

Technical Memorandum No. 101


036040-13-M

4K-CPS

A PROGRAMMING SYSTEM FOR THE PDP-8 SIDE
OF THE DEC LINC-8 COMPUTER

by

K. Metzger
G. Cederquist
B. Barton

Approved by: 
Theodore G. Birdsall

COOLEY ELECTRONICS LABORATORY

Department of Electrical and Computer Engineering
The University of Michigan
Ann Arbor, Michigan

for

Contract No. N00014-67-A-0181-0032
Office of Naval Research
Department of the Navy
Arlington, Virginia 22217

Second Printing
March 1972

Approved for public release; distribution unlimited.

ABSTRACT

The 4K-Cooley Programming System (4K-CPS) is a collection of programs designed to aid in programming the PDP-8 side of Digital Equipment Corporation's LINC-8 computer. This collection of programs consists of a file system, a loader, a symbolic text editor, and an assembler. These programs are structured so that a user can enter, edit, assemble, load, and execute programs entirely from the Teletype keyboard. The system architecture is modular, and future components are easily "plugged-in" as needed.

FOREWORD

This report describes and documents the programming system 4K-CPS. 4K-CPS is a programming and file management system intended for use in programming the PDP-8 side of the Digital Equipment Corporation LINC-8 computer.

4K-CPS was written at the Cooley Electronics Laboratory (CEL) of The University of Michigan and is based on a larger and more comprehensive system (CPS) designed for the 8-K LINC-8 written by Gerald Cederquist of CEL.

When the original CPS was written, it was intended for use only on the CEL 8-K LINC-8. However, CEL is affiliated with the Institute of Marine Sciences (IMS) of The University of Miami in the MIMI project, and IMS has two 4-K LINC-8 computers. It soon became obvious that a 4-K version of CPS would give IMS a greatly increased programming capability and would aid in the transferral of programs between the two facilities. Thus, in the two-month period of June and July, 1969 4K-CPS was developed by CEL for use at IMS.

In producing 4K-CPS the basic CPS data structures were retained; however, because of the smaller amount of available memory the internal coding was extensively restructured. The resulting system closely approximates the external features of the current version of CPS.

The first printing of this memorandum was issued November 1969 under Contract No. Nonr-1224(36), NR187-200.

PREFACE

The 4K-CPS system has been in existence now for over two and one-half years and has been distributed by CEL to over 14 installations. The original printing of the manual consisted of 50 copies, as it turned out a woefully inadequate number. The second printing consists of the corrected first printing plus an additional appendix, Appendix F. Appendix F contains copies of the system alteration memos issued since July 1969 and user descriptions of several new system programs. Of particular note is the addition of FOCAL. FOCAL was incorporated into 4K-CPS through the hard and persistent efforts of CEL's Brian Barton.

It should be obvious from the changes reported in Appendix F that 4K-CPS has not changed greatly since its creation. This is not because the system cannot be improved upon but because of the lack of time and pressure necessary to implement improvements.

TABLE OF CONTENTS

	<u>Page</u>
ABSTRACT	iii
FOREWORD	iv
PREFACE	v
CHAPTER 1: INTRODUCTION	1
1.1 Why 4K-CPS ?	1
1.2 4K-CPS General Structure	3
1.3 Operational Procedure	4
CHAPTER 2: BASIC 4K-CPS	8
2.1 File System	8
2.1.1 Getting Into the File System	8
2.1.2 Input Line Editing	9
2.1.3 Prefix Characters	9
2.1.4 Commands	10
2.2 Symbolic Text Editor	13
2.2.1 Operating Modes	13
2.2.2 Commands	14
2.2.3 Special Features	16
2.3 Assembler	17
2.3.1 Running *ASM	17
2.3.2 Listing Control	18
2.3.3 Reference Material	19
2.4 Loader	23
2.4.1 Running *LOAD	24
2.4.2 Operation	24
2.4.3 Commands	24
CHAPTER 3: ADVANCED 4K-CPS	27
3.1 Tape Organization	27
3.2 Structure of 4K-CPS File Indexes	28
3.3 Key for Updating the Free Names, -S and -B	29
3.4 File System Command Line Structure	29
3.5 Communication Area Structure	30

TABLE OF CONTENTS (Cont.)

	<u>Page</u>
3.6 Special Commands	32
3.7 4K-CPS Data Structure	32
3.7.1 Symbolic Text Files	32
3.7.2 Absolute Binary Files	33
3.7.3 Core Image Files	33
 CHAPTER 4: SYSTEM DOCUMENTATION	 35
4.1 Bootstraps	35
4.1.1 SCLD	35
4.1.2 SBOOT	35
4.1.3 SBOOT 2	36
4.1.4 TAPEMIX	36
4.2 The File System	36
4.3 The Loader	37
4.4 The Editor	37
4.5 The Assembler	38
 CHAPTER 5: CONCLUSION	 39
 APPENDIX A	 40
 APPENDIX B	 42
 APPENDIX C	 43
 APPENDIX D	 54
 APPENDIX E	 58
 APPENDIX F	 63

1. INTRODUCTION

The 4K - Cooley Programming System (4K-CPS) is a collection of programs designed to aid in programming the PDP-8 side of Digital Equipment Corporation's LINC-8 computer. This collection of programs consists of a file system, a loader, a symbolic text editor, and an assembler. These programs are structured so that a user can enter, edit, assemble, load, and execute programs entirely from the Teletype keyboard. The system architecture is modular and future components are easily "plugged-in" as needed.

The 4K-CPS, as described in this report, uses the two tape units present on all LINC-8 computers. However it should be easy to modify 4K-CPS to run on any PDP-8 using two TU-55 tape drives or a disc.

The programming system 4K-CPS is based on an operating system (CPS) developed and in use at the Cooley Electronics Laboratory of the University of Michigan. CPS itself is based on the Michigan Terminal System for the IBM 360/67 and requires 8-K words of memory. 4K-CPS is intended to represent a benchmark in the external structure of CPS and serve as a relatively fixed programming tool. Care has been taken to allow for limited future expansion.

The 4K in 4K-CPS refers only to the minimum required memory size. 4K-CPS is fully capable of producing programs that occupy thorough 16-K of memory and of loading these programs properly.

Although 4K-CPS and CPS are almost identical externally, the internal structures differ radically. This is mainly due to the more severe space limitation imposed by the smaller memory.

*In order to run 4K-CPS a minor hardware (or software) modification is required. Appendix A contains a description of both alterations. In addition 4K-CPS uses LINC tapes marked using 128 word blocks. (A utility program is included for marking such tapes.)

1.1 Why 4K-CPS?

Or for that matter, why an operating system? As an answer to this question, look at the steps involved in preparing a program using paper tape. It is assumed that the reader knows how to write programs in PDP-8 machine language and wishes to use the PDP-8 side of the LINC-8.

*No longer required.

- (1) Write the program down on paper.
- (2) Load in the symbolic editor, and enter the program into the computer.
- (3) When satisfied with the program, punch a paper tape.
- (4) Load and start the assembler.
- (5) Feed the paper tape prepared in step 3 through the assembler at least 2 times.
 - (a) If any errors occur, reload the editor, read the source tapes back in, then go to step 3.
 - (b) If the program assembles, punch a binary paper tape.
- (6) Load the binary loader.
- (7) Load the binary paper tapes.
- (8) Try the program.
 - (a) If it is bad, go back to step 5(a).
 - (b) If it works, save the binary paper tape.

With the basic PDP-8 using an ASR-33 Teletype, paper tape is read and punched at 10 characters per second. The time spent in steps 3, 5, 5(a), 5(b), 7, 8(a) (even assuming the editor, assembler, and loader are loaded in zero time) can easily run into hours for even a small program of one or two memory pages in length. DEC's 8-LIBRARY SYSTEM offers some speed-up but still requires paper tape input and output for the assembler.

All of the steps are still required when using 4K-CPS; however, those steps involving paper tape only require typing short command lines on the Teletype. The hours become fractions of a minute. This speed-up is due to the use of high speed bulk storage (LINC tape in this case) at each stage where paper tape would have been used before. However, just as the programmer had to properly store, handle, and load the paper tapes, the proper sections of magnetic tape must be kept track of, filed, and loaded when required. This could all be done by a programmer with a scratch pad, a simple tape routine, and a lot of patience. 4K-CPS is an attempt at providing an automatic scratch pad and tape routine.

4K-CPS allows the user to store program segments in files, to call them into memory at will, edit them, save them in files, assemble programs from files, and to run programs stored in files. In addition, these programs may operate on data stored in other files. Each one of these steps requires only a single typewritten command from the user. The file system is intelligent enough to know, on its own, whether a file contains symbolic text, binary frames, or a core image.

1.2 4K-CPS General Structure

The heart of 4K-CPS is a powerful, comprehensive, easy-to-use file system. In fact, the file system along with its conventions could be considered to be 4K-CPS in its entirety. This system allocates file storage, transfers files, and links files as well as loading and starting program execution. It also maintains the system indexes, keeping track not only of where data and program files are located but also remembering the type of material stored in each file. All other system functions, such as assemblies, are performed by programs stored in program files using information supplied by the user and the file system. The structure of 4K-CPS is illustrated in Fig. 1.

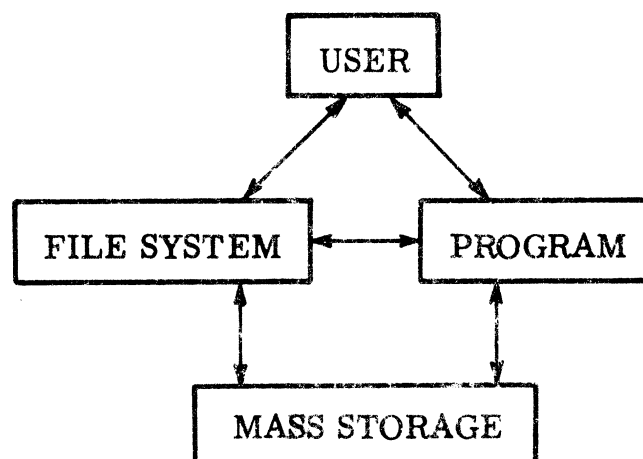


Fig. 1. 4K CPS organization

In the current version of 4K-CPS magnetic tape is used to provide mass storage for programs, data and for the file system.

The three main system programs provided with the 4K-CPS file system are:

(1) A loader, *LOAD. This loader combines core images and absolute binary (terms to be defined later) into core images which are in the form required by the file system "RUN" command.

(2) A symbolic text editor, *ED. This is a greatly modified version of Digital-08-ESAA supplied by DEC.

(3) An assembler, *ASM. This is a cut-down version of DEC's MACRO-8 assembler. The macro and number processors have been deleted because of the support code required for interfacing to 4K-CPS. The symbol table capability is the same as for the standard version of MACRO-8 with all options present.

In putting these programs into 4K-CPS it was decided to borrow as much as possible of DEC's software and modify it in order to speed system development.

1.3 Operational Procedure

This section illustrates the procedures involved in entering, assembling, loading and executing programs using 4K-CPS. Typical commands are used in order to aid in illustrating the basic system operation. Section 2 contains a more detailed description of the various system commands and facilities.

The following operations are performed when using 4K-CPS to prepare and run programs.

(1) The file system must be loaded into core. (See Section 2.1.1 for instructions.) (Mount tapes, lift LOAD.)

(2) The user's program must either be typed or read from paper tape into the system. Symbolic programs are entered into 4K-CPS through the use of the symbolic text editor, *ED. To invoke the editor, use the file system command

RUN *ED

This command causes the editor to be loaded into core and started in the command mode with an empty text buffer.

The use of the editor is described in Section 2.2. When the desired program (or as much of it as the editor can handle in a single text buffer) has been typed in, and is in satisfactory shape, the editor should be placed in the command mode and the "S" command given. This command causes the editor to write its current text buffer into the temporary file -S and then return control of the computer to the file system.

(3) Since *ED always stores its text buffer in the file named -S, the contents of -S should be saved into a permanent file for later use and modification. This is done by giving the file system the command

SAVE -S SFNAME

This command causes the contents of -S to be copied into the file named SFNAME. SFNAME is a file name of 8 characters or less in length supplied by the user. If this file does not exist, the file system will create it on the proper tape unit and then copy the contents of -S into it. If SFNAME already exists on either tape the question

#REPLACE
?

will be typed. The only affirmative response is !(CR). A lone (CR) is the proper negative response. All other responses cause the comment ILLEGAL COMMAND to be typed; the command is then ignored. If the ! is given, the contents of -S will replace the contents of SFNAME. If it becomes necessary to modify the contents of SFNAME it can be re-loaded into *ED through the use of the command

RUN *ED SFNAME

(4) Once the symbolic version of the program has been stored in the file SFNAME it can be assembled (converted into binary form). This involves using the program *ASM which is a modified version of DEC's MACRO-8 assembler. Note: it is assumed that the symbolic program stored in SFNAME conforms to the rules imposed by MACRO-8. (See Section 2.3 and the MACRO-8 manual.) The assembly procedure converts the contents of SFNAME into a binary representation that ends up in the temporary file named -B. The contents of SFNAME are unaffected. To run the assembler use the command

RUN *ASM SFNAME

Often it is necessary to partition a program into several segments because it cannot be contained within a single editor text buffer; in fact, a good rule of thumb is to allot one page of code per editor buffer.

The individual segments can effectively be combined into one large file through the use of the concatenation feature of 4K-CPS. The use of concatenation will allow up to 16 symbolic files to be assembled together as if they were one large file. For example, if four files were to be combined in this manner the resulting command would take the form

```
RUN *ASM FN1+FN2+FN3+FN4
```

Control automatically returns to the file system upon completion of assembly with the resulting binary output stored in the temporary file -B. The contents of -B should be saved in a permanent file through the use of the command

```
SAVE -B BFILENAME
```

The operation of the above command is the same as in Part 3 above.

An assembly listing will be provided if the parameter string construction P=L is used. For example,

```
RUN *ASM SFNAME P=L
```

See Section 2.3 for further details.

