET commands is suggested.

#### *START Command*

The START command is of the form:

1-52

Middle of the page, remove the paragraph that begins "The user may write his own CCL commands. . ." and insert:

Sophisticated users who wish to add their own CCL commands should refer to the OS/8 V3 source listing of CCL(DEC-S8-OSYSB-A-LA18).

- 1-53            After *BACKSPACE* add  
**BASIC**  
Delete *CORE*  
After *DIRECT* add  
**DUPLICATE**  
After *MAP* add  
**MEMORY**  
Delete *RES* and replace with  
**RESOURCE**
- 1-55            In Table 1-16 add options
- N    NULL  
-D    DUMP
- 1-56            Add the following as the last paragraph under the section entitled “INDIRECT  
COMMANDS”:
- A single quote is permitted after file specification; if used, it is ignored. A single quote  
prevents the next letter from being considered part of the indirect command file name.
- 1-57            In the third paragraph, change “64” to “40”. In the example above the paragraph,  
change “0102036404000506” to “0102034004000506”.
- 1-58            Before the title “BOOT Command”, insert the following:
- BASIC COMMAND**  
The BASIC command requests execution of the BASIC editor.
- Format:
- .BASIC
- Example:
- .BASIC  
NEW OR OLD---
- For additional information on BASIC, refer to Chapter 6.
- This command runs CCL.SV and BASIC.SV.

1-59 Add the following to the section entitled "COMPILE Command", after ".COM file.ex<file.ex":

or  
 .COM A,B<C,D  
 for multiple Input/Output files.

The COMPILE command produces similar results when chaining and specifying either the /T option or the /T/F options. For example, both of the following commands produce a RALF symbol table only:

```

  _COM LPT FL1/T/G
  _COM LPT FL1/T/F/G

```

1-61, 62 Delete the CORE command and its description.

1-64 Before the EDIT command, insert the following:

#### DUPLICATE COMMAND

The DUPLICATE command copies or transfers the entire contents of one diskette to another diskette.

Format:

```
.DUPLICATE outdev:<indev:/options
```

In addition to the necessary arguments in the command line, DUPLICATE options can be used to affect the DUPLICATE operation.

Example:

```
_.DUPLICATE RXA1:<RXA0:
```

The contents of input device RXA0 is copied onto output device RXA1.

**Changing Devices Before and After Executing the DUPLICATE Command** — You can only duplicate from RXA0 to RXA1 or RXA1 to RXA0. Driver RXA2 and RXA3 is not supported by the DUPLICATE command. Since the Monitor resides on the system device, the system device must remain on line when interacting with the Monitor and any OS/78 system programs.

If you want to transfer the contents of a diskette containing only files (one that does not contain a system), use the /P option. The /P option pauses before and after its execution of the DUPLICATE command. A ready message followed by a question mark is displayed. This pause provides time to remove the system device and mount a device onto which the contents are to be transferred, to or from. To start the DUPLICATE operation, type a Y after the question mark and press the RETURN key. After the DUPLICATE operation is completed, a message is displayed asking if the Monitor is remounted. The

1-64 (cont.)

second pause provides time to remove the new device and remount the system device so control can return to the Monitor. After remounting the system device, type a Y after the question mark and press the RETURN key, to return control to the Monitor.

Example:

```
DU RXA0:<RXA1:/P
READY?Y
IS MONITOR REMOUNTED?Y
```

#### Performing a Read Check

To check the integrity of a diskette, use the /R option and specify only the input device. By specifying the /R option, every block of the specified device is read and checked for bad sectors. If bad sectors exist, a message with the device, track number and sector number is displayed. If none exist, control returns to the Monitor.

Example:

```
DU <RXA1:/R
INPUT DEV READ ERROR TRACK:nn SEC:nn
```

#### Transfer Without Checking for Identical Contents

To transfer the contents of one device to another without performing any checks, use the /N option. By specifying the /N option, the contents of the input device is transferred or copied to the output device. If /N is not specified and the DUPLICATE operation is completed, the contents of the output device is compared to that of the input device to assure accuracy.

#### Check for Identical Contents Without Transferring

To check the contents of devices to see if they are identical without transferring, use the /M option. By specifying the /M option the contents of both devices are read and checked for a match. If they do match, control returns to the Monitor. However, if they differ in any way, a message, the device name, the track number, and the sector number of the sectors or blocks that do not match are displayed.

Example:

```
DU RXA1:<RXA0:/M
COMPARE ERROR TRACK nn SECTOR nn
```

This command causes the execution of both the CCL.SV and the RXCOPY.SV programs.

1-65

Add the following to the section entitled "EXECUTE Command", after ".EXE file.ex,file.ex":

or

```
.EXE A<B
to execute file "B" producing file "A".
```

1-66

Delete the HELP command description and insert:

#### HELP Command

The HELP command sends information on OS/8 programs to the output device, usually the terminal. There is a HELP program (HELP.SV) that is executed every time the HELP command is used. There is also a HELP file (HELP.HL) that contains a list of all the HELP subfiles available and the actual HELP text itself. Both the HELP.SV and HELP.HL files must be on the system device.

Format:

```
.HELP outdev:file.ex<argument
```

where:

argument is usually an OS/8 program or CCL command.

Example:

```
._HELP RXCOPY or ._HELP DUPLICATE
```

Note that the default output device is TTY.

The OS/8 software for the HELP files can be supplied on any device. Following is a list of all the arguments that can be used with the HELP command.

ABSLDR	BASIC	BCOMP	BRTS	BOOT
BUILD	"NONE"	CCL	CREF	DIRECT
EDIT	CREATE	EPIC	FORT	FRTS
F4	FORTRAN	F4ERR	LIBRA	LOAD
LOADER	MAP	BITMAP	ODT	PAL8
PAL	PALERR	PIP	PIP10	DUPLIC
RXCOPY	SABR	SET	SRCCOM	COMPAR
BATCH	SUBMIT	TECO	MAKE	MUNG
FOTP	LIST	COPY	RENAME	TYPE
DELETE	ASSIGN	DATE	DEASSIG	GET
MEMORY	R	RUN	SAVE	START
SQUISH	UA	ZERO		

There is a HELP file which contains all CCL commands. This file is displayed by typing the HELP command without any arguments as follows:

```
._HELP
```

If a HELP file for a specific command is desired, type the name of the command for which the information is desired. For example,

```
._HELP PAL
```



1-66 (cont.)

To obtain a list of all legal arguments for HELP, type the HELP command followed by an asterisk or type the HELP command followed by 'HELP' as follows:

```
.HELP *
or
.HELP HELP
```

Note that the HELP.HL file must be on the system device.

1-68

Both examples of the MAKE Command should specify "MAK" not "MA". Also, change "% SUPERCEDING" to "%SUPERSEDING".

1-70

In the first line of the MUNG command example, change "HX!" to "HX1!".

Before the PAL Command insert the following:

#### MEMORY COMMAND

The MEMORY command is used to find the highest field available in hardware or to limit that value in software.

Format:

```
.MEMORY n or .MEMORY
```

where:

n is an octal number representing the number of fields (4K) available to OS/8.  
It is in the range of 0-7.

Example:

```
_.MEMORY 3
16K MEMORY
```

The following table lists the values of n and their meanings.

n	memory
0	all available memory
1	8K
2	12K
3	16K
4	20K
5	24K
6	28K
7	32K

1-70 (cont.)

To find the amount of memory actually being used by OS/8, type the command with no argument.

.\_MEMORY

20K/32K MEMORY

In this example, a 32K system has been restricted to only 20K of available memory. This was done by using a MEMORY 4 command.

If all available memory is being used, the total amount of memory is printed.

Example:

.\_MEMORY

32K MEMORY

This command causes the execution of the CCL.SV program.

1-74

Add the following note to the end of the section "UA, UB, UC Commands":

"The CCL commands UA, UB, and UC are used to remember and recall arguments. These arguments are not deleted when the system date is changed. Most other CCL commands do not remember commands typed at sessions on previous days."

1-77

In "Table 1-18 CCL Error Messages (Cont.)", change "% SUPERCEDING" to "% SUPERSEDING".

1-79

Delete the /A option from Table 1-19.

1-81

In Table 1-21, insert a dot (.) in the lefthand column ("Character") of the first line.

1-99

In the second paragraph under the /I option, change "-n" to "=n". The missing example should be:

\*IMPORT.PA[23]</I=27

Delete /L option and its description.

**Page**

**Addition/Correction**

1-102

Add the following NOTE to the end of Table 1-24:

“PIP does not ask the question ZERO SYS for a handler that is co-resident with the SYS: handler. For example, if both SYS: and LTA0 are LINCtape 0, a request to zero LINCtape 0 will not produce the question. This is a potentially dangerous command.”

1-120

In the last line of the first paragraph, change “0007” to “0006”.

1-122

In Table 1-28, insert a slash (/) in the lefthand column opposite the last line.

## CHAPTER 2

### CHANGES AND ADDITIONS

Page	Addition/Correction																								
2-3	<p>In Table 2-1 Run Time Options, append the following:</p> <p>/H Process the batch input file without echoing and without sending the \$JOB and \$END batch monitor commands to both the terminal and batch log.</p>																								
2-34	<p>In the first paragraph of the section entitled "OS/8 Device Handlers", change "Appendix H" to "Appendix G".</p>																								
2-36	<p>At the top of the page, change (DEC-S8-OSYSB-A-UC2) to:</p> <p>AL-4712C-BA</p> <p>After Table 2-7 change (DEC-S8-OSYSB-A-TC4) to:</p> <p>AR-4588C-BA</p>																								
2-37	<p>After Table 2-8 change (DEC-S8-OSYSB-A-PB2) to:</p> <p>AK-4660C-BA and (DEC-S8-OSYSB-A-PB3) to:</p> <p>AK-4671C-BA</p>																								
2-38	<p>In Table 2-9 after the entry "DF32 disk nonsystem handler" add:</p> <table border="0" style="width: 100%;"> <tr> <td>RX01SY disk system handler</td> <td>RX8E</td> <td>SYS</td> <td>RX01SY.BN</td> </tr> <tr> <td>RX01NS disk nonsystem handler</td> <td>RX01</td> <td>RXA0,RXA1</td> <td>RX01NS.BN</td> </tr> <tr> <td>VT50 VT-50 input handler</td> <td>VT50</td> <td>LST</td> <td>VT50.BN</td> </tr> <tr> <td>LQP line printer handler</td> <td>LQP</td> <td>LPT</td> <td>LQP.BN</td> </tr> <tr> <td>Octal block DUMP handler</td> <td>DUMP</td> <td>DUMP</td> <td>DUMP.BN</td> </tr> <tr> <td>RX78B disk nonsystem handler (for VT-78 only)</td> <td>RX01</td> <td>RXA2,RXA3</td> <td>RX78B.BN</td> </tr> </table>	RX01SY disk system handler	RX8E	SYS	RX01SY.BN	RX01NS disk nonsystem handler	RX01	RXA0,RXA1	RX01NS.BN	VT50 VT-50 input handler	VT50	LST	VT50.BN	LQP line printer handler	LQP	LPT	LQP.BN	Octal block DUMP handler	DUMP	DUMP	DUMP.BN	RX78B disk nonsystem handler (for VT-78 only)	RX01	RXA2,RXA3	RX78B.BN
RX01SY disk system handler	RX8E	SYS	RX01SY.BN																						
RX01NS disk nonsystem handler	RX01	RXA0,RXA1	RX01NS.BN																						
VT50 VT-50 input handler	VT50	LST	VT50.BN																						
LQP line printer handler	LQP	LPT	LQP.BN																						
Octal block DUMP handler	DUMP	DUMP	DUMP.BN																						
RX78B disk nonsystem handler (for VT-78 only)	RX01	RXA2,RXA3	RX78B.BN																						
2-47	<p>The last example on the page should be:</p> <p style="text-align: center;">\$UNLOAD TC:DTA3</p>																								

2-48

The first example on the page should be:

```
$UNLOAD TC:DTA0,DTA2
```

2-52

After the section entitled "VERSION" add:

SIZE

Syntax: \$SIZE aname or \$ SIZE aname=new value

aname must be the permanent name of a device currently marked as active.

Example: \$SIZE RF08=1777

changes the length of the RF08 handler to 1777.

Function: The SIZE command modifies word ten of a handler header block. Word ten specifies the size, in blocks, of a single platter on a system device.

2-56

In Table 2-11 after "?PLAT", add:

?SLOTS This error indicates you have inserted more than 8 groups of non-system handlers. Each slot may have more than one entry point. To correct, delete P NAMES until there are 8 or less non-system handlers.

2-60

In Table 2-12 at the top of the page, change "25-26=unused" to 25=RX01 disk; 26=unused.

Change "31-37=unused by Digital" to:

31-35 Unused by Digital

36 Dump Handler

37 Unused by Digital

In the section "ENTRY POINT OFFSET", change all occurrences of "7-23" to "7-24". In the list of devices, change "RK8 disk" to "RK8/RK8E disk", and add the following:

"RF/DF disk	24"
RXA0	30
RXA1	34

Also, change "Thus, the user-coded file . . ." to "Thus, the user-coded file devices should use entry points other than 7-24, 30, 34.

2-61

Add the following information:

#### CREATING A SYSTEM HANDLER

When creating a new system handler, the user must obey the following restrictions:

- (a) The length of a bootstrap must be greater than or equal to 21 (octal) locations. A bootstrap shorter than 21 locations must be padded, otherwise BUILD results are unpredictable.
- (b) The length of the bootstrap must be less than or equal to 177 (octal) locations.
- (c) If the system handler is a one-page handler, only the first 47 (octal) locations of the bootstrap are significant. The remaining locations are ignored and not written on the system device. Also, no handler may have more than 20 (octal) entry points.
- (d) If a system handler is 2 pages long, relative location 12 of the first page must contain a 3. The second page loads into location 27600 and is stored on block 66 of SYS:

2-69

Add the following to the NOTE: RALF is not fully supported by CREF.

2-70

In Table 2-14 (CREF Options), change the /E option to /A.

2-77

After the last paragraph on this page, add the following:

#### NOTE

If you want the date printed in your directory listings, it must be entered into the system prior to the DIRECT command.

2-97

Add the following examples to the "FILE ORIENTED TRANSFER PROGRAM (FOTP)". These examples will help give the user a better understanding of the program.

Transfer the file X.Y from disk to DEctape:

```
*DTAO:<X.Y
```

Transfer the files A, B, C, D and E from SYS: to DTA3:

```
*DTA3:<SYS:A,B,C,D,E
```

2-97 (cont.)

Transfer all FORTRAN source files from one DECtape to another, producing a log of those copied:

```
*DTA2:<DTA5:*.FT/L
```

List all FORTRAN and BASIC files on the line printer in order of appearance on DSK:

```
*LPT:<*.FT,*.BA
```

List all FORTRAN and BASIC files on the line printer listing all FORTRAN files before all BASIC files:

```
*LPT:<*.FT,*.BA/U
```

Copy all files other than .SV and .BN files from DTA3: to DSK: then copy all files other than those whose name begins with a K from DTA2: to DSK: . Log all files copied:

```
*DSK:<DTA3:*.SV,*.BN,DTA2:K?????.*/V/L
```

Copy the file A.B from DSK: to DTA1: changing its name to C. D. Give the new file today's data:

```
*DTA1:C.D<A.B/T
```

Copy all files from LTA2: which have the extension .PA to SYS: changing the extension to .PL allocating storage on SYS: without doing pre-deletions:

```
*SYS:*.PL<LTA2:*.PA/N
```

Find all files on RKA2: with the name FOO and any extension but which have today's date, and copy them to SYS: changing the file name to WXYZ yet keeping the extension:

```
*SYS:WXYZ.*<RKA2:FOO.*/*C
```

Delete all disk files (except those with today's date) which either have the extension .LS, .TM, or .BK and those whose file name begins with TMP:

```
*DSK:<*.LS,*.TM,*.BK,TMP???.*/D/O
```

Delete each .BN file for which there is a corresponding .PA file:

```
**BN<*.PA/D
```

Delete all .LS files on DTA3: for which there is a file on RKA0: with the same name but an extension of either .PA, .RA, or no extension:

```
*DTA3:*.LS<RKA0:*.PA,*.RA,*/D
```

2-97 (cont.)

Delete all files on the disk for which there are already copies on one of the four DECTape drives:

```
*DSK:<DTA0:*.*,DTA1:*.*,DTA2:*.*,DTA3:*.*/D
```

Produce a log of all files on DTA1: that have the file name FOO and an extension which is the same as any file on SYS: that has a one or two-character file name beginning with a "T". Do not perform any transfers or deletions:

```
*DTA1:FOO.*<SYS:T?.*/N/D/L
```

Change the name of the file DSK:FILE.PA to FILE2.PA:

```
*FILE2.PA<FILE.PA/R
```

Rename all files on DTA6: with a .PA extension to have a .PB extension:

```
*DTA6:*.PB<DTA6:*.PA/R
```

Change the extension from .RL to .OL of all files on DTA1: that correspond to files on DSK: with the same name and today's date:

```
*DTA1:*.OL<*,RL/C/R
```

2-106

In "Table 2-20 FOTP Options (Cont.)", add to the "/R" option:

- (a) The rename option (/R) now looks at the /T switch. If /T is typed then not only is the file renamed, but the new file receives today's date. Without /T, the new name has the same date as the old name.
- (b) The rename option (/R) now allows you to rename a file to its own name. This was not previously permitted. It is not very useful unless some other switch is included, for example /T.
- (c) If no output file is specified with /R, then FOTP assumes the same name as the first input file.

Example: To redate all files on a DECTape to Jan. 1, 1976:

```
._DATE 1/1/76
._RENAME DTA0:*.*/T
```

2-113

Add the following options to Table 2-22:

- /I Assume the input device is a cassette drive. An input device must also be specified on the command decoder line, but it is ignored. This option is used when there are no cassette handlers configured into your system. The drive number is specified as an option, e.g., /1 represents drive 1. The /I and /O options must not be used in the same command line.



- 2-113 (cont.)      /O    Assume the output device is a cassette drive. An output device must also be specified on the command decoder line, but is ignored. This option is used when there are no cassette handlers configured into your system. The drive number is specified as an option. The /I and /O options must not be used in the same command line.
- 2-140              In the fourth paragraph change the second sentence to:
- All error messages are of the form:
- ?XXX
- where XXX is a 3-letter mnemonic which references the list of error messages that appears at the end of this chapter.
- After the last sentence on this page, append:
- If TECO runs on a machine with at least 12K of memory, the error message and a description of the error message are displayed on the terminal.
- 2-142              In Table 2-28, under RUBOUT, append:
- If SET TTY SCOPE is typed followed by a carriage return, RUBOUT erases the character from the screen of a VT52.
- 2-143              In Table 2-28, under CTRL/C, append:
- If CTRL/C is not typed as the first character after an asterisk (\*) is printed, the ?XAB error message (execution aborted) is displayed and control returns to TECO.
- 2-146              In Table 2-30, under the Y command, append:
- If this command is issued while an output file containing text in the buffer is open, an error message is displayed. To avoid this situation, use HKY.
- In Table 2-30, under the P command, delete the description and replace with:
- Writes the content of the buffer onto the output file then clears the buffer and reads the next page of the input file into the buffer. A form feed is appended only if one was present when the buffer was read in.
- 2-152              In Table 2-35 delete ↑Rtext1\$text2\$ and the description.

- 2-155 Under COMMAND LOOPS in the sentence "If n is not supplied, a value of 4096 is assumed." change "4096" to "infinity".
- Under Q-Registers, in the sentence "In the number storage area, each Q-register can store one integer in the range  $-2048 \leq n \leq 2047$ " change the range to:
- $-4095 \leq n \leq 4095$
- 2-156 Change Table 2-37 as follows: The last sentence describing n%q\$ should read "If n is not present, it is assumed to be equal to 1."
- Replace the command n%\$ with n%q.
- Under the nUq command change the range from " $-2049 \leq n \leq 2048$ " to:
- $-4095 \leq n \leq 4095$ ".
- 2-159 In Table 2-39 delete the n-m"A and n-m"B commands and their descriptions.
- Under the n"G command, change the range from " $-2048 \leq n \leq 2047$ " to:
- $-4095 \leq n \leq 4095$ "
- 2-160 Under the section entitled Numeric Arguments, delete the second paragraph and insert the following:
- This leads to an important restriction on the maximum size of any numeric argument. Commands which require positive arguments must have an argument in the range  $0 < n < 8195$ , since 8195 is the largest number which may be stored in one TECO word. Commands which may have positive or negative arguments require an argument in the range  $-4095 < n < 4095$ , because -4095 is the smallest number which may be stored in 13 bits using 2's complement notation, while 4095 is the largest number which may be stored in this manner.
- 2-162 In Table 2-40, change the ↑V character to EO and the description to:
- EO is equivalent to the version of TECO which is currently being run. This manual describes TECO version 5.
- 2-163 In the last line before Table 2-41, change 4096 to 8192.
- In Table 2-41, under the - operator, change  $-2=4096$  to  $-2=2$ .

2-175 Replace figures 2-5 and 2-6 with the following:

Figure 2-5:

```
J !!! OUN OUS !
!<QNA-32"E 1% S $"!
!QNA-13"E OJUSTIFY$"!
!1%N$>!
!!JUSTIFY! QN-40"G!
!60-QN-QS<S $I $$^N $>
!0L QS%N$ Q5%$$ OJUSTIFY$"!
!60-QN"G 60-QN<S $I $$^N $>' '!
!L Z-. "G 01$'$
```

Figure 2-6:

```
.R TECO
*ERDTA1:MACRO.TES Y HXI HK$$
*ERDTA1:TEXT.ASS Y MISS
```

2-178 Replace the example for SUPER TECO with:

```
.GET SYS TECO
.ODT
2051/7420 7610
2134/7450 7410
↑C
.SAVE SYS STECO
```

Replace the current example of STECO (Super TECO) with the following example.

```
.R STECO
*ERBBB:$EWDDD:JUNK.$$ (WHERE BBB IS THE DEVICE WITH A LOST
FILE (EMPTY)). READ FROM BAD DEVICE BBB:
WRITE A FILE JUNK ON GOOD DEVICE DDD:

*NABCDE$QLT$$ SEARCH FOR NEXT "ABCDE" THEN TYPE THE
LINE. MULTIPLE SEARCHES MAY NEED TO
BE MADE.

*D. .... DELETE JUNK CHARACTERS FROM THE BE-
GINNING OF THE LINE PARTICULARLY A ^Z.

*EF$$ CLOSE THE JUNK FILE.

*EWDEV:FILEN.EX$$ OPEN A NEW FILE ON THE GOOD DEVICE
WITH THE PROPER FILE NAME.
```

**Page**

**Addition/Correction**

- 2-178 (cont.)      \*NUVWXYZ\$PWEF\$\$      SEARCH FOR THE END OF THE FILE TEXT  
"UVWXYZ" WRITING TO THE NEW FILE.
- \*EX\$\$      EXIT TO OS/8.
- In the section on "Incompatibilities Between OS/8 TECO and DECsystem-10 TECO,"  
change incompatibility #1 to read as follows: "The ^R command does not exist on  
DECsystem-10 TECO."
- 2-179      Step 6 – delete
- Step 7 – delete the word "↑V".
- In Table 2-44, add the commands:
- FS    search for a character string in the current buffer and replace with another string.
- FN    search for a character string in a page of the input file which may not have been  
                     read into the buffer, and replace with another string.
- ↑U    insert the specified character string into the text storage area of the Q-register.
- 2-180      In Table 2-44, under the heading POINTER POSITIONS, change the command mL to  
nL.
- 2-182, 183      In the section "Running TECO on the PDP-12" change all references to "↑W" to "W"
- 2-184      Table 2-45, delete all and replace with: Table 2-45 on next 3 pages.

