Technical Memorandum No. 102-I

03674-23-M

CPS PROGRAM LOGIC MANUAL

Volume I

CPS SYSTEM ARCHITECTURE AND CONVENTIONS

by

G. N. Cederquist
K. Metzger

Approved by: Theodore G Birdsall

for

COOLEY ELECTRONICS LABORATORY
Department of Electrical Engineering
The University of Michigan
Ann Arbor, Michigan

Contract No. Nonr-1224(36)
NR187-200
Office of Naval Research
Department of the Navy
Washington, D. C. 20360

February 1970

PREFACE

TO THE SERIES

This report of four volumes is intended to document the May 21, 1969 version of the Cooley Programming System (CPS) which was developed at the Cooley Electronics Laboratory of The University of Michigan. The four volumes are titled:

Volume 1: CPS System Architecture and Conventions

Volume 2: CPS Basic Programming Package

Volume 3: CPS FORTRAN Package

Volume 4: CPS System Utility Programs

The four volumes were written in order to take a snapshot of CPS at one point in its continuing development. This version of CPS is considered to be a first generation system; successive versions are on the drawing boards and internally resemble their parent less and less every day.

CPS is a generalized programming and file management system written for use on the PDP-8 processor of Digital Equipment Corporation's LINC-8 computer. A minimum memory size of 8192 words is required. Extensive use is made of the two tape units present on every LINC-8 for both file storage and system residence.

Using CPS, programs can be entered, edited, assembled (or

compiled), loaded and executed entirely from the keyboard without the use of paper tape. CPS provides power and flexibility normally only found on larger computers and in fact was modeled after the Michigan Terminal System which operates on an IBM SYSTEM/360 model 67.

In addition to a comprehensive file management and control system CPS contains:

Symbolic Text Editor	8-K FORTRAN Compiler
MACRO-8 Assembler	Two loaders
SABR Assembler	Various utility programs

Each of the above programs contains service routines which permit automatic communication with the central file system and which allow direct access to CPS files. The general policy followed in implementing CPS was to borrow and adapt as much of DEC's software as possible in order to speed system development.

The responsibility (or blame) for various segments of CPS is divided as follows:

Gerald Cederquist	System design and conventions, Control Program, Absolute Assembler, Absolute Loader, MARKP8 (tape marking program), and FILECOPY (file copying program).

Kurt Metzger	SABR, FORTRAN, Relocating Loader, PAPERBIN (binary paper tape input program), TAPCOPY (tape copying program), and assorted tape routines.
joint effort	Text Editor, I/O Control System, and various compromises.

Work started on CPS in November of 1968 with the first workable version being completed in February of 1969. The FORTRAN-SABR package was incorporated in the March-April period of 1969. Since this time CPS has been in use at CEL in the development of digital signal processing programs for project MIMI. It has been found to be a very effective tool and has greatly decreased program development time and programmer frustration. Tasks which formerly took over a month to complete using the DEC 8-LIBRARY System are now routinely completed in one to two weeks.

The bulk of CPS and the associated routines were hurriedly written since the authors were effectively stealing time from their thesis research. Consequently portions of the code were done in a quick and dirty manner. Now that several months have passed, the fact that these portions were quick has dimmed in memory but the dirt remains.

The authors would like to thank Dr. T. G. Birdsall of CEL for his continued encouragement and support and Mr. C. Conley of DEC for his assistance in providing the FORTRAN-SABR package for use in CPS.

G. Cederquist
K. Metzger

Ann Arbor, Michigan
December 1969

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT.)

PREFACE

TO VOLUME I

Volume I of the CPS Program Logic Manual describes the system structure and conventions. It presumes that the reader has a user's knowledge of CPS and is familiar with the CPS User's Reference Manual.

CHAPTER 1

AN OVERVIEW OF CPS SYSTEM ARCHITECTURE

CONTENTS

Preface

This chapter gives an overview of both the external and internal structure of CPS, the Cooley Programming System for LINC-8 computer with 8192 words of core memory. (See Cooley Laboratory Technical Memo 101 for documentation of a related system, 4K CPS, to run on a 4096 word machine.) It is intended as an introduction to CPS architecture for those who will maintain and improve the system. Implementation details and program logic are covered in detail in other chapters and volumes of this report. Familiarity with the CPS SYSTEM REFERENCE MANUAL is presumed.

It will be obvious to the knowledgeable programmer that the author has borrowed notation from many sources (in hopes of relating to past work). One of the most used notation sources is OS/360, the operating system for the IBM System/360 computers.

This publication describes CPS up through release AY219 (May 21, 1969). The system architecture will not remain as specified herein. CPS is a research project, and is subject to change at any time.

1.1 Design Goals

CPS was written to provide a general information processing and programming capability for the PDP-8 processor in the Cooley Electronics Laboratory LINC-8 computer. The original objective in writing the system was to eliminate the need for use of paper tape or the switch register: to allow a user to manipulate symbolic files, assemble these files into machine language, and run the resulting program entirely from the Teletype keyboard. In the early stages of the system design, the original objective was expanded to include building as general a system as possible, consonant with memory limitations. The extent to which the expanded goal has been achieved is indicated by the fact that many new programs and applications not previously envisioned or considered practical have been implemented since CPS became available.

CPS is the most complex set of programs ever written at Cooley Lab. It was recognized early in the system design phase that the only way to bring the project to fruition would be to divide the overall system into many subsystems, each of which could be considered a black box by all the other subsystems. In addition, it was desirable to separate along such boundaries as would minimize the number of bits of information transferred between subsystems. The systems-level programmer will recognize that these goals were in conflict with the fact that writing CPS was in a sense a research project, a project wherein we had much general, but little specific idea of what would be good to implement and

how to do it. Thus CPS consists of very general, flexible, and somewhat redundant support routines which helped to develop the initial system. The later 4K version of CPS does not contain such general routines; rather it is specifically tailored to implement an extension of the system design evolved previously during the writing of CPS for the 8K LINC-8.

An additional design goal was that of delaying as much as possible the binding time of any operation in the system. This has the effect of making the system highly interactive from the user's standpoint. It does however consume memory space, and consequently compromises had to be found to implement some of the features of the system.

1.2 The Resident Supervisor Problem

All of the computer utility systems* from which CPS drew inspiration have in some sense a resident supervisor, a subsystem which does input-output operations, storage allocation, and CPU time allocation. From a systems design standpoint, the need for a resident supervisor in these systems comes about because they serve multiple users simultaneously and it is necessary to have a single procedure to do delicate operations, lest users catastrophically interfere with each other. In addition, the code for doing I/O, servicing the clock, and the like may be written and loaded once, and then be shared among

---

*TSS/360 at IBM, MULTICS at Project MAC, and MTS at The University of Michigan.

many users.

These design considerations were not applicable in CPS since it was decided that, lacking appropriate hardware* to enforce I/O protocol and memory protection, CPS would be a one-user system. Hence there was no classical argument for forcing the CPS user to run his programs with a system-supplied supervisory program in core--the single user would perforce take all the CPU time available, and could certainly do his own I/O. Moreover, an execution-time supervisor would undoubtedly take up core space which many users wou would prefer to have available for their programs.

The approach taken was to do away with all requirements on the user that he give up control of any part of the hardware to programs not of his own choosing. However, there still remained the problem of user program I/O and interface with CPS. Plainly the casual user who wishes to access files will not want to write his own routines, both for lack of knowledge and the possibility that a bad routine might inadvertantly destroy files or cause some other catastrophe. This problem was solved by providing access to procedures such as a series of standardized I/O controllers, a set of file access routines, and the like which the user may load and use as black boxes with known terminal properties. Thus core memory is conserved while standard interface conventions are observed.

---

*such as IOT command trapping which is available on DEC's TSS/8 system.

1.3 CPS Control Program

In spite of the fact that no sort of resident supervisor was to be used in CPS, there was still need for a general program to invoke various service modules and hold a dialog with the user. This program is called the CPS Control Program. It is loaded into core (usually after every job step* and allows the user to direct the system interactively to perform the next job step. The operation of CPS under the CPS Control Program can be envisioned as the traversing under user control of various branches of a tree rooted at the Control Program. (See Fig. 1.1.) Certain service modules are structured as subroutines contained within the Control Program while others are stand-alone modules because of memory space limitations.

It is of interest that in keeping with the objective of finding nice boundaries upon which to divide the system, little information is passed between the various service modules in Fig. 1. This design consideration also dictated that systems programs such as assemblers, compilers, and editors be treated exactly the same as user-generated programs. Thus the system programs may be maintained, moved, replaced, and the like without having to convey any information to the Control Program.

1.4 System Components

CPS contains 2 main components, a filing system and a program

---

*A job step is the action caused by issuing a CPS command; e.g., doing an assembly, saving a text file, printing the file index, and so forth.