(5) Once the binaries required for a particular program have been generated it is necessary to put them into the form required by the 4K-CPS RUN command. The program *LOAD is used for this purpose. To invoke *LOAD use the command

```
RUN *LOAD
```

*LOAD is very similar to the file system in its operation and uses the "." as its command line prefix character. Using this program is analogous to loading paper tapes into core using DEC's binary loader. However, instead of loading programs directly into core, *LOAD loads them into an image of core, which it maintains on magnetic tape. When the loading procedure is completed, this core image is compressed into the format required by the file system RUN command loader. The core images formed in this manner end up in -B and can be saved in permanent files for later use.

The loader GET command is used to load binary files and core image files into the working core image. For example, suppose that in addition to the main program, the floating point package (FPPVB) and a tape routine (RWTAPE) are required to complete a program. These can be loaded using the single command

GET MAIN+FPPVB+RWTAPE

Up to eight files can be loaded with a single GET command. Several GET commands can be used in the course of building up a program using various standard subroutines which have been assembled at some earlier time.

When all of the needed subroutines and subprograms have been loaded, the core image must be compressed and placed into the form required by the RUN command. This involves removing all empty pages and placing a core image descriptor block at the beginning of the compressed core image to form a core image module. A starting address is also supplied at this time. These operations are invoked through the use of the simple command

BUILD SA=XXXXXX

XXXXXX is a starting address furnished by the user; if it is missing, 7777, the address of a HALT command, is used. When the BUILD command is given, the contents of -B are replaced (and not before this time) by the new core image module, and control of the computer is returned to the file system.

(6) The resulting core image module is in -B and can be saved through the use of the command

SAVE -B CFNAME

See part 3 above for an explanation of the SAVE command.

(7) The newly-produced core image module can now be loaded into core and started through the use of the command

RUN CFNAME

or

RUN -B

2. BASIC 4K-CPS

This section is devoted to describing the use of the file system, text editor, assembler and loader.

2.1 File System

The file system is responsible for all file management and is the portion of 4K-CPS with which the user most often "converses."

The file system accepts command lines typed by the user as its input and prefixes each new command line with a # . The user responds by typing a command. Line and character editing facilities are provided in order to allow the user to correct typing errors before he terminates the command line with a carriage return (CR).

2.1.1 Getting Into the File System. The following procedure is used to load and start 4K-CPS:

(1) Mount the two 4K-CPS tapes on units 0 (left) and 1 (right) as marked on the tapes. Spool 10 to 15 feet of tape onto the takeup reels.

(2) Lift the LINC LOAD switch that is on the left side of the machine. This causes the system to be loaded.

(3) 4K-CPS informs the user when it is ready to accept input by prefixing a line with a "#".

(4) 4K-CPS can be restarted at any time through the following procedure:

Press the PDP-8 STOP

Place 7600 in the right switches

Press the PDP-8 LOAD ADDRESS

Press the PDP-8 START

The system will reload if the bootstrap program is present. If it does not, lift the LOAD switch.

4K-CPS should never be manually restarted during a "SAVE" or "DESTROY" operation. This can cause the index to differ from the actual tape content.

2.1.2 Input Line Editing. Commands are entered into 4K-CPS by typing on the Teletype keyboard. A command line is not accepted until it is terminated by a carriage return. In order to aid the user in recovering from typing errors the following special editing characters are provided.

RUBOUT	Deletes the last character (if any) from the current line and types a back slash (\) to record this action.
BACKARROW	Deletes the entire line and starts a new one.
LINEFEED	Echos the current line. The echo uses an "=" for its prefix. This feature is useful if a number of rubouts have been made. Additional input may still be appended.
(CTRL-P)	Causes the line editor to accept the next character (except for leader-trailer). This is useful for placing a BACKARROW in a parameter string.
CARRIAGE RETURN	Terminates the current line.
MINUS	When immediately followed by a carriage return causes the current command line to be extended to the next line. The - is not inserted in the buffer in this case.

A single command entry cannot contain more than 127 characters and spaces although it may extend over many typed lines. Also, when a command is echoed through the use of the LINEFEED, up to 64 characters are typed on the first line with the remainder being placed on the next line.

2.1.3 Prefix Characters. In 4K-CPS extensive use is made of prefix characters typed at the start of every line. The characters used by the file system are:

- # start of a command line
- = start of a command line echo
- ? request for information

2.1.4 Commands. Commands are entered using one of the following structures. (Other structures are available and are described in detail in Section 3.4; the structures presented below represent the standard ones.)

- (1) COMMAND
- (2) COMMAND FNAME
- (3) COMMAND FNAME1+FNAME2+...+FNAMEN
- (4) COMMAND FNAME FNAME1
- (5) COMMAND FNAME FNAME1+...+FNAMEN P=...

Spaces in front of a file name are ignored but extraneous spaces after a file name and in front of a "+" change the character of the command.

The 11 commands implemented in 4K-CPS are described below. Only the first two letters are used by the system to determine the requested command. File names may be any length, however, only the first eight letters are used and retained in the index. File names are terminated by pluses (within a string) or by blanks (at the end of a string). Files are automatically generated through the use of the "SAVE" command.

The following characters should not be used in making up file names:

- CR (carriage return)
- LF (line feed)
- leader-trailer
- @
- +
- =
- non-printing characters

The file system recognizes the following commands:

COMMENT

This command line is ignored.

SAVE FROMFILE TOFILE [!]

The contents of FROMFILE are copied into TOFILE. If TOFILE does not exist, then the required file is created on the appropriate tape (binary and core images on unit 1, symbolic on unit 0). The proper index entry is also made.

If TOFILE exists and the ! is supplied, the copy operation will proceed. If TOFILE is too small or on the wrong unit, it will automatically be destroyed and a new file with the same name will be created. The only exceptions to this are -S and -B. These cannot be destroyed but may be enlarged through the use of the]SET command. If TOFILE exists and is larger than necessary, the extra space will not be returned to the system.

If TOFILE exists and the ! is not given, the question

#REPLACE
?

will be typed. The only affirmative response is !(CR). All other responses are negative. Responses other than !(CR) or just (CR) cause the comment "ILLEGAL COMMAND" to be typed. The command is then ignored.

DESTROY FILE1+FILE2+...+FILE8

The files FILE1 through FILE8 will be destroyed. A maximum of eight files can be destroyed with a single command. It is not possible to destroy the file names -B and -S. File names starting with an * cannot be directly destroyed. (These are system files and so are protected.) Such files must be renamed before they can be destroyed.

Destruction is most rapid if the list starts at the end of the index list and works its way upward. This is not a necessity.

INDEX [A] [B] [S] [D] [0] [1]

This command types out the index information specified by the trailing parameters. The parameter list should be separated from the command by one blank and is terminated by a blank or carriage return. If no parameter string is present, BS is assumed.

Option

A	List all files including those starting with an *
B	List binary (non-system) files; i. e., unit 1
S	List symbolic (non-system) files, i. e., unit 0
D	Type the associated tape location data
0	same as S
1	same as B

Typing (CTRL-C) will terminate an index listing if the user becomes impatient.

FIND FILE1+...+FILE17

This command is used to verify the presence of a file name in an index without having to type out the entire index. The detailed index parameters are typed out. If a file name is not present, a line is not typed for it.

RENAME OLDNAME NEWNAME

All files except -B and -S can be renamed through the use of this command. Files starting with an * can be destroyed if they are renamed.

RUN COREIMAGE FNAME1+...+FNAME16 P=...

This is the standard construct for the RUN command. Advanced users should see Section 3.4 for an explanation of other possible constructs. The RUN command is used to load user and system programs into core and to start them.

This command loads the specified core image and provides it with access to the designated files. The P= construct allows the user to transmit to the core image the characters immediately following the equal sign. The maximum number of characters which can be supplied is $10 + 32 - 2 * (\# \text{ file names present})$. This provides for a minimum of ten characters. Additional characters are ignored.

SIGNOFF

This command shuts 4K-CPS down gracefully and in the process rewinds the tapes and copies the indexes into "safe" areas at the front of the tapes. (This is useful in restoring wiped out tapes.)

The SET IDENTIFY and NUMBER commands are for use by the experienced user and are described in Section 3.6.

2.2 Symbolic Text Editor

The file named *ED contains a modified version of the symbolic editor (Digital-8-1-S, also called DEC-08-ESAA) as supplied by DEC. This program allows the user to type symbolic text into the PDP-8 and to perform line editing operations. This section contains a brief summary of the available commands and indicates their affect on the stored text. For further information consult the appropriate DEC manual.

The editor is invoked from 4K-CPS by the command

```
RUN *ED [ FNAME ]
```

where FNAME consists of a concatenated string of file names which have been assigned to files containing symbolic text.

2.2.1 Operating Modes. Two modes of operation are available, the command mode and the text mode.

Command Mode:

In this mode all characters typed on the keyboard are interpreted as commands to the editor. When the editor is in the command mode, it prefixes all lines with a ":" . The editor is started in the command mode.

Commands must be entered in one of the following forms:

X

NX

N, MX

where "X" represents a general command; N and M represent decimal numbers specifying lines in the editor text buffer. All commands are terminated by a carriage return (CR).

Text Mode:

This mode can only be entered by an appropriate command. When in the text mode, the editor prefixes lines with a "%". In this mode, the editor accepts all characters [except for (CTRL-C), LINEFEED, LEADERTRAILER, BACKARROW and RUBOUT] and places them into the text buffer in accord with the immediately preceding command. When the *ED text buffer becomes full, the control mode is immediately (and automatically) re-entered.

2.2.2 Commands. The commands marked with an asterisk (*) leave the editor in the text mode. To return to the command mode the user must type a (CTRL-C) (no CR required). Commands without an "*" remain in the command mode. The prefix character typed by the editor is a ready guide in determining the editor mode. All commands should be terminated by a CR.

Input:

- | | |
|-----|--|
| R * | Read incoming text from the reader. Append to the text buffer. When a (CTRL-C) is encountered, operation is resumed in the command mode. |
| A * | Append the incoming text to the text buffer. (CTRL-C) immediately returns the editor to the command mode. |

Editing:

- | | |
|-------|------------------------------|
| L | List the entire text buffer. |
| NL | List line N. |
| N, ML | List lines N through M. |

NC * Change line N (decimal). (CTRL-C) returns the editor back to the command mode. This instruction is equivalent to deleting line N and inserting in its place the lines of text which follow until a (CTRL-C) is entered.

N, MC * Change lines N through M. It is not necessary to furnish the same number of lines to the editor as were changed.

NI * Insert the following lines in front of line N.

ND Delete line N.

N, MD Delete lines N through M.

Output:

P Punch the entire buffer. This command halts the machine first to allow the user to turn on the paper tape punch. Press "CONT" to start the punching.

NP Punch line N.

N, MP Punch lines N through M.

F Punch LEADERTRAILER code and a (CTRL-C). This command halts the machine before and after operation. Pressing "CONT" restarts the machine.

Special:

E Exit immediately to CPS without modifying -S.

S Store the current text buffer in the temporary file -S and return to 4K-CPS

T Compress the text buffer and type out the approximate number of decimal locations available for storing text. Text is stored 2 characters to a location.

2.2.3 Special Features.

Special Characters:

BACKARROW	Cancel the present line. (BACKARROW = SHIFT-0)
CTRL-TAB	Tab six spaces. This is suppressed on input and output if right switch 1 is up.
/	Decimal value of the last line present.
.	Decimal value of the current line.
=	Type decimal value of preceding symbol.
- and +	Allow line specification relative to line (.) or (/).

Errors:

If illegal commands or meaningless arguments are entered, the editor responds with a ? .

The editor may be halted at any time during a listing by pressing the PDP-8 "STOP" switch. Restarting the editor at location 177 will preserve the text in the buffer.

Line Editing:

The following capabilities are available in the text mode only:

(1) Typing RUBOUT deletes the last character on a line and echoes a back slash (\) for each deletion. When all of the characters on a line have been deleted, back slashes are no longer echoed.

(2) Typing a line feed echoes the current line. Text may be appended.

(3) The read option (R) does not halt the machine when a (CTRL-C) is entered. However the command mode is re-entered.

(4) A (CTRL-P) serves as a literal next character. This may be used to enter the back-arrow (←) into editor text. Avoid using it for anything else!

(5) The back-arrow still restarts the current line when it does not follow a (CTRL-P).

(6) A (CTRL-C) will return the user to the command mode whenever it is typed.

2.3 Assembler

The assembler supplied in the file *ASM is a cut-down version of DEC's MACRO-8 assembler. *ASM is used to convert symbolic text stored in 4K-CPS files into the binary form acceptable to the loader program *LOAD. The binary output of *ASM is placed in the temporary file named -B.

*ASM differs from MACRO-8 in that the following operations have been removed:

- (1) FLTG pseudo-op
- (2) DUBL pseudo-op
- (3) PAUSE pseudo-op
- (4) The MACRO processor

These operations were removed in order to allow room for the support routines required to interface *ASM with 4K-CPS. The symbol table capability of *ASM is the same as for MACRO-8 with all of the above features present. The symbol table supplied with *ASM is found in Appendix B.

2.3.1 Running *ASM. The assembler is invoked by giving the following command to the file system:

```
RUN *ASM F1+...+F16 P=...
```

It is assumed that the files F1 through F16 contain a symbolic program written using the standard MACRO-8 conventions. A maximum of 16 files can be assembled as one program (a restriction imposed by the file system--however, this many files usually causes the symbol table to overflow anyway). The last file must contain a \$ in order to terminate the assembly.

If the P= construct is not used, passes 1 and 2 of the assembler will automatically be provided. The temporary file -B will contain the resulting binary output. If no error messages are typed out, then the

binary can be assumed to be "good." Error messages should be taken as being fatal.

Pass and listing control can be exercised by the user through the parameter string. *ASM recognizes the following control characters in the parameter string (all others are ignored):

- | | |
|----|--|
| B | Generate a binary and place it in -B, i. e., pass 2. |
| NB | Skip pass 2. |
| L | Generate a listing on the Teletype, i. e., pass 3. |
| S | List the symbol table. |
| NS | Do not list the symbol table. |

Not typing P=... is equivalent to typing P=BNLNS. Pass 1 is always provided.