Table 2-45 Summary of TECO Error Messages

Printout Abbreviation	Printout Message	Meaning
?ARG	IMPROPER ARGUMENTS	Number missing before comma, or two arguments specified to D, or three numeric arguments
?BNI	NOT AN ITERATION	Iteration close (>) without matching open (<)
?CCL	CCL.SV NOT FOUND OR EG ARGUMENT TOO BIG	CCL not found or EG argument too long
?FER	FILE ERROR	FILE ERROR can mean: 1) input file not found on "ER" command 2) cannot enter output file on "EW" or "EB" command 3) device specified for file does not exist 4) "EB" command given on non-file structured device
?FUL	OUTPUT COMMAND WOULD HAVE	Output command would have overflowed output file (panic mode)
?IEC	ILLEGAL CHARACTER X* AFTER E	E followed by an illegal character
?IFC	ILLEGAL CHARACTER X* AFTER F	F followed by an illegal character
?IFN	ILLEGAL CHARACTER X*** IN FILE NAME	Illegal file name in "ER", "EW" or command
?ILL	ILLEGAL COMMAND X*	Illegal command
?INP	INPUT ERROR	Parity error on input file
?IQC	ILLEGAL CHARACTER X* AFTER " "	" " followed by an illegal command
?IQN	ILLEGAL Q REGISTER NAME X**	Non-alphanumeric Q-register name
?MEM	STORAGE CAPACITY EXCEEDED	Text buffer overflow
?NAC	NEGATIVE ARGUMENT TO,	Negative argument to comma
?NAE	NO ARGUMENT BEFORE =	No numeric argument to the left of an equal sign

(continued on next page)

Table 2-45 (cont.)

Printout Abbreviation	Printout Message	Meaning
?NAP	NEGATIVE OR ZERO ARGUMENT TO P	Negative or zero argument to P
?NAQ	NO ARGUMENT BEFORE QUOTE	No numeric argument to the left of a quote
?NAS	NEGATIVE OR ZERO ARGUMENT TO S	Negative or zero argument with a search
?NAU	NO ARGUMENT BEFORE U	No numeric argument to the left of a U
?NAY	NUMERIC ARGUMENT TO Y	Numeric argument specified with Y command
?NFO	VERSION NUMBER TO FILE FOR OUTPUT	Attempt to output without opening an output file
?NYI	CASE SUPPORT NOT IMPLEMENTED	Case support not implemented (use EO for version)
?NYI	CASE SUPPORT NOT IMPLEMENTED	Case support not implemented (use W for watch)
?OUT	OUTPUT ERROR	Output file too big or output parity error
?PDO	INTERNAL PUSHDOWN OVERFLOW	Pushdown overflow (macros and iterations nested too deeply)
?POP	ATTEMPT TO MOVE POINTER OFF PAGE	Attempt to move pointer outside of text buffer
?QMO	Q REGISTER MEMORY OVERFLOW	Q-register storage overflow
?SNI	NOT IN AN ITERATION	Semicolon on command level
?SRH	SEARCH FAILED	Failing search at command level
?STL	SEARCH STRING TOO LONG	Search string too large (greater than 31 characters)
?UTC	UNTERMINATED COMMAND	Incomplete command (PDL not empty at end of command string)
?UTM	UNTERMINATED MACRO	Incomplete command (PDI not empty at end of macro)
?WLO	CANNOT WRITE OUT ERROR MESSAGE OVERLAY	Write locked system device

(continued on next page)

Table 2-45 (cont.)

Printout Abbreviation	Printout Message	Meaning
?XAB		Execution aborted
?YCA	Y COMMAND ABORTED	Y (or .) command aborted because data would be lost

Page

Addition/Correction

2-184

Add the following information about the DECTape Formatting and DECTape Copying Programs at the end of Chapter 2.

**DTFRMT**

This program records the required timing and mark tracks on a DECTape mounted on the TC01-TU55 unit or a TC08-TU56 DECTape unit.

The program interacts with you via the terminal to obtain the necessary data for each set of DECTapes to be formatted. As soon as one set of tapes is formatted, the program is ready to format another set.

Two full passes are required to completely format each DECTape, and up to eight DECTapes may be formatted at a time (assuming that the user has eight tape transports). Upon completion of a cycle, new tapes may be mounted and formatted as the last, with a minimum of operator-program communication.

**PRELIMINARY REQUIREMENTS**

**Equipment**

PDP-8, terminal, TC01-TU55 DECTape Control.

**LOADING PROCEDURE**

Load the program into core using the standard Binary Loader.

**USING THE PROGRAM**

**Starting Procedure**

- a. Key 1000 into the SWITCH REGISTER. Depress LOAD ADDRESS and depress START. "DTA?" is printed on the terminal.

Mount the DECTapes to be marked onto the tape transports, with just enough turns of tape on the right hand reel of each transport to provide a grip. Make sure that no two tape units are set to the same unit number. Set the RDMK-WRTM-NORMAL switch located on the TC01 maintenance control panel to the WRTM position; for each transport to be used, set the WRITE ENABLED-WRITE LOCK switch to WRITE ENABLED, and the REMOTE-OFF-LOCAL switch to REMOTE.

2-184(cont.)

## Operating Procedures

The user type: R DTFRMT in response to the monitor dot (.). The program and operator now converse. The printout "DTA?" is asking which DECTape units will be used. The operator types a unit number or series of unit numbers, corresponding to the DECTape units upon which he has mounted tapes. For instance, if the operator has mounted tapes on units 2, 5, 7, and 8, he would type 2 5 7 8 ↵ (where ↵ signifies carriage return). Spaces are ignored, so it makes no difference if the operator types spaces between the unit numbers. Only one specification of a unit is significant, i.e., typing 2 2 5 7 7 5 8 2 8 ↵ has the same effect as typing 2 5 7 8 ↵.

Once the operator has specified the units he wishes to use, the program types "DIRECT?". The operator responds by typing MARK ↵ or MARK XXXX ↵. If he types MARK ↵, the program assumes 201g words, 2702g blocks (standard PDP-8 format). Otherwise, XXXX is accepted as a decimal number of words per block, and must be divisible by 3. Note that typing MARK 384 ↵ will cause the program to generate a standard PDP-10 format DECTape (1102g blocks of 600g words, which is equivalent to 1102g blocks of 200g words, where each word is 36 bits rather than 12 bits).

The program now types "XXXX WORDS, YYYY BLOCKS OK? (YES OR NO)." This serves as a final check for block count. XXXX and YYYY are octal values representing the final outcome of a formula solved by the program, determining the number of blocks that may be written on a DECTape knowing the number of words. If a NO ↵ answer is given, the program reverts to "DIRECT?". Otherwise (if YES ↵), the tape on the first unit specified begins to move.

Once all of the tapes specified have been marked, the printout "SET SWITCH TO NORMAL" appears. Then the operator returns the "RDMK-WRTM-NORMAL" switch to NORMAL, and strikes the RETURN key on the terminal, starting the second pass. Note that during the second pass with multiple DECTape units, as soon as one tape stops and the next tape starts, the first tape is completed and may be replaced with a fresh tape in preparation for recycling.

The program continues by itself until completed, at which time the "DIRECT?" printout occurs. Typing "SAME ↵" repeats the entire process with the original constants. The new DECTapes must be mounted and ready to write timing and mark tracks before "SAME ↵" is typed. Also, in response to "DIRECT?", typing "RDR ↵" causes the printout of the unit numbers of the DECTapes and the last twelve block numbers; "RDF ↵" causes the printout of the unit numbers and the first twelve block numbers; and "RESTART ↵" returns the program to "DTA?". Unit numbers are printed as "N000", where N is the unit number (0 means DECTape unit 8). Once formatting begins, control C will cause the program to restart at "DTA?". If the user wishes to return to the monitor another control C may be typed at this time.

Following are several examples of successful operation. The underlined portions are printed by the program. ALL operator responses should be followed by a carriage return.



2-184 (cont.)

- a. Create a standard tape on unit 4.

DTA? 4  
DIRECT? MARK  
0201 WORDS, 2702 BLOCKS.OK? YES OR NO  
 YES  
SET SWITCH TO NORMAL  
DIRECT?

- b. Create 16 standard PDP-10 format tapes – eight at a time, on units 1-8.

DTA? 12345678  
DIRECT? MARK 384  
0600 WORDS, 1102 BLOCKS OK? YES OR NO  
 YES  
SET SWITCH TO NORMAL (USER TYPES ↘ )  
DIRECT? SAME  
SET SWITCH TO NORMAL (USER TYPES ↘ )  
DIRECT?

#### Errors

Errors Types to “DTA?” and “DIRECT?” – Revert back to “DTA?” or “DIRECT?”

Error Messages for Response to MARK XXXX –

NOT DECIMAL	A character in XXXX is not 0-9.
NOT DIVISIBLE BY 3	XXXX cannot be divided evenly by 3.
TOO MANY WORDS	The number of words plus 15 exceeds 7777 <sub>8</sub> .
TOO MANY BLOCKS	The number of blocks generated by XXXX exceeds 7777 <sub>8</sub> .

Error Messages for Response to YES (After message – revert back “DTA?”)

SETUP?	Indicates an error in the DECTape setup –
	Unit in WRITE LOCK
	Nonselectable unit
	Switch not in WRTM position

Error Messages for Marking and Verifying a Tape

XXXX SHOULD BE YYYY BLK ERROR PHASE X  
 XXXX SHOULD BE YYYY DATA ERROR PHASE X  
 END TAPE ERROR PHASE X  
 MARK TRACK ERROR PHASE X  
 PARITY ERROR PHASE X

2-184 (cont.)

SELECT ERROR PHASE X  
 TIMING ERROR PHASE X  
 LAST INT NOT END ZONE

**Recovery**

Although error should cause doubt concerning the entire process, restarts may be made by phases (except when in phase 0). Restart the phase by typing "RETRY ↵". Type "RESTART" to return to "DTA?"

PHASE 0: MARK TRACK WRITE  
 PHASE 1: WRITING LAST REVERSE BLOCK NUMBER FORWARD  
 PHASE 2: WRITING BLOCK NUMBERS AND DATA IN REVERSE  
 PHASE 3: READING AND CHECKING BLOCK NUMBERS AND DATA

An error that should be considered catastrophic is LAST INT NOT END ZONE. This indicates that between the last (or first) block number and the end zone, something caused an interrupt (DTF).

The entire program may be restarted at 1000g any time.

**DETAILS OF OPERATION AND STORAGE**

The program writes timing and mark tracks on a DECTape, then inserts block numbers and parity correct information, checking the results of all operations.

The number of block frames to be written is a function of the number of words per block. The formula

$$\text{blocks per tape} = \frac{212080}{NW+15} + 2$$

where NW equals the number of words to be written, is used by the program to compute the number of blocks, but is adjusted by the program to provide the standard PDP-8 format of 129 (12-bit) words, 1744 blocks, and standard PDP-10 format of 128 (36-bit) words, 578 blocks.

Two full passes are required to mark and verify a tape.

Pass 1      Marks the tape forward, inserts block numbers and parity correct data in reverse.  
 Pass 2      Reads and checks block numbers and data forward and reverse.

During the forward direction of the first pass, the TC01 is switched into WRITE TIMING AND MARK TRACKS, CONTINUOUS MODE, FORWARD. The program manipulates data to be written by monitoring the word count register and the DTF, (DECTape flag). Initially, ten feet of end-zone code is written, and abutting the end zone are about two standard block lengths of interblock sync. To the TC01, this interblock sync acts as no operation, but guarantees that at turn-around time, block 0 is read first (or 2701 if turning out of the forward end zone). Now the remainder of the tape is written creating block frames. The number of such frames is determined by the above formula. Upon completion of the block framing; another extended interblock sync zone is written as well as ten feet of end zone.

2-184 (cont.)

Pass 1 forward is now complete (timing and mark tracks are written). The tape is ordered to MOVE in reverse for three seconds, thus moving it out of the end zone and onto the marked section. The tape is once again moved forward, and the last REVERSE BLOCK NUMBER is written until the forward end zone is sensed. Now the tape is turned out of the end zone in SEARCH, and the program waits for a block interrupt (first reverse block number). When the DTF rises, the TC01 is switched into WRITE ALL, CONTINUOUS, REVERSE; thus the system is synchronized and all block numbers and data are written: until the forward end zone is sensed. This completes the marking and blocking of the tape. Pass 2 in CONTINUOUS MODE checks the data and block numbers to be certain they are correct. When multiple DECTape units are specified, Pass 1 forward is completed for each tape before Pass 1 reverse is begun.

### TDFRMT

The TD8-E DECTape formatter program records the timing and mark tracks on a DECTape mounted on the TU56 DECTape transport.

The program interacts with the operator via the terminal to obtain the necessary data for each set of DECTapes to be formatted. As soon as one set of tapes is formatted, the program is ready to format another set.

Three full passes are required to completely format each DECTape, and up to two DECTapes may be formatted at a time (units 0 and 1). Upon completion of a cycle, new tapes may be mounted and formatted as the last, with a minimum of operator-program communication. One tape excluding tape setup time, requires three minutes from start to finish.

Mount the DECTapes to be marked onto the tape transports with just enough turns of tape on the right hand reel of each transport to provide a grip. Make sure that no two tape units are set to the same unit number. Set the switch on the TD8-E to WTM position. For each transport to be used, set the WRITE-ENABLED-WRITE LOCK switch to WRITE ENABLED, and the REMOTE-OFF-LOCAL switch to REMOTE.

### OPERATING PROCEDURES

The user types .R TDFRMT in response to the monitor dot (.). The program and operator now converse. The printout "UNIT?" is asking which DECTape units will be used. The operator types one or two unit numbers corresponding to the DECTape units upon which he has mounted tapes. For instance, if the operator has mounted tapes on units 0 and 1, he would type 0 1. Spaces are ignored, so it makes no difference if the operator types spaces between the unit numbers. Only one specification of a unit is significant, i.e., typing 000111 has the same effect as typing 01.

Once the operator has specified the unit(s) he wishes to use, the program types "FORMAT?". The operator responds by typing MARK or MARK XXXX. If he types MARK, the program assumes 201 words 2702 blocks (standard PDP-8 format). Otherwise XXXX is accepted as a decimal number of words per block and must be divisible by 3. Note that typing MARK 384 will cause the program to generate standard PDP-10 format DECTapes (1102(8) blocks of 600 words, which is equivalent to 1102(8) blocks of 200 words where each word is 36 bits rather than 12 bits).

2-184 (cont.)

The program now types "XXXX WORDS, YYYY BLOCKS OK? (YES OR NO)". This serves as a final check for block count. XXXX and YYYY are octal values representing the final outcome of a formula solved by the program, determining the number of blocks that may be written on DECTape knowing the number of words. If a no answer is given, the program reverts to "FORMAT?". Otherwise (IF YES), the program types out "SET SWITCH TO WTM". Then the operator hits carriage return on the teletype and the tape on first unit specified begins to move if the switch is set.

Once all of the tapes specified have been marked, the printout "SET SWITCH TO OFF" appears, then the operator resets the WTM switch to off, and strikes the return key on the terminal starting the second pass. Note that during the second pass with multiple DECTape units, as soon as one tape stops and the next tape starts, the first tape is completed and may be replaced with a fresh tape in preparation for recycling.

The program continues by itself until completed at which time the "FORMAT" printout occurs. Typing "SAME<" repeats the entire process with the original constants. The new DECTapes must be mounted and ready to write timing and mark tracks before a carriage return is hit on the teletype after the typeout "SET SWITCH TO WTM". Also, in response to "DIRECT?", typing "RDR" causes the printout of the unit number of the DECTape and the last 22 block numbers; "RDF<" causes the printout of the unit number and the first 22 block numbers; and "RESTART<" returns the program to "UNIT?". Unit numbers are printed as "000N" where N is the unit number.

Following are several examples of successful operation. The underlined statements are printed by the program. All operator responses should be followed by a carriage return.

A. CREATE A STANDARD PDP-8 TAPE ON UNIT 1

UNIT? 1  
FORMAT? MARK  
0201 WORDS, 2702 BLOCKS, OK? (YES OR NO)  
 YES  
SET SWITCH TO WTM  
SET SWITCH TO OFF  
FORMAT?

B. CREATE 4 STANDARD PDP-10 FORMAT TAPES, TWO AT A TIME ON UNITS 011

UNIT? 01  
FORMAT? MARK 384  
0600 WORDS, 1102 BLOCKS OK? (YES OR NO)  
 YES  
SET SWITCH TO WTM  
SET SWITCH TO OFF  
FORMAT? SAME  
SET SWITCH TO WTM  
SET SWITCH TO OFF  
FORMAT?

2-184 (cont.)

**ERRORS**

Errors typed to "UNIT" and "FORMAT" revert back to "UNIT?" or "FORMAT?".

Error messages for response to MARK XXXX

NOT DECIMAL	A CHARACTER IN XXXX IS NOT 0-9
NOT DIVISIBLE BY 3	XXXX CANNOT BE DIVIDED EVENLY BY 3
TOO MANY WORDS	THE NUMBER OF WORDS PLUS 15 EXCEEDS 7777(8).
TOO MANY BLOCKS	THE NUMBER OF BLOCKS GENERATED BY XXXX EXCEEDS 7777

Error messages for response to "SET SWITCH TO WTM".

1. SETUP? indicates an error in the DEctape setup. One of the units specified is in write lock position, not selected, or the write flip-flop is unable to be set, or there may be a timing error. (After message revert back to "UNIT".)
2. Switch not set to WTM or single line flag failed to set. Set switch to WTM.

This type out says that either the switch on the M868 modules is not set to the WTM position or the timing generator for writing the mark and timing tracks is not setting the single line flag.

**RECOVERY:**

If the switch was not set to WTM position set the switch and hit carriage return on the teletype.

If the switch was set to WTM position and this type out occurred, try again or examine the timing generator circuit.

Error messages for marking and verifying a tape

PC	XXXX	MARK TRACK ERROR PHASE Y
PC	XXXX	BLOCK NUMBER ERROR PHASE Y
PC	XXXX	DATA ERROR PHASE Y
PC	XXXX	CHECKSUM ERROR PHASE Y
PC	XXXX	TIMING ERROR PHASE Y
PC	XXXX	WRITE ERROR PHASE Y

XXXX equals the program counter at time of the failure. Y equals the pass which it was in.

2-184 (cont.)

Although an error should cause doubt concerning the entire process, a restart may be made (except in phase 0) by typing "RETRY<". Retry causes the program to go back to phase 1, type "RESTART<" to return to "UNIT?".

PHASE 0: WRITE TIMING AND MARK TRACK FORWARD  
 PHASE 1: READS MARK TRACK REVERSE  
 PHASE 2: WRITE DATA, FORWARD BLOCK AND REVERSE BLOCK  
 NUMBERS FORWARD AND WRITES THE CHECKSUMS  
 PHASE 3: DISPLAYS BLOCK NUMBERS IN AC REVERSE  
 PHASE 4: READS DATA, FORWARD BLOCK AND REVERSE BLOCK  
 NUMBERS FORWARD AND CALCULATES THE CHECKSUM  
 PHASE 5: READS REVERSE BLOCK NUMBERS IN REVERSE

The entire program may be restarted at 0200 any time.

#### DETAILS OF OPERATION AND STORAGE

The program writes timing and mark track on a DECtape forward with WTM switch set. Then it reads the mark track in the reverse direction with the switch set to off. The program checks all of the mark track once it is in sync. (see flow figure 1) when it finishes reading the mark track reverse, it bounces off the end zone and starts writing zeroes to the first block mark. The program is now in sync. The program now continues writing forward block numbers, reverse checksum, data, checksum, and reverse block numbers for the rest of tape. When it sees the end zone, it turns around and starts displaying the reverse block number in the accumulator until it hits the end zone again. Now the tape turns around and starts reading and comparing all forward block numbers, reverse checksum, all data, checksum and reverse block numbers that was written in Phase 2. This comparison is done on all blocks until the end zone is reached. The tape turns around in the end zone and starts looking for reverse block numbers and comparing them all the way down tape to the end zone. The formatting is now complete, the tape stops, and "FORMAT" is typed out waiting for new directions.

The number of block frames to be written is a function of the number of words per block.

The formula

$$\text{BLOCKS PER TAPE} = [(212080)/(\text{NW}+15)]+2$$

where NW equals the number of words to be written, is used by the program to compute the number of blocks, but is adjusted by the program to provide the standard PDP-8 format of 129(10) (12-bit) words, 1474(10) blocks, and standard PDP-10 format of 128(10) (36-bit) words, 578(10) blocks.

#### Theory

The writing of the mark track is done through AC bits 0, 3, 6 and 9. The following description is how the mark track is written.

2-184 (cont.)

- A. Install the tape with enough turns to create a pull. The reverse end zone requires a sequence of three data words for its pattern.

4044  
0440  
4404

In the mark track the words appear at 101101101101101 (5555(8)). The reverse end zone should cover about 10 feet of tape. Write the above three words 4096(10) times.

- B. Write the below three words (see C) or expand code 99 times.  
C. Expand code, three words of expand code should immediately follow each block,

0404  
0404  
0404

In the mark track the words appear as 010101010101 (2525(8)).

- D. The forward block mark and reverse guard require three words.

0404  
4004  
4040

Which appear on the mark track as 010110011010 (2632(8)).

- E. The lock mark, reverse checksum, reverse final, reverse prefinal consist of six PDP-8 memory words,

0040  
0000  
4000  
0040  
0000  
4000

These words appear on the mark track as 001000001000001000001000 (10101010(8)).

- F. Mark track code for data is generated by

4440  
0044  
4000

These three words appear as 111000111000(7070(8)) and are repeated 41(10) times for a 129 word block.

2-184 (cont.)

G. The prefinal, final, checksum, and reverse lock consist of six PDP-8 words.

4440  
4444  
4044  
4440  
4444  
4044

These words appear on the mark track as 111011111011111011111011  
(73737373(8)).

H. The guard and reverse block mark consist of three words

4040  
0440  
0404

which appear as 101001100101 (5145(8)).

I. Generate 2702(8) block patterns. Repeat C through H. 2702(8) times.

J. 100 expand codes (see C).

K. The end zone pattern consist of three words,

0400  
4004  
0040

which appears on the mark track as 010010010010 (2222(8)). Repeat these 3 words  
4096(10) times. See Figure 2 for a diagram of the mark track and data tracks.