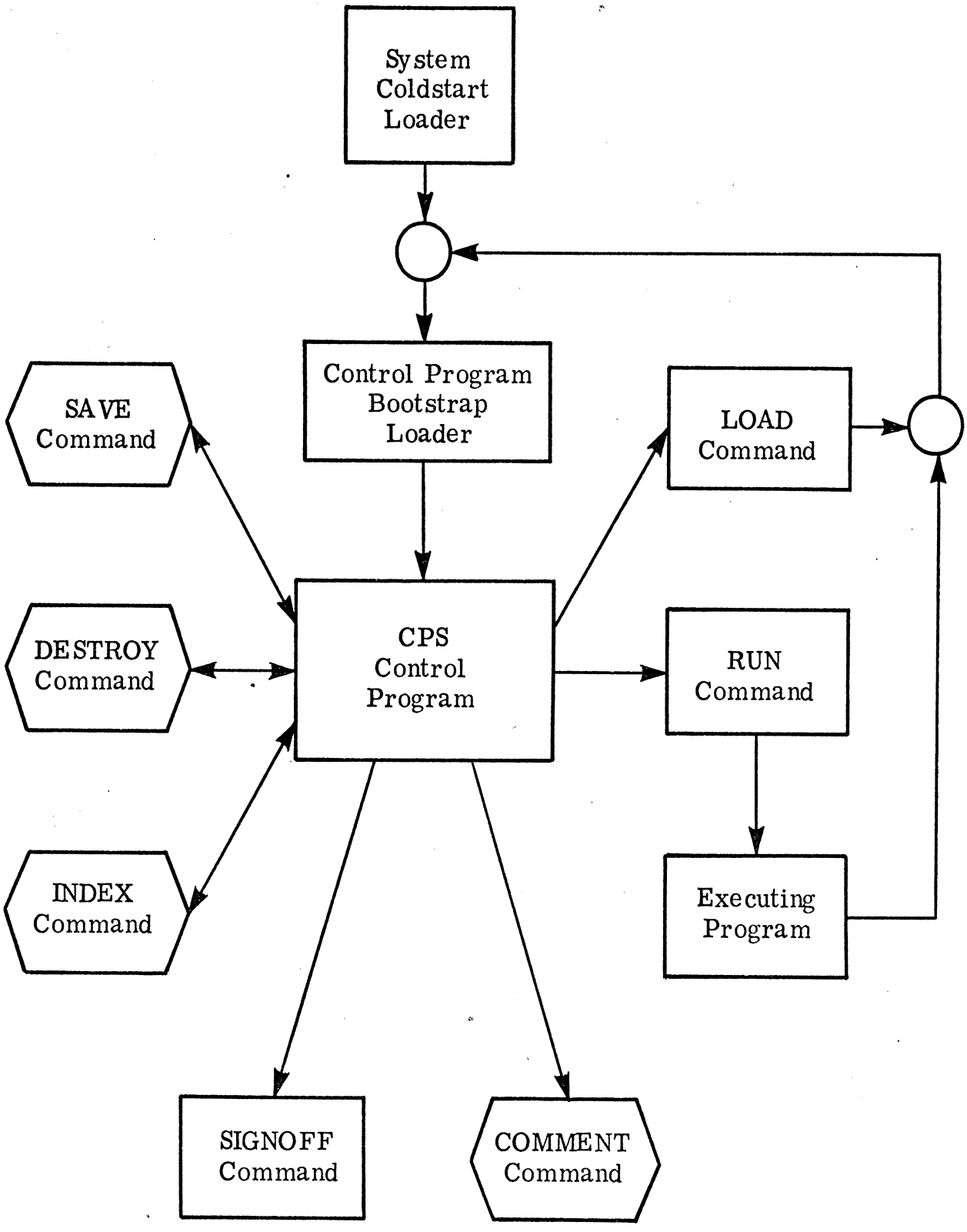


Fig. 1.1. CPS CONTROL STRUCTURE

execution system. Provision is made for interfacing these systems with each other, with the user, and with executing programs by means of (among other things) a command language and a system communication area. These are discussed in the following sections.

1.4.1 File System. A file is a collection of information in machine-readable form; for example, it may be the output of the assember operating upon a particular text string. Using the CPS filing system, the user may name this collection of information and store it on LINC tape. The file may then be referred to by name for later use. A user may also change the contents of files, destroy files, and copy files from other users' CPS tapes. All routines which are part of the filing system (with the obvious exception of user program file access routines) are currently resident within the Control Program.

1.4.1.1 File Structure. Files consist of a number of PDP-8 pages of information and are stored on LINC tape which has been marked at 128 words per block. (Some thought was given earlier to having the system handle files in partial page form or in line file form such as in MTS, MULTICS, or TSS/360, but this was rejected as being too slow* and too consumptive of high speed memory.)

CPS requires two tapes for filing purposes, and they are organized as shown in Table 1.1. As can be seen, the tape allocation is complicated by the fact that the tapes are used for system residence as

---

*This was based on tape spooling times. Use of disk memories would permit a much more flexible file organization.

Tape Drive	Block Nos.	Contents
0	0	System Coldstart Loader
	1- 377	Symbolic working area
	400- 477	Reserved for system functions
	500-2000	User files
1	0	Coldstart Error Program
	1- 147	Binary working area
	150- 237	Reserved for system functions
	240-2000	User files

TABLE 1.1 CPS TAPE ORGANIZATION

well as for filing purposes. The file space allocation algorithm is quite simple; newly-created files are placed at the end of the tape. This algorithm trades decreased program complexity for increased elapsed time in the form of tape spooling and is only marginally acceptable. The next scheduled change to CPS is modification of this algorithm to decrease tape spooling time.

1.4.1.2 Working Areas. Many programs produce files of information, and some programs need access to mass storage while they are in execution. To provide for both these requirements, a portion of each file tape has been set aside as a working area. The working areas have names, "-S" for the symbolic working area and "-B" for the binary working area, by which the user may reference them in CPS commands as if they were files. Operationally, programs which produce files put these files into the working areas. Provision exists within the CPS Control Program to modify the file description of the working areas so that they will reflect the information deposited there by a previously-run program. This modification is transparent to the user. Except for the (possible) modification of the file description of the working areas upon entry to the CPS Control Program, the working areas behave within the file system exactly the same as permanent, user-created files.

The symbolic working area, -S, is located on drive Ø along with user-created symbolic files. For this reason, the tape mounted on drive Ø is often called the "symbolic tape" of the 2 tapes necessary to

run CPS. Similarly the drive 1 tape contains -B, and is often called the "binary tape."

1.4.1.3 File Attributes. As each file is created, it is assigned an attribute according to its contents; e.g., a text file produced by the symbolic editor has the "compressed symbolic" attribute assigned to it. This attribute is inspected when the file is to be assigned to a tape drive. If the attribute falls into the "symbolic" equivalence class, the file is assigned to drive Ø (the left-hand drive); otherwise it is assigned to drive 1. The contents attribute and length of a newly created file (as well as its contents) are derived from the existing file specified by the user in the SAVE command (see Section 1.4.1.6).

An additional feature of the filing system in CPS is the capability to discern if a file may be used as an information source or sink or both. This capability is utilized during user specification of the interface between a program to be run and the file system.

1.4.1.4 Access Methods. The accessing of information in files is the responsibility of the program needing the information in the file, and though most access is currently sequential, random access is certainly allowable subject to mechanical limitations of the tape drives.

1.4.1.5 Critique of the Filing Structure. At this point, it should be apparent that the file structure chosen for CPS was strongly

influenced by the requirement that the user need have no system code resident with his program. This point and the inherent sequential structure of the tape storage medium lead to knotty problems when writing into a permanent file. The working areas have worked out well in practice, although they make filing proceed somewhat more slowly than it would if writing into a permanent file were possible. Again, the file structure could be improved if a disk were available.

1.4.1.6 User Creation of Files. The user creates files by means of the SAVE command. This command simply copies an existing file into a file which is created to hold the copy; the name of the newly-created file is designated by the user.

If the user specifies as the name of the file to be created a name already in the file index, he will be prompted to determine if he wishes to have the existing file replaced by the new one. If he indicates replacement, the old file is destroyed and the SAVE operation proceeds as if the old file had never existed.

It will be useful to follow through an example of a typical filing operation; suppose the situation is as follows. The user invokes the text editor stored in the file "*ED" and specifies the name of a file, say "QQSV", containing the text to be edited. The editor is brought into core memory and started. The input file support routine within the editor spools down the symbolic tape and reads from the tape the contents of QQSV. (See the RUN command, Section 1.4.2.2, for an

explanation of how the editor knew where QQSV was located.) The user is then prompted for editing commands. When the user is done editing, he signifies his desire to return to the CPS Control Program by issuing a command to the editor. The editor then writes the text file onto -S and leaves -S updating information in a communication area for the Control Program. This information consists of a starting tape block number, a length, and a contents attribute for the file of edited text. The Control Program is then bootstrapped into core and started. If certain key words are properly set (the editor will have set them), the Control Program will update the file description of -S to reflect the fact that the edited copy of QQSV is stored there. The user may then issue a SAVE command to save the edited file contained in -S into a permanent file, replacing the old copy of QQSV if he sees fit.

Let us suppose the user does desire to replace the old copy of QQSV with the edited version stored in -S. He will issue the command

SAVE -S QQSV !

causing the SAVE service module within the Control Program to be invoked. The SAVE module will destroy the old copy of QQSV by calling upon the DESTROY service module, and will then create a copy of -S at the end of the sumbolic tape. The copy will have name QQSV, and will possess the length and contents attribute of -S. It will also be flagged as an information source only, since write access to

permanent files is not allowed.

Note that at no point during the file manipulation does the user have to supply information to the file system other than a name, a large improvement over previous file management systems for the PDP-8. Note also that it is possible to store information of virtually any type in such a filing system, a necessity if the system is to expand to fill applications not yet envisaged.

1.4.1.7 Destruction of Files. The user may delete files from his file tapes by issuing the DESTROY command to the Control Program which then invokes the DESTROY service module. The DESTROY module removes the file entry from the volume table of contents (VTOC) for the tape and puts the tape space occupied by the file back into the free storage pool by simply moving all files further down the tape toward the front of the tape, overwriting the destroyed file on tape in the process. The names of the two working areas may not be deleted from the VTOC.