Errors encountered in the assembly do not terminate the assembly and all passes requested are supplied. Exceptions to this consist of internal table overflows which immediately return the user to 4K-CPS.

2.3.2 Listing Control. Listing control for assembly listings has been incorporated in the I/O support code for *ASM. Pagination and print control are exercised through the inclusion of listing control lines within the input text stream.

Listing control lines are defined as lines whose first character is either an (ALT-MODE) or (CTRL-H). The remaining characters are used to determine the desired control action. Listing control lines are not passed to the assemblers. The valid control characters are:

- | | |
|----|--|
| E | Start the next line on a new page. |
| P | Turn the print operation on (normally on). |
| NP | Turn the print off. |

The following structure has been found to provide "nice-looking" listings.

*xxx / the first line starts on a new sheet
/

} text

PAGE / dumps the literal pool on the current sheet
(ALT-MODE)E

*xyz / now on a new sheet
/

} text

PAGE / dump literal pool
(ALT-MODE)E

.
. .
.

2.3.3 Reference Material. The following information has been taken from DEC's "Introduction to Programming." For further information on the MACRO-8 language consult the DEC MACRO-8 manual.

*ASM SYMBOLIC ASSEMBLER

The *ASM Symbolic Assembler is a service program used to translate symbolic programs that are written in the MACRO-8 symbolic language into binary programs. The MACRO-8 Language as found in 4K-CPS can be generally considered as PAL III with the following additional features:

Operators--Symbols and integers may be combined with a number of operators.

Literals--Symbolic or integer literals (constants) are automatically assigned.

Text Facility--There are text facilities for single characters and blocks of text.

Link Generation--Links are automatically generated for off-page references.

LITERALS. Since the symbolic expressions which appear in the address part of an instruction usually refer to the address of locations containing the quantities being operated upon, the programmer must explicitly reserve the locations holding his constants. The MACRO-8 programming language provides a means for using a constant directly. Suppose, for example, that the programmer has an index which is incremented by two. One way of coding this operation would be as follows.

```

      . . .
      CLA
      TAD INDEX
      TAD C2
      DCA INDEX
      . . .
C2,   2

```

Using a literal, this would become

```

      . . .
      CLA
      TAD INDEX
      TAD (2)
      DCA INDEX
      . . .

```

The left parenthesis is a signal to the assembler that the expression following is to be evaluated and assigned a location in the constants table of the current page. This is the same table in which the indirect address linkages are stored. In the above example, the quantity 2 is stored in a location in a list beginning at the top of the memory page (page address 177), and the instruction in which it appears is encoded with an address referring to that location. A literal is assigned to storage the first time it is encountered, and subsequent references will be to the same location.

If the programmer wishes to assign literals to page 0 rather than the current page, he may use square brackets, [and], in place of the parentheses. However, in both cases, the right or closing member may be omitted, as in the example below.

Literals may be nested. For example:

```
*200
TAD (TAD (30
```

will generate

```
0200      1376
. . .      . . .
0376      1377
0377      0030
```

This type of nesting may be carried to as many levels as desired.

PSEUDO-OPS. *ASM has available a number of useful pseudo-ops, which are listed and briefly explained below:

- PAGE** When used without an argument, the current location counter is reset to the first location on the next succeeding page. With an argument (**PAGE n**), the current location counter is reset to the first location on the specified page, page *n*.
- DECIMAL** When this pseudo-op occurs, all integers encountered in subsequent coding will be taken as decimal until the occurrence of **OCTAL**,
- OCTAL** which will reset the radix to its original base.
- EXPUNGE** When used, the entire permanent symbol table (excluding pseudo-ops) is erased. This pseudo-op is used when the programmer wishes to provide more core for the user program symbols. Then, with **FIXTAB**,
- FIXTAB** the programmer can build a customized permanent symbol table, containing only those permanent symbols required for his user program.

The preceding pseudo-ops and a few others are described in detail in the MACRO-8 Assembler manual.

OFF-PAGE REFERENCING. During assembly, the page bits of the address field are compared with the page bits of the current location counter. If the page bits of the address field are nonzero and do not equal the page bits of the current location counter, an off-page reference is being attempted. If the reference is to an address not on the page where the instruction will be located, the assembler will set the indirect bit (bit 3) and an indirect address linkage will be automatically generated on the current memory page.

Although the assembler will recognize and automatically generate an indirect address linkage when necessary, the programmer may still indicate an explicit indirect address using the special symbol "I" between the operation code and the address field.

ERROR MESSAGES. The assembler is constantly checking for assembly errors, and when one is detected, an error message is printed on the teleprinter. Error messages are printed in the following format.

xx yyyyyy

where xx is a two-letter code which specifies one of the type of errors listed below, and yyyyyy is either the absolute octal address where the error occurred or the address of the error relative to the last symbolic tag (if there was one) on the current page.

Following is a descriptive list of the error messages, some are printed out during Pass 1, others during Pass 2, and some of which are printed out during both passes.

<u>Error Code</u>	<u>Meaning</u>
BE	MACRO-8 internal tables have overlapped
IC	Illegal character
ID	Illegal redefinition of a symbol
IE	Illegal equal sign
II	Illegal indirect address
ND	No \$
PE	Current, nonzero page exceeded
SE	Symbol table exceeded
US	Undefined symbol
ZE	Page zero exceeded

For a thorough description of MACRO-8, see MACRO-8 Assembler, Order No. DEC-08-CMAA-D.

2.4 Loader

The loader stored in the file *LOAD is used to combine binary and core image files into new core image files which can be loaded into core and started by the file system RUN command.

Instead of loading into core as does the paper tape binary loader, *LOAD places the contents of designated files into an image of core which it maintains on magnetic tape. In this core image one tape block (128 words long) corresponds to one page of memory and 32 (decimal) tape blocks correspond to one memory field. Capacity for up through four memory fields is provided.

When loading *ASM output (absolute binary files), the loader addresses the core image in the same manner as the paper tape binary loader addresses memory. (In fact, it is essentially the same loader.) Each computer word content is placed into the specified location of the core image without disturbing the contents of any other location. However, when loading a core image file the data transfer consists of copying entire pages from the designated file directly into the core image; the previous contents of the corresponding pages in the core image are lost.

It is possible to patch existing programs by loading their core images first and then overlaying them with the desired patches which are stored in files produced by *ASM.

When a value is stored in any location of a particular memory page, this page is said to be "referenced." Referenced pages contain a core constant in all locations not stored into. This core constant is 7402 (PDP-8 halt).

When all of the desired binary and core image files have been loaded, the loading procedure is terminated and the core image module is produced by the use of the BUILD command.

The desired core image module is formed by deleting all unreferenced pages from the core image and placing a core image descriptor block at the front of the resultant compressed core image. The core image descriptor block provides the file system RUN loader with sufficient information to allow reconstruction of the original uncompressed core image. In no case will a core image module contain the top page in memory field 0. The system boot and system communications area reside in this page and the loader protects it.

The final core image module is left in -B (-B is not altered until the BUILD command is given) and may be saved in a permanent file.

The 4K-CPS loader is designed to load into as much as 16-K of memory. It is up to the user to insure that all addresses and origins are within the capacity of his own particular machine.

2.4.1 Running *LOAD. The loader is loaded into core and started by the following file system command

RUN *LOAD

2.4.2 Operation. The operation of *LOAD is very similar to that of the file system. The input to *LOAD consists of command lines typed by the user. The same line editing facilities are available in *LOAD as are available in the file system (Section 2.1.2). The loader uses the "." as its prefix character.

2.4.3 Commands. The loader recognizes the following commands (these may be abbreviated to two characters):

GET FILE1+...+FILE8

The contents of files FILE1 through FILE8 are placed into the core image. A maximum of eight files can be loaded with a single GET command. Files are loaded in the order in which they are named. Several GET commands can be used during the formation of a core image module. Only valid core images or *ASM produced files resident on tape unit 1 can be loaded.

In many cases the user may have page relocatable subroutines which he may wish to load into a page other than that for which the subroutine was originally assembled. This type of relocation can be accomplished using the construct, e.g.,

GET TAPEROUT@7000

This command causes TAPEROUT to be loaded into the core image starting at location 7000. All origins contained in the file TAPEROUT (assuming it to an absolute binary file) are taken modulo-200 and are added to the relocation constant, in this case 7000. When core images are relocated this way, each segment is relocated starting at location 7000, thus only single segment core images should be relocated using this facility. Any number of files named in a GET command may be relocated in this fashion.

REPLACE LOCN CONT CONT CONT ...

This command is used to modify the contents of individual locations in the core image from the keyboard.

LOCN is the first location in the core image to be modified.

CONT means the octal numbers to be placed in successive locations starting with location LOCN.

Replacement proceeds until all values have been placed or until an invalid value is encountered.

DUMP { LOCN }
 START END

This command causes the contents of LOCN, or the contents of locations START through END (inclusive), to be typed out on the Teletype. Typing a (CTRL-C) will terminate a dump for the impatient user.

INDEX

This command causes a list of all loadable files contained on tape unit 1 to be typed out. Each name is followed either by an A or a C depending upon whether the file contains absolute binary (*ASM output) or a core image (*LOAD output). Typing (CTRL-C) causes the typing to be terminated.

UNREFERENCE LOC1 LOC2 LOC3...

This command causes the pages containing the designated locations to be erased from the core image under construction.

EXIT

This command causes the 4K-CPS file system to be restarted with the contents of -B unaltered.

MAP

The referenced sections of PDP-8 memory are listed on the Teletype. Each line represents a segment in the present core image. A maximum of 42 (decimal) segments can be incorporated into one core image module.

BUILD [[SA=] LOCN]

This command causes the loader to form a core image module from the current core image and place it into -B. LOCN is an optionally specified starting address supplied by the user. If no starting address is specified, value of 7777 (which is a HLT in the system boot) is used. The resulting core image module is placed in -B and is in the proper format for being loaded directly into core using the file system RUN command.

3. ADVANCED 4K-CPS

This section is intended for the programmer who wants to interface additional system program modules to the 4K-CPS file system. It is not the intention of this section to provide detailed documentation of the internal structure of 4K-CPS. The program listings and Appendix C should be consulted for this purpose.

3.1 Tape Organization

The LINC tapes used for 4K-CPS are marked using 128 (decimal) word blocks and are written using a checksum which is the arithmetic sum of the words in the blocks. Each tape contains 1024 (decimal) blocks.

Unit 0 Structure

<u>Block Numbers</u> (octal)	<u>Contents</u>
0	cold start loader
1-6	"safe" index copy for unit 0
7	sign off boot
10-307	scratch area
310	system boot
311	system loader
312-333	file system
334-341	unit 0 index
342-377	reserved for system expansion
400-1777	file storage

Unit 1 Structure

0	load error routine
1-6	"safe" index copy for unit 1
7	spare
10-217	scratch area
220-225	unit 1 index
226	RUN command loader
227-246	*LOAD
247-266	*ED
267-326	*ASM
327-1777	file storage

3.2 Structure of 4K-CPS File Indexes

INDEX HEADER

The first six words of each index describe the current state of the index and the file tape.

<u>Word #</u>	<u>Contents</u>
0	2's complement of the number of index entries
1	next free block on file tape
2	space remaining on tape (in blocks)
3	tape identification number
4	unused
5	unused

INDEX ENTRIES

The total index size is six pages giving room for 127 entries. The first file entry in each index is protected and cannot be renamed or destroyed. Each index entry is structured in the following manner:

first word	first two characters in file name, packed using stripped ASCII
second word	third and fourth characters in name
third word	fifth and sixth characters of name
fourth word	seventh and eighth characters of name
fifth word	4000*unit file is on + starting block number
sixth word	1000*file descriptor + file size in blocks

For file names shorter than eight characters in length, trailing blanks are supplied by the file system.

File descriptors currently used are:

0	packed ASCII - using *ED packing conventions
1	absolute assembler output - *ASM output packed 3 paper tape frames per two 12 bit words
2	reserved for SABR output if ever placed in 4K-CPS

3	core image loadable by the RUN command loader
4	straight ASCII packed 3 frames per two 12 bit words
5	FOCAL
6	unassigned
7	unassigned

3.3 Key for Updating the Free Names, -S and -B

In order to alter the attributes of -S and -B the following procedure is used:

A "key" word whose structure is described below must be placed in location 7715 and its 1's complement must be placed in location 7724 before reloading the system. This informs the 4K-CPS file system that an update is desired. The file system then takes the new attributes for -S (if it is to be updated) from locations 7725 and 7726 and the new attributes for -B (if it is to be updated) from locations 7727 and 7730 and places them in the appropriate indexes without modification.

The contents of the attribute words are precisely those parameters used in the actual index, $4000*UNIT+BLN$ and $1000*FILE\ DESCRIPTION+SIZE\ IN\ BLOCKS$.

The "key" constant is

$$0250 \begin{cases} +1 & \text{if the 4K-CPS identification header is to be typed} \\ +4000 & \text{if -S is to be updated} \\ +2000 & \text{if -B is to be updated} \end{cases}$$

or any combination.

3.4 File System Command Line Structure

In its most general form, the 4K-CPS input command line consists of a command verb followed by three groups or strings of file names. These groups are ordered by the numbers 0, 1 and 2. These numbers may in fact explicitly appear before the file name groups:

COMMAND 0=NAMESTRING0 1=NAMESTRING1 2=NAMESTRING2 P=...

For example

RUN *ASM F1+F2 P=L

is equivalent to

```
RUN 0=*ASM 1=F1+F2 P=L
```

and

```
DES QA+DE
```

is equivalent to

```
DES 0=QA+DE
```

The only restriction in this scheme is that the groups must be assigned in order; i. e., 0 must precede 1, etc.

The meaning and utilization of the various file strings are dependent upon the specified command. From the command descriptions given in Section 2.4.1, it should be apparent which commands use which group numbers.