### DTCOPY

The TC01, TC08, TU-55 Copy Program is controlled through a dialog on the terminal. The responses to the questions are in the form of octal numbers followed by a carriage return. Where more than one answer is required to a question, the answers are separated by semicolons. Alphabetic or other illegal characters will cause an error message to be generated and the question to be repeated. If too many digits are typed for the response expected, only the last ones typed will be used. If the response was to be either 0 or 1 (YES or NO), a non-zero final digit will be assumed to be 1.

Before answering the dialog's questions, the user must ensure that all the DECTapes involved are mounted on their respective drives. All the drives must be set to REMOTE. The input drive may be set to WRITE LOCK or WRITE ENABLE; all output drives must be set to WRITE ENABLE. No two drives may have the same unit number.

The user types R DTCOPY in response to the monitor dot (.). The program types DECTape COPY V10A.



2-184 (cont.)

For each set of copies, the dialog is as follows (the user's response is underlined; (CR) means carriage return):

```

DECTAPE COPY V10A
FROM UNIT 0

TO UNIT 2

FIRST BLOCK TO COPY (OCTAL) 0

FINAL BLOCK TO COPY (OCTAL) 700

PDP-8 WORDS PER BLOCK 0201

VERIFY OUTPUT? (0=YES, 1=NO): 0

```

When all specified copies have been finished, the tapes are rewound and the dialog continues:

```

DONE
DECTAPE COPY V10A
FROM UNIT

```

The user may return to the monitor by typing CTRL/C at anytime. (Control characters are not echoed printed.)

#### ERROR MESSAGES

- ILLEGAL RESPONSE**      The user's response to the dialog was not correct; for example, an alphabetic character was typed or carriage return was typed before an octal number was given where one was required. The question will be restated and any previous answer ignored. Nothing should be typed until the terminal has stopped printing.
- SELECT ERROR UNIT n**      During attempted data transfer, unit n was not found. The program waits for the user to correct the cause of the error. The user should check to see that:
1. if unit n is an output drive, it is set to WRITE ENABLE.
  2. unit n is set to REMOTE.
  3. there is only one unit n.
  4. all units are set to numbers appropriate to their TD8E internal wiring.
- When the cause of the error has been corrected, the user may type CTRL/R to resume transfer or he may type CTRL/S to restart the dialog.

2-184 (cont.)

**TAPE ERROR BLOCK x UNIT n**

During attempted transfer, a parity error or timing error was detected, or too great a block number was requested near block x on the tape on unit n. The tapes are rewound and the dialog is automatically restarted at DONE, REPEAT (YES=1, NO=0).

**VERIFY ERROR BLOCK x UNIT n**

The data on the input tape does not match the data which was written on block x of the output tape on unit n. The user may type CTRL/R to ignore the error and continue with the transfer, CTRL/T to try the last transfer again and continue if the error does not recur, or CTRL/S to restart the dialog.

**ILLEGAL FORMAT UNIT n**

Either the number of words per block on unit n does not agree with the number of words per block on the input unit or, when the number of blocks on the tape was calculated from the block length of the input tape, the length was found to be illegal. The number of blocks is only calculated if the whole tape copy option is requested. In either case, when the error has been corrected, the user may type CTRL/R to check the formats of all tapes again and continue, or CTRL/S to restart the dialog.

**TDCOPY**

TD8E Copy is controlled through a dialog on the terminal. The response to the questions are in the form of octal numbers followed by a carriage return. Where more than one answer is required to a question, the answers are separated by semicolons. Alphabetic or other illegal characters will cause an error message to be generated and the question to be repeated. If too many digits are typed for the response expected, only the last ones typed will be used. If the response was to be either 0 or 1 (YES or NO), a non-zero final digit will be assumed to be 1.

Before answering the dialog's questions, the user must ensure that all the DECtapes involved are mounted on their respective drives. All the drives must be set to REMOTE. The input drive may be set to WRITE LOCK or WRITE ENABLE; all output drives must be set to WRITE ENABLE. No two drives may have the same unit number.

The user types R TDCOPY in response to the monitor dot(.). The program prints:

```
TD8E COPY V4A
HIGHEST FIELD AVAILABLE:
```

The user response with the number of the highest field he wishes used for buffer space. This response may allow data to be preserved in any higher field or may make full use of the memory available. This question is asked only once, immediately after the program has been loaded. To change the response, the program must be executed again. If 4K of memory is to be used, the response is 0; if 8K, the response is 1, and so forth.

2-184 (cont.)

For each set of copies, the dialog is as follows (the user's response is underlined>; (CR) means carriage return):

Dialog	Comments
FROM UNIT: 0 (CR)	User may specify one unit number.
TO UNITS: 1; 2; 3; 4; 5; 6; 7 (CR)	User may specify up to 7 unit numbers, separated by semicolons.
FIRST INPUT BLOCK: 100 (CR)	User may supply any legal DECTape block number.
FIRST OUTPUT BLOCK: 200 (CR)	User may supply any legal DECTape block number.
NUMBER OF BLOCKS TO COPY: 50 (CR)	User may supply appropriate number of blocks.
VERIFY OUTPUT (YES=1, NO=0): 1 (CR)	
0201 12-BIT WORDS PER BLOCK	Determined by program from tape on input unit.

The block length of all the specified tapes is checked. If any are found to be different from the input tape, the ILLEGAL FORMAT UNIT n error message is generated.

When all specified copies have been finished, the tapes are rewound and the dialog continues:

DONE

REPEAT (YES=1, NO=0):

If there are more tapes to be copied with the same set of specifications, they should be placed on the drives before typing 1 to repeat the previous operation. If a different set of specifications is desired, 0 should be typed to restart the dialog.

Occasionally a TD8E drive will not stop fast enough after the tapes have been rewound and the end of the tape will spin off the reel. If this should happen, the drive may be stopped manually by setting it to OFF and stopping the reel by hand. This will not affect the validity of the copy. If the dialog does not continue properly after one or more tapes have spun off, the program may be restarted.

In response to any question in the dialog, the user may type either CTRL/S to restart the dialog at REPEAT (YES=1, NO=0) or CTRL/C to exit the monitor. Either CTRL/S or CTRL/C may also be typed during a small amount of further motion. If CTRL/S is typed during the dialog the response to the REPEAT question should be NO; this option is mainly for cases where a complete set of specifications is already available.

(CTRL/ characters are typed by holding the CONTROL key down while typing the character. The procedure is similar to that used with the SHIFT key on a typewriter. CTRL/ characters are not echoed (printed).)

2-184 (cont.)

A special case of the dialog allows the entire input tape to be copied onto the output tape with a minimum of effort. This case eliminates the need to specify the starting block numbers and number of blocks to copy. In this case, the answer to FIRST INPUT BLOCK: is only a carriage return. The shortened dialog will be as follows:

```

TD8E COPY
FROM UNIT: 0 (CR)
TO UNITS: 1;2;3;4;5;6;7 (CR)
FIRST INPUT BLOCK: (CR)
VERIFY OUTPUT (YES=1, NO=0): 1 (CR)
0201 12-BIT WORDS PER BLOCK

```

The preceding sample dialog will cause the entire tape on unit 0 to be copied onto the other 7 tapes and verified.

#### ERROR MESSAGES

##### ILLEGAL RESPONSE

The user's response to the dialog was not correct; for example, an alphabetic character was typed or carriage return was typed before an octal number was given where one was required. The questions will be restated and any previous answer ignored. Nothing should be typed until the terminal has stopped printing.

##### SELECT ERROR UNIT n

During attempted data transfer, unit n was not found. The program waits for the user to correct the cause of the error. The user should check to see that:

1. if unit n is an output drive, it is set to WRITE ENABLE.
2. unit n is set to REMOTE.
3. there is only one unit n.
4. all units are set to numbers appropriate to their TD8E internal wiring.

When the cause of the error has been corrected, the user may type CTRL/R to resume transfer or he may type CTRL/S to restart the dialog.

##### TAPE ERROR BLOCK x UNIT n

During attempted transfer, a parity error or timing error was detected, or too great a block number was requested near block x on the tape on unit n. The tapes are rewound and the dialog is automatically restarted at DONE, REPEAT (YES=1, NO=0).

##### VERIFY ERROR BLOCK x UNIT n

The data on the input tape does not match the data which was written on the block x of the output tape on unit n.

2-184 (cont.)

The user may type CTRL/R to ignore the error and continue with the transfer, CTRL/T to try the last transfer again and continue if the error does not recur, or CTRL/S to restart the dialog.

#### ILLEGAL FORMAT UNIT n

Either the number of words per block on unit n does not agree with the number of words per block on the input unit or, when the number of blocks on the tape was calculated from the block length of the input tape, the length was found to be illegal. The number of blocks is only calculated if the whole tape copy option is requested. In either case, when the error has been corrected, the user may type CTRL/R to check the formats of all tapes again and continue, or CTRL/S to restart the dialog.

#### DETAILS OF OPERATION

After the answers to the dialog have been stored, the following procedure is used:

1. The number of words per block is determined from the input tape. All output tapes are checked to see if they have the same format as the input tape. If the shortened dialog option was used, the number of blocks on the tape is determined using the formula:  
$$\# \text{ of blocks} = (636,160 / (\text{words per block} + 17)) + 2 \text{ (octal)}$$

or

$$\# \text{ of blocks} = (212,080 / (\text{words per block} + 15)) + 2 \text{ (decimal)}$$
2. The response to the VERIFY question is checked. The copying loop is set up to verify or not, as was requested.
3. The loop is entered which copies the input tape, using the same set of specifications for each output tape.
  - a. The buffers are filled from the input tape.
  - b. All output tapes are written with the contents of the buffers.
  - c. If verification was requested, a separate set of buffers is filled from the output tape and the two sets of buffers are compared. If there are any discrepancies a VERIFY ERROR has occurred.
  - d. If more blocks remain to be copied, the loop is entered again.
4. When all the specified blocks have been copied onto the output tapes, all the tapes are rewound.
5. The REPEAT option is offered.

2-184 (cont.)

The number of fields to be used for buffer space is determined immediately after loading. As soon as the question has been answered, it is removed from the program.

If the output tape is to be verified, each available field, including that part of field 0 not occupied by the program, is divided in half. The lower half is used as the input and output buffer; the upper half is used for verification. The output tape is read back into the upper half and the two halves are compared. If they are not identical, a VERIFY ERROR has occurred.



## CHAPTER 3

### ADDITIONS AND CHANGES

Page	Addition/Correction
3-29	<p>End-Of-File paragraph should read:</p> <p><b>End-Of-File</b> PAUSE signals the assembler to stop processing the file being read. A PAUSE should only be used at the physical end of a file and with two or more segments of one program. When a PAUSE statement is reached, the remainder of the file is ignored and processing continues with the next input file. In such a case PAUSE must be present or a PH error will occur. The PAUSE pseudo-op is present mainly for compatibility with paper tape assemblers, and its use is optional.</p>
3-30	<p>After the TEXT string example change the words "IF option" to "/F option".</p> <p>The second sentence in the second paragraph under "Suppressing the Listing" should read "XLIST may also be used with expression as an argument; a listing will be inhibited if the expression is not equal to zero, or allowed if the expression is equal to zero".</p>
3-31	<p>Insert the following example just before the section entitled "Controlling Binary Output":</p> <pre>IFZERO      A&lt; . . (code) . . &gt; ...</pre>
3-41	<p>Under "Memory Reference Instructions" change "JSM" to "JMS".</p>





## CHAPTER 5

### ADDITIONS AND CHANGES

Page

Addition/Correction

5-22

Add the following note on RALF assembly at the end of the page:

“RALF code that includes forward reference to the base page should employ pseudo-ops # and ' as the first character of the symbol; this permits RALF to generate symbols that do not conflict with programmer-generated symbols that are also on the base page. The # pseudo-op can also be used following FPP memory reference instructions to indicate use of the 2-word form of the instruction. Likewise, the ' pseudo-op indicates use of the single-word direct form of the instruction.



## CHAPTER 6

### ADDITIONS AND CHANGES

Page	Addition/Correction
6-35	<p>Randomize example (bottom of page) is incorrect.</p> <p style="text-align: center;">1 PRINT "A"</p> <p>should be added as the first line.</p>
6-61	<p>In the paragraph above the section entitled "File Statements" change "PS/8" to "OS/8" and change the order number of the manual to "DEC-S8-OSSMB-A-D".</p>
6-64	<p>The last sentence in the first paragraph should read: the subsequent reading of numbers from the file in line number 80 shows the use of a dummy argument (C) to compensate for the carriage return and line feed since they would otherwise be read as numeric data with a value of 0. Line 80 in the example should be: 80 INPUT #1:J,C.</p>
6-66	<p>Add the following option designations (bottom of page):</p> <p>/C In BCOMP, the /C option is used in conjunction with the /K option to create a file that can be chained to from a non-BASIC file. For example:</p> <p style="text-align: center;">.R BCOMP *EXAM.BA/C/K</p> <p>/V In BCOMP, the /V option is used to obtain the current version number of COMP, BLOAD, and BRTS. For example:</p> <p style="text-align: center;">.R BCOMP *EXAM.BA/V</p> <p>This causes the system to print at the console the current version numbers for BCOMP, BLOAD, and BRTS as part of the output of the file being compiled.</p>

6-69 Change the sentence preceding NOTES to:

In general, any departure from these procedures will cause a CX error.

Add the following to the List of CHAIN restrictions:

- 3. When chaining BASIC core image files, the program being chained to must be on the system device. This is a restriction of the USR CHAIN function.

6-76 Under the title THE STRING ACCUMULATOR (SAC), and starting on line 4, delete the sentence:

The SAC starts at location SAC for 36 words (72 characters), and the length of the string currently in the SAC is stored as a negative number in STRLEN.

and insert the following:

The SAC starts at location SAC in BRTS. The SAC is 80 words long and contains one 6-bit character per word. The length is stored as a negative number in SACLEN.

6-83 In the middle of the page, delete:

BRTS maintains links for FGET and FPUT on page 0 of field 0, providing convenient access to these frequently used routines.

Page 0	
Link Name	Routine Linked
FGETL	FFGET
FPUTL	FFPUT

and insert the following:

BRTS contains Page Zero literals used by the FGET and FPUT routines. These Page Zero literals can be found in the BRTS source listing. Page Zero literals reference the following routines. For more information on Page Zero literals, refer to the section on BRTS Subroutines.

Page Zero Link	Routine
FNEGL	FFNEG
FNORL	FFNOR
FCLR	FACCLR

6-85 (cont.)

In the middle of the page under the title FLOATING POINT OPERATIONS, and in the second paragraph, delete:

Page 0 links are maintained for negate, normalize, and clear.

Page 0 Link	Routine
FNEGL	FFNEG
FNORL	FFNOR
FCLR	FACCLR

At the bottom of the page before the title SUBROUTINE ARGPRE, insert the following:

Many routines are now addressed with Page Zero literals that can be found in the BRTS source listing. Note that explicit references to Page Zero pointers by name no longer apply. The purpose is to shorten the size of the BRTS symbol table.

6-90

Towards the bottom half of the page, delete the section entitled SUBROUTINE BSW.

6-108

In the example form for the CHAIN command (Table 6-1) the device and file name specification must be enclosed in quotation marks as follows:

CHAIN "dev:filename.ex"

6-117

In Table 6-3 Run-Time Diagnostics after the CI diagnostic code, insert the following:

CX Incompatible file extensions were used in BASIC CHAIN statement.

6-118

At the bottom of the page following the first paragraph, delete the column entitled "Distributed on:"

At the bottom of the page following the first paragraph, delete:

BASIC.BN	Binary for editor
BCOMP.BN	Compiler binary
BLOAD.BN	Loader binary
BRTS.BN	Run-time system binary (any PDP-8 or PDP-12)

6-119

At the top of the page following BLOAD.SV, insert the following:

EABRTS.SV KE8/EAE Version of Run-time System save image

At the top of the page under the column entitled "Component" that describes BRTS.SV, delete:

(from BRTS.BN)

6-119 (cont.)

In the middle of the page, delete:

**Making SAVE Images from Binary Files:**

To create SAVE images of each of the OS/8 BASIC binaries, perform the following OS/8 commands.

and insert the following:

**Making SAVE Images from Binary Files:**

To create SAVE images for each of the OS/8 BASIC binaries, use the following BUILD procedure for OS/8 BASIC non-EAE.

In the middle of the page delete:

1. For the editor:

```
.R ABSLDR
*DEV:BASIC.BN$
.SAVE SYS:BASIC;3011
```

and insert the following:

1. For the editor:

```
.PAL BASIC
.LOAD BASIC
.SAVE SYS:BASIC;3211
```

Towards the bottom of the page, delete:

2. For the compiler:

```
.R ABSLDR
*DEV:BCOMP.BN$
.SAVE SYS:BCOMP;7000
```

and insert the following:

2. For the compiler:

```
.PAL BCOMP
.LOAD BCOMP
.SAVE SYS:BCOMP;7000
```

6-119 (cont.)

at the bottom of the page, delete:

3. For the loader:

```
.R ABSLDR
*DEV:BLOAD.BN$
.SAVE SYS:BLOAD;7605
```

and insert the following:

3. For the loader:

```
.PAL BLOAD
.LOAD BLOAD
.SAVE SYS:BLOAD;7605
```

At the bottom of the page, delete:

4. For the run-time system:

```
.R ABSLDR
*DEV:BRTS BN$ (without KE8/E EAE option)
or
*DEV:BRTS.BN,DEV:EAEOVR.BN$
(PDP-8/E, PDP-8M or
PDP-8F with KE-8E EAE)
```

and insert the following:

4. For the run-time system:

```
.PAL BRTS/W
.LOAD BRTS
.SAVE SYS:BRTS 0-6777;7605
.SAVE SYS:BASIC.AF 3400-4577;7605
.SAVE SYS:BASIC.SF 12000-13177;7605
.SAVE SYS:BASIC.FF 13400-14577;7605
```

Append the following at the end of the page

**Making SAVE Images from Binary Files:**

To create SAVE images of each of the OS/8 BASIC binaries, use the following BUILD procedure for OS/8 BASIC EAE.



6-119 (cont.)

## 1. For the editor:

```
.R PAL8
*DEV:BASIC.BN<DEV:BASIC.PA
.R ABSLDR
*DEV:BASIC.BN$
.SAVE SYS:BASIC;3211
```

## 2. For the compiler:

```
.R PAL8
*DEV:BCOMP.BN<DEV:BCOMP.PA
.R ABSLDR
*DEV:BCOMP.BN$
.SAVE SYS:BCOMP;7000
```

## 3. For the loader:

```
.R PAL8
*DEV:BLOAD.BN<DEV:BLOAD.PA
.R ABSLDR
*DEV:BLOAD.BN$
.SAVE SYS:BLOAD;7605
```

## 4. For the run-time system:

```
.R PAL8
*DEV:EARBRTS.BN<TTY:.,SYS:BRTS.PA/W
  (pause)
EAE=1
↑Z
  (pause)
↑Z
.R ABSLDR
*DEV:EABRTS.BN$
.SAVE SYS:BRTS 0-6777;7605
.SAVE SYS:BASIC.AF 3400-4577;7605
.SAVE SYS:BASIC.SF 12000-13177;7605
.SAVE SYS:BASIC.FF 13400-14577;7605
```

6-120

At the top of the page on lines 1, 2, 3, and 4, append the following:

;7605

6-120 (cont.)

In the middle of the page under the column entitled "Name", delete all the .03 extensions and replace them with the following extension:

.PA

In the middle of the page, following "The OS/8 BASIC sources are named as follows": delete "NAME.MM" and the sentence "where MM represents the version number."

At the bottom half of the page change the .03 extensions on the input files to .PA.

6-121

At the top of the page, delete:

To assemble for PDP-12, PDP-8, PDP-8/I or PDP-8/L, or PDP-8E without EAE, create a source file named NOEAE.PA with EDIT that works as follows:

```
EAE=0
PAUSE
```

Then

```
.R PAL8
*DEV:BRTS.BN<DEV:NOEAE,DEV:BRTS.03/K
```

To assemble the run-time system overlay for PDP-8E, PDP-8F or PDP-8/M with KE-8/E EAE option, prepare a file called EAE.PA that looks as follows:

```
EAE=1
PDP8E=1
PAUSE
```

Then:

```
.R PAL8
*DEV:EAEOVR.BN<DEV:EAE,DEV:BRTS.03/K
```

and insert the following:

To assemble for PDP-12, PDP-8, PDP-8/I or PDP-8/L, or PDP-8E without EAE, type the following command:

```
.R PAL8
*DEV:BRTS.BN DEV:BRTS.PA/W
```

**Page**

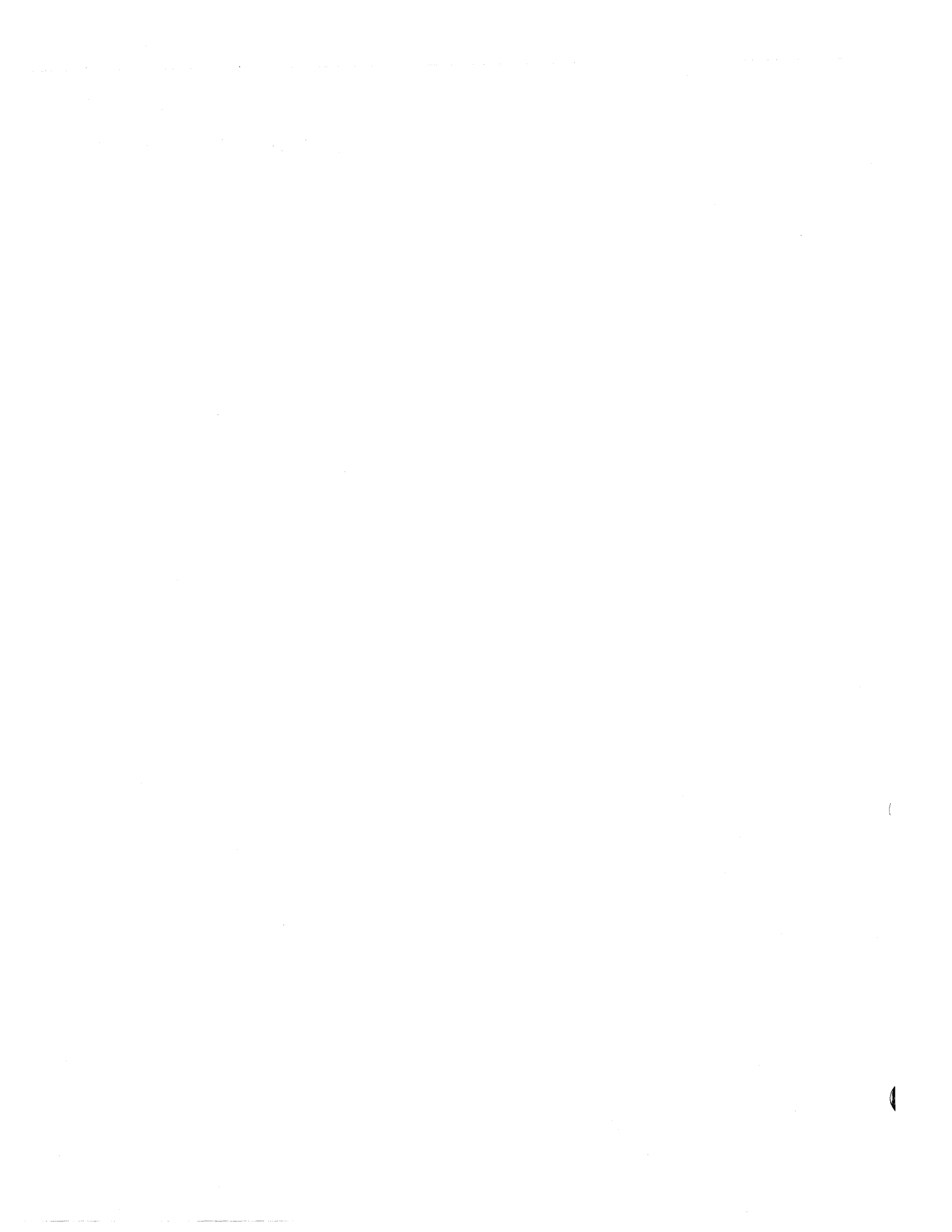
**Addition/Correction**

- 6-121 (cont.) In the example command lines for PAL8, change the extensions “.03” to “.PA”.
- 6-126 In the list of addresses, change “01566/\*\*\*\* 3541” to “01566/\*\*\*\* 3542” and change “01567/\*\*\*\* 3521” to “01567/\*\*\*\* 3522”.
- 6-127 In command line 11 (top of page), change “ADH(N)” to “ADC(N)”.