1.4.1.8 Listing of File Names. The user may request a listing of names of files on his file tapes by issuing the INDEX command. The listing may be qualified as to level of detail, type of file, and file attribute. The information displayed in the listing is derived from the VTOC of each file tape.

1.4.1.9 System Files. By convention, a file whose name begins with an asterisk * is designated a system file. The system

file is meant to hold system programs, such as assemblers, compilers, tape utilities, and the like. These files differ from other files in only 2 respects. The syntax for the command to destroy a system file differs from the command to destroy other files. This renders the system file somwhat immune to destruction by the casual user. Also, the names of system files are not printed in a listing of file names unless specifically requested.

1.4.2 Program Execution System. The program execution system includes a set of loaders* and provision for executing a program. In a broader sense, this system also includes the various language translators within CPS since they make program execution possible.

The program execution system gives the user the capability to bring together many translator output files and build their contents into a single file, called a core-image file, structured as an executable program. The RUN command can be used to invoke the program in a core-image file and put it in execution. At the same time, much information may be passed to the program to be invoked through the communication area. These aspects are discussed in more detail below.

1.4.2.1 The Loaders. Each language translator in CPS produces a file whose contents attribute falls into the binary equivalence

*More nearly, primitive linkage editors. However, the name Loader has stuck.

class and which contains the object program produced as a result of the translation process. The contents of these files are in a form close to machine language, the binary numbers which the PDP-8 interprets as instructions, but need to be further processed before they can be deposited into core memory in the form of a binary number program. This processing is done by the loaders.

The loaders have many capabilities and require much memory space, and so are run as independent programs though they are invoked by the Control Program. To conserve core memory during the loading process, loaders use the working areas for mass storage. -S is used while building the core image, and the completed core image appears in -B, from which it may be SAVEd or RUN. The core-image file contains the complete set of binary numbers constituting a program along with a descriptor table indicating where the numbers are to be placed in memory. It should be emphasized that in using the core-image file as an intermediate step in running a program, early binding time of the program linkages was traded for substantial flexibility in the loaders.

There are presently 2 loaders available in the system, one to load the output of DEC's absolute assemblers (e.g., MACRO8) and the other to load the output of the SABR relocating assembler. Two loaders are needed since the absolute and relocating assemblers have completely different object file formats and philosophies. Since the

loaders are in reality run as separate programs, their structure, data formats, user interaction, and the like are covered in a separate chapter in later volumes.

1.4.2.2 The RUN Command. The RUN command service module is invoked by the Control Program to load into core memory and place into execution a program contained in a core-image file. The user may specify the following items in the RUN command: 1) the name of the file containing the core image to be run, 2) a number of interface connections between the program to be executed and the filing and program execution systems in CPS, and 3) a character string to be passed to the program to be executed. This last is referred to as the parameter string. The RUN command service module deposits the user-specified information and linkages in a system communication area before loading into core the core-image file containing the program to be run. In keeping with the CPS design philosophy, responsibility for using this information lies with the program to be executed.

1.4.2.2.1 Core-Image File Specification. The file tape VTOC's are searched for the name of the core-image file to be run. If all is in order, the physical address of the file is left in a communication area for the core image loader. At present, only one file may be specified to be run; i.e., the system contains no explicit provision for overlaying programs. A possible future extension would

be to allow concatenated core-image files to be run with the understanding that this construct would in reality leave information for program overlays in the communication area.

1.4.2.2.2 Logical I/O Unit Specifications. The interface between an executing program and the file system is specified by the user-provided logical I/O unit specifications in the RUN command.

Logical I/O units provide a mechanism to delay the I/O device binding time when writing a program until the program goes into execution. For example, when running a FORTRAN program under the IBSYS operating system for the IBM 7000 series computers, the convention is made that the logical tape drive (read "I/O unit") which is named "6" is the system output tape. Thus a FORTRAN programmer may write the statement

```
WRITE(6,100) A
```

and rest assured that at the time of execution of his program the value of the variable A will be written upon the physical tape drive which mounts the system output tape, the correspondence between the name "6" and the physical tape drive address being established by IBSYS.

The important points in the above example are 1) by convention, a certain logical I/O unit fulfills a particular purpose in the context of a specific program; it is certainly possible that a SNOBOL program

running under IBSYS could consider logical tape drive 6 as the system input device. 2) The actual assignment of a logical device to a physical I/O device is delayed as late as possible. Within early operating systems, it was not possible to change the assignment of logical drives to specify different information sources and sinks than were set at system assembly time, and thus device binding time occurred at the beginning of program execution. With later executive systems the user had complete control over logical/physical device correspondence and so binding time was delayed right up until device use. Consonant with the CPS design philosophy of delaying binding time as late as possible, a variation of this second approach was used.

Programs which need to operate with files stored in CPS do their I/O through a set of logical device support subroutines. At RUN time the user must specify which CPS files* or physical I/O devices are to be attached to the available logical I/O units. This information is left in the system communication area. When the executing program does an I/O operation involving a logical I/O unit, the information left in the system communication area is interpreted to gain access to the file or device specified by the user.

As long as the linkage information is being interpreted, it is a simple extension to the logical device support routines to handle

---

*A file is an information source or sink, just as an I/O device is.

more than one file or device attached to an I/O unit. Thus concatenated files are easily incorporated into the I/O structure, giving the user the capability to effectively create one "superfile" from a number of smaller files insofar as an executing program is concerned. The concatenation capability has proven extremely useful in practice.

Currently CPS supports 6 logical I/O units, which have names "1", "2", ..., "6". Units 1, 2, and 3 are constrained by the Control Program to specify files or devices which are information sources, while units 4, 5, and 6 are constrained to specify information sinks. The user, when writing a program, must take these constraints into account. The constraints were made originally to prevent writing into permanent files. However, they also prevent specification of read/write access files* within the structure of the CPS RUN command. If the direction of data transfer were bound later+ within the logical device support routines rather than earlier in the Control Program, read/write access files would easily fit into the RUN command semantics. This problem is undergoing study and will probably result in system changes sometime in the future.

In order to specify physical devices in an I/O unit specification, the teleprinter has been given the name "PTR" and the reader/keyboard the name "RDR". A pseudofile named "*DUMMY*" is also

---

*Or if you speak PL/I, "update files".

+As the old philosopher says, "Do what I say, not what I do!"

recognized by the Control Program and has the following properties: When reading from *DUMMY*, an END OF FILE indication is always returned. When writing to *DUMMY*, information is accepted but never stored; thus it serves as the circular file, the system waste-basket.

Although this I/O linkage mechanism does not contain provision for getting location information for a file dynamically during program execution, it does move the logical I/O unit binding time from program assembly time to user program execution time. The author feels that a large part of the utility of CPS is due to this logical device mechanism.

1.4.2.2.3 Parameter String Specification. The user may specify up to 32 ASCII characters as a parameter string to be passed in the system communication area to the program to be run. The program may then use these characters in any way it desires. (Some past uses have been headings for listings and program behavior modification.) The parameter string is thus a general mechanism available to all programs to delay binding of some aspect of their behavior until execution time.

1.4.3 Command Language. The user communicates interactively with the Control Program through the CPS Command Language. Each command line input to the system causes execution of a job step; hence operation of CPS insofar as the user is concerned consists of

inputting a command, waiting until it is carried out, inputting the next command, and so forth.

1.4.3.1 Diagnostics and User Aids. The Command Language has purposely been kept simple and is quite easily learned, desirable features that certain widely-used third-generation operating systems notoriously lack. The entire command entry structure has been designed to minimize user frustration in entering commands. At times command lines can grow to be quite long (up to 255 characters are allowed) and so it is desirable to be able to fix errors in the middle of lines and not have to retype the whole line. Two facilities are provided for correcting errors, input line editing and dynamic replacement of bad strings.

The input line editor allows users to delete lines, delete characters, echo lines for checking, enter an END OF FILE status, enter control characters into command lines, and continue command lines over many physical lines on the Teleprinter. Thus a user may edit a line while he is forming it, eliminating bad characters as he goes.

If the user does not fix all the mistakes in a command line before he gives it to the Control Program for execution, the various command service modules will of course find the mistakes and issue a diagnostic. However in many cases, it is possible to fix the mistake on the fly by having the user enter a replacement string for the bad string in the command. This is done in several places, most

extensively in the RUN command where it saves copious amounts of retyping.

In order that the user know where he is within a job step, the Control Program (and most of the systems programs) issues a prefix character before each line typed to the user and before each input line solicited from the user. This character informs the user which part of CPS is talking to him or wants input from him.

1.4.3.2 Command Line Parsing. The first characters of an input line are assumed to be a command verb. Only the first three characters of this verb are significant and so commands may be abbreviated by their first three characters. When scanning an input command line, the first three characters are compared against an internally stored table of valid command verbs. If a match is found, the command service module which handles that verb is called. If no match is found, the command line is flagged as invalid.

Each service module consists of two routines in cascade. The first processes the remainder of the input line according to the syntax and semantics of the particular command. It is this routine which issues diagnostics concerning the user input line and which does dynamic replacement of bad input strings if possible. The second routine actually performs the function called for in the command using the parameters extracted from the input string by the first routine. If the command does not require leaving the Control Program, a return is made to the command input routine after the action is completed.

CHAPTER 2

COLD START PROGRAMS

2.1 CSBL - CPS Cold Start Bootstrap Loader

The source code for the Cold Start Bootstrap Loader (CSBL) is stored in the file "CLDSTART". Execution of CSBL is the means by which CPS is started when the LINC-8 has just been turned on.