Further exposition on the structure of the RUN command is in order at this point. The action of the RUN command is to put into execution the program stored in the file assigned to group 0. This command also passes to the program a string of file addresses corresponding to the file names specified in groups 1 and 2. (This is done via the communication area, Section 3.5.) Routines in the specified program can then access these user specified files. Thus specifying file names in groups 1 and 2 in a RUN command, in a sense, sets up a logical I/O interface between the program and the file system. It is for this reason that it is often said that the files in group 1 are "attached to Logical I/O Device 1," etc.

At the present time no program uses a device number greater than 1. However, at some future time it may become necessary to use device 2 for assigning output files or for establishing explicit program overlays.

3.5 Communication Area Structure

When a core image module is loaded into core and started by the file system RUN command, the following information is available:

<u>Name</u>	<u>Location</u>	<u>Function</u>
STRBLN	7723	The location of the descriptor block for the current core image in core. This is in the form $4000*UNIT+BLOCK\ NUMBER$.
BLN	7711	The location of the next tape block immediately after the last one loaded. This is in the form $0*UNIT+BLOCK\ NUMBER$.
NUMD1	7721	The number of files assigned to device 1.
NUMD2	7722	The number of files assigned to device 2.
FILEADR	7725	The start of the file address chain. The following information is stored sequentially in the same order as named in the invoking RUN command. The device 1 assignments are immediately followed by the device 2 assignments. The file pointers are in the form $4000*UNIT+BLOCK\ NUMBER$ $1000*ATTRIBUTE+SIZE$. . . A combined total of 16 (decimal) files can be assigned to devices 1 and 2.
PARLIST		The parameter list starts immediately after the file string. Thus PARLIST starts at location $FILEADR+2*NUMD1+2*NUMD2$. A minimum of ten locations is available if all 16 available file pointers are used.

The system boot tape routine shares the top page of memory of field 0 with the communication area.

It should be pointed out that the updating of -S and -B alters the contents of the input file string starting in location FILEADR. This can cause problems in multi-pass programs such as *ASM

3.6 Special Commands

The following commands have been included in 4K-CPS in order to aid in the system development and maintenance.

]SET FNAME

The next two lines are input using a 4-digit octal input routine (NOT the normal input line editor). The first line contains $4000 * \text{UNIT} + \text{BLOCK NUMBER}$ and the second line $1000 * \text{ATTRIBUTE} + \text{SIZE}$. NO editing is available on these numbers other than the back-arrow (\leftarrow) which deletes the entry (setting it to 0) and allows input of another number. Any other non-octal character terminates the line.

This command replaces the descriptors of the designated file with those entered from the keyboard. This command is primarily intended to allow changing the attributes of -B and -S from the keyboard. Do not attempt to use this command to alter the location of the last used block on either tape. Use at your own risk.

]IDENTIFY

This command causes the first six locations in each index to be typed. See Section 3.2 for a description of these locations.

]NUMBER

The next line is read in using the 4-digit octal input routine described along with the]SET command. This value is placed in the tape user identification number location of both indexes. CEL uses $4000 + \text{TAPE NUMBER}$ to identify its 4K-CPS tapes.

3.7 4K-CPS Data Structures

This section describes the data structures used in symbolic text files, absolute binary files and core image files.

3.7.1 Symbolic Text Files. Symbolic text files are line oriented and use the 6-bit compressed format followed in DEC's symbolic text editor. Each ASCII character is coded into either one or two groups of 6 bits. Each 6-bit group is put into the file immediately after the last group, and groups are packed two per computer word. The only exception is the line terminator, the carriage return. By convention each line starts at the left-end of a computer word.

The following encoding scheme is used:

ASCII	6-BIT
$\left. \begin{array}{l} 240-276 \\ 300-337 \end{array} \right\}$	the lower 6 bits in the ASCII representation
$\left. \begin{array}{l} 200-237 \\ 340-376 \\ 277 \end{array} \right\}$	$\left\{ \begin{array}{l} \text{a 77 group followed by the lower 6 bits in} \\ \text{the ASCII representation} \end{array} \right.$

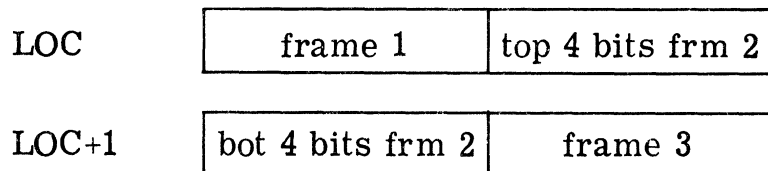
The RUBOUT character is not included in the above scheme.

For example, the two lines I(CR) DO(CR) would be encoded as

1177
1500
0417
7715

Symbolic text files are terminated by the occurrence of the (CTRL-C) character. (*ED automatically provides this character on its own line.)

3.7.2 Absolute Binary Files. Absolute binary files contain binary paper tape frames packed three 8-bit frames per two computer words.



Absolute binary files start with a 200 frame (leader-trailer) and terminate with a 200 frame.

3.7.3 Core Image Files. Core image files must be structured in the following manner in order to be properly loaded by the file system RUN command.

The first block of the core image contains a set of descriptors which describe the following core image segments. Three words are used to describe each segment except for the last segment which is described by five words. The regular 3-word descriptor is structured as follows:

word 1	The number of tape blocks contained in the segment. This number should be less than or equal to 40 (octal) except for field 0 where it must be less than or equal to 37 (octal). Bit 0 of this word is set equal to 1 to indicate when the last segment in the current core image module is being described.
word 2	Bits 6 through 8 of this word contain the memory field into which the segment being described is to be loaded. All other bits in the word must be zero since this entry is added to a CDF instruction.
word 3	The full 12-bit address within the designated field where the described segment is to be placed.

The fourth word in the descriptor for the last segment contains the field in which the program is to be started. The format of word two is used. The fifth and final word contains the full 12-bit starting address in the designated field.

Memory fields can be loaded in any order with the exception of field 0 which must be the last field loaded. Loading should proceed from the bottom of a memory field to the top. (This is a must for field 0.)

4. SYSTEM DOCUMENTATION

The Master System Development tapes contain the symbolics for the entire 4K-CPS system. Instructions are given below for assembling and updating the system. Appending a P=L to the assembler commands will provide the user with a complete set of listings. Appendix C contains additional details on the internal structure of the file system, *LOAD, *ASM and *ED.

4.1 Bootstraps

There are four programs which are termed bootstraps. The symbolic versions are found in files:

SCLD
SBOOT
SBOOT2
TAPEMIX

4.1.1 SCLD. SCLD is the cold start boot routine located in block 0 of unit 0. This routine is loaded by lifting the LINC LOAD switch, and reads in the system boot SBOOT.

```
RUN *ASM SCLD
RUN *LOAD
GET -B
BUILD
RUN *TCOPY (copy 1 block from block 11 of
            unit 1 to block 0 of unit 0)
```

4.1.2 SBOOT. This boot is the resident system boot and is located in the top page of memory field 0. Its starting address is 7600 and its purpose is to read in the main system loading routine SBOOT2. SBOOT does not provide checksum protection and is only used to read in the SBOOT2 or the page starting at 7400 in memory field 0, if required, during a RUN command. The SBOOT tape search and wait routines are used by *ED and *ASM in their tape operations. This routine is stored in block 310 of unit 0.

```
RUN *ASM SBOOT
RUN *LOAD
GET -B
BUILD
RUN *TCOPY (copy 1 block from block 11 of
            unit 1 to block 310 of unit 0)
```

4.1.3 SBOOT2. This boot loads in the file system and the unit 0 index. It is stored in block 311 of unit 0 and is read into the page starting at location 7200 by SBOOT.

It should be noted that the tape routines in these boots are arranged so that the tape operation involved in loading the system into core, started by lifting the LOAD switch, is one smooth continuous forward motion.

The unit 1 index is loaded by the system and overlays SBOOT2.

```
RUN *ASM SBOOT2
RUN *LOAD
GET -B
BUILD
RUN *TCOPY (copy 1 block from block 11 of
            unit 1 to block 311 of unit 0)
```

4.1.4 TAPEMIX. This routine is not really a boot but an anti-boot. It is called into action only if the tapes are mounted on the wrong units. This fact is pointed out to the user by TAPEMIX.

```
RUN *ASM TAPEMIX
RUN *LOAD
GET -B
BUILD
RUN *TCOPY (copy 1 block from block 11 of
            unit 1 to block 0 of unit 1)
```

4.2 The File System

The file system is assembled in four separate parts. These parts are later combined using the loader *LOAD.

```
IOC          RUN *ASM SIOC1+SIOC2+$
            SAVE -B IOC

SPART1       RUN *ASM S00A+S06+S10+S12+S14+$
            SAVE -B SPART1

SPART2       RUN *ASM S00B+S16+S20+S22+S24+S26+S30+
            S32+S34+S36+S40+$
            SAVE -B SPART2

RUNLDR       RUN *ASM S74+$
            SAVE -B RUNLDR
```

```
RUN *LOAD
GET IOC+SPART1+SPART2+RUNLDR
BUILD
```

To update the main body of the system use *TCOPY to copy 22 (octal) blocks from block 11 of unit 1 to block 312 of unit 0.

To generate a "clean" index on unit 0 (rarely done), copy 6 blocks from block 33 of unit 1 to block 334 of unit 0.

To generate a "clean" index on unit 1 (rarely done), copy 6 blocks from block 41 of unit 1 to block 220 of unit 1.

To update the RUN loader, copy 1 block from block 47 of unit 1 to block 226 of unit 1.

4.3 The Loader

The loader is an overlay onto the file system and uses many of its routines.

```
RUN *ASM L00+L06+L20+L22+L24+L26+L30+L32+L34+L36+
                                         L40+L74+$
RUN *LOAD
GET -B
UNREF 100
BUILD SA=4000
SAVE -B *LOAD
```

4.4 The Editor

The editor and its support package are assembled separately.

```
RUN *ASM E0+E1+E2+E3+E4+E5+$
SAVE -B BINED
RUN *ASM EDMA+EDMB+EDMC+$
RUN *LOAD
GET BINED+-B
BUILD SA=2000
SAVE -B *ED
```

4.5 The Assembler

The assembler and its support package are assembled separately.

```
RUN *ASM A0+A1+A2+A3+A4+A5+A6+A7+A8+A9+AEND
SAVE -B BINASM
RUN *ASM AS0+AS1+AS2+AS3+AS4+AS5+AS6+$
RUN *LOAD
GET BINASM+-B
BUILD SA=4600
SAVE -B *ASM
```

5. CONCLUSION

This report and the symbolics contained on the 4K-CPS Master System Development tapes are intended to serve as the total 4K-CPS system documentation, although this is admittedly incomplete.

The following procedure is recommended in generating and maintaining 4K-CPS tapes:

(1) Upon receipt of a set of 4K-CPS Master tapes make two copies using *MARKP8 and *TCOPY.

(2) Keep one copy as a backup to the Master set and update this copy each time the system is modified.

(3) On the other copy delete all files which are only relevant to the system. This would include all the files named in Section 4 with the possible exceptions of the files SIOC1, SIOC2, and \$. Add any useful "standard" routines. Use this set of tapes as the Distribution and Library Master tapes. (It may be desirable to have a backup copy of this set also.)

(4) Using the]NUMBER command it should be possible to keep track of the various sets of tapes and their owners.

At present there is no convenient way to transfer files between sets of 4K-CPS tapes. One approach to this problem is to use the program *ED to transfer symbolic files by loading a symbolic file into *ED from one set of tapes and then writing it out onto another set of tapes using the "S" command. Another approach is to use *TCOPY to copy the desired programs into the scratch areas of the new tapes, use the]SET command to make -S or -B correspond to the data being transferred and then save -S or -B in the normal manner. Using this method it is possible to transfer several files at one time. A general file transfer program is in development.

In addition to the material presented in the preceding sections, Appendix D contains the operating instructions for three additional utility programs and Appendix E describes the 4K-CPS I/O controller.

It is hoped that DEC's 4K-FORTRAN-D will eventually be incorporated into 4K-CPS.

APPENDIX A

The Cooley Electronics Laboratory LINC-8 computer has been modified in order to allow proper I/O processing when reading paper tapes under interrupts. This minor modification permits clearing the reader flag without advancing the reader. IOT device code 13 (which conflicts with DEC's real-time clock) is used for this purpose. The operation of the device 13 op-codes is identical to that of the standard device 03 codes except that the reader is not advanced when the reader flag is cleared.

This modification is reflected in 4K-CPS by the occurrence of a 6132 instruction in the I/O controller, IOC. A device 35 op-code (6354) is also used in IOC.

If the hardware modification described below is found to be impractical, the following procedure can be used to remove these two instructions from 4K-CPS.

- (1) Mount the 4K-CPS tapes on the proper units
- (2) Lift the LINC LOAD switch
- (3) After 4K-CPS identifies itself and the tapes stop moving, press the PDP-8 STOP switch
- (4) Toggle in and verify the following small program

LOC	CONT	
0212	7200	was 6354
0224	6032	was 6132
4200	4431	reset IOC
4201	4427	write tape
4202	0313	unit 0 block 313
4203	0001	one block
4204	0200	starting at loc 200
4205	4430	wait for tape to finish
4206	6002	IOF
4207	5610	JMP I + 1
4210	7600	reload the modified system

- (5) Place 4200 in the right switches

- (6) Press the PDP-8 LOAD ADDRESS
- (7) Press the PDP-8 START
- (8) The system will be modified and reloaded.

This fix takes care of all system programs using the I/O controller, IOC. No further modification (except to the IOC symbolics) should be necessary.

Modification to LINC-8

MB5(0) is disconnected from the keyboard select AND gate (teleprinter drawing, location ME17, Pin D). This enables the use of both a 603X and 613X since the AND gate would see a 13 code as an 03 code.

The ground is then disconnected from the DCD gate on the "READER RUN" flip flop (teleprinter drawing, location ME 21, Pin V). MB5(1) is then connected to this point. This will not allow the flip flop to be set on a 613X code.