## CHAPTER 7

### ADDITIONS AND CHANGES

Page	Addition/Correction
7-1	After “Calling and Using the OS/8 FORTRAN Compiler”, insert the following:  Before calling the FORTRAN compiler, make sure that LIB8.RL is on the system device.
7-13	In the paragraph “Zero raised to a power . . .” change the sentence “A negative number . . .” to read as follows:  A negative number raised to a floating point power causes an error message and uses the absolute value; calculation continues.
7-18	Change the example of an implied DO LOOP from “WRITE (1,100) I, (A(J,I)J=1,3)” to “WRITE (1,100)I,(A(J,I),J=1,3)”.
7-44	In the section on “DECTAPE I/O ROUTINES”, add a note that the RTAPE and WTAPE are for TC08 DECTapes only.



## CHAPTER 8

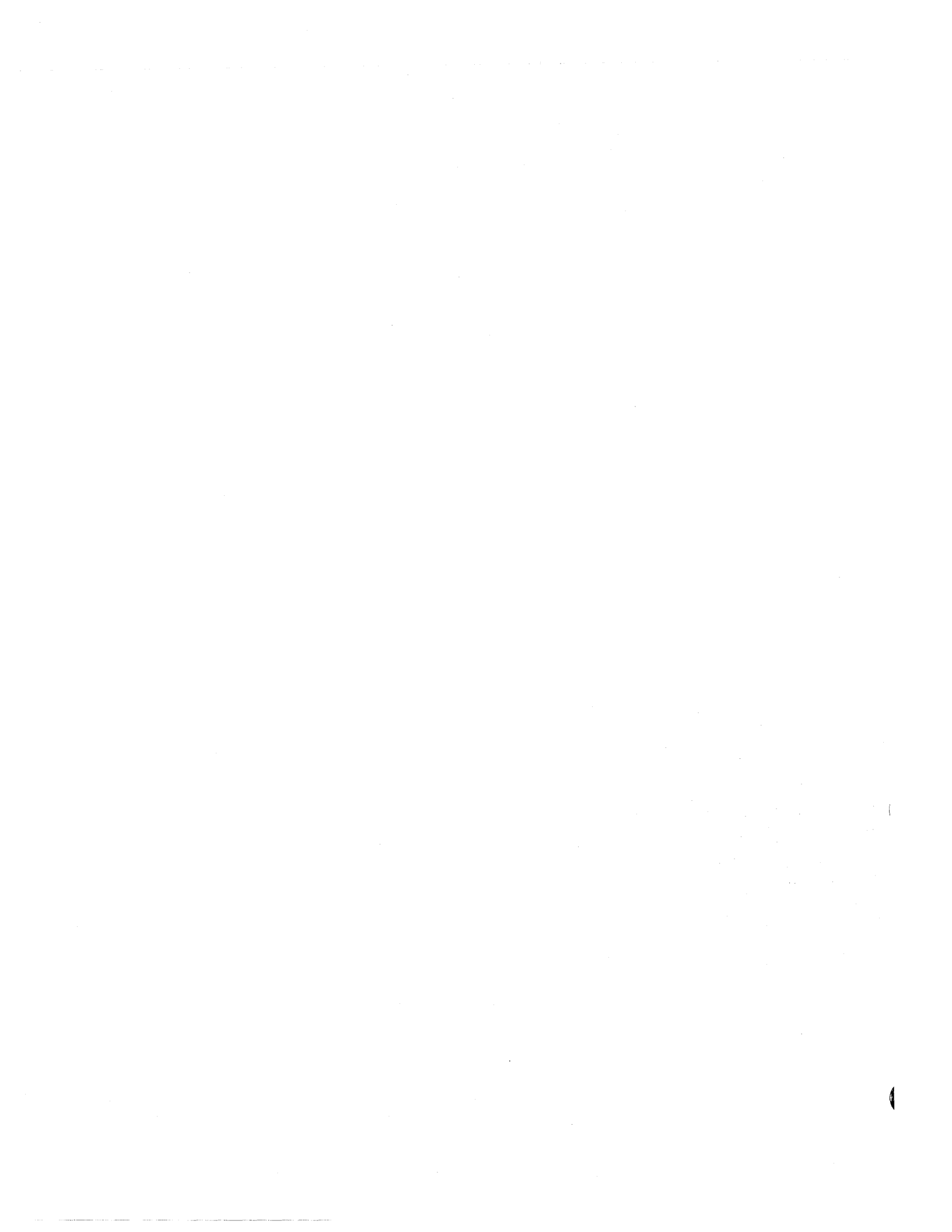
# ADDITIONS AND CHANGES

Page	Addition/Correction
8-10	In the paragraph after the example command line, change “(terminated by a carriage return)” to “(terminated by a carriage return or altmode)”.
8-29	Add the following to Table 8-6:  EX The symbol is referenced but not defined.  ME Multiple Entry. The symbol is multiply defined.  MS Multiple Section. A section is multiply defined.  * The symbol is referenced illegally. Generally this symbol is an overlay and is either referenced as data from another overlay (only CALL references are allowed) or called from the same or a higher-number overlay level, violating the overlay rules defined on page 8-22.
8-37	Add the following restriction to the first paragraph (concerning error traceback):  When a statement is reached through any form of GOTO, the line number for error traceback is not reset. Thus, an error in such a line will give the number of the last executed line in the error traceback.
8-62	In the section “NCHANL =”, change “(8 images)” to “(FUNCTN=8)”.
8-66	In the example FORTRAN IV Coding Form, change “TYPE 105, I” to “WRITE (4,105) I”.
8-80	In the second paragraph, change “An integer variable which . . .” to “A variable that . . .”.

- 8-81 Change the first two lines of the section "Computed GO TO Statement" to read:
- Form GO TO(n1,n2,.. . .,nK),e
- A comma must follow the right parenthesis.
- 8-83 Add the following information to the section entitled "Arithmetic IF Statement":
- Logical expressions may be used in an arithmetic IF statement. In such a case, the logical expression is first converted to an integer.
- 8-88 At the top of the page in the section on PAUSE STATEMENT, under Effect, change the second sentence as follows:
- Execution is suspended until the user types a character on the console.
- 8-91 Change "FORMAT (F7.2,3(I2,2(I3,E9.3)I7))" to  
"FORMAT (F7.2,3(I2,2(I3,E9.3),I7))"
- 8-98 In the last example on the page, note that the first character output after the colon is a "space".
- 8-100 Lower third of page, change "Where a is an integer constant or variable name that is . . ." to "Where a is an integer constant or integer variable..".
- 8-106 In the first full paragraph, which begins "A direct access WRITE statement . . .", add the following sentence before the words "For example:": "This means minimum record size is one block."
- In the first example, change "WRITE (8) X" to "WRITE (6) X".
- 8-107 Change all references (under FORTRAN) to the maximum size of ARRAYS from "4096" to "4095".
- 8-112 Under the section entitled "THE DATA STATEMENT", change "DATA var list1/val list1/  
var list2/val list2/, . . ." to "DATA var list1/val list1/, var list2/val list2/, . . .".

- 8-112 (cont.) Change the second paragraph from the bottom to read:  
  
Elements in the variable list may be either single subscripted or unsubscripted variables, or the name of an entire array.
- 8-113 Change example line "DATA A(1),A(2),A(3),/3\*0./" to  
"DATA A(1),A(2),A(3)/3\*0./"
- 8-125 The middle paragraph, which begins "PDP-12 users . . ." should be changed to begin "PDP-8 and PDP-12 users . . .".
- 8-127 In the example at the top of the page, after "LVL OVLY LENGTH", change "0 00 10270" to "0 00 10371".
- 8-129 After the second sentence, insert the following:  
  
A minimum of 12K of memory is required for the plotting routines.
- 8-138 In the description for "L" under the subject CALL PSCALE (A,L,N,I), change "greater than or equal to 1" to:  
  
. . . greater than or equal to 2.





## APPENDICES E-H

### ADDITIONS AND CHANGES

Page	Addition/Correction
E-10	<p>Insert the following message after the CORE? message:</p> <p>CX BCOMP Incompatible file extensions were used in BASIC CHAIN statement.</p>
E-31	<p>After the error message "SL" add the following:</p> <p>?SLOTS BUILD More than 8 groups of non-system handlers have been inserted.</p>
F-1	<p>Add the following to the table:</p> <p>.CM Indirect file used by CCL.</p>
G-4	<p>Insert the following sentence at the end of the paragraph in the section entitled "MAGNETIC TAPE".</p> <p>Normal use of magnetic tape by OS/8 V3 involves reading and writing records of 128 words.</p> <p>At the end of page, insert the following information:</p> <p>NULL HANDLER</p> <p>The NULL handler is included in the source of RF, the nonsystem RF08 handler, only because there was extra room there. It has nothing to do with RF08's. These handlers share space since there is a limitation on the number of handler slots available in OS/8. NULL is very useful when debugging a program with voluminous output which you do not need. On output, NULL ignores data sent to it. On input, NULL returns an immediate end-of-file.</p>
H-1	<p>In the table under program, change "POTP" to "FOTP".</p>

H-2

Add to the table:

HANDLERS Run RESORC with the /E option.

BCOMP /V to .R BCOMP

BLOAD /V to .R BCOMP

BRTS /V to .R BCOMP

# APPENDIX I

## RXCOPY PROGRAM

The RXCOPY program is used to copy or transfer the entire contents and system head of one RX floppy disk to another RX floppy disk. This program can be used only with RX permanent device names or a user defined name that has been assigned to an RX device. Specification of filenames in the I/O specification line results in an error message and, therefore, is not permitted.

The following command and I/O specification line will load and run RXCOPY under OS/8:

```
_R RXCOPY
*output dev:<input dev:/options
```

Example:

```
_R RXCOPY
*RXA1:<SYS:
```

When RXCOPY is loaded and the I/O specification line is entered at the keyboard, the input device is copied to the output device on a sector by sector basis. When the operation is complete, the Monitor dot is printed on the screen and the specified output device becomes an exact duplicate of the input device.

Table I-1 lists the options available for use with the RXCOPY program. These options modify the RXCOPY operation.

**Table I-1 RXCOPY Options**

Option	Meaning
/P	Pause and wait for user response before and after execution of RXCOPY program.
/N	Copy the contents of one device to another but don't check them for identical contents unless otherwise specified.
/M	Check both devices for identical contents and list the tracks and sectors that do not match but do not perform a transfer unless otherwise specified.
/R	Read every block on the specified device and list the bad tracks and sectors but do not perform a transfer unless otherwise specified.
/V	Print the current version number of the RXCOPY program.

If no options are specified, RXCOPY assumes both the /N and /M options.

If an error occurs during the execution of RXCOPY, the current job is aborted and control returns to the Monitor.

Table I-2 lists the RXCOPY error messages and their meanings.

**Table I-2 RXCOPY Error Messages**

<b>Message</b>	<b>Meaning</b>
NO INPUT DEVICE	No input device is specified.
CAN'T LOAD INPUT DEVICE	The name of the input device specified in the command line is not a permanent device name.
CAN'T LOAD OUTPUT DEVICE	The name of the output device specified in the command line is not a permanent device name.
COMPARE ERROR	When using the /M option all the areas that do not match are printed as COMPARE ERRORS. Since this is a non-fatal error, the RXCOPY operation continues.
INPUT DEVICE READ ERROR	Bad input, bad tracks or sectors. Since this is a non-fatal error, the RXCOPY operation continues.
OUTPUT DEVICE READ ERROR	Bad data on output device, tracks and sectors bad. Since this is a non-fatal error, the RXCOPY operation continues.
OUTPUT DEVICE WRITE ERROR	Fatal output error. Since this is a non-fatal error, the RXCOPY operation continues.

# APPENDIX J

## SET PROGRAM

The SET program permits the operating characteristics of OS/8 to be modified, according to the attributes that are specified. The SET program is used to make frequently-required standard changes to system programs, especially I/O handlers. These changes are identified by specifying certain attributes in the SET command string which has the following format:

```

_R SET
#SET device [NO] attribute [argument]
  
```

where:

- SET is the operation to be performed.
- device indicates the handler of the device you want modified.
- [NO] indicates that the attribute specified does not apply. [NO] cannot be used with every attribute.
- attribute is the characteristic to be modified. (See Table J-1.)
- [argument] is an optional parameter that the user must supply for certain SET commands.

SET error messages are listed in Table J-2.

**Table J-1 SET Command Attributes**

TTY	Card Reader	Mag Tape	SYS	LPT	Any Device
ARROW CODE n COL n ECHO ESCAPE FILL FLAG HEIGHT m LC PAGE PAUSE n SCOPE TAB WIDTH n	CODE n	PARITY x FILES	INIT xxxxx OS8 OS78	LA78 LA8A LC LV8E WIDTH n	FILES DVCODE LOCATION n=m READONLY VERSION x BLOCK b

**Table J-2 SET Error Messages**

<p>? SYNTAX ERROR Incorrect format used in SET command or NO specified when not allowed.</p> <p>? UNKNOWN ATTRIBUTE FOR DEVICE dev An illegal attribute was specified for the given device.</p> <p>? CAN'T – DEVICE IS RESIDENT No modifications are allowed to the system handler.</p> <p>? CAN'T – OBSOLETE HANDLER The handler has an old version number.</p> <p>? CAN'T – UNKNOWN VERSION OF THIS HANDLER The version of the handler is not one recognized, possibly because it is a newer version.</p> <p>? ILLEGAL WIDTH A width of 0 or a width too large was specified? or for the TTY, a width of 128 or one not a multiple of 8 was specified.</p> <p>? NUMBER TOO BIG The number specified was out of range.</p> <p>? CAN'T – DEVICE DOESN'T EXIST A nonexistent device was referenced.</p> <p>? I/O ERROR ON SYS:</p>
---

## **J.1 TERMINAL ATTRIBUTES**

### **J.1.1 Arrow**

Specifying this attribute causes each control character typed by the KL8E handler to be printed in the form:

↑x

where:

↑ indicates that a control character is being typed, and

x specifies which control character is being typed (100 + code for control character).

Format:

`_SET TTY [NO] ARROW`

Example:

`_SET TTY ARROW`

If a CTRL/E character is typed, by the KL8E handler, an ↑E is printed on the terminal. Note that ARROW is the default.

Using this attribute with the NO modifier causes each control character typed by the KL8E handler to print with no modification.

Example:

```
._SET TTY NO ARROW
```

Now if a CTRL/E character is typed by the KL8E handler, it will actually send a CTRL/E (ASCII code 5) to the terminal. The result is that no character is printed.

#### NOTE

On some terminals, the arrow (↑) is replaced by the circumflex (ˆ).

#### J.1.2 CODE n

where n is an octal number in the range

$$1 \leq n \leq 77$$

This command changes the internal IOT code for keyboard to n. The internal device code for the teleprinter is set to n+1. For example, if you have a VT05 hooked to your system with device codes of 40 and 41, you would say SET TTY CODE 40. The NO restriction is not permitted.

Example:

```
._SET TTY CODE = 3
```

#### J.1.3 COLumn n

Specifying this attribute changes the default number of columns used to print the directory (using the DIRECT command) to the decimal number you specify as n. The initial default number of columns is equal to one and the decimal number you specify should be in the range of 1-7.

Format:

```
._SET TTY COL n
```

Example:

```
._SET TTY COL 3
```

Note that specification of this attribute does not actually change the behavior of the KL8E handler. Also, the NO modifier is not permitted to be used with this attribute.

#### J.1.4 ECHO

Specifying this attribute causes all TTY characters typed at the keyboard as input or received on the terminal as output to be printed. Specification of this attribute affects the KL8E handler only and not character echoing by the keyboard monitor.

Format:

```
._SET TTY [NO] ECHO
```

Example:

```
._SET TTY ECHO
```



If it is desired that no character echoing take place, use the NO modifier in the command line. By specifying the NO modifier, all TTY characters on input or output are not printed and do not appear on the terminal screen.

Example:

```
._SET TTY NO ECHO
```

### J.1.5 ESCape

Specifying this attribute causes the escape character (ASCII code 33) to print as a control character (see also ARROW attribute).

Format:

```
._SET TTY [NO] ESC
```

Example:

```
._SET TTY ESC
```

Specifying the NO modifier, in the command line, causes escape to print as a dollar sign (\$).

Example:

```
._SET TTY NO ESC
```

Escapes can also be affected by the ARROW attribute. Specifying NO ARROW causes escapes to be sent to the terminal with no modification. This is useful for sending escape sequences to CRT terminal.

### J.1.6 FILL

Specifying this attribute causes two fill characters to be typed following a tab. This attribute should only be used in conjunction with the TAB attribute.

Format:

```
._SET TTY [NO] FILL
```

To remove these fill characters, use the NO modifier in the command line.

Example:

```
._SET TTY NO FILL
```

### J.1.7 FLAG

Specifying this attribute causes the handler to flag lower case characters on output by printing them as upper case characters preceded by a quote.

Format:

```
._SET TTY [NO] FLAG
```

Example:

```
._SET TTY FLAG
```

If it is desired to remove the quote preceding upper case characters, use the **NO** modifier in the command line.

Example:

```
._SET TTY NO FLAG
```

#### **J.1.8 HEIGHT m**

Specifying this attribute changes the number of lines that are printed on the terminal between pauses. Twenty-four lines is the default value of m.

Format:

```
._SET TTY HEIGHT m
```

Example:

```
._SET TTY HEIGHT 12
```

This attribute has no meaning unless the **PAUSE** attribute is also specified.

#### **J.1.9 LC**

Specifying this attribute causes the **KL8E** handler to accept lower case characters on input.

Format:

```
._SET TTY [NO] LC
```

Example:

```
._SET TTY LC
```

Specifying the **NO** modifier in the command line causes lower case characters on input to be converted to upper case.

Example:

```
._SET TTY NO LC
```

#### **J.1.10 PAGE**

Specifying this attribute adds both the **CTRL/S** and **CTRL/Q** features to the keyboard monitor.

Format and example:

```
._SET TTY PAGE
```

When used with the **NO** modifier, this attribute removes the **CTRL/S** and **CTRL/Q** features.

Example:

```
._SET TTY NO PAGE
```

For more detailed information on the control characters, see the *OS/8 Handbook*.

### **J.1.11 PAUSE n**

Specifying this attribute sets the pause time between terminal output frames to the decimal number you specify as n. The exact time depends on the cycle time of your machine.

Format:

```
._SET TTY PAUSE n
```

Example:

```
._SET TTY PAUSE 5
```

If it is desired that no pause takes place, specify the NO modifier in the command line or specify zero as n.

Example:

```
._SET TTY NO PAUSE
```

or

```
._SET TTY PAUSE 0
```

### **J.1.12 SCOPE**

Specifying this attribute causes characters that are erased via the rubout or delete key by the KL8E handler, keyboard monitor, and command decoder to physically disappear from the CRT screen. This attribute should not be specified if you do not have a CRT.

Format and Example:

```
._SET TTY SCOPE
```

### **J.1.13 TAB**

Specifying this attribute causes the handler to print real tabs (ASCII code 211). This can only be used if your handler has the TAB feature.

Format:

```
._SET TTY [NO] TAB
```

Example:

```
._SET TTY TAB
```

If your handler does not have the TAB feature, use the NO modifier in the command line. By specifying the NO modifier, all tabs are simulated as spaces.

Example:

```
._SET TTY NO TAB
```

### **J.1.14 WIDTH n**

Specifying this attribute changes the width of the terminal to the decimal number you specify as n. The decimal number you specify should be a multiple of eight and in the range of 1-255. However, n must not be 128. If your TTY handler does not have the tab feature, the width you specified in the command line may not be your final result. The NO modifier is not permitted to be used with this attribute. Placing an equal sign (=) between the attribute and the decimal number you specify is optional.

Format:

`._SET TTY WIDTH n`

Example:

`._SET TTY WIDTH 64`

## **J.2 CARD READER ATTRIBUTES**

### **J.2.1 CODE n**

Specifying this attribute causes the card reader to use the card code you specify.

Format:

`._SET CDR CODE n`

where:

n is a decimal number having a value of either 026 or 029.

Example:

`._SET CDR CODE 026`

Note that the NO modifier is not permitted with this attribute.

## **J.3 MAGNETIC TAPE ATTRIBUTES**

### **J.3.1 PARITY x**

Specifying this attribute causes the parity check to be either even or odd.

Format:

`._SET MTxx: PARITY x`

Example:

`._SET MTA0: PARITY EVEN`

Note that the NO modifier is not permitted with this attribute.

### **J.3.2 FILES**

Specifying this attribute causes the handler not to issue an automatic rewind when referencing block 0.

Format:

`._SET MTxx [NO] FILES`

Example:

`._SET MTA1: FILE`

If it is desired that the automatic rewind take place when block 0 is referenced, use the NO modifier in the command line.

Example:

```
._SET MTA0: NO FILES
```

## J.4 SYSTEM ATTRIBUTES

### J.4.1 INITIAL xxxxx

Specifying this attribute causes the system device to execute the command you specify as when the system is bootstrapped. This command can contain a maximum of five characters excluding a RETURN key.

Format:

```
._SET device INIT xxxxx
```

Example:

```
._SET SYS INIT HELP
```

If xxxxx is not specified, @INIT is assumed, and causes the system to execute the command in the file INIT.CM when bootstrapped. Note that the INIT.CM file must be created prior to bootstrapping.

If it is desired that no special commands be executed at system bootstrap, use the NO modifier in the command line. By specifying the NO modifier, the system prints the monitor dot immediately after bootstrapping.

Example:

```
._SET SYS NO INIT
```

If an initial command is specified and the system is bootstrapped, anything previously in memory is destroyed.

### J.4.2 OS8

Specifying this attribute modifies the system handler to be OS/8.

Format and example:

```
._SET SYS OS8
```

Note that the NO modifier is not permitted with this attribute.