CSBL is resident on tape block Ø of a unit Ø CPS tape. It is loaded into core and started at location zero whenever the "LOAD" switch on the LINC-8 console is lifted.

2.1.1 Program Operation. After chasing through the string of NOP's (which are just space fillers), a -1 is deposited into memory location 17774 in the communication area (see Section 2.3 CSBL Code). This will inform the Control Program that it should type out the identifying heading "CPS(MMDDY)" when it is activated. Currently only CSBL calls for typing of the identifier heading, although any program may do so. The "MMDDY" portion of the heading may be decoded to give the month, day, and year of the genesis of the version of CPS being loaded. Note that the characters "MM" are the second and third characters in the month name.

The read-only tape subroutine is next invoked to read one block from tape (128 words) from tape block 451 on tape unit Ø into memory starting at location Ø76ØØ. This block contains the regular

system bootstrap routine, BOOT. (See BOOT writeup, Chapter 3.) CSBL then jumps to the starting location of BOOT, Ø76ØØ, to bring the control program into core.

The read-only tape routine was lifted from BOOT, and has the same calling sequence as the one in BOOT. The reader should see the BOOT writeup for an explanation of the tape routine.

2.1.2 Listing. The annotated assembly listing for CSBL follows. It was produced by issuing the CPS command

```
RUN *ASM CLDSTART  P=L
```

```
                    /CPS
                    /COLD START BOOTSTRAP ROUTINE
                    /GNC 12/18/68
                    /
                    *0000
                    /
0000  7000          NOP
0001  7000          NOP
0002  7000          NOP
0003  7000          NOP
0004  7000          NOP
0005  7000          NOP
0006  7000          NOP
0007  7000          NOP
0010  7000          NOP
0011  7240          STA
0012  6211          CDF 10
0013  3424          DCA I KEYADR
0014  6201          CDF
0015  4025          JMS RDTAPE / read tape
0016  0451          451 /SYSTEM BOOT SITS HERE (2 blocks)
0017  0001          001 / read 1 block
0020  0000          000 / unit 0 and field 0
0021  7600          7600/ location 7600
0022  5423          JMP I .+1 / go to BOOT
0023  7600          7600
                    /
0024  7774          KEYADR, 7774 / NOP's keep this away from AX register
                    /
                    /READ-ONLY TAPE ROUTINE
                    /GNC 12/17/68
                    /
0025  0000          RDTAPE, 0          / for documentation, see "BOOT"
0026  7300           CLA CLL           / writeup
0027  1425           TAD I RDTAPE
0030  3170           DCA BLN
0031  2025           ISZ RDTAPE
0032  1425           TAD I RDTAPE
0033  7041           CIA
0034  3171           DCA NUMB
0035  2025           ISZ RDTAPE
0036  1425           TAD I RDTAPE
0037  3173           DCA LOC
0040  1173           TAD LOC
0041  7112           CLL RTR
0042  1165           TAD C2
0043  3172           DCA UNIT
0044  1173           TAD LOC
0045  0073           AND MASK
0046  7112           CLL RTR
0047  7010           RAR
0050  1130           TAD CDFD
0051  3126           DCA RCDF
0052  2025           ISZ RDTAPE
```

space filler

start here for real - no particular reason why 11 was chosen; I just put in that # of NOP's

set 7774 to -2 to type out heading

PAGE 02

```
0053  1425   TAD I RDTAPE
0054  3173   DCA LOC
0055  2025   ISZ RDTAPE
0056  7120   STL
0057  2170  SERCHA, ISZ BLN
0060  3174  SERCHB, DCA CSUM
0061  1147   TAD C7600
0062  3175   DCA CNTR
0063  1172   TAD UNIT
0064  6141   ICON
0065  7201   CLA IAC
0066  7430   SZL
0067  6141   ICON
0070  1132   TAD C6
0071  6141  B, ICON
0072  4146  A, JMS WAIT
0073  7500  MASK, SMA
0074  7120   STL
0075  1170   TAD BLN
0076  7650   SNA CLA
0077  5113   JMP THERE
0100  6147   INTS
0101  7010   RAR
0102  0166   AND M4000
0103  7460   SZA SNL
0104  5072   JMP A
0105  7020   CML
0106  7520   SNL SMA
0107  5072   JMP A
0110  6141   ICON
0111  7001   IAC
0112  5071   JMP B
0113  6147  THERE, INTS
0114  7012   RTR
0115  7620   SNL CLA
0116  5072   JMP A
0117  1167   TAD RFUNCT
0120  6141   ICON
0121  4146   JMS WAIT
0122  4146  RDTA, JMS WAIT
0123  1174   TAD CSUM
0124  3174   DCA CSUM
0125  6171   IAAC
0126  6201  RCDF, CDF
0127  3573   DCA I LOC
0130  6201  CDFD, CDF 0
0131  2173   ISZ LOC
0132  0006  C6, 6
0133  2175   ISZ CNTR
0134  5122   JMP RDTA
0135  4146   JMS WAIT
0136  7041   CIA
0137  1174   TAD CSUM
0140  7650   SNA CLA
0141  5160   JMP WAIT2
```

PAGE 03

```
0142  1147   TAD C7600
0143  1173   TAD LOC
0144  3173   DCA LOC
0145  5060   JMP SERCHB
0146  0000  WAIT, 0
0147  7600  C7600, 7600
0150  6147  C7, INTS
0151  7700   SMA CLA
0152  5147   JMP C7600
0153  1150   TAD C7
0154  6141   ICON
0155  7300   CLA CLL
0156  6171   IAAC
0157  5546   JMP I WAIT
0160  7300  WAIT2, CLA CLL
0161  2171   ISZ NUMB
0162  5057   JMP SERCHA
0163  6141   ICON
0164  5425   JMP I RDTAPE
0165  0002  C2, 2
0166  4000  M4000, 4000
0167  0003  RFUNCT, 3
0170  0000  BLN, 0
0171  0000  NUMB, 0
0172  0000  UNIT, 0
0173  0000  LOC, 0
0174  0000  CSUM, 0
0175  0000  CNTR, 0
```

2.2 UNITBAD - CPS Cold Start Error Program

The source code for the CPS Cold Start Error Program (UNITBAD) is stored in the file "UNITBAD". UNITBAD is a program which informs the user if he mounted a unit 1 tape on drive Ø and then tried to load it by lifting the LOAD switch on the LINC console.

UNITBAD resides on tape block zero of a unit 1 CPS tape. It is loaded into core and started at location zero only if a unit 1 CPS tape has been mounted on unit Ø and the LOAD switch has been operated. It causes the following message to be typed on the system teleprinter:

"You have mounted a unit 1 tape on unit Ø."

after which the CPU is halted. The code should be easily understandable as it stands, even to the novice. The listing which follows was produced by issuing the CPS command

RUN *ASM UNITBAD P=L

```
                    /UNITBAD - A PROGRAM TO TELL THE
                    /USER THAT HE'S MOUNTED HIS TAPES
                    /ON THE WRONG UNITS
                    /GNC  2/21/69
                    *0
0000  7000          NOP
0001  7000          NOP
0002  7000          NOP
0003  7000          NOP
0004  7000          NOP
0005  7000          NOP
0006  7000          NOP
0007  7000          NOP
0010  7300          CLA CLL
0011  2024    LOOP, ISZ MSGPTR / get char
0012  1424          TAD I MSGPTR
0013  6046          TLS
0014  6041          TSF / typeit
0015  5014          JMP .-1
0016  7300          CLA CLL
0017  2023          ISZ COUNTER / done?
0020  5011          JMP LOOP / no
0021  7402          HLT / yes - quit
0022  5021          JMP .-1
0023  7717    COUNTER, -END+BEG
0024  0024    MSGPTR, BEG-1
0025  0215    BEG, 215
0026  0212          212
0027  0212          212
0030  0331          "Y
0031  0317          "O
0032  0325          "U
0033  0240          240
0034  0310          "H
0035  0301          "A
0036  0326          "V
0037  0305          "E
0040  0240          240
0041  0315          "M
0042  0317          "O
0043  0325          "U
0044  0316          "N
0045  0324          "T
0046  0305          "E
0047  0304          "D
0050  0240          240
0051  0301          "A
0052  0240          240
0053  0325          "U
0054  0316          "N
0055  0311          "I
0056  0324          "T
0057  0240          240
0060  0261          "1
```

```
PAGE 02

0061  0240   240
0062  0324  "T
0063  0301  "A
0064  0320  "P
0065  0305  "E
0066  0240   240
0067  0317  "O
0070  0316  "N
0071  0240   240
0072  0325  "U
0073  0316  "N
0074  0311  "I
0075  0324  "T
0076  0240   240
0077  0260  "0
0100  0215   215
0101  0212   212
0102  0212   212
0103  0212   212
0104  0212   212
0105  0212   212
0106  0212  END, 212
```

CHAPTER 3

BOOT - CPS CONTROL PROGRAM LOADER

3.1 Introduction

The source code for the CPS Control Program Loader (BOOT) is stored in the file "BOOT". BOOT serves 3 purposes: (1) the code starting at Ø76ØØ loads and starts the CPS Control Program; (2) the read-only tape routine within BOOT is used by the core-image loader, CILDR; (see CILDR writeup, Chapter 4); (3) several locations within BOOT, starting at Ø7774, are used by CILDR to put a core image program into execution.