APPENDIX B

This appendix contains the symbol table used in the CEL version of *ASM . This symbol table can be tailored to any installation's needs by making the desired modifications in the assembler symbolics and re-assembling the assembler (see Section 4.5). There is room for 251 user defined symbols in the CEL version of *ASM .

Assembler Symbol Table

AND=0000	ION=6001	IACA=6167
TAD=1000	IOF=6002	ISSP=6165
ISZ=2000	KSF=6031	IACS=6163
DCA=3000	KCC=6032	IACB=6161
JMS=4000	KRS=6034	IMBS=6155
JMP=5000	KRB=6036	ICS2=6153
NOP=7000	TSF=6041	ICS1=6151
CLA=7200	TCF=6042	INTS=6147
CLL=7100	TPC=6044	ILES=6145
CMA=7040	TLS=6046	IBAC=6143
CML=7020	FEXT=0000	ICON=6141
RAR=7010	FADD=1000	EXIT=5400
RTR=7012	FSUB=2000	CALL=4400
RAL=7004	FMPY=3000	PLS=6026
RTL=7006	FDIV=4000	PPC=6024
IAC=7001	FGET=5000	PCF=6022
SMA=7500	FPUT=6000	PSF=6021
SZA=7440	FNOR=7000	RFC=6014
SPA=7510	RDF=6214	RRB=6012
SNA=7450	RIF=6224	RSF=6011
SNL=7420	RMF=6244	
SZL=7430	RIB=6234	
SKP=7410	CDF=6201	
OSR=7404	CIF=6202	
HLT=7402	ITAC=6177	
CIA=7041	IACF=6175	
LAS=7604	IZSA=6173	
STA=7240	IAAC=6171	
STL=7120		
GLK=7204		

APPENDIX C

This appendix contains material descriptive of the internal structure of the 4K-CPS system.

Detailed 4K-CPS File System Core Map

<u>Page Location</u>	<u>Tape Location</u>	<u>Contents</u>
0000-0177	0312	Constants, variables and pointers
0200-0377	0313	I/O controller
0400-0577	0314	I/O controller
0600-0777	0315	TCOPY, CALCPY
1000-1177	0316	PLIB, DPOSIT, CRLF, RDRIN, PREF
1200-1377	0317	OBI, EXCHR, GWFLB, SPRSPA, TABTST
1400-1577	0320	MESAGE, OCTIN, OCTOUT
1600-1777	0321	ISRCH, GETW, PUTW, SYSTST, SVS, FIND command
2000-2177	0322	WVTOC, SIGN command, CCOMAR
2200-2377	0323	Messages
2400-2577	0324	DESTROY command, DELETE
2600-2777	0325	SAVE command
3000-3177	0326	SAVE command, RENAME command, TSTCTC
3200-3377	0327	INDEX command, INTYP
3400-3577	0330	RUN command, SEEK
3600-3777	0331	Main entry point (internal), TERMTST
4000-4177	0332]SET command,]NUMBER command,]IDENTIFY command, D01CHK
4200-4377	0333	System start (from boot).. Command line buffer
4400-4577	0334	Start of unit 0 index
4600-4777	0335	.
5000-5177	0336	.
5200-5377	0337	.
5400-5577	0340	.
5600-5777	0341	End of unit 0 index
6000-6177	4220	Start of unit 1 index
6200-6377	4221	.
6400-6577	4222	.
6600-6777	4223	.
7000-7177	4224	.
7200-7377	4225	End of unit 1 index
7400-7577	4226	RUN command loader
7600-7777	0310	System boot

(4000=Tape unit 1)

A brief description of each routine used in the 4K-CPS file system is given below.

CALCPY

Makes calls to TCOPY using a list of copy operations previously set up by the DESTROY and SAVE commands. When the list has been exhausted, CALCPY restores the system (TCOPY reads over most of the system) and returns to the caller.

CCOMAR

Zeros the communication area and places a HLT in location 7777. Also clears the write VTOC (index, if you prefer) switches.

CRLF

Uses IOC to type out a carriage return followed by a line feed.

DELETE

Deletes the file specified in locations W1-W4 from the appropriate index, determines the required tape operation and sets the appropriate write VTOC switch. If the specified name is not present (i. e., appears twice in a destroy string), the system is reloaded thus restoring the indexes to their original state.

DESTROY Command

Destroys up through 8 file names assigned to logical device 0 (all files must be on device 0). Absence of input file names causes the ILLEGAL COMMAND comment to be typed. DESTROY uses the DELETE routine and sets up CALCPY for the actual tape operations. Files are destroyed in the specified order as if each had been named in a separate DESTROY command (lack of room necessitated a dumb command).

DPOSIT

Places the contents of the AC into the command line input buffer. If the buffer overflows, an error message is typed and the current line is lost. This routine must be initialized.

D0CHK

Checks to see if all assigned file names are present, the ILLEGAL COMMAND comment is typed and the current command is ignored.

D01CHK

Checks to see if precisely 1 file name has been specified on logical device 0 and precisely 1 file name has been specified on logical device 1. If not, the ILLEGAL COMMAND comment is typed. Device 2 is not checked.

EXCHR

Extracts characters from the input command line buffer. This routine is initialized by OBI. Separate return points are provided for a normal return with a character in the AC and for an end of buffer return.

FIND Command

Searches the indexes (VTOCs) for the file names attached to device 0. If an entry is located, the name along with the index parameters are typed. If a name is not located, it is ignored. All file names must be attached to device 0.

GETW

Transfers the contents of W1-W4 to the four locations starting at the contents+1 of auto-index register 10.

GWFLB

Gets the first eight characters of the next word in the command line buffer. Separate return points are used for a normal return and for an end of command line return. GWFLB recognizes (space), + and = as break characters. The eight text characters are left in locations W1-W4. Trailing blanks are provided if necessary.

INDEX Command

Types out the indexes as requested by the specified parameters.

INTYP

Does the typing for the INDEX command.

ISRCH

Searches the index specified by the contents of the AC for the name contained in W1-W4. Separate return points are used depending on whether or not the specified file name is found in the specified index. ISRCH also sets up various pointers if a file name is located.

IOC. . I/O Controller

See Appendix E for a description of the 4K-CPS I/O controller.

Main Entry Point (internal)

This is the point to which all commands (except RUN) and errors eventually return. A new command line is read in, decoded, checked for gross errors and the command dispatched.

MESAGE

This is a modified version of DIGITAL 8-18-U. It is used to type messages produced using the MACRO-8 TEXT command.

OBI

Initializes the EXCHR routine.

OCTIN

Accepts a four digit octal number typed from the keyboard. Used in the]SET and]NUMBER commands. Any non-octal input except for the back-arrow terminates the input number. Back arrow restarts OCTIN.

OCTOUT

Used to type four-digit octal numbers.

PLIB

Used to input a line into the command line input buffer. Also contains the line editing support code.

PREF

Types a carriage return-line feed followed by the current prefix character.

PUTW

Transfers the contents of the four locations starting at the contents+1 of auto-index register 11 into locations W1-W4. If the first word in this string is zero, a special return is made to the caller. Otherwise the return is normal.

RENAME Command

This command allows the user to rename the file whose name is assigned to device 0 using the name assigned to device 1.

RDRIN

Used by PLIB to read characters.

RUN Command

Used to load programs. Sets up the file and parameter strings in the communication area, reads in the core image descriptor block of the specified program file and exits to the RUN Command Loader with the tapes moving forward.

RUN Command Loader

Performs the actual booting in of core image modules. Uses the system boot search and wait routines. If the last page to be loaded overlays this loader, it sets the system boot to read in this page and goes to the boot. When started, it assumes the proper core image descriptor is in core and the tapes are moving forward.

SAVE Command

Copies the file designated on device 0 into the file designated on device 1. Creates new files and allocates storage on tapes as required. Performs the actual copying through the CALCPY-TCOPY routines.

SEEK

Searches both indexes for the file name contained in W1-W4. Two return options are possible. If SEEKSW=0 and file name cannot be located, a replacement or carriage return is requested. If SEEKSW has been set to -1, then a special return is made. SEEK always resets SEEKSW to 0 upon exiting.

SIGN Command

Causes the current indexes to be copied to the front of the tapes and types out BYE. This is done in order to rewind the tapes for the user and to provide a backup copy of his indexes in case the primary versions are accidentally destroyed.

SPRSPA

Places four words containing 4040 starting at a specified location. This is a relatively worthless routine.

SVS

Used to set the write VTOC switches.

System Bootstrap

Loads in the system load tape routine from block 311 of unit 0 and starts it.

System Start (external)

The point at which the file system is started after being loaded. This code updates the index entries for -S and -B as needed and types the 4K-CPS identification. This code resides in the command line buffer and is lost after start-up.

SYSTST

Tests file names for * as the first character. This routine is invoked when it is not desired to alter files whose names start with an * .

TABTST

This is a routine which checks the value of the AC against the values stored in a specified table. The specified return point is determined by where the match (if any) occurs.

TCOPY

Performs a tape copy operation determined by the parameters stored in locations immediately following the call to TCOPY. Uses locations 1000 through 7577 as a buffer.

TERMTST

Used to check the word terminator encountered by GWFLB.

TSTCTC

Checks to see if a (CTRL-C) has been typed. Used to terminate various operations at the users "request."

WVTOC

Checks the write VTOC switches and writes out the current specified VTOCs (indexes) onto tape.

] IDENTIFY Command

Types out the first six locations in each index.

] NUMBER Command

Used to number each set of 4K-CPS tapes. Intended for use in keeping records of who has what set of tapes.

] SET Command

Used to change the index parameters for the file specified on device 0. Primarily intended to be used on -S and -B.

4K-CPS Loader

The loader, *LOAD, is an overlay onto the file system and uses several of the file system utilities.

Detailed *LOAD Core Map

<u>Page Location</u>	<u>Contents</u>
0000-0177	file system..pointers
0200-0377	file system..IOC
0400-0577	file system..IOC
0600-0777	TCOPY, PUTSEG
1000-1177	file system
1200-1377	file system
1400-1577	file system
1600-1777	file system
2000-2177	BUILD command, BLDFLD
2200-2377	GET command, BINARY
2400-2577	COREIM, READIN
2600-2777	GETFRM, GNXL0C, REFTST
3000-3177	TSTCOR, WBUF, DPIPT, OCTTST, REFER
3200-3377	INDEX command, LOCDTA, REPLACE command UNREFERENCE command, UNRF, DPOSIT

<u>Page Location</u>	<u>Contents</u>
3400-3577	DUMP command, LHEAD, TSTCTC, EXIT command
3600-3777	LDRSTR, LDR
4000-4177	Loader system entry
4200-4377	
4400-4577	
4600-4777	Buffers
5000-5177	
5200-5377	
5400-5577	
5600-5777	
6000-6177	Start of unit 1 index
6200-6377	.
6400-6577	.
6600-6777	.
7000-7177	.
7200-7377	End of unit 1 index
7400-7577	SETLOC, SETFLD, MAP command, LOCATE, PRNTADR
7600-7777	System boot

4K-CPS Assembler

The assembler furnished with 4K-CPS is a modified version of DEC's MACRO-8 assembler. The following operations have been removed:

PAUSE
 FLTG
 DUBL
 The MACRO processor

The core is used as follows:

<u>Page Locations</u>	<u>Contents</u>
0000-3577	MACRO-8 + some interface routines
3600-3777	Input routines
4000-4177	Pass control and tape routine
Pass 1	
4200-4777	Symbolic input buffer
Pass 2 (overlay)	
4200-4377	Binary output routines
4400-4577	Binary output buffer
4600-	Symbolic input buffer. . max of 4 pages. . min of 1 page. Checks bottom of symbol table
Pass 3 (overlay)	
4200-4377	Listing control routines
4400-4777	Symbolic input buffer
All Passes	
5000-7577	Available for symbol table
7600-7777	System boot

The assembler buffer management is a little sloppy in that *ASM always completely fills its symbolic input buffers. This can cause problems only if the unit 0 tape does not have the correct checksum in each block (*MARKP8 does not provide the proper checksum). Since the maximum buffer size is 4 blocks and blocks 2000 through 2007 really do exist, no problems would be encountered at the end of a tape.

4K-CPS Symbolic Text Editor

The editor supplied with 4K-CPS is an extensively modified version of DEC's editor DIGITAL 8-1-S.

The available core is used as follows:

<u>Page Locations</u>	<u>Contents</u>
0000-1377	Modified editor
1400-1777	Input-output tape routine
2000-7577	Text buffer + starter (used once at start-up)
7600-7777	System boot + tape rewinder

A special tape routine is used to load and write out the editor's text buffer. When loading the buffer, the placement of lines and line pointers is automatically done by the tape routine. Similarly, when writing the buffer back out onto tape, the tape routine traces out the scrambled line sequence, deletes the line pointers, writes tape, and provides the terminating 7703 character.

The "T" command uses the section of tape starting at block 0200 on unit 0 as a temporary buffer. This command is used to return all unused buffer space to the user (the *ED internal line management leaves something to be desired).

The "S" command writes the text buffer starting in block 0240 of unit 0 and returns to the file system with the -S update switches set properly.

APPENDIX D

This appendix contains descriptions of three 4K-CPS utility programs.

*PBIN

This program reads binary paper tapes into 4K-CPS from either the LINC ASR-33 or from any remote 8-level Teletype through the data phone. The binary paper tape is read into the binary working area and can be saved in a file by 4K-CPS.

Instructions

To load and start *PBIN use the command

RUN *PBIN

The program is self-starting and will type out

4K-CPS PAPER BINARY INPUT PROGRAM

When *PBIN is ready to start reading in the binary paper tape, the message

START READER

will be typed. This is the time to turn on the paper tape reader, not before!

The program ignores leader code (200 octal) and also throws away the first character read in. Always start reading the paper tape in the leader section.

The program terminates when 1 trailer (also 200 octal) code frame has been read. The binary working area then contains a copy of the information on the paper tape, properly formatted for the 4K-CPS loader. 4K-CPS is automatically reloaded after the comment

DONE!

is typed. Turn off the paper tape reader when this message appears.