### J.4.3 OS78

Specifying this attribute modifies the system handler to be OS/78.

Format and example:

```
._SET SYS OS78
```

Note that the NO modifier is not permitted with this attribute.

## **J.5 LINE PRINTER ATTRIBUTES**

### **J.5.1 LA78**

Specifying this attribute modifies the LPSV handler to handle an LA78 line printer.

Format and example:

```
._SET LPT LA78
```

### **J.5.2 LA8A**

Specifying this attribute restores the LPSV handler to its original state.

Format and example:

```
._SET LPT LA8A
```

### **J.5.3 LC**

Specifying this attribute causes the handler to print lower case characters. This attribute must only be used with line printers that have the physical ability to print lower case characters.

Format:

```
._SET LPT: [NO] LC
```

Example:

```
._SET LPT: LC
```

Specifying the NO modifier in the command line converts lower case characters to upper case prior to printing.

Example:

```
._SET LPT: NO LC
```

### **J.5.4 LV8E**

Specifying this attribute modifies the LPSV handler to work on an LV8E line printer.

Format:

```
._SET LPT: [NO] LV8E
```

Example:

```
._SET LPT: LV8E
```

By specifying the NO modifier in the command line, this will work on an LP08 and LS8E line printer.

Example:

```
._SET LPT: NO LV8E
```

### **J.5.5 WIDTH n**

Specifying this attribute sets the width of the line printer to the decimal number you specify as n.

Format:

```
._SET LPT WIDTH n
```

where:

n is a decimal number in the range of 1-256

Example:

```
._SET LPT WIDTH 80
```

Note that the NO modifier is not permitted with this attribute.

## **J.6 ANY DEVICE ATTRIBUTES**

### **J.6.1 FILES**

Specifying this attribute causes the handler to handle a file-structured device.

Format:

```
._SET device [NO] FILES
```

Example:

```
._SET MTA0: FILES
```

If it is desired that the handler handle non-file structured devices, use the NO modifier in the command line.

Example:

```
._SET DTA1: NO FILES
```

#### **NOTE**

This attribute remains in effect until the next time you bootstrap, at which time the original status is restored.

### **J.6.2 DVCode nn**

Specifying this attribute sets the IOT device code used by the handler to the decimal number you specify as nn. This number should be in the range of 30-77.

Format:

```
._SET device DVC nn
```

Example:

```
._SET RXA0 DVC 64
```

This example could be used if your diskettes were hooked up to the non-standard device code of 64. Note that the NO modifier is not permitted with this attribute.

### J.6.3 LOCation n=m or LOCation n

Specifying the first argument changes the contents of the location in the handler you specify as n to contain the value you specify as m. Both n and m are octal numbers.

where:

n is an octal number and must be in the range of 0-177 for one-page handlers and in the range of 0-377 for two-page handlers.

m is an octal number and must be in the range of 0-7777.

Format:

```
._SET device LOC n=m
```

Example:

```
._SET LPT LOC 37-1234
```

Specifying the second argument causes the system to print the current contents of the location in the handler you specify as n. This is then followed by a slash. The user can now enter a new value in that location by typing that value followed by a carriage return. If it is desired to leave the contents of that location unchanged, the user just types a carriage return.

Format:

```
._SET device LOC n
```

Example:

```
._SET PTP LOC 144
```

### J.6.4 READOnly

Specifying this attribute causes the device specified to become a read-only device. Therefore, any output sent to this device causes an error message informing you that the output device is a read-only device.

Format:

```
._SET device [NO] READO
```

Example:

```
._SET TTY READO
```

To remove the READONLY attribute, use the NO modifier in the command line.

Example:

```
._SET TTY NO READO
```

#### NOTE

The READONLY attribute remains in effect only until the next time you bootstrap, at which time its original status is restored.



### J.6.5 **VERSION x**

Specifying this attribute changes the version number of the handler to the letter you specify as x.

Format:

```
._SET device VERSION x
```

Example:

```
._SET TV: VERSION G
```

Note that the NO modifier is not permitted with this attribute.

### J.6.6 **BLOCK b, LOCation n=m or BLOCK b, LOC n**

Specifying the first attribute changes the contents of the location in the handler you specify as n that is located in the block you specify as b. The contents of that relative location is changed to the value you specify as m.

where:

- b is an octal number
- n is an octal number and must be in the range of 0-177 for one-page handlers and in the range of 0-377 for two-page handlers
- m is an octal number and must be in the range of 0-7777.

Format:

```
._SET device LOC n=m
```

Example:

```
._SET RKB1 LOC 10 = 2420
```

Specifying the second attribute causes the system to print the current contents of the location in the handler you specify as n that is located in the block you specify as b. This is then followed by a slash. The user can now enter a new value in that location by typing that value followed by a carriage return. If it is desired to leave the contents of that location unchanged, the user types a carriage return.

Format:

```
._SET device LOC n
```

Example:

```
._SET LPT LOC 175
```

## APPENDIX K

# FUTIL OS/8 FILE UTILITY PROGRAM

FUTIL was originally developed by Jim Crapuchettes of Menlo Computer Associates, Inc., Menlo Park, CA. It is now included within the OS/8 Extension Kit for the convenience of PDP-8 users.

### K.1 INTRODUCTION

#### K.1.1 Description

FUTIL enables a user to examine and modify the contents of mass storage devices. It is the only program currently available that can be used to patch programs containing overlays (F4/LOAD outputs). Other possible uses include examination and repair of OS/8 directories; bad block checking and correction; decimal/octal conversion of double precision numbers; output of the Core Control Block (CCB) of “.SV” files and the HEADER of “.LD” files; and the creation of special directories. Supporting these functions is signed double-precision arithmetic expression evaluation that can be used in the command syntax whenever a numeric value is needed.

FUTIL's command set is divided into two groups of commands. The first group uses single letters to direct the program in the examination and modification of single words on the device specified. The second group of commands uses command words to direct the program in the dumping, listing, modifying and searching of the device on a block-by-block basis. Also included in this group is a series of commands to direct the program in some auxiliary functions including setting and resetting switches and variables within the program, showing current FUTIL parameters.

Several examples are given in Section K.4. The first two examples, especially, are simple and well-documented to acquaint you with the features of FUTIL. You may want to look at them at this point to get a better understanding of the material that follows.

#### K.1.2 Special Characters Used in This Appendix

To help increase clarity, the characters single quote ('), double quote ("), angle brackets (< and >) and square brackets ([ and ]) have been used to help separate special items from the words around them. The single quote character is used to surround a word-type command; for example, the 'FORMAT' option 'SET's up the format in which output is to be done. The double quote is used to surround an item whose actual name is being used; for example, the "RETURN" key is the key on the Teletype that has that word printed on it. The angle brackets are used to surround the name of a type of item (a syntactical type); for example, "<n>" means that a NUMERIC ITEM is to be used. The square brackets are used to surround optional items; for example, "w[ord]" would indicate that the characters "ord" may be supplied optionally.

#### K.1.3 Special Characters Used in FUTIL

Several characters, when keyed, cause immediate action from the program. Typing either CTRL/P (which prints "^P") or CTRL/C (which prints "^C") will immediately cause the program to stop whatever it is doing. CTRL/P then causes the program to go back to command input mode and wait for you, while CTRL/C calls the OS/8 Monitor (as it does with most system programs). CTRL/S and CTRL/Q control program execution (including all I/O). Typing CTRL/S at any time will cause the program to pause and wait for either CTRL/C, CTRL/P or CTRL/Q. Typing CTRL/Q will then allow program execution to resume. Any other characters entered at this point will be simply ignored. If a CTRL/Q is typed by itself at any time, it is simply ignored.

#### NOTE

CTRL/S and CTRL/Q are active at all times, not just during console output. The result is that both input from the console and program execution with no console interaction (such as 'SCAN', 'WORD' and 'STRING' command execution) can and will be paused and re-started with these keys.

During console terminal input, three other keys can be used to help with editing the input string of characters. These keys are RUBOUT, CTRL/U (which prints "`^U`") and CTRL/R (which prints "`^R`"). The action of RUBOUT and CTRL/U is exactly the same as it is for the OS/8 Monitor and Command Decoder (including usage of "scope mode" operation to change the action of the RUBOUT key from echoing the rubbed out characters between backslashes to erasing the characters from the screen). The action of CTRL/R is the same as that of the LINE-FEED key for the Monitor and Command Decoder.

For those users with upper-lower case terminals, the program translates all lower case characters received from the keyboard to upper case. The characters are echoed and handled internally as upper case characters. While this makes use easier, it does not allow any lower-case characters to be input directly. In those cases where lower-case codes are needed in the modification of a file, either use the codes directly or use a text editor. Note that this translation occurs only on input. Lower case characters in a file will be printed to the best ability of the output device. This may produce incorrect results on upper-lower case line printers.

All of the commands are taken in context, that is, many of the characters which are used in the single character command set will not be considered to be commands if they are included in a line which begins with a command word or if they are embedded within expressions.

The carriage-return ("`RETURN`") always starts command execution, and is the terminator for all word-type command lines.

#### K.1.4 Running FUTIL

FUTIL is run using the OS/8 Monitor command "`R FUTIL`" (or "`RU dev:FUTIL`").

When started, FUTIL is set up to access the system device, the 'ERROR' message output mode is set to 'LONG', the access 'MODE' is set to 'NORMAL' and no file is known. To access some other device, give the command 'SET DEVICE dev'. To set the 'ERROR' mode to 'SHORT', give the command 'SET ERROR SHORT'. To use some other access mode, give a 'SET MODE <mode>' command with a <mode> of 'LOAD', 'OFFSET' or 'SAVE'. When in 'OFFSET' mode, the 'OFFSET' to be used can be specified by the command 'SET OFFSET nnnn'. Lastly, a file lookup can be performed by giving a 'FILE' command (with three default extensions).

#### K.1.5 Access Method

The program accesses the OS/8 device one OS/8 block (256 words) at a time. For every location specified, the real block and word are determined and compared with the current block in memory. If the desired block and current block are not the same, the <something-changed> flag is checked to see if anything has been changed in the current block. If nothing has been changed, the new block is read in. If something has been changed, the current (modified) block is first written out and then the new block is read in. This action happens correctly even when the access mode is changed because it is done at the level of the OS/8 block number right before calling the current 'DEVICE' handler. The status of the <something-changed> flag can be determined by simply 'SHOW'ing 'ABS', 'REL' or 'ODT' locations. If the flag is set, the word "`MOD`" will be output following location information.

The contents of the OS/8 device are therefore not changed unless the block in which changes are made is written out either implicitly, as described above, or explicitly, using the 'WRITE' command (which is discussed near the end of the section on word-type commands). The result is that typing CTRL/C before writing out the current block (assuming it has been modified) will return to the Monitor without actually modifying the contents of the device

itself. Note, also, that only one implicit write attempt is ever made by the program. Should an error occur when the write is attempted (for example, write-locked device), an explicit 'WRITE' command must be given to actually write out the block.

If the words within some blocks are changed accidentally, the <something-changed> flag can be reset by using the 'SET' command to reset the 'DEVICE' (described further along in this writeup) to the same device currently being used. This will reset the <something-changed> flag, the current block in memory, and the file start block and core-control-block/header-block (if they had been set by a 'FILE' command). The resetting of the current block in memory will cause the next access to the device to read in the block desired. The resetting of the file information will require a new 'FILE' command to be given to set it back up. If you can't remember what is the current setting of the 'DEVICE', use 'SHOW DEVICE' first and then 'SET' it the same.

Files stored on an OS/8 mass-storage device generally fall into one of four categories. The program has four corresponding modes for accessing the device. The current 'MODE' of the program can be set by the 'SET' command or by chaining (as described previously) and examined by the 'SHOW' command (to be described later).

The four categories and their corresponding modes are:

1. General (binary, ASCII and data) files – 'NORMAL' mode
2. Core image (save) files – 'SAVE' mode
3. FORTRAN IV load modules – 'LOAD' mode
4. System overlays – 'OFFSET' mode

The actual operation of the program for each of these modes is as follows:

'NORMAL'	The high order 7 bits of the 15 bit address are added to the current block number to get the actual block number. The low 8 bits of the 15 bit address are used to specify the desired word within that block.
'SAVE'	The file to be examined must be set up by a 'FILE' command. "Block" numbers are used to specify an overlay number (future MACREL/LINK support) and must be exactly zero ("0") for files without overlays (generated by the monitor "SAVE" command). The core segment data (pages and fields) from the file's CCB (core-control-block) is used to determine where on the device the desired word is to be found. This is done by first determining the correct block from the file's CCB and then using the low 8 bits of the address to specify the desired word within that block. Specifying a nonexistent address or overlay for one of the single-character (ODT) commands will cause an error. Specifying a nonexistent address or overlay for any of the word-type commands will cause the program to ignore the address and access no data.
'LOAD'	The file to be examined must be set up by a 'FILE' command. Block number specifications are actually taken as FORTRAN IV overlay specifications and must be contained within the file. The information from the OIT (overlay-information-table) in the header block of the file is used to determine where on the device the desired word is to be found. Nonexistent addresses are handled the same way as for 'SAVE' mode.

#### NOTE

Because the "block" part of the location specification changes definition depending on the mode in use, it is recommended that the first operation following a switch to 'SAVE' or 'LOAD' mode explicitly specify a "block" part of 0. Otherwise a previously specified "block" part will be taken to mean a non-existent overlay number, causing an error.

## 'OFFSET'

The 12-bit 'OFFSET' (which is set by the 'SET' command and examined by the 'SHOW' command) is subtracted from the low order 12 bits of the address and then the same arithmetic as with the 'NORMAL' mode is used. This mode is used mostly with system overlays whose start block number and actual loading address is known. By setting the 'OFFSET' to the loading address (which can only be a 12 bit number), the 12 bit "actual" addresses of the overlay can be used.

The 'SAVE' and 'LOAD' modes are mentioned together throughout this appendix as MAPPED modes because their method of address translation uses a descriptor block from the file of interest to control access to the file in a non-contiguous manner.

### NOTE

For all access modes, the OS/8 "actual" block number for the block to be read is stored (for display) in the computer MQ register (if present). The value is stored before checking if the current block needs to be written. It is particularly useful for following the progress of the 'SCAN' command.

#### K.1.6 Referencing Words on the Device

The words on the OS/8 device are referenced by their <location> (often abbreviated as <1>). This <location> consists of an optional <block> or <overlay> number (which must be followed by a "." if present), and an <address> or <displacement>. The <block>/<overlay> number is a 12-bit number which must be in the range 0 thru 7776 (octal), or 4094 (decimal). Block number 7777 (or 4095, decimal) does not exist under OS/8, and the program will ignore this number. The <overlay> number is further limited to the number of overlays at a given address. Whenever the <block>/<overlay> part of the <location> is not used, the program will use the last specified value. The <address>/<displacement> is a 15 bit number (5 octal digits), but leading 0's need not be specified. Thus, the forms and their corresponding examples are as follows:

Form	Example
<block>.<displacement>	1201.37524
<overlay>.<address>	3.57633
<address>	15721
<displacement>	223

### CAUTION

Neither this program nor the OS/8 device handlers generally include checking for legal block numbers! It is simply assumed that all accesses to the device will be done after checking with the directory for legal file start blocks and lengths, which is the normal mode of operation under OS/8. This can have very interesting results with this program; for example, the RK8/E handler, given a block number greater than 6257 (octal) on device RKA0, will simply continue on into device RKB0.

For the rest of this document, unless otherwise stated, block will mean <block> or <overlay> and address will mean <address> or <displacement>, depending on usage. Therefore the definition will be:

[block.] address=<location> = <1>

Since these location references are numeric input, all of the characteristics described next can also be used when specifying locations.

### K.1.7 Numeric Items (Or Numbers)

Two “switches” are used by the program to allow the input of either octal, decimal or mixed numeric input where ever numeric input is used. Each new command line always resets the input mode to octal. The character CTRL/D (printed as “^D”) switches the input mode for any following input to decimal. The character CTRL/K (printed as “^K”) switches the input mode back to octal. These two switches may be located anywhere in numeric input.

For example, when inputting a string of numbers, the input would be alternately decimal and octal if it were

```
^D100,^K100,^D200,^K200,^D300,^K300
```

Two other characters, the double quote (") and apostrophe ('), may be used for numeric input. The double quote functions the same way in this program as it does in PAL8 in that the 8-bit ASCII value of the following character is used as a number. As with all character input, the special characters described earlier cannot be used. The apostrophe functions in a way similar to the way that the “TEXT” pseudo-op operates in PAL8 in that the following two characters are masked to 6-bits each and packed into a 12-bit word. There must always be exactly two characters following the single quote. If it is desired to pack one half of the word with a 6-bit 00, use the character “@”. For example, a string equivalent to the file-name “PIP.SV” would be represented by the string

```
'PI,'P@,0,'SV
```

Expressions may also be used for numeric input when enclosed in parentheses. The parentheses pair “( and )” must surround the expression. When this is so, all the options of the ‘EVAL’ command are available for numeric input. For example, the contents of the switch register can be used for a number by the expression “(S)”, or the current block number +5 could be used by the expression “(B+5)”. See the discussion of the ‘EVAL’ command for the other options available.

#### NOTE

The opening and closing parentheses must completely surround the expression. Neither digits nor the switch characters may be outside of the parentheses or an error will result. This is required because many of the non-alphabetic characters have multiple meanings (commands or operators) so the use of the parentheses pair “(. . .)” provides the necessary context to remove ambiguity.

### K.1.8 Errors (And Error Messages)

Whenever the program recognizes an error of some type, it outputs an error message to inform you what went wrong. The message tells both what went wrong and where in the command line the error was made. Depending on the setting of the ‘ERROR’ mode switch, either ‘SHORT’ or ‘LONG’ messages are output.

The error messages have the forms:

```
“?<ee> at <cc> <error message>” — ‘LONG’
```

or

```
“?<ee> at <cc>” — ‘SHORT’
```

where <ee> is the error code, <cc> is the number of the column in the command line where the program stopped scanning and <error message> is the message itself. There are currently 45 error conditions with corresponding codes and messages to assist the user of this program. The error codes and their messages can be printed out by the 'SHOW' 'ERRORS' command. The 'ERROR' mode is set by the 'SET' command.

The error messages are swapped with the USR, but not in the normal manner, allowing write locked startup with the loss of the message text (see the section on program execution for more information).

## K.2 SINGLE CHARACTER (ODT-LIKE) COMMANDS

These commands allow the examination and modification of words on an OS/8 device in the same way that ODT allows the examination and modification of the memory in the computer.

In all of the following commands where <n> – a numeric item – is specified, the operation of “closing” the location is to place the value of <n> into the word if it is open. If the current location is not open, or if <n> is not specified, no change takes place. Refer to the “Introduction to Programming” and the OS/8 Handbook section on ODT for more information if needed. Note that (as mentioned previously) “[<n>]” with the following commands means that a numeric item may be optionally supplied.

<1>/	Open and output the contents of location <1> in the current 'OUTPUT' mode.
/	Reopen the last location opened by one of these commands and output its contents in the current 'OUTPUT' mode.
[<n>]#	Close the current location, reopen it and output its contents in 'BCD' (3 digit binary-coded decimal).
[<n>]\$ (dollar sign)	Close the current location, reopen it and output its contents in 'OS/8' ASCII.
[<n>]%	Close the current location, reopen it and output its contents in 'BYTE' octal (8 bits with OS/8 packing).
[<n>]&	Close the current location, reopen it and output its contents in 'XS240' format packed ASCII.
[<n>]:	Close the current location, reopen it and output its contents in 'SIGNED' decimal.
[<n><	Close the current location, reopen it and output its contents in 'OCTAL'.
[<n>]=	Close the current location, reopen it and output its contents in 'UNSIGNED' decimal.
[<n>>	Close the current location, reopen it and output its contents in 'PDP' (symbolic).
[<n>]?	Close the current location, reopen it and output its contents in 'DIRECTORY' format [negated DECIMAL, DATE (see "@" next) and PACKED (ASCII)].
[<n>]@	Close the current location, reopen it and output its contents in 'DATE' format: dd-mmm-yy 2 digits each for the day and year and 3 alphabetic characters for the month (except for illegal month numbers, which are output as a space and 2 decimal digits).
[<n>] [	Close the current location, reopen it and output its contents in 'ASCII'.

[<n>]\	Close the current location, reopen it and output its contents in 'FPP' (symbolic).
[<n>]]	Close the current location, reopen it and output its contents in 'PACKED' ASCII.
[<n>]\$ ("ALTMODE" or "ESCAPE" key)	Close the current location, reopen it and type its contents as specified by the current 'FORMAT'.
[<n><cr>	Close the current location.
[<n>;	Close the current location and open the next sequential location. Neither address nor contents are output, but one space is echoed.

#### NOTE

The ";" command can be used to advance through addresses without outputting their value in octal when some other format is really more helpful. For example, when examining a directory, the file name and extension can be output using the "]" command (PACKED ASCII), the date can be output using the "@" command and the file length can be output using the ":" command and all of this information can be made to appear on one line by simply using the ";" command to do the incrementing between each of the output commands. The result would look something like this:

```
2.5/ 2317 ]SO ; ]UR ; ]CE ; ]PA ; @30-AUG-72 ; : - 0071
```

For the following commands, the location of the newly opened word is output before the contents are output. This location is composed of the 12-bit block number (4 octal digits), a "." for a separator, and the 15 bit address (5 octal digits). This is immediately followed by "/" to separate the contents from the address.

[<n><line feed>	Close the current location, open and output the contents of the next sequential location in the current 'OUTPUT' mode.
[<n>]!	Close the current location, open and output the contents of the previous sequential location in the current 'OUTPUT' mode.
[<n>]^(circumflex or up-arrow)	Close current location, open the location that would have been referenced if the contents were a PDP-8 memory reference instruction, and output the contents of the new location in the current 'OUTPUT' mode. Note: this command works like the stand-alone version of ODT, not like the OS/8 version. Even if bit 3 of the word (the indirect bit of a PDP-8 instruction) is a 1, this command will not do the equivalent of an indirect reference.
[<n>]_(backarrow or underline)	Close the current location, take its contents as an address, open that location and print its contents in the current 'OUTPUT' mode. This operates as an indirect address into the current field would. The field currently being examined (the high octal digit of the 5 digit location) will not be changed by this operation.



- <1>+           Open the location <1> locations forward from the current location and output its contents in the current 'OUTPUT' mode. 15 bit arithmetic is used and the block part is ignored, so this will operate across field boundaries, i.e., within a 32K area.
- <1>-           Open the location <1> locations backward from the current location and output its contents in the current 'OUTPUT' mode. Same restrictions as with the '+' command.

The "current 'OUTPUT' mode" has been mentioned several times above. The program will output the contents of a location either as a four-digit octal number, or as a four-digit octal number with two spaces and the "symbolic" representation ('PDP' or 'FPP') of the word. See the 'SET' and 'SHOW' commands as well as the following section.

### K.2.1 "Symbolic" Output Formats

The "symbolic" typeout is in approximately the format that input to an assembler would need to be in order to generate the contents of the current location. It is assumed, of course, that these contents are either a PDP-8 or an FPP-12/8A instruction, depending on the output selected. If the word to be output is not an instruction, as is the case for the second word of all 2-word instructions (EAE and FPP), the decoding will obviously be meaningless.