BOOT rèsides on tape block 451 of tape unit Ø and is read into core under 2 circumstances. (1) When the LOAD switch is lifted, the cold start loader reads BOOT into core. (2) Every time the user issues a RUN command, the copy of BOOT which read in the Control Program is refreshed as a precautionary measure. The BOOT code was origined at 200 to allow the binary loader to load it, but it is page relocatable and resides at 07600 without any fuss.

3.2 Control Program Loading

The first part of BOOT (Ø76ØØ - Ø762Ø) consists of 3 calls to the read-only tape routine which load the Control Program into core. The utility subprograms and tape unit 1 file index (or VTOC - volume table of contents) are first loaded into field 1, followed by the tape

unit Ø file index which also goes into field 1. The third call loads field Ø with the I/O controller and Control Program field Ø module. The Control Program is then entered via a JMP instruction.

3.3 Read-only Tape Routine

The read-only tape routine will read from either tape drive (Ø or 1) into either memory field (Ø or 1). The calling sequence is

```
JMS RDTAPE
tape block number
# of blocks
100 * field + tape unit
starting memory location
return call + 5 when done
```

This says, when executed, to read from the specified tape unit starting at the specified block number the specified number of blocks (128 words per block), depositing the blocks in core in the specified memory field starting at the specified memory location within that bank.

The tape routine resembles one written by R. J. Clayton of DEC for the LINC-8, and is very close in structure and philosophy to DECUS L-62 submitted by one of the authors (Metzger). After making copies of the calling parameters, the routine moves the tape toward the desired block via the code starting at SERCHA. When the tape is on block, THERE is entered and if the drive is moving forward, data is transferred via the code starting at RDRA. After all specified

blocks have been read, control returns to the caller.

3.4 Core Image Program Startup Locations

Locations Ø7774 - Ø7777 are used by the core image program loader (CILDR) to set a program into execution after it has been loaded into core memory. CILDR diddles the read-only tape routine return address to cause control to return to location Ø7774 after the last portion of the core image program has been read in. At 7774, the proper data and instruction memory fields are set up and a jump is made to the starting address previously inserted by CILDR into STADR, locn 7776. A halt instruction is provided in location 7777; if the user wishes the CPU to halt after loading the core image, he may specify 7777 as a starting address.

3.5 Listing

An annotated assembly listing for CILDR follows. This listing was obtained by issuing the CPS command

```
RUN *ASM BOOT    P=L
```

```
                    /SYSTEM BOOTSTRAP AND LOADER ROUTINE
                    /RESIDES AT 7600
                    *200 /FOR BIN LOADER
                    /
0200  4221  BOOT,   JMS RDTAPE
0201  0150          150 /UNIT 1 UTILITIES AND VTOC  BLKNO 150
0202  0030          030 /30 blocks
0203  0101          101 /unit 1, ~~unused~~ field 1
0204  0000          000 / location 0
0205  4221          JMS RDTAPE
0206  0400          400 /UNIT 0 VTOC / block 400
0207  0010          010 / 10 blocks
0210  0100          100 / unit 0, field 1
0211  2000          2000 / location 2000
0212  4221          JMS RDTAPE
0213  0410          410 /CPS  block 410
0214  0037          037 / 37 blocks
0215  0000          000 / unit 0, field 0
0216  0000          000 / location 0
0217  5620          JMP I .+1
0220  2000          SYSTEM   Put Control Program Prologue into execution
                    /
                    /
                    /READ-ONLY TAPE ROUTINE
                    /GNC 12/17/68
                    /
0221  0000  RDTAPE, 0   / must be called from field 0 !
0222  7300          CLA CLL
0223  1621          TAD I RDTAPE   Copy Tape Block #
0224  3366          DCA BLN
0225  2221          ISZ RDTAPE
0226  1621          TAD I RDTAPE   get # of blocks
0227  7041          CIA
0230  3367          DCA NUMB
0231  2221          ISZ RDTAPE
0232  1621          TAD I RDTAPE   get unit and field word
0233  3371          DCA LOC
0234  2373          ISZ CNTR /LET TAPE DRIVE SETTLE
0235  5232          JMP .-3   / necessary due to mechanical limitation of tape drive
0236  1371          TAD LOC
0237  7112          CLL RTR   set up word to select
0240  1363          TAD C2    tape unit
0241  3370          DCA UNIT
0242  1371          TAD LOC
0243  0271          AND MASK  set up CDF for
0244  7112          CLL RTR   destination field
0245  7010          RAR
0246  1326          TAD CDFD  note use of SMA as a mask
0247  3324          DCA RCDF
0250  2221          ISZ RDTAPE
0251  1621          TAD I RDTAPE   get starting location
0252  3371          DCA LOC
0253  2221          ISZ RDTAPE
0254  7120          STL            start tape
```

PAGE 02

```
0255  2366  SERCHA, ISZ BLN / block # is in 1's compliment on tape
0256  3372  SERCHB, DCA CSUM / checksum = 0
0257  1345   TAD C7600
0260  3373   DCA CNTR  set up word counter (200g words/block, -200g = 7600g)
0261  1370   TAD UNIT
0262  6141   ICON
0263  7201   CLA IAC   start proper tape
0264  7430   SZL       drive in backwards direction
0265  6141   ICON
0266  1330   TAD C6
0267  6141  B, ICON    set tape motion
0270  4344  A, JMS WAIT await block # from tape
0271  7500  MASK, SMA  / check to see if
0272  7120   STL       / we're there
0273  1366   TAD BLN
0274  7650   SNA CLA
0275  5311   JMP THERE / if desired block # found
0276  6147   INTS      / desired block not yet
0277  7010   RAR       / under head - set tape
0300  0364   AND M4000 / direction
0301  7460   SZA SNL
0302  5270   JMP A
0303  7020   CML
0304  7520   SNL SMA
0305  5270   JMP A
0306  6141   ICON
0307  7001   IAC
0310  5267   JMP B
0311  6147  THERE, INTS / have found block - are we going forward?
0312  7012   RTR
0313  7620   SNL CLA
0314  5270   JMP A / no
0315  1365   TAD RFUNCT
0316  6141   ICON      set up to read a block
0317  4344   JMS WAIT
0320  4344  RDTA, JMS WAIT / get next word
0321  1372   TAD CSUM
0322  3372   DCA CSUM / add into checksum
0323  6171   IAAC
0324  6201  RCDF, CDF  / deposit word into core
0325  3771   DCA I LOC /
0326  6201  CDFD, CDF 0/
0327  2371   ISZ LOC
0330  0006  CS, 6 / just in case we skip - "AND 6" does not hurt
0331  2373   ISZ CNTR / done?
0332  5320   JMP RDTA / no
0333  4344   JMS WAIT / yes - get checksum
0334  7041   CIA
0335  1372   TAD CSUM / check it
0336  7650   SNA CLA
0337  5356   JMP WAIT2 / good
0340  1345   TAD C7600  NO GOOD -
0341  1371   TAD LOC      try again
0342  3371   DCA LOC
0343  5256   JMP SERCHB
```

PAGE 03

```
0344  0000  WAIT, 0  / Wait until DEC tape block mark or word mark is
0345  7600  C7600, 7600 /sensed
0346  6147  C7, INTS
0347  7700   SMA CLA
0350  5345   JMP C7600
0351  1346   TAD C7
0352  6141   ICON
0353  7300   CLA CLL
0354  6171   IAAC
0355  5744   JMP I WAIT
0356  7300  WAIT2, CLA CLL  Come here when done with a block
0357  2367   ISZ NUMB
0360  5255   JMP SERCHA /Do another block
0361  6141   ICON /stop tape-done
0362  5621   JMP I RDTAPE /return to caller
0363  0002  C2, 2
0364  4000  M4000, 4000
0365  0003  RFUNCT, 3
0366  0000  BLN, 0
0367  0000  NUMB, 0
0370  0000  UNIT, 0
0371  0000  LOC, 0
0372  0000  CSUM, 0
0373  0000  CNTR, 0
            /
            /
            /LOADER WILL USE THE NEXT 4 LOCNS TO
            /STARTUP A USER PROGRAM
            /
0374  6203  STARTUP, CDF CIF /will be set to starting memory field
0375  5776   JMP I .+1
0376  0377  STADR, NO /start execution
0377  7402  NO, HLT
            SYSTEM=2000
```

CHAPTER 4

CILDR - CPS CORE IMAGE PROGRAM LOADER

4.1 Introduction

The source code for the Core Image Program Loader (CILDR) is stored in the file "CILDR". CILDR is invoked by the RUN command module in the CPS Control Program to load into core and put into execution a program stored in a core-image file. It is resident on tape block 450 of a CPS unit Ø tape and is overlayed on top of page Ø74ØØ of the Control Program when invoked. It operates as follows.

4.2 Program Operation

Before CILDR is put into execution, the RUN command module of the Control Program will deposit 2 words of information describing the core-image file to be run into page zero, bank Ø: 4000* tape unit + block # will go into ØØ134 and file control bits + file length in blocks will go into ØØ135. These two words are copies directly from words 5 and 6 of the file index entry for the core-image file.

The first task done by CILDR after it is invoked is to use the information in 134 and 135 to construct a unit and block number word for the file immediately following on tape the core-image file to be run. The word is deposited for safekeeping into 17773 in the communication area. This information is used in only one case in the

present version of CPS, and that is to invoke the SABR assembler to do pass 2 of a FORTRAN compilation. Of course we have carefully insured that the SABR assembler always directly follows the FORTRAN compiler on tape. (A much more general scheme for transfer of control, allowing both explicit and implicit specification of overlays, has since been worked out, but has not yet been incorporated into the system.)