Once back in 4K-CPS the binary can be saved using the command

SAVE -B FNAME

where FNAME is a file name chosen by the user.

***TCOPY**

*TCOPY is a general purpose magnetic tape copying utility program. It can be used to copy blocks of PDP-8 formatted LINC magnetic tapes onto another or the same tape.

Instructions

The program *TCOPY is loaded and started by the command

RUN *TCOPY

The following heading is typed

**4K-CPS TAPE COPY
CTRL-C EXITS
RUBOUT RESTARTS**

The following control options are provided to the user on the input lines.

A (CTRL-C) on any line will immediately terminate program execution and will automatically return control of the computer to 4K-CPS (provided the proper 4K-CPS tapes are mounted at the time).

A RUBOUT on any line will immediately delete all previously entered data and will cause a return to the first input request.

A back-arrow (shift-0) deletes the information entered on the current line. The request is not retyped.

Lines should be terminated by a carriage return or a (CTRL-C) or a RUBOUT.

The following is a description of the input requests on a line by line basis. These requests are typed by the program and the user must type the appropriate responses. Minimal error checking is done by this program. Be careful! All numbers used below are octal.

Request 1 FROM BLOCK =

The user should type in the tape block number on the source tape at which the transfer is to start. Inputs on this line must be less than or equal to 3777.

Request 2 FROM UNIT =

Type in the tape unit upon which the source tape is mounted. Only units 0 and 1 are valid inputs.

Request 3 AMOUNT =

Type in the octal number of blocks to be transferred. Values of 1 through 7777 are acceptable. (PDP-8 format tapes contain 2000 blocks.)

Request 4 TO BLOCK =

Type the starting block number of the sink tape to which the transfer is to be made. This number must be less than or equal to 3777.

Request 5 TO UNIT =

Type the unit number upon which the sink tape is mounted. Only units 0 and 1 are valid inputs.

At this point the program types

CR STARTS COPY

This is the last chance to back out.

*MARKP8

*MARKP8 is a tape marking program which was generated by overlaying the program MARKL8 with a program called XMARK, available from DECUS as DECUS number L-32. XMARK was itself modified so that *MARKP8 will mark tapes with 2000₈ blocks of 200₈ words apiece.

*MARKP8 is interactive and is run using the LINC display and the 33ASR keyboard. It may be invoked from 4K-CPS by the command

RUN *MARKP8

To return to 4K-CPS after marking a tape, be certain that both 4K-CPS system tapes are mounted, and then load address 7600 into the program counter and press START.

Note that 4K-CPS tapes must be marked using *MARKP8.

The tapes generated by *MARKP8 do NOT have the proper check sum. This can, and has, caused problems. To correct this, use the program *TCOPY (supplied with 4K-CPS) to either copy a good tape onto the one just marked or to copy the tape onto itself. The latter approach is somewhat slower since *TCOPY tries two times to get a good check-sum before giving up.

It is strongly urged that all newly marked tapes contain the proper checksum.

APPENDIX E

Introduction

This appendix describes the small general purpose I/O control package (IOC) used in 4K-CPS. This package is intended for use on the basic 4-K LINC-8 computer and permits operation of the low-speed reader-printer and magnetic tapes under interrupts. Although this package is a part of the 4K-CPS system, it can also be used for general programming applications. The symbolics for IOC can be found on the 4K-CPS Master Development Tapes stored in the files SIOC1 and SIOC2.

Requirements

IOC occupies locations 0-2, 20-77, and 200-577.

The CEL LINC-8 computer has been modified to allow the clearing of the reader flag without advancing the reader. This was done in order to allow the proper handling of paper tapes when running under interrupts. The required modification can be found in DECUSCOPE, Vol. 6, Nos. 5 and 6 (also Appendix A). This modification is relatively minor and results in the use of device code 13 to clear the reader flag without advancing the reader.

This modification is reflected in IOC by the use of a RCC (6132) instruction at location 0224. If it is not possible to make the required hardware modification, then location 0224 may be changed to a KCC (6032). The 6354 instruction at location 0212 can also be removed or NOP'ed (this is a special code used at CEL).

Non-Magnetic Tape Operation

IOC should be reset before any I/O operations are attempted. This procedure assures the user that all of the internal IOC flags and switches are in their proper state when I/O operations begin. It also causes the PDP-8 interrupt system to be enabled.

Resetting is done through a subroutine call

```
JMS I 31 (RESET=JMS I 31)
```

IOC only needs to be reset once at program start-up.

The teleprinter I/O routines are not buffered and are divided into two parts, interrupt time and task time. The interrupt time portions are invoked when an interrupt occurs. The proper hardware flag is cleared and the corresponding software flag is set. Control is then returned to the user at the point at which he was interrupted. The task time routines are called by the user to either fetch or type a character. The task time routines are coded the same as "normal" Teletype routines except that software flags are used in place of hardware flags.

The subroutine names used in the following illustrative program do not necessarily correspond to those used within IOC.

To read a character

```
JMS I 25 (RDRGET=JMS I 25)
```

The associated routine is equivalent to

```
RGET, 0
      KSF
      JMP .-1
      KRB
      JMP I RGET
```

To type a character (PDP-8 AC not cleared)

```
JMS I 23 (PTRPAS=JMS I 23)
```

The associated routine is equivalent to

```
PPAS, 0
      TSF
      JMP .-1
      TLS
      JMP I PPAS
```

To type a character (PDP-8 AC cleared)

```
JMS I 24 (PTRPUT=JMS I 24)
```

The associated routine is equivalent to

```
PPUT, 0
      JMS PPAS
      CLA CLL
      JMP I PPUT
```

Magnetic Tape Operation

The magnetic tape routines in IOC operate under interrupts and use LINC tapes marked with 128 word blocks (plus 1 word for checksum). IOC allows overlapping magnetic tape operations with normal Teletype I/O operations. This feature is used in 4K-CPS to allow the user to enter command lines while the unit 1 index is being read in. If swing buffers are used, it is possible to accept data continuously from the reader while writing processed input out on magnetic tape. (This is done in the program *PBIN.) This mode of operation is particularly important when input is being accepted from a Data-Phone since, in this case, the computer cannot exert reader start-stop control.

The following calling sequences are used in IOC magnetic tape operation:

To read tape

```
JMS I 26          (TREAD=JMS I 26)
4000*UNIT+BLN    UNIT=1 or 0
AMOUNT
LOCATION
```

To write tape

```
JMS I 27          (TWRITE=JMS I 27)
4000*UNIT+BLN    UNIT=0 or 1
AMOUNT
LOCATION
```

To wait for the previous tape operation to be completed

```
JMS I 32          (TCLOSE=JMS I 32)
```

If no tape operation is in progress, the return is immediate.

The IOC tape routines automatically wait for any previous tape operation to be completed before attempting to start a new operation.

Control is returned to the user immediately after the tape routines have been set up to perform the requested operation. The specified operation will not have been completed at this time but will take place behind the user's program. The status of the current tape operation can be checked by the user in two ways. Location 22 (TFLAG) is 0 if the current tape operation has been completed and -1 if it is still in execution. The TCLOSE operation may be performed to cause a wait-state to be

entered until the current operation is ended. (This amounts to going into a loop until TFLAG is 0, then exiting.) The following code could be used to return to 4K-CPS from a user program.

```
TCLOSE
IOF
JMP 7600
```

Unless otherwise specified by the user, IOC clears the tape motion bits (M0 and M1) at the end of each read-write operation. Because of timing restrictions in the LINC tape hardware, if the IOC tape routines are called and the motion bits are zero, then a 0.05 second delay is provided in order to allow the tape units to settle.

This delay can be avoided in two ways. The user can select and start the appropriate tape unit before entering IOC, or he can leave the tapes moving after each tape operation.

To select and start the desired tape unit (assuming it is not moving)

```
CLA CLL
TAD 4000*UNIT+0002    UNIT=0 or 1
ICON
CLA
TAD 4000*DIR+0001    DIR=0=FWD, DIR=1=REV
```

To leave the tapes running after an operation make location 44 (TMOTION) nonzero. This location is not reset by IOC. The user may reset location 44 to zero at any time. Once location 44 is reset to 0 the tapes will be properly stopped.

The IOC tape routine is moderately "intelligent." It remembers the last used tape block number and starts the specified tape unit moving in the direction required by the current call. However, because of space limitations, the tape routines do not remember which tape unit was used last. It is felt that the resulting occasional wrong start is better than the tapes always starting in either the forward or reverse direction.

If a checksum error is encountered in reading in a tape block, one additional attempt is made to read it properly. (This retry can cause the tapes to "hang up" searching for the block.) If the second attempt is not successful, the results of this try are left in core and the contents of location 34 (ERRCNT) are incremented by 1. (ERRCNT is cleared by IOC whenever a read or write tape operation is started.)

The following symbol definitions have been found to be useful:

TREAD=4426
TWRITE=4427
TCLOSE=4430
RESET=4431
RDRGET=4425
PTRPUT=4424
PTRPAS=4423
RFLAG=21
PFLAG=20
TFLAG=22
ERRCNT=34
TMOTION=44

APPENDIX F

F.1 Alterations

4K-CPS Alterations

December 3, 1969

The following alterations in 4K-CPS have been made:

4K-CPS (OV249): The 6132 instruction described in Appendix A has been replaced by a 6032 instruction. The 6354 instruction has been retained.

The System core map contained in Appendix C has been slightly modified.

The [] notation used in the report is IBM'esse for optional. For example, the [!] in the SAVE command means that typing the ! is optional.

Files which cannot be located by the FIND command are typed followed by a ? .

Block 7 of the unit 0 tape contains a "loader" tape routine which is read into the top page of memory when the SIGNOFF command is given. When this routine is present, starting at location 7600 is equivalent in operation to lifting the LINC load switch. This routine has been incorporated into 4K-CPS in order to make up for the lack of a load switch on the CEL PDP-8 which has a LINC tape capability.

*ASM(EC019): Two additional parameters are recognized in the invoking RUN command. These are

P: Punch the binary on the ASR-33 instead of placing it into -B.

H: Same as P but use the high speed punch.

Typing CTRL-C causes an attention interrupt and returns control back to the file system. This feature is effective during all assembler passes.

System code numbers are formed by combining the second two letters of the month with a two digit date and one digit year. Hence November 24, 1969, becomes (OV249).

4K-CPS Alterations

February 17, 1970

The following alterations in 4K-CPS have been made:

4K-CPS (EB160): Additional protection has been provided for files whose names begin with an * (system files). It is no longer possible to use the SAVE command to create or alter the contents of such files. Files whose names start with an * can only be generated by renaming an existing file using the RENAME command.

For example assume that a new version of the text editor has been produced and is in -B . The following sequence of command will cause *ED to be updated.

```
# RENAME *ED ED
# SAVE -B ED !
# RENAME ED *ED
```

ED(EB160): A timing problem has been eliminated in the magnetic tape routines in order permit use on the CEL PDP-8/L .

ASM (EC019): The high speed reader-punch OP-codes are NOT included. However some of the reader MNEMONICS have been used for CEL's device 13 low speed reader control.

MEMO TO: 4K-CPS Users
FROM: K. Metzger
SUBJECT: System Alterations
DATE: March 2, 1971

Introduction

I have gone through 4K-CPS and corrected all known errors. In addition several features have been added. Because of changes to the control program a new version of *FCOPY was necessary. (Furnished by B. Barton) The old version of *FCOPY should be replaced when your tapes are updated.

The following changes have been made:

Control Program:

1. The I/O controller tape routine has been modified so that it can be used to read into and write out of extended memory.
2. Lower case input is accepted and is used internally as if it were upper case. Thus typing in the file name "punt" is the same as the name "PUNT". All characters in the parameter string are translated to upper case before being passed to an invoked program.
3. The parameter A in the index command now defaults to units 0 and 1 if no units are specified by the user.
4. A bug in the SAVE command algorithm was fixed.
5. A CTRL-C on an input line causes the line to be treated as an empty line.

*ED:

1. The Delete and Change commands now check to insure that both line numbers specified are valid.

2. Unless overridden by a `P = L` on the invoking run command the editor translates lower case text input to upper case (echoed). Commands may be entered either in lower or upper case.
3. The halts in the editor for the `P` and `F` commands have been replaced by waits for an input. Striking any key (except `CTRL-C` and `NULL`) will cause the editor to proceed. (The key struck is not echoed.) This change should make remote operation less painful.
4. To allow for lower case input use the command

```
RUN *ED  FILE+ ...      P = L
```

The text capacity is reduced approximately by a factor of 2 when lower case input is used.

***ASM:**

1. Symbol table listings now respond to attention interrupts (`CTRL-C`)
2. Attention interrupts cause the message


```
AI SYMBOL + NUMBER
```

 to be typed. Thus an assembly is successful (hopefully) if and only if it lacks any error messages.
3. The assembler no longer halts on illegal input characters. Illegal characters cause an error message to be typed and are then ignored.
4. The `FIELD` pseudo-op now causes the following operations
 - a) The field setting is evaluated mod-8
 - b) The current page literals are dumped
 - c) The page 0 literals are dumped
 - d) The internal literal tables are reset
 - e) The desired field frame is put out

- f) The radix is reset to octal
 - g) An origin at 200 is put out
 - h) The assembly process is then re-entered.
5. The " operator now works properly.
 6. The † now denotes unsigned multiplication.

NOTE: The rule used by *ASM in evaluating expressions is to proceed from left to right. Each binary operator combines the value of the symbol immediately to its right with the result of the evaluation of the expression to its left. For example

$$2 + 3†2$$

gives the value ten, not eight! (FORTRAN programmers beware!)

7. On listings, lines containing instructions which caused *ASM to generate an off-page linkage are flagged by an * typed immediately after the location number.
8. Device 35 and device 13 op-codes have been replaced by the op-codes for the high speed reader and high speed punch.

***FCOPY:**

This routine is documented in a separate memo. The primary alteration consists of removal of the ability to transfer files between 4K-CPS and 8K-CPS.

***LOAD:**

An error which would only occur for oddly-constructed 16K word core images was fixed.