For PDP-8 instructions decoding into mnemonics is done for all memory reference instructions, for all legal operate instructions (including 8/E EAE instructions except for "SWAB"), for all 8/E processor, extended memory and memory parity IOTs, for teletype and high-speed paper-tape IOTs, for 8/E redundancy check option IOTs, for programmable real-time clock IOTs and for FPP IOTs. There are currently a total of 96 IOTs and space has been provided in the program for an additional 32 IOT codes and their mnemonics. These can be patched directly into the program using itself. The first word of each four-word entry is the exact IOT code (for example, 6221 for "CDF 20"), followed by 3 words containing up to 6 packed ASCII characters padded with trailing 0's. No attempt is made to decode any micro-coded IOTs. Either an exact match for the current contents will be found in the table or the program will output "IOT nnnn where nnnn is the octal typeout of the low 9 bits of the code. The next free location in the table (which is in field 1) is pointed to by the contents of location 10000. The table is terminated by the first 0 for an IOT code, so additions must be contiguous and added directly at the current end of the table.

For FPP instructions, the full FPP-8/A instruction set is decoded except for "IMUL", which is actually an integer mode "LEA". For the data manipulation instructions, the op-code mnemonic is followed by a "#" for the long-indexed format, by a "%" for the indirect-indexed format and by a space for the base addressing format. For the indirect-indexed and base addressing formats, the operand address is output as "B+nnn", where nnn is the 3 digit octal value of the displacement (3 or 7 bits) multiplied by 3. These formats are those used by the RALF assembler. This is also true for "LEA" instructions (i.e., "LEAI" is decoded as "LEA%"). Both jump and load-truth instruction decoding is done as a single mnemonic whose last two characters indicate the specified condition. All instructions which use 2 words are decoded with an "\*" in the location in the normal assembler format where the value of the second word would go. Index register number and "+" for auto-increment (if used) are also shown in the assembler format. Any combinations which are not in the FPP-8/A instruction definitions are output as "UNUSED".

#### NOTE

For both of these output formats, the use of the mapped access modes (and the 'OFFSET' mode for PDP decoding) allow the use of the "actual" addresses when decoding the instruction.

### K.3 WORD-TYPE COMMANDS

These commands are grouped by function, as follows:

#### Group 1:

- |        |  |
|--------|--|
| DUMP   | type/list out the contents of one or more blocks.    |
| LIST   | type/list out the contents of one or more locations. |
| MODIFY | modify one or more locations.                        |

- Group 2:
- WORD word search
  - STRING string search
  - SMASK set up string search mask
- Group 3:
- SET set up program switches & variables
  - SHOW show settings of program switches & variables
  - FILE look up file(s) on device
  - WRITE write out current buffer
  - SCAN scan for bad blocks
  - REWIND move device to block 1 & reset directory segment
- Group 4:
- OPEN open an output file on a file-structured device
  - CLOSE close the open output file
- Group 5:
- IF cause command skipping based on expression value
  - END resume command execution after unsatisfied 'IF'
  - COMMENT pass user commentary to output device
  - EXIT exit to OS/8 (same as CTRL/C)
- Group 6:
- EVAL evaluate a signed, double-precision expression.

Command words may always be abbreviated to their first two characters, as with the Monitor and BUILD, and some of the commands and their options may also be abbreviated to only one letter. When this is true, the command forms given will include the one-letter form, and the option forms will give the one-letter form directly under the full word form.

#### NOTE

In many cases, two or more words start with the same letter. In these cases, only one of these words may be abbreviated to one letter.

The descriptions for each command include each of the possible forms of the command, with an example of that form following it on the same line.

#### K.3.1 Output Formats

The 'FORMAT' option is used to 'SET' up the output format for the "\$" ('ALTMODE' or 'ESCAPE') command (single-character) described earlier and the default format for the 'DUMP', 'LIST' and 'MODIFY' commands described below. The syntax of this command is shown with the other 'SET' commands but is described here to make the descriptions of the following three commands more understandable. The <format> may be one of the following:

- ASCII output each word as a single ASCII character.  
A
- PACKED Output each word as two 6-bit trimmed and packed ASCII characters. This is the  
P format of PAL8 TEXT strings.
- OS Output each word as 1 or 2 OS/8 packed ASCII characters. The even address words  
output 1 character and the odd address words output 2 characters.

XS240	Output each word as two 6-bit packed ASCII characters by adding a space (240 octal) to the contents of each 6-bit byte. This is the format of PAL12 SIXBIT strings.
BYTE	Output each word as 1 or 2 OS/8 packed bytes of 8 bits each as a 3-digit octal numbers. The even address words output 1 number and the odd address words output 2 numbers.
UNSIGNED U	Output each word as an unsigned decimal number.
SIGNED S	Output each word as a signed decimal number.
OCTAL O	Output each word as a 4 digit octal number.
BCD B	Output each word as 3 bcd digits. The digits 0 through 9 are followed by “:” (10), “,” (11), “<” (12), “=” (13), “>” (14) and “?” (15).
PDP FPP	Output each word as an octal number, followed by 2 spaces and its mnemonic representation, assuming it to be a PDP-8 or an FPP-8A instruction. See the “symbolic” output description.
DIRECTORY	Output each word in octal, decimal (signed), date (see “@” command) and packed ASCII formats.

The ‘FORMAT’ is initialized to ‘PACKED’ ASCII.

The output from the ‘DUMP’ and ‘LIST’ commands for each of these formats is set up as follows:

1. At the beginning of each line the current location is output in <location> format with a 4 digit block number and a 5 digit address, both in octal, as

<block>.<address>:

For example, “1271.17205: “—location 17205(8) relative to block 1271(8).

2. The maximum number of words per line is set up as follows:
  - A. The four character formats output 16 words per line with no extra characters.
  - B. The five numeric formats output 8 words per line with 2 spaces between each number.
  - C. The “symbolic” and directory formats output 1 word per line.

For ‘LIST’ with A or B, the first line may be shorter than succeeding lines to force the second and following address outputs to be even multiples of 10 (octal).

**K.3.1.1 DUMP** — The ‘DUMP’ command is used to output one or more whole 256 word device blocks in the default or an optionally supplied format. This command has the following forms:

DUMP [<format>] <block string>

DUMP <block string>	DU 100,200-213,250
D <block string>	D (B)-(B+10),(S)
DUMP <format> <block string>	DU PA 212
D <format> <block string>	D OS 514

where the optional <format> is one of those given for the 'FORMAT' option above, and the <block string> is one or more numeric items separated by ","s and "-"s. The "-" is used when it is desired to dump a group of blocks, and is used as

<start block> - <end block>

the "," is used to separate single blocks or groups of blocks if there is more than one per line.

#### NOTE

When in a mapped ('SAVE' or 'LOAD') mode, the 'DUMP' command cannot dump any block except the block containing location 0. To eliminate the confusion that this would produce, the command will simply output an error message reminding the user that the proper command to use in a mapped mode is the 'LIST' command.

The output from the 'DUMP' command is sent to the 'DDEV' ("dump" device), which can be either the console terminal, the line printer, or a file. See the 'SET' command for setting the "dump" device and output mode.

**K.3.1.2 LIST** - The 'LIST' command is used to output the contents of one or more words on the device in the default or an optionally supplied format. This command has the following forms:

LIST [<format>] <location string>

LIST <location string>

L <location string>

LIST <format> <location string>

L <format> <location string>

LI 123.200-517,200.0

L 312.10-17,100-117,176

LI UN 200-227

L SI 200-277

where the optional <format> is one of those given for the "FORMAT" option above, and the <location string> is one or more <location>s, separated by ","s. When it is desired to list a group of words, the "-" is used to separate the start and end addresses as

[<block>.]<start address>[-<end address>]

If the block part is not specified, the last block number specified to the program will be used. If an end address is specified, the start address is assumed to be in the same field as the end address (i.e. the highest octal digit of the 5-digit address), so a maximum of 4096 words can be specified by each group.

As with the 'DUMP' command, the output from the 'LIST' command is sent to the 'DDEV'. For more information see the last paragraph of the 'DUMP' command, the 'SET' command, and the miscellaneous information section.

**K.3.1.3 MODIFY** - The 'MODIFY' command allows a string of locations on the device to be changed in an easy way. This is done by specifying the format of the input and letting the program do the work of storing the data properly. This command has the following forms:

MODIFY [<format>] <location string>

MODIFY <location string>

M <location string>

MODIFY <format> <location string>

M <format> <location string>

MO 200.0-17,35-43

M 32745-32777

MO PA 12342-12360

M AS 367.7261-7275

where the <location string> has exactly the same format as for the 'LIST' command and the <format> options are shown below. If the <format> is not specified (as with the first form), the program will pick the one of the formats below which corresponds to the current setting of the 'FORMAT' option. The corresponding formats are shown below.

'MODIFY' format	'FORMAT' setting and 'MODIFY' action.
ASCII A	ASCII – one character of input is stored in each word to be modified.
PACKED P	PACKED – two characters of input are packed as trimmed 6-bit characters, padded with trailing 00's. Control characters (those with codes less than 240 octal) are packed as a 6-bit 77 (flag) and the low-order 6-bits of the character. Note that this means that "@" is packed as a terminator (00) and that "?" is not unique.
OS	OS – three characters of input are packed into two words to be modified. When using this format, the start address must be even and the end address must be odd.
XS240	XS240 – a space (240 octal) is subtracted from each character and then it is packed as 6-bit bytes. Control characters are handled as with 'PACKED' format.
NUMERIC N	SIGNED & UNSIGNED decimal, BCD, OCTAL, BYTE, PDP, FPP and DIRECTORY formats – the input is a string of numeric items which are stored one per 12 bit word. See the section on numeric items. Note that bcd, byte, directory and "symbolic" are not included, that decimal or octal input are determined by the "CTRL"- "D" and "CTRL"- "K" switches and that signed numbers must be input enclosed in parentheses, e.g., 17, (- 10), ^D200, (- ^K312), 40, (- ^D35*129).

For each location or group of locations specified by the <location string>, the program will prompt for the input by printing the start location in the same format as described under the output format options above.

#### CAUTION

The program always modifies exactly the number of words specified by each item in the <location string>. If you input extra characters for the character formats or extra numeric items for the numeric format, they will be ignored. If you input not enough characters or items, the rest of the words to be modified will be set to the 'FILLER' value (see the 'SET' command). The program will not output any message if either of these things takes place. This does, however, make it possible to fill from 1 to 16 blocks on a device with zero or some other value by specifying all the words to be filled in 'NUMERIC' format and then responding to the prompt with a single "(F)" (the value of the 'FILLER') and "RETURN".

Input to the program is always terminated by a carriage-return ("RETURN"). It is therefore not possible to insert a carriage-return into a word using this command. All of the editing keys are available for use during input, therefore the CTRL/C, CTRL/Q, CTRL/S, CTRL/R, CTRL/P, CTRL/U and "RUBOUT" characters cannot be entered using this command either. For all of the character input formats, spaces (excluding leading spaces, which are ignored) and tabs in the input string are packed as they are seen. For numeric input, spaces are ignored and the numeric items must be separated by commas.

The command can always be aborted by CTRL/P if you change your mind before the "RETURN" key is pressed.

### K.3.2 Search Limits:

There are two search commands in the program, the 'WORD' search and the 'STRING' search. They both search from a lower to an upper limit. The limits are either the 'LOWER' and 'UPPER' limits set by the 'SET' command (the default) or the limits set up by the "'FROM' <1>" (which overrides the 'LOWER' limit) and/or "'TO' <1>" (which overrides the 'UPPER' limit) clauses which can optionally follow the command word. Leaving out the block parts of either of the two temporary limits will cause the program to use the block part of the corresponding default limit set by the 'SET' command. When in a mapped ('SAVE' or 'LOAD') access mode, searching through non-existent locations or overlays will never produce a match. Whenever a match is found, the program outputs the location where the match occurred, followed by the word or string that matched.

#### NOTE

It is not possible to search through more than one overlay per search command. To do so would require different and separate handling of the "block" and "address" parts of the limits when in the mapped modes including the resetting of the "address" part. The result is that in the mapped modes the "block" parts are used to set the overlay to be searched (lower limit only) and only the "address" parts are used in the determination of the number of words to be searched.

**K.3.2.1 WORD (Search)** – The 'WORD' search command is used to search for a word for words which, masked by the 'MASK' (which is set by the 'SET' command), will match the search word (also masked). This command has five options and therefore has the forms:

```
WORD [UNEQ] [ABS] [MEM] [FROM <1>] [TO <1>] <n>

WORD <n>                WO 217
W <n>                   W (S)
WORD UNEQUAL <n>       W UN 0
WO U <n>                WO U (C&377)
WORD ABSOLUTE <n>     WO AB 7402
W A <n>                 W A 7000
WORD MEMREF <n>       WOR MEM 41
WO M <n>                WO M 40
WORD FROM <1> <n>     WO FR 213.0 2317
W F <1> <n>            W F 1.35 (S)
WORD TO <1> <n>       W TO 213.345 1111
W T <1> <n>           WORD T 6257.377 7777
... and any combination and order of the above options.
```

where <n> is the bit pattern being searched for, 'UNEQUAL' means that all words which are not equal to <n> under the mask do match, the temporary limits clause is as described above, 'ABSOLUTE' means that the location where the match occurred is to be output as an absolute block number and displacement rather than as a relative location, and 'MEMREF' means that only words whose high-order octal digit is 0 thru 5 (i.e. the PDP-8 memory reference op-codes) are allowed to match, independent of the setting of the 'MASK'.

When you want to search for those words which reference a specific location, 'SET' the 'MASK' to 377 (octal) and then use the 'MEMREF' option. This will exclude all Operate (op-code 7) and IOT (op-code 6) "instructions" from the output and can make it considerably easier to find the desired information (e.g. you will not output the location of every "CIA", 7041 octal, when you are looking for references to location 41 octal).

#### NOTE

'UNEQUAL' has a higher priority than 'MEMREF', so first each word is tested under the mask for equal/ 'UNEQUAL' and if the specified condition is true, then the word is tested for the 'MEMREF' condition.

**K.3.2.2 STRING (Search)** — The 'STRING' search command is used to search for a string of numbers (bit patterns) under an optional string mask. This command has four options and therefore has the forms:

**STRING [MASKED] [ABS] [FROM <1>] [TO <1>] <numeric string>**

STRING <numeric string>	ST 4557,0,0
STRING MASKED <numeric string>	ST MA 4577,0,1203
ST M <numeric string>	ST M 5566,0
STRING ABSOLUTE <numeric string>	ST AB 'PI,'P@
ST A <numeric string>	ST A "A,"B
STRING FROM <1><numeric string>	STR FR 100 1,4000,2
STR F <1><numeric string>	ST F 123.4567 (S),(-S)
STRING TO <1><numeric string>	STR T 7577 'ER,'RO,'R@
ST F <1> T <1><numeric string>	ST F 1.0 T 7.0 'FO,'TP

... and any combination and order of the above options.

where the <numeric string> is simply a string of numeric items separated by commas, 'MASKED' specifies that the search is to be done under the string mask, 'ABSOLUTE' is as for the 'WORD' search, and the temporary limits clause is as described above.

When the 'MASKED' option is used, each item of the <numeric string> is masked by a separate mask word from the string mask. If the string mask is shorter than the search string, it is used in a circular fashion (the first word follows the last) as many times as necessary to mask all of the items of the search string. If the string mask is longer than the search string, the extra words are not used. This feature allows for very complex searches to be done.

For example: Suppose it is desired to find all calls to a certain subroutine in a file and also see their arguments. This could be done as follows:

FILE FUTIL	— look up file to be searched
FUTIL.SV 6070-6120 ^P	— you stop typeout
SE MODE SAVE	— set access mode to mapped
SMASK (-1),0,0	— set mask for 2 arguments per call
ST M 4547,0,0	— search for 4547 and 2 dummies

The output will give the address of the subroutine call (which requires an exact match due to the mask of 7777) and the contents of the two following words (which can be anything, since they are masked by 0).

Using the mask specified above, a search could be made for an exact match, 2 "don't care words" and another exact match by simply specifying a search string with 4 arguments. The first item of the string mask will be used to mask both the first and the last items of the search string.

This command can be particularly useful when trying to find certain kinds of references in programs for which no CREF listing (or perhaps no listing at all) is available.

**K.3.2.3 SMASK** — The 'SMASK' command is used to set up the string mask. It has the following form:

**SMASK <numeric string> SM (-1),0,0,7000,0**

where the <numeric string> is the same as for the 'STRING' search command above. The current contents of the string mask may be examined by the 'SHOW' command.

**K.3.2.4 SET** — The ‘SET’ command is used to set up various switches and variables within the program. It has many options, each of which is the name of the switch or variable and is always followed by a word or number describing how it is to be set. All items are separated by spaces. The command has the following two forms:

```

SET <option(s)>
S <option(s)>
SE OU PDP ERR LONG MODE SAV
S LO 100.0 UP 123.377 1DEV LPT

```

where the options are as follows:

OUTPUT OCTAL	Set the output mode for the single-character commands. Initialized to ‘OCTAL’.
OUTPUT O	
O PDP	
O P	
OUT FPP	
O F	
ERROR SHORT	Set the mode for error message output. The ‘SHOW’ ‘ERRORS’ command will list all error messages. Initialized to ‘LONG’. Also set to ‘SHORT’ by write-locking system device.
E S	
E LONG	
ERROR L	
FORMAT <format>	Set output format for ‘LIST’, ‘DUMP’, etc. The formats have been described previously. Initialized to ‘PACKED’ ASCII.
OFFSET <1>	Set the offset to the low 12 bits of <1>. Initialized to 0.
FILLER <n>	Set the filler to the low 12 bits of <n>. Initialized to 0.
LOWER <1>	Set the lower search limit. Initialized to 0.200.
UPPER <1>	Set the upper search limit. Initialized to 0.17577.
DEVICE <device name[:]>	Set up the OS/8 device for access. The handler is fetched at this time. Initialized to “SYS” (device 01). “:” In <device name[:]> is optional. <device name> is an assigned or permanent OS/8 mass storage device name.
DDEV <device name[:]>	Set up the “dump” device. Initialized to ‘SYS’. See also ‘DMODE’ below and ‘OPEN’ and ‘CLOSE’ commands.
MODE NORMAL	Set up the device access mode. These have been described previously. Initialized to ‘NORMAL’.
MODE N	
MODE SAVE	
MODE S	
MO LOAD	
MO L	
MO OFFSET	
MO O	
DMODE NONE	Set up the “dump” output mode. Initialized to “NONE”, which sends all output to console only. ‘PART’ sends ‘DUMP’, ‘LIST’ and ‘SHOW ERRORS’ output to the ‘DDEV’ (perhaps to a file). ‘ALL’ sends all output to both the console device and to the ‘DDEV’. (See section on file output.)
DMODE PART	
DMODE ALL	



MASK	<n>	Set the 'WORD' search mask to the low 12 bits of <n>.
M	<n>	Initialized to 7777.
TEMP	<n>	Set the 'TEMP' storage to the 24-bit value of <n>. Value is returned by subsequent use of the 'T' in expressions.

As many options as desired may be specified on one command line, separated by spaces. In the event of an error, none of the options past the point where the error occurred will have been set. If you have any question, use the 'SHOW' command.

**K.3.2.5 SHOW** — The 'SHOW' command is used to list the current setting of any of the program switches and variables set by the 'SET' command and other information. The program outputs either words or numbers to best describe the current settings. As with the 'SET' command, as many of the options for this command as desired may be specified on a single command line, separated by spaces. This command has the form:

SHOW <option(s)> SH BL CCB LOW UP ODT REL ABS

where the <options> are as follows:

BLOCK B	Output in octal the start block number of the last file specified by the last 'FILE' command.
CCB C	Output the core control block of the last file specified by the 'FILE' command. If the file is not a 'SAVE' file, an error will occur. The start address of the file is output as a 5-digit octal number, the job status word (JSW) is output in octal, and the core segments are output as 5-digit octal addresses.
HEADER H	Output the header block information for the last file specified by the last 'FILE' command. If the file is not a 'LOAD' file, an error will occur. The start address is output as a 5-digit octal number, followed by the next free address as a 5-digit octal number, the loader version number in octal and a message if Extended Precision is required. Then, for each level, a line is output with the number of overlays, the 5-digit start address, the relative start block and the length of the overlays (in blocks) for this level.
ABSOLUTE A	Output the absolute location of the last word accessed on the device in <location> format (a 4 digit octal block number, a "." and a 5-digit octal address) and the word "MOD" if the current block has been changed (the <something-changed> flag is set).
RELATIVE R	Output the relative location (what you specified) of the last word accessed on the device in <1> format and the word "MOD" if the current block has been changed.
ODT	Output the relative location of the last word accessed by one of the special-character commands in <1> format and the word "MOD" if the current block has been changed . .
LOWER	Output the search lower limit in <1> format.

UPPER	Output the search upper limit in <1> format.
FILLER	Output the value of the filler in octal.
MASK M	Output the 'WORD' search mask in octal.
SMASK	Output the current contents of the 'STRING' search mask as a string of octal numbers.
OFFSET	Output the value of the offset in octal.
MODE	Output the name of the current setting of the device access mode switch ('NORMAL', 'SAVE', 'LOAD' or 'OFFSET').
DEVICE	Output the OS/8 device name and number.
DDEV	Output the name of the "dump" device.
OUTPUT O	Output the name of the current single-character (ODT) command 'OUTPUT' mode (OCTAL, PDP or FPP).
FORMAT F	Output the name of the current output format.
VERSION	Output the current version number of FUTIL.
ERRORS E	Output a complete list of all error codes and their corresponding messages. Note: This list is output to the 'DDEV' ("dump" device) so that it can be output using the "LPT" handler for your system. Note that Version number is also output with errors.

**K.3.2.6 FILE** – The 'FILE' command is used to locate files on the OS/8 device and to set up the start block of a file for the mapped access modes, 'SHOW CCB', etc. This command has the forms:

```
FILE <file name string>      FI FUTIL PIP.SV
F <file name string>         F MICRO.LD
```

where the <file name string> is a string of one or more OS/8 file names, separated by spaces. Any other characters except "." will be taken as part of the file names. The program assumes extensions of ".SV", ".LD" and null (in this order) when looking up the file. This can lead to a substantial amount of time when a large directory is searched three times for a file that does not exist. Specifying an extension will cause only one lookup attempt to be made. A null extension, if desired, may be specified by making the "." the last character of the file name. The program does one (or more) separate lookup(s) for each file name specified and outputs either

```
<file name> ssss-eeee oooo (dddd) b.111 dd-mmm-yr
or
<file name> ssss-eeee oooo (dddd) b.111
or
<file name> LOOKUP FAILED
```

where "sss" is the start block of the file in octal, "eeee" is the last block of the file in octal, "oooo" is the length of the file in octal, "dddd" is the length of the file in decimal, "b.111" is the block (segment) and location within that block of the first word of the file entry (the first two characters of the name) in the directory, and dd-mmm-yy

is the file date. If the directory does not contain the extra word required for the date or the date word of the file is 0, the second form with no date will be output rather than the first form. The "LOOKUP FAILED" message means either that the file name was not found on the device or that the device is a write-only device.