CILDR now concerns itself with loading the core-image file. Recall that the first tape block in any core-image file contains a table describing the placement of the remaining blocks in core as well as the program starting address. (See Core Image File Structure, Section 7.3 of this volume.) CILDR picks up the tape address of this descriptor block from location 134 and separates the unit and block number into parameter words appropriate for calling RDTAPE, the read-only tape routine within BOOT in page Ø76ØØ. (See BOOT write-up, Chapter 3.) RDTAPE now reads the core-image descriptor block into core page Ø72ØØ.

The code beginning at the symbol LOADER (Ø7425) executes successive calls to RDTAPE to read into core all but the last core subimage. When the descriptor for the last subimage is encountered (the top bit in the first word of a subimage descriptor table entry is set to one), control transfers to LASTMOD, where a call to RDTAPE to read in the last subimage is faked in such a manner as to cause

RDTAPE to return to the code in BOOT which will start the loaded program into execution. The starting memory field and address are deposited into BOOT before RDTAPE is called.

By convention core-image files are structured so that code destined for field zero will always be loaded last. Moreover, since within a memory field code loads from low address to high address, the scheme used in CILDR will allow code to be loaded all the way up to 7677 in bank zero.

4.3 Listing

An annotated assembly listing for CILDR follows. This listing was produced by issuing the CPS command

```
RUN *ASM CILDR    P=L
```

```
                    /CORE IMAGE FILE LOADER - CPS
                    /
                    *7400
7400  7300          CLA CLL
7401  1135          TAD 135 /GET FILELEN
7402  0377          AND (377 /STRIP OFF CRUD
7403  1134          TAD 134 /ADD FILE START BLKNO
7404  6211          CDF 10 /=NEXT FILE FOR SYSTEM
7405  3776          DCA I (7773 /OVERLAYS
7406  6201          CDF
7407  1134          TAD 134 /GET BLN OF CI FILE
7410  7006          RTL
7411  0375          AND (1
7412  3221          DCA DESUNIT / descriptor block tape unit
7413  1134          TAD 134
7414  0374          AND (1777
7415  3217          DCA DESBLK / descriptor table block# on tape
7416  4773          JMS RDTAPE
7417  0000        DESBLK, 0
7420  0001        DESLEN, 1
7421  0000        DESUNIT, 0
7422  7200          7200        / read descriptor table into page 7200
7423  1372          TAD (7177
7424  3260          DCA DESPTR / set descriptor table pointer
7425  2260        LOADER, ISZ DESPTR
7426  1660          TAD I DESPTR
7427  7510          SPA    / is This the last subimage
7430  5261          JMP LASTMOD /yes
7431  3254          DCA MODLEN / non-std call to RDTAPE
7432  2260          ISZ DESPTR
7433  1660          TAD I DESPTR /GET FLD
7434  7106          RTL CLL
7435  7004          RAL /SHIFT FOR RDTAPE CALL
7436  1221          TAD DESUNIT /ADD UNIT
7437  3255          DCA MODUFW  / unit and field word for RDTAPE
7440  2260          ISZ DESPTR
7441  1660          TAD I DESPTR
7442  3256          DCA MODADR
7443  1217          TAD DESBLK / do bookkeeping on tape block numbers
7444  1220          TAD DESLEN
7445  3217          DCA DESBLK
7446  1217          TAD DESBLK
7447  3253          DCA MODBLK
7450  1254          TAD MODLEN / also on subimage length
7451  3220          DCA DESLEN
7452  4773          JMS RDTAPE / read in subimage
7453  0000        MODBLK, 0
7454  0000        MODLEN, 0
7455  0000        MODUFW, 0
7456  0000        MODADR, 0
7457  5225          JMP LOADER
7460  0000        DESPTR, 0
7461  0371        LASTMOD, AND (37 / come here for last subimage
7462  7141          CIA CLL  / # of tape blocks
```

PAGE 02

```
7463  3770   DCA NUMB    / set up parameters in RDTAPE
7464  2260   ISZ DESPTR
7465  1660   TAD I DESPTR
7466  1367   TAD (CDF
7467  3766   DCA RCDF    / set up destination field
7470  1221   TAD DESUNIT
7471  7112   RTR CLL
7472  1365   TAD (2
7473  3764   DCA UNIT    / set up unit word
7474  2260   ISZ DESPTR
7475  1660   TAD I DESPTR
7476  3763   DCA LOC     / set up core location
7477  1217   TAD DESBLK
7500  1220   TAD DESLEN
7501  3762   DCA BLN     / set up tape block #
7502  1361   TAD (STARTUP / fake return address
7503  3773   DCA RDTAPE
7504  2260   ISZ DESPTR
7505  1660   TAD I  DESPTR
7506  1761   TAD STARTUP (CIF CDF / set up starting field
7507  3761   DCA STARTUP
7510  2260   ISZ DESPTR
7511  1660   TAD I DESPTR
7512  3760   DCA STADR   / set up starting address
7513  2322  TIMEOUT, ISZ LOCNTR /LET TAPE SETTLE
7514  5313   JMP TIMEOUT / necessary due to mechanical
7515  2321   ISZ HICNTR  / response time limitations of DEC's tapes
7516  5313   JMP TIMEOUT
7517  5720   JMP I .+1
7520  7654    /go to RDTAPE
7521  7772  HICNTR, -6
7522  0000  LOCNTR, 0
            /
            RDTAPE=7621
            NUMB=7757
            RCDF=7724
            UNIT=7770
            LOC=7771
            BLN=7756
            STARTUP=7774
            STADR=7775
7560  7776  $
7561  7774
7562  7756
7563  7771
7564  7770
7565  0002
7566  7724
7567  6201
7570  7767
7571  0037
7572  7177
7573  7621
7574  1777
7575  0001
```

PAGE 03

7576 7773
7577 0377

CHAPTER 5

CPS CONTROL SYSTEM

CONTENTS

5.1 Program Residency and Loading

The CPS Control Program is the vehicle through which the user controls and directs his programming session. It is loaded into core memory by the system bootstrap loader routine, BOOT (see Chapter 3) and occupies about 7000_8 words of memory in the version of May 21, 1969. The Control Program is broken into 3 parts insofar as its residency on tape is concerned. All code (instructions) which will be loaded into bank 1 sits on a unit 1 tape from block 150 to 155 inclusive; BOOT loads this code into core starting at location 10000. It also reads the VTOC (file index) for unit 1 from blocks 170 - 177 into location 14000 upwards. (The blocks 156 - 167 are transferred to core, but this area is overwritten by a later read from unit 0.) This read operation for the bank 1 code and unit 1 VTOC is performed in a single sweep in order to speed up Control Program loading. BOOT next loads the unit 0 VTOC from blocks 400 - 407 into location 12000 upwards, and finally finishes off by reading the bank 0 portion of the Control Program from unit 0, tape blocks 410 - 446 inclusive, into core starting at location 0. The broad-brush memory map of the Control Program after loading is complete is shown in Fig. 5.1. (A more detailed memory map is given just before the code in Section 5.4.)