MEMO TO: 4K-CPS Users
FROM: B. Barton
SUBJECT: New *FCOPY
DATE: March 5, 1971

A new *FCOPY is necessary to run with the updated 4K-CPS system described in a memo by K. Metzger dated March 2 1971. The new version is similar to the old one except that the ability to transfer files between 4K-CPS and CPS has been removed. This was done for a good reason and anyone desiring to make such a transfer should use *FILCOPY in CPS.

When *FCOPY is run, the operator is asked to designate the type of tapes present on the two LINCTAPE units. Proper replies are:

S	symbolic (unit 0)
B	binary (unit 1)

Operation is then directed by the program and the user is requested to furnish the source file name (FROM:), the unit the source tape is presently on (UNIT:) and the sink file name (TO:) into which the designated file will be copied. The corresponding index entry will also be made by the file transfer program.

*An empty line allows the operator to either return to 4K-CPS or to restart *FCOPY. The operator should not mount tapes of different types from the last ones he specified without going through this restart procedure and specifying the correct types!*

MEMO TO: 4K-CPS Users
FROM: K. Metzger
DATE: April 22, 1971
SUBJECT: Assembler Alteration

Introduction

The assembler used in 4K-CPS has been modified in order to provide an additional error check. Copies of the new assembler (*ASM(PR211)) can be obtained from Brian Barton or myself.

Modification

In the earlier versions of *ASM the following code would not produce an error message (undefined symbol):

```
TAD A  
A = B  
B, NOP
```

This problem was compounded by the fact that A would be defined properly on pass 3 and thus the correct output would appear on the listing but not in the binary file.

Ideally the assembler should have listed the symbol A in line one as being undefined. (Because on pass 1 A was equivalenced to an undefined symbol. Interchanging lines 2 and 3 would yield a valid instruction sequence.) *ASM has been modified so that it will identify line one as containing an undefined symbol (on pass 2).

Although the above example may appear contrived this problem has occurred at least three times in much more elaborate fashion in very large programs. Since the listing used in debugging is correct, locating the use of such accidentally "undefined" symbols can be very difficult.

F.2 *TRW2

*TRW2 is a utility program for use in reading and writing selected portions of LINC tape into and out of the memory of the LINC-8 computer. The program resides in locations 07000-07577 and allows reading into and out of locations 00000 through 06777. Operation is controlled from the keyboard. The starting address of *TRW2 is 07000.

To load and start *TRW2 give the command

```
RUN *TRW2
```

The dialogue which follows is described below:

TRW2 (xxyz)	identifies the version of the program being used
R OR W	type R for read and W for write, no line terminator is used on this line
BLK =	the tape block (octal) at which the transfer is to start
UNT =	the tape unit involved, 0 or 1
AMT =	the number of tape blocks involved, must be less than 34 octal
LOC =	where in core the transfer is to start
CR COPYS	a carriage return starts the copy operation

When the indicated operation has been completed, *TRW2 halts. Pressing continue restarts. This program checks the values entered on the AMT and LOC lines to see if the requested transfer will clobber *TRW2. If it could, then *TRW2 is restarted and the request is ignored. All invalid requests cause *TRW2 to be restarted.

All numeric inputs must be made in octal and are terminated by a non-octal character. The typing of the backarrow (←) on any numeric input line deletes the number typed to that point and allows a new value to be input.

F.3 *PUNCH

The program *PUNCH is intended for use in punching out the contents of 4K-CPS symbolic files. *PUNCH is invoked from 4K-CPS by giving the command

```
RUN *PUNCH FILE1+FILE2+ ... +FILE16 P= ...
```

The characters recognized in the parameter field are:

P low speed punch

H high speed punch

C include 4K-CPS command lines (first character is
ALT-MODE or CTRL-H)

N negate the immediately following parameter

The default set is CP .

*PUNCH allows the punching of up to 16 files onto a single paper tape. Each file is separated from the previous one by 256 (decimal) frames of leader trailer code.

After being loaded the program goes into a wait loop in order to permit the user to turn on the proper punch. Typing (CTRL-C) starts the punching. Each output file is provided with 128 (decimal) frames of leader and 128 (decimal) frames of trailer.

*PUNCH only punches symbolic files. All other types are ignored and the files are skipped over. The program halts when it has punched all of the designated files. Typing (CTRL-C) returns control to the 4K-CPS file system.

F.4 *TAPMARK

F.4.1 Introduction. The *TAPMARK program was developed at Cooley Electronics Laboratory to mark magnetic tapes for use on the Linc-8 and PDP-8/L computers. The program's primary objective is to replace the MARKL8 and MARKP8 programs that are now used at CEL. A special feature of this program is that it allows manual entry of any desired block and word size format.

F.4.2 Operation. To use *TAPMARK on CPS the command is:

RUN *TAPMARK

All input lines are terminated by a carriage return. The User supplied numbers are to be in octal. An illegal entry automatically restarts the program. The program provides a line editing feature: typing a back arrow (←) erases all previously entered data on the current line. New data is entered immediately after back arrow.

Once *TAPMARK is invoked it rewinds unit 1, and types the following:

CEL *TAPMARK(AN140)

LINC, PDP-8 OR MANUAL ? TYPE =

After this the user types in one of three possible entries. All that is necessary is the first letter of the name, but the user may enter the complete word if he wishes. The three entries are:

- P- This commands the program to mark the tape in PDP-8 format. This format has 2000₈ data blocks per tape and 200₈ data words per block. The checksums for these blocks are computed as the arithmetic sum of the words in each block.
- L- On this command the program marks the tape in LINC format. This format has 1000₈ data blocks of 400₈ words each. The checksum for these blocks are computed using the two's complement of the arithmetic sum of the words in each block.
- M- This command signifies manual specification block and word size is desired. The computer then types two questions on the teletype. These

questions are:

NUMB OF BLKS =

NUMB OF WORDS =

After each question the user types in the desired number of blocks or words in octal. The maximum number of blocks or words the program will accept without restarting is 3777_8 . The word and block length should be geared to the tape length (see appendix). The minimum number of words that can be written is 15_8 because of the simultaneous writing of the first 14_8 data mark and backward block marks. The minimum number of data blocks that can be written is 20_8 because of the program's rewind routines.

The number of blocks and words having been entered the computer then types on the teletype the following message:

HIT CTRL-C TO GO ;

ANY OTHER REPEATS QUESTIONS.

This is a check point intended to allow the user to see if his tape format has been entered correctly. If the format is correct, he should strike (CTRL-C) to initiate the marking of the tape. If the format is incorrect, he should strike any other character on the keyboard and the above two questions will be repeated. Manual format uses PDP-8 checksum computation.

When the above sequence of operations are completed, the commands to MOUNT VIRGIN TAPE ON UNIT 1 and LIFT MARK are printed on the teletype. In the case of manual format, the mount virgin tape command appears before entry of any block or word sizes to prevent accidentally destroying the unit 1 CPS tape.

F.4.3 Error Responses. *TAPMARK has three error diagnostics:

BAD TAPE NO MARK - The tape was not marked.

BKWD BLOCK NO. WRONG - Backward block numbers are either not marked or are incorrectly marked.

CHECKSUM FAILURE - The checksums on the read and write test were incorrect.

After all error responses, the program is automatically restarted. User should erase tape and mark again.

At the end of the marking and testing sequence if tape is good, the comment shown below is printed and the program is restarted.

GOOD TAPE

If the user is finished marking tapes, he should press the stop switch, remount his unit 1 CPS tape, and lift load to return to the CPS system.

This program was written by John Nepiuk for use at CEL.

F.5 *FOCAL

F.5.1 System Usage. Two commands, a source file capability and a parameter string specification procedure have been introduced in *FOCAL to make use with the filing system easier.

(A) LEAVE

This command will wait for any teletype output to be completed and then return immediately to the system through the bootstrap tape routine in page 7600 octal.

(B) UPDATE

To save a *FOCAL program the operator should use this command. It causes the program to be written into the temporary binary file -B of the system, and a return to the system to be made through the boot in page 7600 octal. The operator may then use -B just as any other system file such as

SAVE -B FOCLFILE

to save his *FOCAL program into the file named FOCLFILE.

(C) To run a *FOCAL program which has been saved in the system under the name FOCLFILE, it must be specified as a source file. To do this, the operator would give the system the command

RUN *FOCAL FOCLFILE

*FOCAL would then be loaded along with the program FOCLFILE and with no variables initially defined.

(D) The parameter string capability allows *FOCAL to reconfigure itself to meet the operator's requirements. The following characters are legal:

4	4K configuration of *FOCAL
8	8K configuration of *FOCAL
E	extended precision (known as 4-word or 10-digit)
L	allow lower case character input
F	allow extended functions (FLOG, FATN, FEXP)
T	allow trig functions (FSIN, FCOS)
N	logical "NOT" of next character in par. string.

If the 4K configuration has been provided, *FOCAL will default to

4NENLFT

or the 8K, if provided, will default to

8NENLFT

This means that the extended precision is not desired, lower case input is not allowed and the extended and trig functions are available. Using the parameter string, a command to the system would appear as

```
RUN *FOCAL FOCLFILE P=8NEF
```

Note that the trig functions cannot be deleted without also deleting the extended functions and any attempt to do so will be ignored.

The default core requirement for *FOCAL may be altered by using (octal)

3217 as 0000 for 4K default,
3217 as 7777 for 8K default.

And these must be changed before starting *FOCAL (starting address is 200 octal).

F.5.2 Miscellaneous Notes of Interest.

(A) Plotting with *FOCAL

The use of an X-Y plotter with *FOCAL has been found extremely useful in the graphical display of calculations. The following is information and suggestions to generalize the plotting procedures.

(1) Use on the LINC-8

(1.1) The FDIS (X, Y) function is used to load the LINC A-register with Y and the LINC B-register with X+400 octal. The two D/A's on the LINC-8 continually operate on these registers (see 1967 Small Computer Handbook for further information).

(1.2) The FRLY (P) function may be used to load the LINC-8 Relay register with P. It is suggested that the normally open side of relay number 0 be used for pen up, pen down control of the X-Y plotter. Since most plotters call for an open relay to obtain pen up and a closed

relay to obtain pen down this would mean

P=0 decimal for pen up
P=32 decimal for pen down

where control is available from the CR relay socket labelled

RO - NORMALLY OPEN

(2) Line approximation - Most X-Y plotters are unable to go from one point to the next in a reasonably straight line unless the move is a small one, so a software procedure must be used to achieve this. It is suggested that a line number group (part 15 has been selected) be devoted to this task although it means that plotting will be slowed down by such a routine. A routine is enclosed which requires that all use of the D/A's be made through it. It also requires setting three variables

PN<0 for pen down, >=0 for pen up
X=absolute X-coordinate
Y=absolute Y-coordinate

and calling it by means of a DO 15 instruction. An example of a calling sequence is

SET PN=-1; SET X=0; SET Y=0; DO 15

which would put the pen down and draw a line from its last position to the position (0,0) .

These three variables remain unchanged after using part 15, however two more variables, WX and WY, must be devoted to holding the last X and Y values respectively and should not be changed by the user. Part 15 also uses variables ZX, ZY and ZZ as temporaries which may be altered.

When the pen is up, part 15 will move the pen to its new coordinates without any line approximations so that this may be as fast as possible.

The suggested routine is

15.01 C ROUTINE TO MOVE PEN.

15.02 C S PN(>0=UP, <0=DOWN); S X=-256->255; S Y=-256->255;
D 15

15.10 I (PN) 15.5; F ZX=1, 20; S ZZ=FRLY ()

15.11 S ZZ=FDIS(X, Y); S WX=X; S WY=Y; F ZX=1, 100; S ZZ=0

```

15.12 G 15.99
15.50 S ZZ=FRLY(32); S ZX=X-WX; S ZY=Y-WY; I (FABS(ZX)
                                     - FABS(ZY)) 15.53
15.51 S ZZ=FDIS(WX, WY); I (FSGN(ZX) * (X-WX) ) 15.99, 15.99
15.52 S WX=WX+FSGN(ZX); S WY=WY+ZY/FABS(ZX); G 15.51
15.53 S ZZ=FDIS(WX, WY); I (FSGN(ZY) * (Y-WY) ) 15.99, 15.99
15.54 S WY=WY+FSGN(ZY); S WX=WX+ZX/FABS(ZY); G 15.53
15.99 C

```

Different plotters, however, call for different delay values and trial and error methods must be resorted to to obtain the optimized routine for a particular plotter.

(B) *FOCAL Summary Table

ASK	Input value from teletype		
CONTINUE/COMMENT	Dummies		
DO	Execute DO argument and return		
ERASE/ERASE ALL	Erase variables/Erase text		
FOR	Repetitive operation		
GO	Program control transfer		
IF	Conditional		
LEAVE	Back to system		
MODIFY	Edit text line		
QUIT	Return control to operator		
RETURN	Terminates DO operation		
SET	Define a variable		
TYPE	Output value on teletype		
UPDATE	Save program into -B and return to system		
WRITE	Output program on teletype		
FSQT	Square root	FCOS	Cosine
FABS	Absolute Value	FATN	Arctangent
FSGN	Sign part of the expression	FDIS	Scope function
FITR	Integer part	FADC	Analog to digital input
FRAN	Random number	FRLY	Relay output
FEXP	Natural base to the power	FRSW	Right switch register
FSIN	Sine		

?00.00	Manual start given from console.
?01.00	Interrupt from keyboard via CTRL/C.
?01.40	Illegal step or line number used.
?01.78	Group number is too large.
?01.96	Double periods found in a line number.
?01.:5	Line number is too large.
?01.;4	Group zero is an illegal line number.
?02.32	Nonexistent group referenced by 'DO'.
?02.52	Nonexistent line referenced by 'DO'.
?02.79	Storage was filled by pushdown list.
?03.05	Nonexistent line used after 'GOTO' or 'IF'.
?03.28	Illegal command used.
?04.39	Left of "=" in error in 'FOR' or 'SET'.
?04.52	Excess right terminators encountered.
?04.60	Illegal terminator in 'FOR' command.
?04.:3	Missing argument in display command.
?05.48	Bad argument to 'MODIFY'.
?06.06	Illegal use of function or number.
?06.54	Storage is filled by variables.
?07.22	Operator missing in expression or double 'E'.
?07.38	No operator used before parenthesis.
?07.:9	No argument given after function call.
?07.;6	Illegal function name or double operators.
?08.47	Parentheses do not match.
?09.11	Bad argument in 'ERASE'.
?10.:5	Storage was filled by text.
?11.35	Input buffer has overflowed.
?20.34	Logarithm of zero requested.
?23.36	Literal number is too large.
?26.99	Exponent is too large or negative.
?28.73	Division by zero requested.
?30.05	Imaginary square roots required.
?31.<7	Illegal character, unavailable command, or function.