The actual lookup operation is performed by the OS/8 USR, which is swapped as needed (see section on program execution). Since the USR keeps track of the current device once the first 'FILE' command is given, it will have the wrong directory in memory if the medium (tape or disk) is changed on the physical device. This can be solved one of three ways:

1. Use the 'REWIND' command to rewind the device being removed and clear the directory segment from the USR.
2. Do a 'SHOW ERRORS' and abort the output when the message output begins. This will have swapped out the USR. If messages are not available, use 1 or 3.
3. Use EXIT or CTRL/C to return to OS/8 and then directly restart FUTIL with the OS/8 START command. This will have swapped out both error messages and USR from memory.

Any of these methods should be followed by a 'SET' command to reset the 'DEVICE' and the rest of the I/O parameters desired.

The last file name specified that did not have a LOOKUP FAIL will be the file used in the mapped access modes, 'SHOW' 'CCB', etc. The program is initialized with no known file, so attempting to access any location in a mapped access mode or attempting to 'SHOW' 'CCB' or 'SHOW' 'HEADER' without giving a valid 'FILE' command will cause an error.

**K.3.2.7 WRITE** — The 'WRITE' command is used to force the program to write out the block currently in memory. It has the form:

```
WRITE [<block>]
```

where the optional <block> overrides the default number of the block that was read to specify where the current block is to be written. This obviously dangerous operation does allow a limited amount of copying in a special situation, e.g., allowing a directory to be backed up by moving a copy to the end of the device (see the examples section) or copying a single block from one device to another by changing the 'DEVICE' and then doing a 'WRITE' (with or without an argument). Again, as stated in the section on accessing the device, caution must be used because attempting to write beyond the end of a device may not be checked by the handler.

**K.3.2.8 SCAN** — The 'SCAN' command is used to do a rapid scan for read errors on the current 'DEVICE'. It has the form:

```
SCAN <block string>          SC 0-6257
```

where the <block string> is of the same form as for the 'DUMP' command. Each block is simply read. If an error occurs, it is reported as:

```
oooo BAD BLOCK
```

where "oooo" is the block number in octal, and the scan continues. This is the only FUTIL command that will continue on a read error. Should the current block have been changed, and any other blocks be included in the scan, an implicit write will be attempted by FUTIL. An error on this implicit write will be reported and then the command will be aborted. This is the only time that this command will attempt a write. The command can then be repeated if it is desired and it will execute (only one implicit write attempt is ever made by FUTIL).

#### NOTE

The OS/8 “actual” block number for the block to be read is stored (for display) in the computer MQ register (if present). It is particularly useful for following the progress of this command. The value is stored before checking if the current block needs to be written.

**K.3.2.9 REWIND** — The ‘REWIND’ command is used to move a tape back to block 1 and to reset the USR directory segment. It has the form:

**REWIND**

and must be terminated by the “RETURN” key. It causes a read of block 1 of the device and resets the directory segment in the USR (if in memory). Any subsequent ‘FILE’ command will cause the directory to be read.

#### K.3.3 File Output

Output to file-structured or non-file-structured “dump” devices is provided through two commands, ‘OPEN’ and ‘CLOSE’, and two ‘SET’ options, ‘DDEV’ and ‘DMODE’. They can be used to simply make fast hard copy output from the ‘DUMP’, ‘LIST’ and ‘SHOW ERRORS’ commands, to provide a hard copy log of all operations carried out with a video terminal, to provide an ASCII file output of some data for later processing by another program, etc.

Output to file-structured and to non-file-structured devices (serial devices) is handled in two separate ways. Output to the file-structured device is done by first setting the ‘DDEV’ and ‘DMODE’ and then ‘OPEN’ing an output file. No output to the device will be done until the file is open (to protect your directories), and then output will be done one block at a time. When output to the file is complete, ‘CLOSE’ your file to make it a permanent file (properly terminated with a CTRL-Z and padded with nulls).

Output to a non-file-structured device is done by simply setting the ‘DDEV’ and ‘DMODE’. Output to the device will be done one line at a time, as soon as specified by the ‘DMODE’, and neither the ‘OPEN’ nor the ‘CLOSE’ commands are needed. The output is done by padding the buffer with nulls after each line is ready and then calling the output device handler, so the handler used should ignore nulls (which leaves out the “PTR:” handler, for example).

**K.3.3.1 OPEN** — The ‘OPEN’ command is used to open an output file on file-structured devices for partial or total output from the program. It has the form:

**OPEN <file name>                      OPEN OUT.DA**

where the <file name> should be a standard OS/8 file name. The extension defaults to “.DU” (for “dump”) if none is supplied.

#### WARNING

FUTIL gives significance only to the characters space, carriage-return and “.” when scanning file names. It is therefore the responsibility of the user not to include characters that are not legal to other OS/8 programs or the files will be able to be accessed only through FUTIL or the CCL command decoder.

This command must be given after the “dump” device is ‘SET’ by the ‘DDEV’ option. The output specified by the ‘DMODE’ will then be sent to this file, one block at a time (packed only 8 bits per word), until either the ‘DMODE’ is changed or the file is ‘CLOSE’d.

Files can be opened at will without closing any previous file. This gives the user additional flexibility, but at the expense of possibly losing an output file if it is not ‘CLOSE’d.

Should an error occur on the output device while doing output, the file is simply thrown away (it cannot be ‘CLOSE’d).

**K.3.3.2 CLOSE** — The ‘CLOSE’ command is used to close an output file previously ‘OPEN’ed. It has the form:

CLOSE

and must be on a line by itself. If given with no file open, it is simply ignored.

### K.3.4 Batch Operation

Operation of FUTIL under BATCH allows repeated operations to be done without re-entry. All of the operations provided under interactive operation are provided except that the “RUBOUT” character is simply ignored, input is taken directly from the BATCH stream and console output goes to the log output device.

Four commands have been added specifically to support use of FUTIL under BATCH: ‘IF’, ‘END’, ‘COMMENT’ and ‘EXIT’. These commands are also available for interactive use, but are not as important in that mode.

**K.3.4.1 IF** — The ‘IF’ command was implemented specifically to allow FUTIL, when operating under BATCH to be sure that the correct operations are proceeding before modifying something incorrectly. It has the form:

IF <expression> IF C-3575

where <expression> is a general expression of the same form as used by the ‘EVAL’ command. If the expression evaluates to exactly zero (as a 24-bit integer), command execution will continue as though the command had not been seen. If the result is not exactly zero, command skipping will begin and will continue until a line containing the single word ‘END’ is found. Command execution will then resume.

This command was set up to test only for zero under the assumption that a test is to be made for some exact quantity. However, the capabilities of the expression evaluator can be used to generate sufficiently complex expressions for other tests. For example:

IF 40000000&(.....)	will test for positive
IF -(40000000&(..))-1	will test for negative
IF 10000&(-(77770000!(...)))	will test for 12-bit non-zero

**K.3.4.2 END** — The ‘END’ command is used to re-enable command execution following an unsatisfied ‘IF’ command. It has the form:

END

and must be on a single line by itself. When encountered during command execution, it is ignored. Note that the ‘IF’/‘END’ commands cannot be nested because the first ‘END’ found will re-enable command execution for any number of previous ‘IF’ commands. For example:

IF ...	
IF ...	
IF ...	
END	will terminate all three!

**K.3.4.3 COMMENT** — The ‘COMMENT’ command allows optional comments in command input which will simply be ignored during execution. It has the forms:

```
COMMENT [<comment>]          COMMENT THIS IS ONE
C      [<comment>]          C
```

where [<comment>] is an optional comment. Note that blank lines may also be used for formatting of the output log but that they will also close any open location.

**K.3.4.4 EXIT** — The ‘EXIT’ command provides a method of return to OS/8 other than “CTRL/C”. It has the form:

```
EXIT
```

and the rest of the line is ignored. Exit does not write out the last block modified. Use ‘WRITE’ to make changes permanent.

**K.3.4.5 EVAL** — The ‘EVAL’ command is used to evaluate a parenthesized expression of signed double-precision integers. It has the forms:

```
EVAL <expression>          EV S^D4096+D
E <expression>            E B*400+L
```

where the <expression> follows the normal rules for arithmetic expressions. Legal operators, in their order of precedence are:

(	evaluate inner expression
/	signed division
*	signed multiplication
-	subtraction
+	addition
&	logical product (“and”)
!	logical sum (“or”)
)	expression end

Besides 24 bit numeric input (which can be octal, decimal or mixed octal and decimal under the control of the CTRL/D and CTRL/K switches and ASCII and packed ASCII using ” and ’, the following “variables” may be used:

C	current contents (of location “L”).
L	current location (15 bit, same value as is output by the ‘SHOW’ ‘RELATIVE’ command).
B	current block number (as for “L”).
F	contents of ‘FILLER’ (12 bits).
T	contents of ‘TEMP’ (24 bits).
S	contents of the console switch register.
R	the remainder of the last division or the high product of the last multiplication. 24 bits, the sign may not be correct.
D	contents of OS/8 Monitor “date” word.

Overflow on addition, subtraction and multiplication are ignored, but trying to divide by 0 will cause an error.

If no errors occur, the program evaluates the expression and types out the results in the form:

```
= 00000000 (sddddddd)
```

where "00000000" is the double precision result in octal and "sddddddd" is the signed double precision result in decimal (the sign is either "-" or " ").

#### K.4 EXAMPLES

These examples are to help provide an overview of the use of the program and to stimulate the thoughts of the user. Example 3 and those that follow are not as well commented as the first two examples since it is intended to show concepts of what can be done with the program rather than the mechanics of the operations. Should questions arise on the mechanics, it is suggested that the first two examples and the discussions of the commands in question be reviewed.

##### EXAMPLE 1:

Assume that you would like to know what CCL remembers of your last ".UA" command. The remembrances are stored on block 65 (octal) of the system device. As described in the source of CCL, each of the remembrances is allocated 40 (octal), or 32 (decimal) words in this block, the first four of which contain binary information and the last 34 of which contain the last input command, stored as packed ASCII characters. The lines contain the inputs for the commands as follows: TECO and MAKE (line 0), EDIT and CREATE (line 1), COMPILE and EXECUTE and PAL (line 2), UA (line 3), UB (line 4), and UC (line 5). Thus, the saved ".UA" command can be listed by outputting the contents of the 4th through 37th words of area 3 in block 65 as packed ASCII characters as follows:

```
.R FUTIL <cr>           - call FUTIL from OS/8
EVA 3*40+4<cr>         - calculate start displacement
= 00000144 ( 0000100)  - of the 3rd "line" (=144[8])
```

Now list the words of this line with the LIST command, specifying the output format to be PACKED ASCII characters and the words to list to be block 65 locations 144 (from above) through 144+33 (the expression for the location of the last word of this "line"). FUTIL responds with the start location and a line of characters, and the next location with a multiple of 10[8] as an address and a line of characters.

```
LIST PACKED 65.144-(144+33)<cr>  - list the words wanted
0065.00144: DIR R:FUT???.*/E/R=3
0065.00160:                       - that's it!
```

#### NOTE

For the examples above and below, the symbol "<cr>" is used to show that you need to terminate your command lines with a "carriage return". All other lines above are output by the program.

##### EXAMPLE 2:

Now assume that you would like to make the simple patch for OS/8 FORTRAN IV users with an FPP-8/A to use the lockout feature of the FPP-8/A, as given in the August 1976 DIGITAL Software News. This requires changing the contents of location 15776 of FRTS (the Fortran Run Time System) from 400 to 410 (which adds the lockout bit). You also want to update the date word of the directory entry for FRTS (the 4th word beyond the start of the entry) to show that the file has been updated. This is done as follows:

```
.R FUTIL<cr>           - call it
```

SET MODE SAVE <cr>                   – set FUTIL to a mapped mode  
FILE FRTS<cr>                       – look up the file to map  
FRTS.SV 0671-0722 0032 (0026) 1.327 31-DEC-75  
                                     – “1.327” is start of entry!

Now use “ODT” command “/” to open and change one word.

15776/ 0400 410<cr>                   – add LOCKOUT bit  
SET MODE NORMAL<cr>                   – switch to unmapped

Now use “ODT” command “/” with an expression to open the date word, command “@” to output it in “date” format and then put today’s date (as an octal value) in its place.

1.(327+4)/6375  
@31-DEC-75 (D)<cr>                   – change file date to today’s date  
WRITE<cr>                           – send out this change

#### NOTE

First the file FRTS.SV is changed, and then the OS/8 directory is updated to the current date. Changing the address desired from FRTS to the directory automatically writes out the modified block of FRTS before reading in the directory segment that contains the file name. However, the changed directory segment must be written out explicitly because there are no other blocks to examine for this example.

#### EXAMPLE 3:

While doing a “/S” transfer with PIP, PIP gives a read error in your file “SOURCE.PA”. Attempting to read it with EDIT causes EDIT to type “?0^C” and return to the Monitor. Find out what is wrong as follows:

```
.R FUTIL
FI SOURCE.PA                       – look up the file
SOURCE.PA 0243-0351 0107 (0071) 2.005 30-AUG-74
SE MASK 0 LO 243.0 UP 351.377       – set up mask & limits
W UNE 0                           – search the file
?ee AT 08 FATAL READ ERROR       – here is the problem
[Note: “ee” may change with version, so is left out.]
SH ABS                           – find out where it is
ABS. LOC = 0271.00000
WR                               – attempt to clear error
DU OS (B+L/400)                   – it worked, now dump it
0271.00000: . . . ^P           – change your mind
W UN FR 272.0 0                   – check the rest of the file
^C                               – ok, now go fix the source
```



This sequence can also be carried out using the SCAN command as follows:

```
.R FUTIL
F1 SOURCE.PA          -- use CCL to call & lookup

SOURCE.PA 0243-0351 0107 (0071) 2.005 30-AUG-74
SCAN 243-351         -- scan the area

0271 BAD BLOCK      -- here is the problem!

271.0/ ?ee AT 07 FATAL READ ERROR -- get block with trouble

WR                  -- attempt to clear error

DU OS (B+L/400)    -- it worked, now dump it

0271.00000: . . . . ^P -- change your mind

^C                  -- ok, now go fix the source
```

If the error had been of some type other than a clearable error, the 'WR' command might also have failed.

#### EXAMPLE 4:

After using BUILD to change your system, find out the device number for "DTA1":

```
.R FUTIL

SE DEV DTA1         -- fetch the device handler
SHOW DEV
DEVICE = DTA1 (06) -- number is decimal
```

#### EXAMPLE 5:

By accident you zero a DECTape directory which contains the only copy of a file you need. You have the PIP "/E" listing of the directory but only want to re-build it enough to get the wanted file. The name of the file is "LOST.FI":

```
.R FUTIL

SE DEV DTA1         -- it was here
EV ^D5+14+11+10+16+13+8+5 -- lengths of all preceding
= 00000122 ( 0000082)    -- files
EV ^D730-^K61-^D82-25  -- rest of DECTape room
= 00001076 ( 0000574)

1.0/ 7777 (-3)        -- now 3 files
4/ 7777              -- 1 extra word per entry
0001.00005\ 0000 'DU  -- set up a "DUMMY" file
0001.00006\ 7556 'MM  -- over the old <EMPTY>
0001.00007\ 1752 'Y@
0001.00010\ 3451 0    -- a null extension
0001.00011\ 6234 (D)  -- put in today's date
0001.00012\ 4235 (-^D82) -- length
0001.00013\ 5761 'LO  -- the desired file
0001.00014\ 3341 'ST
0001.00015\ 2371 0
```

0001.00016\ 1107 'FI	– the extension
0001.00017\ 1366 (D)	
0001.00020\ 3015 (- ^D25)	– its length
0001.00021\ 3415 0	– an <EMPTY> to end it
0001.00022\ 2713 (- ^D574)	– the rest of the tape
WRITE	– now write it out
^C	– & exit to use it

The “LINE-FEED” key was used to advance through the words.

The above example is exactly the same as hand calculating the required length of the “DUMMY” file and then doing the following sequence using PIP:

```
.R PIP
*DTA1:DUMMY</I=122           – enter the DUMMY file
*DTA1:LOST.FI</I=31         – enter the LOST.FI
*^C
```

Note that the lengths of the files are specified for PIP in octal.

#### EXAMPLE 6:

Search for the end of each page of text in the file “WRITE.UP”. Since the file is an OS/8 ASCII file, which has two characters packed in the low 8 bits of two words and a third character packed in the high 4 bits of both of the two words, the form-feed character (^L) may be packed as the third character in some cases. So it is necessary to search both through the low 8 bits of each word and through the high 4 bits of each pair of words. Do it as follows:

```
.R FUTIL
FI WRITE.UP
WRITE.UP 0301-0437 S^P       – typeout stopped
SE MA 377
SE LO 301.0 UP 437.377      – char mask & limits set
WA “^L                       – search for form-feed
..... typeout occurs here
SMASK 7400,7400             – set up string mask
STM A (“^L*20), (“^L*400)   – search for 3rd char f-f
..... more typeout here    – only even addresses are real
                             – parts of form-feed pair!
```

In the string search, both the string and the data searched are “masked” by the string mask.

#### EXAMPLE 7:

You just assembled and saved PROG.SV but forgot to use the “/P” switch to ABSLDR. Fix the CCB (core control block) as follows:

```
.R FUTIL
```

```

FI PROG.SV
PROG.SV 0341-^P          - stop output
341.1/ 6203              - the "CDF CIF" part &
0341.00002\ 6400        - the address
0341.00003\ 0000 400    - change the JSW

WR                          - write the new CCB

SHOW CCB                  - check it this way
CCB:
  SA = 06400, JSW = 0400
  CORE^P                  - ok, output stopped

```

EXAMPLE 8:

The CREF listing file for your source file is about 732 blocks long (just over one full DECTape). If you do want to CREF the file onto a DECTape, you must do it either with the "/X" (don't process literals) switch or else you could use FUTIL to set up the directory with 735 blocks (by starting at block 2) as follows:

```

.R PIP
*DTA1:</Z                - zero the directory
*^C

.R FUTIL

SE DEV DTA1              - * * see WARNING below * *
1.1/ 0007 2              - change first block number
6/ 6446 (C-5)            - 5 more blocks
WR                          - write it out
^C                          - now CREF it . . . .

```

**WARNING**

Do not copy files onto a device that has been fixed this way with FOTP (COPY command) because it writes out a directory of six blocks after the transfers are finished and this will zap blocks 2 through 6 (the first 5 blocks of the first file) after the copy is done! PIP and other processors do not monkey around with the directory and will handle this correctly.

EXAMPLE 9:

Something is extremely flaky in your system and you have been losing your directory repeatedly. After fixing it up with both PIP and FUTIL, you just want to back it up while you generate your output files onto another device. Since your system device has a total of 6260 (octal) blocks (an RK8E) you back up the directory as follows:

```

.R FUTIL

1.0/ 7714 WR 6251        - transfer blocks up by
2.0/ 7740 WR 6252        - 6250 blocks.
3.0/ 7770 WR 6253
4.0/ 0000 3.2/ 0000     - block 3 was last, so
^C                          - all done

```

Shortly after this, everything crashes totally, i.e., directory smashed, system gone from disk. Rebooting from DECTape you use PIP to restore the system area and then use FUTIL to restore the directory:

.R FUTIL

SET DEV RKA0	– load non-system device
6251.0/ 7714 WR 1	– transfer by 6250 blocks
6252.0/ 7740 WR 2	– the other way
6253.0/ 7770 WR 3	– the last one
SCAN 0-6250	– do a SCAN for good luck

EXAMPLE 10:

During a SCAN of a device a bad block is found in an important data file and you would like to know just how far the read of that block really succeeded (e.g., on a DECTape, the type of error will determine whether the read will abort immediately or wait until the end of the physical block). The following commands assume that the block number is “bbbb” and set the input/output buffer in FUTIL to zeros before doing the read:

bbbb.0/ ?ee AT 07 FATAL READ ERROR	- do read to set up
MOD NUM 0-377	
bbbb.00000: 0	– set whole buffer to 0
SET DEV same	– set to device now in use
/ ?ee AT 01 FATAL READ ERROR	– force the read again
DUMP OC bbbb	– dump & examine the block

This example makes use of the fact that changing the DEVICE resets the status of the buffer without changing its contents. This status includes the block number known and the <something-changed> flag. Therefore the next access to the block causes the block to be re-read without attempting to write it out. Following the second error, as much as possible of the block will have been read into memory and can now be examined for non-zero values (assuming that the data itself was not all zeros). If the read terminated before the end of the block, there should be an obvious separation between the zero and non-zero values.

EXAMPLE 11:

Your system has a line printer which can output 132 characters per line and 68 lines per page and you would like to change PAL8 and CREF to make use of this to use less paper. Allowing two lines at the bottom of the page, the lines per page should be set to 66 (call this “nl”). Three changes need to be made to PAL8 to change the global number of lines per page (nl), the number of items per column of the symbol table (-nl+1) and the number of symbols per page (3\*[nl-1]). One change needs to be made to CREF to change the number of lines per page (nl) and three changes need to be made to change the number of items per line of cross references. Since CREF uses 10 characters for the symbol name and 6 characters per line number, 19 references can comfortably fit on one line (19\*6+10 = 124). The following changes to these two programs will increase the number of lines per page and the number of items per line in the cross-reference outputs and then update the dates of the two programs in the directory:

.R FUTIL FILE PAL8.SV

PAL8.SV 0200-0217 0020 (0016) 1.057 03-APR-76	
SET MODE SAVE	
1104/ 0070 ^D66	– global lines per page

1256/ 7711 (-^D65)	– symbol table column size
1273/ 0245 (3*^D65)	– symbols per page
FILE CREF	– * * SEE NOTE BELOW * *
CREF.SV 0220-0234 0015 (0013) 1.065 18-JAN-74	
2564/ 7704 (-^D66)	– lines per page as above
2017/ 1102 1366> TAD 2166	– change instructions here
2132/ 1102 1366> TAD 2166	– and here to get new
2166/ 0077 (-^D19)	– references per line
SET MODE NORM	– reset access mode
1.(57+4)/ 2036 (D)	– change dates of PAL8
(65+4)/ 0624 (D)	– and CREF.
WRITE	– output the last changes

Location 2166 was not used previous to this patch. Note that the first reference to the word in CREF will cause the last block that was modified in PAL8 to be written out. Similarly, the first reference to the directory will cause the last block that was modified in CREF to be written out.

#### NOTE

These patches were empirically determined and applied to PAL8 V9H and CREF V3C. They have been applied to some other versions of both programs but have not been tested with OS/8 V3D. USE THESE WITH CAUTION!

## K.5 MISCELLANEOUS INFORMATION

### K.5.1 Program Execution and Memory Allocation

The start address is 06400. When the program is started here, it resets the internal CCB buffer, resets the start address to 00200, tests the “scope mode” status (changing the action of “RUBOUT” if it is set), performs initialization for the extended date format, attempts to write out the error messages (resetting the ‘ERROR’ mode control if unsuccessful), tests the “BATCH-in-progress” status (changing all console I/O to BATCH I/O if it is set) and jumps to 00200. If, for some reason, it is desired to manually re-start the program after it has been loaded, it can be re-started at 00200.