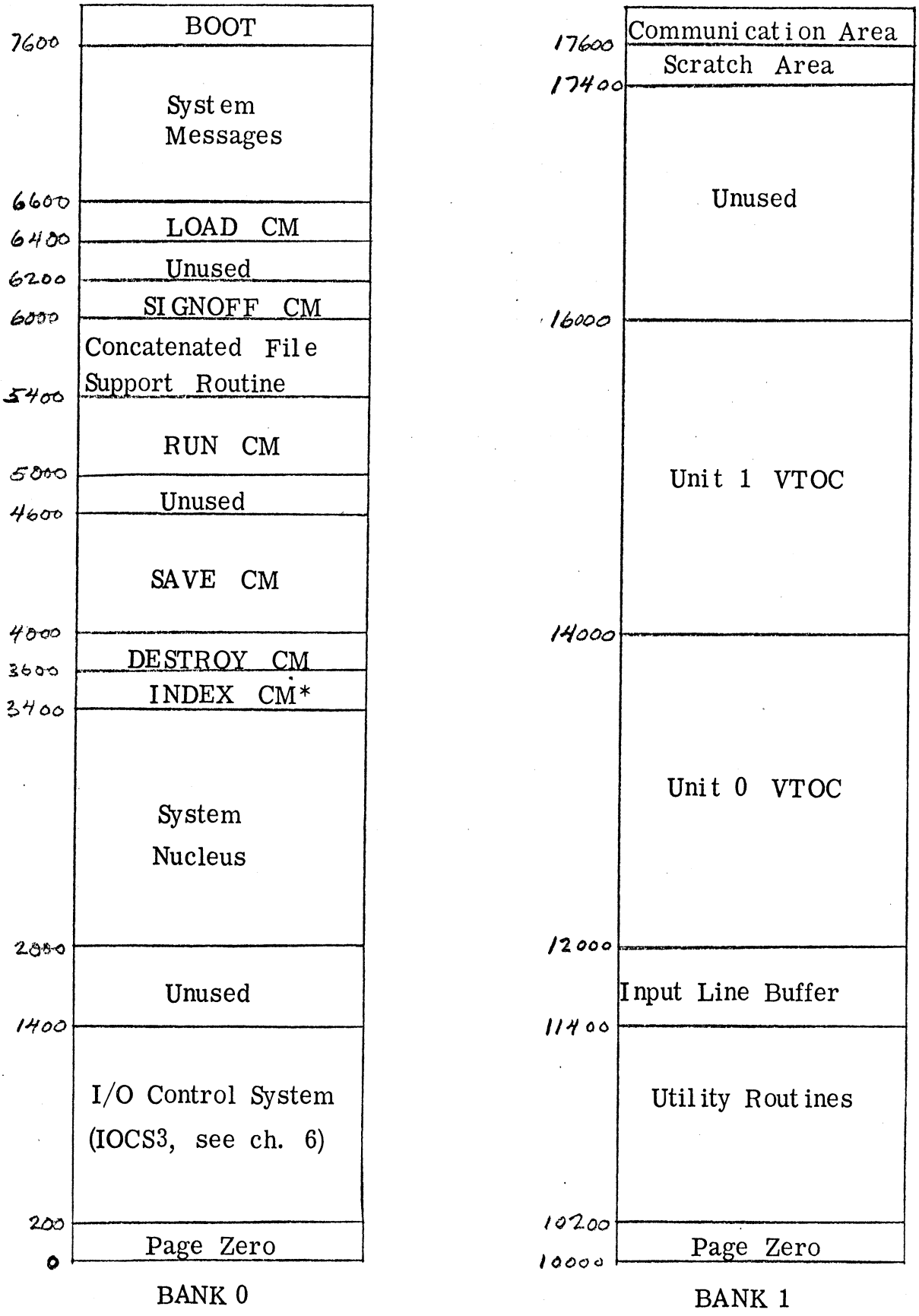


FIG. 5.1. MEMORY MAP of the CONTROL PROGRAM

* CM means COMMAND MODULE

5.2 System Prologue and Command Language Interpreter

BOOT passes control to the system prologue which starts at location 2000. The prologue code initializes the I/O system and sets up the base address on page zero for the system message routine.

The prologue code then checks the key area in the communication area to see if the program which caused the Control Program to be loaded and started has passed any messages. The possible messages in the current version of CPS are (1) a request to type out the system ID ("CPS(MMDDY)") or (2) information about the disposition of -S or -B as files. (See Section 7.4.4 for the structure of the key area and file information areas.) After the prologue code has been completely executed, it is never used again until the system is reloaded.

Control is then given to the command language interpreter (CLI) on page 3000. The CLI calls the standard line-input routine, GETLINE, to obtain a command line from the user. It then scans the command table, COMTBL, in page 10600 to look for a match between the first 3 characters in the line received from GETLINE and an entry in the command table. If a match is found, the CLI jumps to a command service subroutine (whose address was also designated in the COMTBL) which handles the remainder of the input line. If a match is not found, the CLI flags the command line as invalid. After either of the above actions has been completed, the CLI

prompts the user for another input line.

5.3 System Utility Routines

Before studying the action of the various command service subroutines which actually implement the available CPS commands, it will be useful to look at the more complex of the various utility routines within the system. These are covered in this section in the order of the core addresses at which they are situated in the CPS version of 21 May 1969.

5.3.1 IOCS3 - System I/O Supervisor. IOCS3, a generalized I/O package for LINCtapes and 33ASR Teletype, is the I/O supervisor for the control program. All I/O done by the control program is via subroutine calls to IOCS3. IOCS3 is of sufficient complexity so that Chapter 6 in this volume has been entirely devoted to it; see that chapter for further information.

5.3.2 PACK - Pack a File Tape. PACK is located at 2115_8 and is called only by RELTAP (release tape). It is used to move the information on a segment of a file tape further down toward the beginning of the tape - it is this routine which actually copies over a file being destroyed on tape.

PACK sits in the lower memory field because the whole upper memory field is used as a buffer when packing. Thus RELTAP calls PACK to pack a tape; PACK sets up the proper parameters and calls

TPCOPY (tape copy). TPCOPY saves the upper memory field, does a copy operation from the old locations on the tape to the new locations, and then returns to PACK. PACK waits until the upper memory field has been restored, and then returns to RELTAP which is stored in the upper field. If the space occupied by RELTAP were not needed to buffer the tape copying operation, PACK could be eliminated entirely and RELTAP could call TPCOPY directly.

PACK is rather a simple-minded routine. It copies parameters supplied by RELTAP from the upper to the lower memory bank and deposits them into the parameter locations for TPCOPY. It then calls TPCOPY, waits for its operation to be completed, and then returns to RELTAP in the upper field.

5.3.3 Message Printing Routine. The message printing utility routine (MPUR) is located at 2200_8 and is called by PRTNAM (print file name) and SYSMSG (system message routine). The MPUR has two entry points, MESAGI and CELMSG. If called at MESAGI, the routine behaves exactly as DIGITAL-8-18-U. If called at CELMSG, it is assumed that the address of a 2 character-per-word message to be typed is in the accumulator. The internal logic of the routine is very close to that of DIGITAL-8-18-U except for some switching necessary to insure that the routine returns through the proper entry point.

5.3.4 TPCOPY - Tape Copying Routine. TPCOPY is located at 2262_8 and is called by SAVE and PACK. It is a general-purpose tape copying routine and may be used to copy any number of blocks from any tape location to another. The user specifies the source and sink tape locations (i.e., unit and block number) and the number of blocks to copy by depositing them into locations reserved for the purpose on page zero. (See the listing at the end of the chapter for particulars.) TPCOPY is then entered via a JMS.

TPCOPY uses all 40_8 pages in memory field one for a buffer and so must first save field one in the stack area--tape unit 1, blocks 200 - 237 inclusive. The routine then proceeds to copy from the source area to the sink area, transferring up to 40_8 blocks at a time. The actual tape operations are handled by calls to IOCS3 however. Every block transferred is read after it has been written to see if it checksums properly. After the transfer is complete, TPCOPY initiates the tape operation to read field one back into core and returns to the caller. Note that this last tape operation will not have been completed when TPCOPY returns.

5.3.5 GETLINE - General Line Input Routine. GETLINE occupies page 2400 and part of page 2600. In addition it needs a line buffer of N words where -N+1 is specified by the contents of location 00110_8 and the buffer address is in locations 00101_8 and 00102_8. Its only entry point is at 2400_8. It is called by CLI, the

command language interpreter, and QUIZUSER, a routine which solicits replies from the user to questions. GETLINE is the only line input routine in CPS; it does all input line editing, continuation, and the like, and in general is charged with presenting its caller with a cleaned-up input line in the line buffer.

In order to use GETLINE, the caller must set the buffer length and buffer address into the page zero locations previously mentioned. He then does a subroutine jump to location 2400. Upon return, the line buffer will contain up to N-1 characters, which were input by the user from the Teletype, in full 8-bit ASCII form, 200_8 bit set, one character per word. If the user input fewer than N-1 characters, the buffer will be padded to the right with blanks. In any event the very last word in the buffer will contain a zero -- 000_8 -- in order to mark the end of the buffer and eliminate the need for keeping character counts when scanning the input line.

The operation of GETLINE is as follows. The CDF instruction stored in 102_8 is put into the 3 locations within the routine where it will be needed. The line buffer is blanked and the line pointer set to zero. The main character handling loop is then entered.

In the main loop, a character is obtained from the IOCS3 teleprinter buffer, its 200_8 bit is set, and it is deposited into a temporary. The processor then hops down a long skip chain, testing to see if the character is one of a set of special characters. The

special characters are leader/trailer code (200_8), back arrow, rub-out, line feed, carriage return, device control 0 (ctrl P), minus sign, and data separator (ctrl Y). The action of these various characters is described in the following paragraphs.

If the input character is leader/trailer (200_8) code, then GETLINE echos the character to the teleprinter. (This echoing is in keeping with the design assumption that whenever the user strikes any key, the teleprinter will cycle to let the user know the control program is alive.) After echoing however, control returns immediately to the main loop, thus effectively ignoring the leader/trailer character.

If a back arrow is input, GETLINE echos the back arrow, issues a carriage return, line feed, and prefix character, and then jumps to the GETLINE initialization routine, thus effectively erasing the whole line which has been input up to the back arrow.

Rubout is used to erase the last character input. If rubout is encountered, GETLINE tests the line pointer for the current line. If the line pointer indicates that no characters have been input on the current line, the main character handling loop is entered. Thus hitting a rubout to erase characters from an empty line results in no erasure and no echo character to the teleprinter. This is the only case where no character is echoed for a key depression, but it serves to convey the information to the user that the line is empty,

and we feel that because it conveys information, it is justified. If, on the other hand, at least one character is currently in the input line, a blank is deposited over the top of the last character typed, thus erasing it, and a reverse slash, \ , is echoed to indicate the erasure. The line pointer is then decremented by one so that the next character input will overwrite the blank just deposited in the buffer.

The line feed is used to indicate the user's desire to see an echo of the whole line he has input to date. GETLINE starts a new line on the teleprinter with the prefix character "=" and then prints the contents of the line buffer. There is a difficulty in this last section of the code in that the routine does not count the characters in the line buffer as it prints them. Thus if there are more than 71 characters to be echoed, the teleprinter will overprint at the end of the line. The fix for this condition, though obvious, has never been put into the code.

The "literal next" character is device control 0 (DC0). Whenever a DC0 character is encountered, the literal-next switch - LNSW - is set to -1 and the main loop is re-entered. Since LNSW is set, the next character retrieved from the IOCS3 input buffer will be put into the line being built in the line buffer no matter what it is. After this next character has been inserted, the input stream will again be tested for the special characters listed above.

The data separator character serves to indicate an end of file condition. Wherever it is encountered, the message "END OF FILE" is echoed to the teleprinter, and a return is made to where GETLINE was called plus two locations, to indicate the end-of-file condition.