F.5.3 FOCAL Features Altered or Added.

(A) FADC

The FADC function works on the LINC-8 with the argument of the function specifying the A/D channel from which the sample is to be taken (channels 0-17 octal; 0-15 decimal). The usage of the FADC is compatible with the DEC write-ups of this function except that the returned value is two times as large as the original 9-bit sample. This was done to remain compatible with other CEL machines. Therefore this function

will return

an even integer from -512 to +510

when used on the LINC-8.

It is possible to generate any LINC instruction using this function (note that many should not be used without `PROGOFOP` and some may cause program control to transfer in such a way as to crash `*FOCAL`). The `FADC` function adds 64 (100 octal) to its argument and executes this as a LINC instruction. Therefore, by supplying the proper argument, certain LINC instructions may be used. The function will always return the contents of the LINC A-register after the execution of the LINC instruction.

It should be noted that the interpretive nature of `FOCAL` does not allow it to sample at a high rate. The user may need to determine the rates attainable by his `FOCAL` program if this becomes critical.

(B) `FDIS`

This function has been modified to handle the LINC-8 oscilloscope and corresponding `D/A`'s. The use of the function is compatible with DEC's write-ups concerning it except for the ranges of the arguments which are

X an integer from -256 to +255

Y an integer from -256 to +255

These arguments do not have to be integers but the `FDIS` will use only their integer parts. When `*FOCAL` is in operation, the LINC A-register will contain the last Y value from `FDIS` and the LINC B-register will contain the last X value from `FDIS` (note there is a +400 octal offset on the X value so that X=0 is the center of the scope). These two registers are changed only for a few microseconds when executing other LINC instructions and are immediately restored. This was done in order to preserve `D/A` positions in the event that an X-Y plotter is being driven by the `FDIS` function.

This function is unable to produce scope sweeps at a reasonable rate, again because of the interpretive design of `FOCAL` and a single scope sweep may take on the order of seconds.

(C) `FRLY`

A function to drive the LINC Relay register and the connected relays has been added. The integer part of the argument of this function is placed in the LINC-8 Relay register.

(D) FRSW

The contents of the Right Switch register on the LINC-8 may be determined using this function. The value returned will be

an integer from -2048 to +2047.

(E) FRAN

A mixed congruential psuedo-random number generator has been included in *FOCAL . It is moved into locations 7725 to 7766 (octal) of the system boot by the *FOCAL initialization routines. This routine obtains random numbers by the formula

$$R=(R*A)+B \pmod{2^{*}24} \quad (\text{mod } 16,777,216 \text{ decimal})$$

where A is a multiplicative constant and B is an additive constant (note each random number is generated from the previous one but in such a way as to seem somewhat random). When *FOCAL is loaded, R is initially zero but this may be altered as it is a double precision number with

R using locations 7756 and 7757 octal

for its high and low order parts respectively. When *FOCAL is loaded, the contents of

A is 6541 octal (3425 decimal) which is 1 mod 4
B is 2263 octal (1203 decimal) which is 1 mod 2.

These, too, may be changed as they are single precision numbers with

A using location 7760 octal,
B using location 7761 octal.

Note random numbers are "uniformly" distributed from

+0.0 to +1.0

(F) UPDATE

This new command causes a *FOCAL program to be saved on magnetic tape using the filing system. Refer to section 4., System Usage, of this write-up for a complete description of this command.

Implementation of this command meant replacing the high speed reader character * in the command table with the character U for UPDATE . This is regrettable but due to the design of FOCAL, unavoidable. The core devoted to the high speed reader routines has also been allocated to other routines since it is disabled.

(G) LEAVE

The command causes immediate return to the system without saving the current FOCAL program. See System Usage of this write-up for a complete description of this command.

(H) Upper/Lower Case Input

FOCAL will allow lower case letters to be used but recognizes them as legal only when used as commands. The packing scheme used by FOCAL also causes lower case characters to use twice the storage that upper case require. *FOCAL allows the capability of mapping lower case characters into upper

ASCII codes 340-376 octal go to 300-336 octal.

See System Usage of this write-up for information on how to use this feature of *FOCAL .

(I) Deletion of Extended Functions

The feature of the FOCAL initial dialogue which deletes certain functions has been implemented in *FOCAL . Refer to System Usage of this write-up for information on how to use this feature.

(J) Suppression of = on Typeouts

The suppression of the character = when outputting a numerical value has been suppressed to provide for more readable table output possibilities. This was accomplished by changing (octal)

location 6002 from a 4551 to a 7200

For certain applications, the user may wish to suppress the leading blank which is typed when a numerical output value is typed. To do this, the user should change (octal)

location 6005 from a 1334 to a 5210.

(K) Two DEC Errors Corrected

Only two errors are presently known in FOCAL and these have been corrected by changing (octal)

location 5333 from a 2401 to a 2501,
location 5525 from a 0005 to a 0004.

(L) Storage Alterations

*FOCAL has in no way reduced the size of core devoted to the user as all alterations and additions were done using various unused core locations in FOCAL (of which there are essentially none in *FOCAL).

(1) The input line buffer when in the 8K mode has been changed from 70 octal to 130 octal locations so that a full line of lower case characters may be used. This reduces the user text space from 7500 octal to 7440 octal locations but it seems very unlikely that anyone would require this much program space anyway. (Programs seem to exceed other capabilities of FOCAL long before the available text storage is full when in the 8K mode.)

(2) There is room in *FOCAL's tables for one more function. A possible implementation of an FTAP function to read and write data using LINCTAPES is considered a future goal.

(3) Due to core requirements, the core locations for the non-interrupt I/O routines described in DEC's FOCAL write-ups has been devoted to other routines and these may no longer be used by resorting to the simple modification procedures in the DEC write-ups.

.5.4 Hardware Configurations. Various types of DEC computers are able to run *FOCAL but certain features may be disabled because of the particular machine configuration. If *FOCAL is run on a standard DEC LINC-8, it will configure itself so that the FADC and FRLY functions are disabled (their use will produce an error) and the actual intensification of the dot for the FDIS command is disabled. *FOCAL will notify the operator of its status in this case by typing

NO ADC, RLY, DIS DOT.

The loading of the D/A's by means of the FDIS instruction, however, is still possible on a standard LINC-8. The reason that *FOCAL must be configured this way is that the LINC Lower Memory Bank (LMB) cannot be set from the PDP-8 side of the LINC-8. The FADC, FRLY and FDIS must execute LINC instructions and control of the LMB is essential since it controls the section of core from which a LINC instruction is to be taken.

A machine modification is necessary to make use of all of these functions. The Cooley Electronics Lab. (CEL) LINC-8 has been modified so that an ICON-16 is used to set the LMB=0 (which is what *FOCAL requires it to be). The hardware on the LINC-8 will always leave the LMB=2 or 0 depending on the state of the machine so if the LMB is not set under program control to something else just prior to running *FOCAL, only one bit need be cleared to insure the LMB=0. This modification involves only

one wire from pin LJ10U to pin LFP3S
for an ICON-16 to set the LINC LMB=0.

The actual modification made to the CEL LINC-8 was such that the least significant three bits were cleared (the LMB could be set to 0-7 octal before running *FOCAL). A suitable modification is described in DECUSCOPE Vol 10, No 5, Page 14.

Note an ICON-16 means to execute a 6141 (octal) instruction with a 16 (octal) in the PDP-8 accumulator. When the LINC LMB=0, it means that all LINC instructions will be taken from the locations known as 0000-1777 octal in the PDP-8.

If all five bits of the LMB are used prior to running *FOCAL, then an even more elaborate modification must be designed and implemented.

Machines which have been modified so that various iot instructions are used and which should not be executed by *FOCAL may require changing locations in *FOCAL so that these are not executed. *FOCAL executes the following iot's intended for use on CEL machines but not defined by DEC ;

6554
6536
6532
6141 with AC=16
6162

F.6 *FCOPY

This program is intended for use in transferring files from one 4K-CPS tape to another. The main purpose of the program is to provide a convenient capability for communication between various users of the 4K-CPS system.

Instructions

Execute the 4K-CPS instruction

RUN *FCOPY

to load and start

*FCOPY

An identification heading will be typed and the user will be asked to specify the type of tape that will be mounted on unit 0 during the transfer. If a symbolic ("unit 0") tape will be present on unit 0 during the file transfer, the user should type S followed by a carriage-return. If a binary ("unit 1") tape will be mounted on unit 0 during the transfer, the user should type B followed by a carriage-return.

The same question will be asked of the user concerning the type of tape which will be mounted on unit 1 during the transfer. Failure to mount the same types of tape specified in answer to *FCOPY's inquiries may result in the destruction of one or both of the tapes; i. e., if the user specified a symbolic tape (S) would be on unit 1 and a binary tape was inadvertently mounted there, the binary tape may be destroyed.

Three more requests are then made of the user.

Request 1

FROM:

The name of the source file should be entered at this point. The tapes to be used in the actual transfer must be mounted before the terminal carriage return is struck in response to this request!

Request 2

UNIT:

The user should enter the unit number that the source tape is on (0 or 1). This does not refer to the type of the source tape. *FCOPY assumes that the destination unit is the opposite as the source

unit.

Request 3

TO:

This entry should be the desired name of the destination file (the name the file is to have on the destination tape). The use of the "!" to denote replacement of an already existing file is compatible with the description of the SAVE command in 4K-CPS . The user should check that his entries have been made correctly before terminating this response as the transfer will begin immediately after this entry is made.

After a transfer is completed, *FCOPY again makes the above last three requests. This allows multiple file transfers to be made between two tapes and it also allows the mounting of different tapes of the same types on each unit (if a binary tape was on unit 0 , only another binary tape may be mounted on unit 0 , etc.)

An empty terminated line causes *FCOPY to ask for a (CTRL-C) to return to the system. If this is desired, the user should mount his 4K-CPS system tapes before striking the (CTRL-C) . Striking any other character restarts *FCOPY . This option allows changing the type of tapes to be mounted on each unit.

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Cooley Electronics Laboratory The University of Michigan Ann Arbor, Michigan		2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	
		2b. GROUP	
3. REPORT TITLE 4K-CPS: A Programming System for the PDP-8 Side of the DEC Linc-8 Computer			
4. DESCRIPTIVE NOTES (Type of report and Inclusive dates) Technical Memorandum No. 101 - 03674-21-M March 1972			
5. AUTHOR(S) (First name, middle initial, last name) Cederquist, Gerald Metzger, Kurt			
6. REPORT DATE March 1972 (2nd printing)		7a. TOTAL NO. OF PAGES 93	7b. NO. OF REFS
8a. CONTRACT OR GRANT NO. N00014-67-A-0181-0032		9a. ORIGINATOR'S REPORT NUMBER(S) TM101	
b. PROJECT NO.		9b. OTHER REPORT NO(S) (Any other numbers that may be assigned (this report) 036040-13-M	
c.			
d.			
10. DISTRIBUTION STATEMENT Approved for public release; distribution unlimited.			
11. SUPPLEMENTARY NOTES		12. SPONSORING MILITARY ACTIVITY Office of Naval Research Department of the Navy Arlington, Virginia 22217	
13. ABSTRACT The 4K-Cooley Programming System (4K-CPS) is a collection of programs designed to aid in programming the PDP-8 side of Digital Equipment Corporation's LINC-8 computer. This collection of programs consists of a file system, a loader, a symbolic text editor, and an assembler. These programs are structured so that a user can enter, edit, assemble, load and execute programs entirely from the Teletype keyboard. The system architecture is modular, and future components are easily "plugged-in" as needed.			

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Digital Computer Programming System PDP-8 LINC-8						

INSTRUCTIONS

1. **ORIGINATING ACTIVITY:** Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization (*corporate author*) issuing the report.
- 2a. **REPORT SECURITY CLASSIFICATION:** Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.
- 2b. **GROUP:** Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.
3. **REPORT TITLE:** Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.
4. **DESCRIPTIVE NOTES:** If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.
5. **AUTHOR(S):** Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.
6. **REPORT DATE:** Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.
- 7a. **TOTAL NUMBER OF PAGES:** The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.
- 7b. **NUMBER OF REFERENCES:** Enter the total number of references cited in the report.
- 8a. **CONTRACT OR GRANT NUMBER:** If appropriate, enter the applicable number of the contract or grant under which the report was written.
- 8b, 8c, & 8d. **PROJECT NUMBER:** Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.
- 9a. **ORIGINATOR'S REPORT NUMBER(S):** Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.
- 9b. **OTHER REPORT NUMBER(S):** If the report has been assigned any other report numbers (*either by the originator or by the sponsor*), also enter this number(s).
10. **AVAILABILITY/LIMITATION NOTICES:** Enter any limitations on further dissemination of the report, other than those

imposed by security classification, using standard statements such as:

- (1) "Qualified requesters may obtain copies of this report from DDC."
- (2) "Foreign announcement and dissemination of this report by DDC is not authorized."
- (3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."
- (4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."
- (5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. **SUPPLEMENTARY NOTES:** Use for additional explanatory notes.
12. **SPONSORING MILITARY ACTIVITY:** Enter the name of the departmental project office or laboratory sponsoring (*paying for*) the research and development. Include address.
13. **ABSTRACT:** Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as (TS), (S), (C), or (U).

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. **KEY WORDS:** Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.

UNIVERSITY OF MICHIGAN



3 9015 03483 5275