The error messages are swapped with the USR, but not in the normal manner, allowing write-locked startup with the loss of the message text. When the program starts execution, it writes the messages onto the system device in the same area used by the USR in swapping. Once this has been done, the USR or error messages need only be read into memory, as needed. In the case where it is not possible to write on the system device, i.e., it is write-locked, the messages are discarded, ‘SHORT’ mode is set permanently and execution continues without a hitch. Similarly, should an error occur when reading the messages, ‘SHORT’ mode is set permanently and an error is given to warn that this has happened (with no message, of course!).

The program uses almost all of the available memory in an 8K PDP-8. It is allocated as follows:

00000-06237	program proper
06240-06577	buffer for arguments
06400-06777	– once only code for chaining –
06600-07177	“dump” device handler area, 2 pages
07277-07577	device handler area, 2 pages

10000-11777	USR area & error messages (swapped)
12000-12577	CCB/header input and test, file output
12600-15700	text strings, lists
15700-16377	string mask, command buffer stack
16400-16577	CCB buffer, 1 page
16600-17177	"dump" device buffer, 2 pages
17200-17577	I/O buffer, 2 pages

The buffer for arguments in field 0 is defined long enough to store 45 numeric string items. The string mask buffer, in field 1, is 66 words long, and the command buffer, also in field 1, is 140 characters long. These lengths were chosen in anticipation of input from console devices with up to 132 characters per line. No checking of any kind is done to protect against overflow of any of these buffers under the assumption that these buffers are large enough for any reasonable input to this program, however, the arrangement of the buffers is set up in such a way that the most valuable data is the farthest distance from a variable buffer.

The expression evaluation stack buffer uses the area in field 1 from the end of the command buffer (approximately location 16130) to the beginning of the CCB buffer (location 16377). This should provide ample room for any expression able to fit on one line. Again, no checking to prevent overflow is done.

## K.6 COMMAND SUMMARY

SINGLE-CHARACTER commands: ([<n>] = optional <item>)

[<1>] / <1>+ <1>-  
 [<n>] with # \$ : % & < = > ? @ [ \ ]  
 \$ (ESCAPE) RETURN ; LINE FEED ! ^ \_

WORD-TYPE commands: (And modifiers, many of which are optional)

ASCII PACKED OS XS240 UNSIGNED SIGNED BCD BYTE OCTAL PDP FPP DIR  
 DUMP           [<format>] <block string>        ([<format>]s above)  
 LIST           [<format>] <location string>    ([<format>]s above)  
 MODIFY         [<format>] <location string>    ([<format>]s below)  
 ASCII PACKED OS XS240 NUMERIC

WORD           <option(s)> <n>  
 UNEQUAL ABSOLUTE MEMREF FROM <1> TO <1>

STRING         <option(s)> <number string>  
 MASKED ABSOLUTE FROM <1> TO <1>

SMASK          <number string>                e.g., 1,34,0,7700,0,(-1),377

SET            <option>                    <setting>

OUTPUT         OCTAL PDP FPP  
 ERROR         LONG SHORT  
 FORMAT        <format>  
 OFFSET        <1>  
 LOWER         <1>  
 UPPER         <1>  
 DEVICE        <device name[:]>  
 DDEV          <device name[:]>  
 MODE          NORMAL SAVE LOAD OFFSET  
 DMODE         NONE PART ALL  
 MASK          <n>  
 FILLER        <n>  
 TEMP          <n>

SHOW <option(s)>  
 BLOCK CCB ABSOLUTE RELATIVE ODT LOWER UPPER  
 MASK SMASK OFFSET MODE DEVICE OUTPUT FORMAT  
 HEADER FILLER VERSION ERRORS DDEV  
 FILE <file name(s)>  
 WRITE [<block>]  
 SCAN <block string>  
 REWIND  
 OPEN <file name>  
 CLOSE  
 IF <expression>  
 END  
 COMMENT [<comment line>]  
 EXIT  
 EVAL <expression> e.g., (1!(S+^D17))\*^K15+(C&7600)  
 ! & + - \* / ( ) C L B F T S R D

Numeric Input:

^D ^K <digits> “<1 character> ’<2 characters>  
 (...all eval options...)

Control Characters:

^P ^C ^U ^R RUBOUT ^S ^Q

## K.7 SINGLE-CHARACTER COMMAND OUTPUT FORMAT SUMMARY

([<n>] = optional numeric item)

Output in octal or octal & “symbolic” (PDP or FPP):

<1>/ / [<n>]“LINE-FEED” [<n>! [<n>]^ [<n>]\_  
 <1>+ <1>-

Output in a specified format:

[<n>] # BCD  
 [<n>] \$ OS/8 ASCII  
 [<n>] : SIGNED decimal  
 [<n>] % BYTE octal  
 [<n>] & XS240 format packed ASCII  
 [<n>] < OCTAL  
 [<n>] = UNSIGNED decimal  
 [<n>] > PDP symbolic  
 [<n>] ? DIRECTORY  
 [<n>] @ DATE format (extended, in alpha)  
 [<n>] [ ASCII  
 [<n>] \ FPP symbolic  
 [<n>] ] PACKED ASCII  
 [<n>] \$ (ESCAPE) As ‘SET’ by last “SET FORMAT x”

No output: [<n>];

# APPENDIX L

## DUMP PROGRAM

The DUMP handler is a new OS/8 2-page handler that obtains blocks of binary data on file-structured devices and sends them to the LP08 line printer to produce a listing. This listing is called a DUMP.

Format:

```
.COPY DUMP:<dev:filename.ex  
or  
.R PIP  
*DUMP:<dev:filename.ex/I
```

Example:

```
.COPY DUMP:<SYS:FL2
```

After typing the command line followed by a carriage return, type the initial block number of the area in the specified file you want dumped. This causes block number 0000 of the file to automatically be dumped. In addition, it also causes the DUMP routine to skip to the block number specified, dump it and any block numbers greater than it.

Because the DUMP handler contains a routine that interacts with the keyboard monitor, you can change the block number previously entered by typing a new block number on the keyboard. When a new block number is specified, the current block number is completely dumped before the new block number takes effect.

If a carriage return is entered following the command line and no block number is supplied, the DUMP handler starts at block number 0000 of the file and dumps all the remaining block numbers in the specified file.

For each block of data (2 memory pages) sent to the LP08 line printer, there is a printed page of data followed by a form feed. If an uneven number of pages is sent to the line printer during the DUMP operation, the odd numbered page printed on the line printer will show only half a block (one memory page) of data.

If an illegal character (excluding 0-7, carriage return, and CTRL/C) is typed while entering the block number, a question mark (?) is echoed on the terminal. This causes any digits typed before it to be ignored and allows you to type in a new block number. If CTRL/C is typed while the DUMP handler is running, control returns to the keyboard monitor.

In addition to the CCL format shown using the COPY command, there is a -D option. When specified, this option forces the output device to be DUMP:.. This option can be used with any CCL command.

### FORM FEEDS

A form feed on the LP08 line printer occurs before block 0 data is sent to the handler, and after the handler is called to do a close (page count of 0).

### ADDING THE DUMP HANDLER TO YOUR SYSTEM

The DUMP handler can be added to your system the same way any other handler is added which is through the BUILD program. Its group name as well as its entry point name is DUMP; and the current version of the handler is A. This handler does not directly interact with the keyboard monitor but contains a routine that performed that function. It is a 2-page handler and has no coresident handler. The keyboard monitor runs completely overlapped with the LP08 handler.



## FORMAT OF THE DUMP

At the top left of every printed page in the DUMP listing is a 4-digit octal number. This number is the relative file block number of the data that is printed on that page. The first column of 4-digit octal numbers represents line numbers. Note that each line number is followed by a slash (/) which distinguishes it from the remaining eight columns. The remaining eight columns represent the actual data words located within a specific block in a file. The next column containing 16 characters is a representation of the eight data words on that line. There are two 6-bit characters packed in one word. That is, each data word is represented by two ASCII characters.

The last column containing 12 characters is another representation of the eight data words on that line. There are three 8-bit characters packed in two words. That is, every two data words is represented by three ASCII characters. Note that some of these spaces in this column could represent non-printable characters. Any character that is not physically on the line printer can be referred to as a non-printable character.

The following listing is an example of a single printed page from a DUMP listing.

```

0004
0000/ 7733 2213 0000 1720 7777 1322 0506 6300 ?[KK@@OP??KREF3@ [ P R F@
0001/ 1501 6264 7653 1322 0506 6300 2202 6264 MA24>+KREF3@R824 A4<+R F@ 4L
0002/ 7767 1420 2326 0000 0216 8304 7777 2213 ?7LPSV@@BN3D??RK V @ D
0003/ 0502 1404 2326 6354 7737 0425 1520 0000 EBLDSV3,?.DUMP@@ B V L P 0
0004/ 0215 6314 7776 0681 0000 0000 1501 6314 BN3L?>F1@@@@MA3L L 1 AL<
0005/ 7777 0662 0000 0000 1501 6314 7777 0664 ??F2@@@@MA3L??F4 2 AL< 4
0006/ 0000 0000 1501 6314 7777 0425 1520 0000 @@@@MA3L??DUMP@@ AL< P 0
0007/ 2001 6304 7761 2425 1520 6200 2001 6314 PA3D?1DUMP2@PA3L DL P < LL
0010/ 7762 0663 0000 0000 1501 6314 7777 0665 ?2F3@@@@MA3L??F5 3 AL< 5
0011/ 0000 0000 1521 6314 7777 0363 0000 0000 @@@@MA3L??C3@@@@ AL<
0012/ 1601 6314 7777 0361 0000 0000 1501 6314 MA3L??C1@@@@MA3L AL< AL<
0013/ 7777 0362 0000 0000 1501 6314 7777 0364 ??C2@@@@MA3L??C4 AL<
0014/ 0000 0000 1501 6314 7777 2262 0000 0000 @@@@MA3L??R2@@@@ AL< 2
0015/ 1501 6314 7777 2261 0000 0000 1501 6314 MA3L??R1@@@@MA3L AL< 1 AL<
0016/ 7777 2263 0000 0000 1501 6314 7777 0365 ??K3@@@@MA3L??C5 3 AL<
0017/ 0000 0000 1501 6314 7777 0425 1520 6300 @@@@MA3L??DUMP3@ AL< P<
0020/ 2001 6324 7762 0425 1520 6300 0216 6324 PA3T?2DUMP3@BN3T TL P@< T
0021/ 7776 2266 8070 1623 0216 6324 7777 0425 ?>RF08NSBN3T??DU 8 C T
0022/ 1520 6400 0216 6324 7776 0425 1520 0000 MP4@BN3T?>DUMP@@ P = T P 0
0023/ 2320 6334 7774 0425 1520 6500 2001 6334 SV3\?<DUMP5@PA3\ V\L P@= \L
0024/ 7762 0425 1520 6600 2001 6354 7761 0425 ?2DUMP6@PA3,?1DU P = L
0025/ 1520 6500 0216 6354 7776 0425 1520 6400 MP6@BN3,?>DUMP4@ P = P =
0026/ 2001 6324 7762 0425 1520 6500 0216 6334 PA3T?2DUMP5@BN3\ TL P@= \
0027/ 7776 1501 0363 6500 0216 6354 7735 1501 ?>MAC35@BN3,?]MA A @ ]A
0030/ 0363 6500 2001 6354 7400 0617 1700 0000 C35@PA3,<@FOO@@@ @ L @ 0
0031/ 1320 6354 7775 1501 0363 6500 1520 6354 MP3,?=MAC35@MP3, P < A @ P <
0032/ 7744 0000 7252 1501 0363 6500 1423 6354 ?$@:*MAC35@LS3, *A @ <
0033/ 5654 0000 7271 6300 0001 6264 7653 1322 2,@:93@PA24>+KR , @9@ 4L+R
0034/ 0300 6300 0216 6264 7767 3000 0000 0000 EF3@BN24?7X@@@@@ F@ 4
0035/ 1501 6264 7777 7252 7277 7304 7440 5300 MA24??:*?:D< +@ A4< * ?D @
0036/ 7666 1234 4000 7235 7241 2422 7666 1234 >6J\ @:] :!TR>6J\ 6 ! 6
0037/ 4000 7235 7241 2422 4000 7235 7241 2422 @:] :!TR @:] :!TR ! !

```

# APPENDIX M

## RKLFMT DISK FORMATTER PROGRAM

### M.1 INTRODUCTION

This appendix describes the procedure for formatting a RK05 disk using the RKLFMT program. The RK8E/RK8L disk formatter program is designed to write and check the format of the complete disk cartridge. Only standard Digital surface format is available (i.e., sectors numbered in the normal numerical sequence 0, 1, 2, 3, 4, 5, etc.). The program occupies locations 0000 to 4177 of the current field.

Restrictions, using this program, are that the RK8L control which can control up to 8 drives, will not run with the DW8E bus adapter. This is because the RK8L control uses IOT0 for extended drives 4-7 which is not available on the DW8E.

Hardware requirements are as follows:

1. PDP-8/E, 8/F, 8/M or 8/A Computer  
Other family of 8-compatible computer with necessary DW8E bus adapter for RK8E control only.
2. At least 4K of read/write memory, and at least 8K of memory is needed for operation of the console package.
3. ASR-33 teletype or equivalent
4. RK8E disk control or RK8L disk control
5. RK05J or RK05F disk drive(s)

#### NOTE

The RK05F drive is considered as two separate units.  
When answering all questions each separate unit must be specified: DSK0?, DSK1?, DSK2?, etc.

### M.2 RUNNING THE PROGRAM

To format an RK05, type the following command:

```
_R RKLFMT
```

Mount the disk (write enabled), and the instructions in the program that follow.

If the formatter program fails to operate correctly, run the following programs:

1. All basic and extended memory diagnostics
2. For the RK8E control, run the RK8E diskless control test and the RK8E drive control test.
3. For the RK8L control, run the RK8L instruction test.

### M.3 STANDARD TEST PROCEDURES

To run the formatter program, follow the procedure in Section M.4. The following two procedures describe the drive setup procedure for the RK05F and the drive cartridge mounting procedure for the RK05J.

### M.3.1 RK05J Drive Cartridge Mounting Procedure

The following is the correct cartridge mounting procedure for the RK05J disk drive. Any deviation encountered during this procedure is considered an error condition.

- a. Set switch labeled "RUN/LOAD" to the "LOAD" position.
- b. Turn AC power on.
- c. Verify that light labeled "PWR" is on.
- d. Wait for light labeled "LOAD" to come on.
- e. Verify that lights labeled "RDY", "ON CYL", "FAULT", "WT", and "RD" are off.
- f. Open access door.
- g. Insert cartridge.
- h. Close access door.
- i. Set switch labeled "RUN/LOAD" to the "RUN" position.
- j. Wait for lights labeled "RDY" and "ON CYL" to come on.
- k. Toggle switch labeled "WT PROT" and verify that the light labeled "WT PROT" goes on and off.
- l. Toggle switch labeled "WT PROT" until light labeled "WT PROT" goes off.
- m. Verify that lights labeled "FAULT", "WT", "RD", and "LOAD" are off.

### M.3.2 RK05F Drive Setup Procedure

The following is the correct drive setup procedure for the RK05F disk drive. Any deviation encountered during this procedure is considered an error condition.

- a. Set switch labeled "RUN/LOAD" to the "LOAD" position.
- b. Turn AC power on.
- c. Verify that light labeled "PWR" is on.
- d. Wait for light labeled "LOAD" to come on.
- e. Verify that lights labeled "RDY", "ON CYL", "FAULT", "WT", and "RD" are off.
- f. Set switch labeled "RUN/LOAD" to the "RUN" position.
- g. Wait for lights labeled "RDY" and "ON CYL" to come on.
- h. Toggle switch labeled "WT PROT" and verify that the light labeled "WT PROT" goes on and off.
- i. Toggle switch labeled "WT PROT" until light labeled "WT PROT" goes off.
- j. Verify that lights labeled "FAULT", "WT", "RD", and "LOAD" are off.

### M.4 FORMAT PROGRAM

- a. Make all drives ready to be formatted:

For RK05J drives use the RK05 drive mounting procedure (M.3.1).

For RK05F drives use the RK05 drive setup procedure (M.3.2).

- b. Set switch labeled "RUN/LOAD" to the "LOAD" position on all drives not being formatted.

The TTY will type the following program name, information, and question.

RK8E/RK8L DISK FORMATTER PROGRAM

For all questions answer Y for YES or N for NO.

FORMAT DISK 0?

- c. If the operator desires to format disk 0, type Y for YES, otherwise, N for NO, on the TTY keyboard. The following question is then typed on the TTY.

FORMAT DISK 1?

- d. If the operator desires to format disk 1, type Y for YES, otherwise, N for NO, on the TTY keyboard. The following question is then typed on the TTY.

FORMAT DISK 2?

- e. If the operator desires to format disk 2, type Y for YES, otherwise, N for NO, on the TTY keyboard. The following question is then typed on the TTY.

FORMAT DISK 3?

- f. If the operator desires to format disk 3, type Y for YES, otherwise, N for NO, on the TTY keyboard. The following question is then typed on the TTY.

FORMAT DISK 4?

- g. If the operator desires to format disk 4, type Y for YES, otherwise, N for NO, on the TTY keyboard. The following question is then typed on the TTY.

FORMAT DISK 5?

- h. If the operator desires to format disk 5, type Y for YES, otherwise, N for NO, on the TTY keyboard. The following question is then typed on the TTY.

FORMAT DISK 6?

- i. If the operator desires to format disk 6, type Y for YES, otherwise, N for NO, on the TTY keyboard. The following question is then typed on the TTY.

FORMAT DISK 7?

- j. If the operator desires to format disk 7, type Y for YES, otherwise, N for NO, on the TTY keyboard. The following question is then typed on the TTY.

ARE YOU SURE?

- k. Typing N for NO causes the repetition of all the previous questions. Typing Y for YES, results in execution of the operation selected.

- l. Program execution is approx. 80 seconds per disk drive. After all disks selected have been formatted and checked, the TTY types the following pass complete message and question.

RK8E/RK8L DISK FORMATTER PASS COMPLETE  
FORMAT SAME DISK(S) AGAIN?

- m. If the operator desires to repeat the operation selected, type Y for YES. Typing N for NO causes the repetition of the initial start-up questions.

## M.5 ERRORS

When a recoverable error occurs the TTY prints an "ERROR HEADER" and error information pertaining to the failure.

Possible error headers are as follows:

DISK DATA ERROR  
READ STATUS ERROR  
WRITE STATUS ERROR  
RECALIBRATE STATUS ERROR

After the error header mentioned above is typed, the TTY prints some of the following error information pertaining to the failure.

PC: Program Location of Failure  
GD: Expected Information  
EX: Extended Drive Bit  
CM: Software Command Register  
ST: Contents of Status Register  
DA: Software Cylinder, Surface, and Sector Register  
CA: Initial Current Address  
AD: Address of Data Break  
DT: Data Found During Data Break

After the error information is typed, the TTY types one of the following questions asking the error recovery desired.

1. If the error was a recalibrate error, the following question is typed.

TRY TO RECALIBRATE SAME DISK AGAIN?

Typing a Y for YES causes the repetition of the recalibrate sequence on the disk in error. Typing N for NO results in progressing to the next available disk.

2. If the error was a write error the following question is typed.

TRY TO FORMAT SAME CYLINDER AGAIN?

Typing Y for YES results in a repeat of the write sequence on the current cylinder. Typing N for NO causes progressing to the next sequential cylinder.

3. If the error was a read or check error the following question is typed.

TRY TO CHECK SAME CYLINDER AGAIN?

Typing a Y for YES causes the repetition in the read and check sequence on the current cylinder. Typing a N for NO results in progressing to the next sequential cylinder.

## M.6 PROGRAM DESCRIPTION

The formatting is actually a function of the RK8E or RK8L control and drive logic. The program writes data on every sector in the "WRITE ALL" mode, then checks the data while in the "READ DATA" mode to verify that the header words written on every sector are also correct. The "READ DATA MODE" automatically performs a check header function.

The first two words of every sector are set to the absolute disk address (i.e. command register bits 9-11 and cylinder, surface, and sector bits 0-11, respectively). The remainder of the data area is set to all zeros when the data is written. Only the first two words of every sector (i.e., the addressing information) are checked when data is read in the "READ DATA" mode.

## M.7 CONTROL CHARACTERS

Control characters are used to give the operator the ability to perform the following functions.

### NOTE

The program will respond to the control character in five seconds or less.

<b>CTRL/C</b>	Starts the monitor at location 7600.
<b>CTRL/R</b>	Restarts the program.
<b>CTRL/E</b>	Continues the program from an error if allowed by the diagnostic or from a waiting statement.
<b>CTRL/L</b>	Switches the terminal messages from the display to a line printer. To restore the messages on the terminal, CTRL/L must be typed again. If no printer is available and CTRL/L is typed the result is that the console package will wait for CTRL/C or R. The CTRL/L sends output to the line printer and the program attempts to continue as if a CTRL/E was typed in.
<b>CTRL/D</b>	Allows you to change the switch register during program operation. Typing this character results in an interrogation of the switch register question.
<b>CTRL/S</b>	Stops program execution and waits in a loop for a continue. The only way to continue is to type a CTRL/Q, R or C. This is a nonprinting character.
<b>CTRL/Q</b>	This causes continuation of a program after a CTRL/S is typed. This is a nonprinting character.

## M.8 MISCELLANEOUS

### M.8.1 Waiting Message

The waiting message allows the operator time to make a decision as to what control character to type. This message appears at the end of pass message if the halt on pass bit is set. The control characters may now be used to perform the needed function.

The waiting message is printed after an error message if the halt on error bit is set. Here again the control characters may be used.

The waiting message is printed if operator intervention is required.

### M.8.2 End of Pass

The normal program pass complete as described in Section M.4 is used.

### M.8.3 Errors

The standard error reports previously described in Section M.5 are used.

### M.8.4 Location Changes

The following locations can be changed to meet the specific need for modification of the diagnostic.

3637 Is the location set for the number of filler characters after a CRLF set to four (4)



READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. Problems with software should be reported on a Software Performance Report (SPR) form. If you require a written reply and are eligible to receive one under SPR service, submit your comments on an SPR form.

Did you find errors in this manual? If so, specify by page.

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

Please indicate the type of user/reader that you most nearly represent.

- Assembly language programmer
- Higher-level language programmer
- Occasional programmer (experienced)
- User with little programming experience
- Student programmer
- Non-programmer interested in computer concepts and capabilities

Name \_\_\_\_\_ Date \_\_\_\_\_

Organization \_\_\_\_\_

Street \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip Code \_\_\_\_\_  
or  
Country

Please cut along this line.



-----  
**Fold Here**  
-----

-----  
**Do Not Tear - Fold Here and Staple**  
-----

FIRST CLASS  
PERMIT NO. 33  
MAYNARD, MASS.

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

Postage will be paid by:

**digital**

Software Documentation  
146 Main Street ML 5-5/E39  
Maynard, Massachusetts 01754