If the carriage return character is encountered, a check is made to see if the preceding character was a minus sign. If it was, the line pointer is decremented by one so that the next input character will overwrite the minus sign in the buffer. The teleprinter is then issued a carriage return, a line feed, and a prefix character, and the main loop is re-entered. Thus a minus sign-carriage return combination continues the current line being built in the line buffer, yet allows the user the convenience of typing on a new line on the teleprinter thus avoiding pile up of characters at the physical end of the printer line. If a minus sign did not precede the carriage return, then a 200_8 character is echoed to the teletype, the number of characters in the line buffer is loaded into the AC, and a return is made to where GETLINE was called plus one location.

If the input character does not fall into the class of special characters, it is simply inserted into the next available position in the line buffer after being echoed to the teleprinter.

5.3.6 QUIZUSER - A Routine to Ask the User a Question.

QUIZUSER is called with a system message number in the accumulator.

QUIZUSER uses EOL and SYSMSG to output this message on a new teleprinter line and then prompts the user with a question mark at the beginning of another teleprinter line. GETLINE is then called to get an input reply line from the user into the reply buffer of 128 characters located at 17400. If GETLINE returns an end-of-file or zero characters input, control is returned to the command language interpreter. Otherwise control is returned to the caller. In either case, the input line buffer is restored to location 11400 before exiting.

5.3.7 PRTNAM - Print a File Name. PRTNAM is used to print a file name which is stored in stripped ASCII form in locations 10020 - 10023. It uses MESAGI, the message printing routine of Section 5.3.3, to print the file name after first transferring it down to within PRTNAM in field zero. (MESAGI will only print messages contained in field zero.) There is a mistake in this routine, in that if a user inserts the illegal character "@" into a file name, the command support routines will try to print the offending file name using PRTNAM. However, "@" encodes to zeros in stripped ASCII so that MESAGI terminates printing of the name prematurely, and returns to execute portions of the name as instructions. The usual result is that the CPU grinds to a halt on page zero. Of course the system may be reloaded, but such is hardly the proper fix. Again this is one of those small errors which has not yet been fixed.

5.3.8 NDXPRT - File Index Printing Routine. NDXPRT is used to print the names of files which are catalogued in each VTOC. Before doing a JMS to NDXPRT, the user must set up a number of switches on page zero. SW0 and SW1 must be set to minus one or zero according to whether the indexes for tape units zero and one are, or are not, to be printed. If ALLFLG is set to -1, then all file names, including system files and working areas, will be printed. If it is zero, then only user file names will be printed. If DETAIL is zero, then only file names will be printed, and if it is minus one, then file names and words 5, 6, and 7 of the FCB will be printed in octal.

5.3.9 GETFD - Find a Control Block for a File/Device. GETFD is located at 10200_8 and may be called from either memory field. When invoked it searches the unit zero and unit one VTOCs as well as an internal device table for a match to the stripped ASCII file or device (FD) name stored in locations 10020 - 10023. If a match is found (that is, the FD name is "known" to the system), then a return is made to the GETFD call plus one with the address of the FD control block (stored in the VTOC or device table) in the accumulator. If no match is found, the FD does not exist, and a return is made to the GETFD call plus two.

GETFD uses the routine SRCHVTOC to search the unit zero and unit one VTOCS. If no match is found in the VTOC search, a

search of the device table within GETFD is done. Note that the device table is structured as a set of device control block entries and hence may be supplemented at any time by inserting additional device control blocks and increasing the number of devices, NDEVS.

5.3.10 MAKFIL - Create a File Entry in a VTOC. MAKFIL is located at 10275_8 and is used to insert an entry for a file into a VTOC. Before calling MAKFIL, the caller must construct a virtually complete file control block in 10020 - 10027 for the file to be created:

Locn	Contents
10020 - 10023	Eight character file name in stripped ASCII
10024	Unused when MAKFIL is called. Upon return from MAKFIL, this location contains the unit and tape block number where the file will be situated on tape.
10025	File length and switches word for the file to be created (See vol. 1, Section 7.2, for the structure of this word.)
10026	File contents attribute word for the file to be created. (See vol. 1, Section 7.2 for the structure of this word.)
10027	Unused at present; however, this word does get copied into the FCB entry which is created in the VTOC.

In addition, the user must specify in location 10000 upon which tape drive the file is to be created (unit 0 or 1).

Note: MAKFIL simply creates a file control block entry in the VTOC for the specified tape unit; it does not copy any information into the file on tape.

If MAKFIL is able to create the file as specified in the partial FCB in 10020 - 27, it returns to where it was called plus one location. If something has gone wrong (no space left on the tape or in the VTOC), it error returns to where it was called plus two.

MAKFIL first sets up pointers to the desired VTOC by calling GABUV. It then checks to see if space is available in the VTOC for at least one more file control block. If no space is available, an error return is made. If space is available in the VTOC, then MAKFIL calls GETTAP to get tape space for the file. If no space is available, an error return is made. Otherwise GETTAP returns the beginning block number of the tape space which was allocated. This block number is used to form the fifth word (unit and block number) of the file control block at 10020 in order to complete it. The completed file control block for the file is then copied into the VTOC, the VTOC is written onto tape to reflect the addition of a file, and control is returned to the caller.

5.3.11 COMPRESS - Squeeze Characters from One to Two per Word. COMPRESS is located at 10400 and is used to encode string of 8-bit ASCII characters into N words of stripped, 6-bit ASCII characters with trailing blanks if necessary. N is a calling parameter. COMPRESS works from an array of 8-bit ASCII characters and outputs encoded characters to another array; both arrays must be in the same memory bank. COMPRESS is used to convert file names supplied by the user on an input line into internal form, suitable for use as a comparand, for example, when searching a VTOC. Because the routine is used for this purpose, it stops taking characters from the input string when it encounters a blank. If less than 2N characters have been taken from the input string at this time, then the output string is padded to the right with stripped blanks (40_8).

5.3.12 DESTRY - Destroy a File. DESTRY is located at 10462_8 and is used to destroy a file. It must be called with a pointer to the FCB of the file to be destroyed in the AC.

The algorithm used is straightforward. First the count of the number of entries in the VTOC is decreased by one. (The stored word is incremented by one since the count is kept as a negative number.) Subroutine RELTAP is then called to release the tape space occupied by the file; the tape unit, beginning file block number, and the length of the file in blocks are supplied as parameters to RELTAP. The VTOC is then packed up by moving all FCB's which

had higher core addresses than the FCB for the destroyed file down in core by 8 locations (i.e., the length of the deleted FCB). Note that during this moving process, the "first block" word in every FCB which is moved is decreased by the length of the destroyed file, in order to reflect the fact that RELTAP packed the file tape. After the VTOC is packed, it is written out via the routine WVTOC, and control is then returned to the caller.

5.3.13 SRCHVTOC - Search a VTOC. SRCHVTOC is located at 11000_8, and is used to search a VTOC for a file control block, given the file name in stripped ASCII in locations 10020_8 - 10023_8. The tape unit number whose VTOC is to be searched is deposited in location 10000 before SRCHVTOC is called.

The logic of the routine is straightforward. GABUV is called to set up address pointers for the search, and a counter is initialized to the negative of the number of valid file control blocks in the VTOC. The first four words of each FCB are then compared against the user-supplied file name using the routine CLW. If a match is found, control is returned to the caller at the SRCHVTOC call plus one, with a pointer to the found FCB in the AC. If no match is found, a return is made to the SRCHVTOC call plus two.

5.3.14 OPSCAN - Find Next Operand. OPSCAN, located at 11067_8, is used to advance a line pointer at 105_8 to point to the next operand in an input line. In a line, an operand string is a group of

non-blank characters with at least one blank at its left and right ends to delimit it from other operand strings. When OPSCAN is invoked, it presumes that location 105 is pointing to some character within an operand string. It advances the line pointer until it encounters a blank; the pointer is then further advanced until it points to a non-blank character, which is presumed to start an operand string.

If, during the line scanning process, an all-zero character (line terminator) is encountered, a return is made to the OPSCAN call plus two. Otherwise the return is made to the call plus one.

5.3.15 GETTAP - Allocate Tape Space. GETTAP is located at 11200_8, and is used to allocate tape storage for a file. It is the only routine used for this purpose.

Since tape storage is organized in a sequential manner (see Chapter 7) allocation of space for a file consists of the following three steps: 1) Save the tape address of the "first free block" on the tape in order to pass it back to the user when finished. 2) Update the "first free block" address by adding to it the number of blocks allocated. 3) Decrease the "number of blocks left" word by the number of blocks allocated.

5.3.16 RELTAP - Deallocate Tape Space. RELTAP, located at 11230_8, is used to deallocate tape storage space which was previously allocated by GETTAP (see preceding section). As is

common in storage allocation systems, one pays the piper here for an easy allocation procedure by being forced to live with a more complex deallocation procedure.

RELTAP is responsible for packing the filing tape so that deallocating space in the middle of the filing area does not leave a "hole" in the filing area at that point. It does this by computing information for the PACK subroutine (see Section 5.3.2) and then calling PACK if in fact the deallocated space was not at the end of the filing area. In the process, it updates the "first free block" address word and the "number of free blocks" word in the VTOC for the tape.

5.4 Detailed Memory Allocation

A listing of the significant entry points in the system nucleus (which is comprised solely of utility routines) according to the page upon which they are located is as follows.

Page	Entry Point Names
2000	SYSTART, PACK
2200	MESAGI, CELMSG, TPCOPY
2400	GETLINE
2600	QUIZUSER, SYSMSG
3000	CLI, OCTOUT, PUSHPFX, EOL, POPPFX, PRTNAM
3200	NDXPRT
3400	TPERR

Page	Entry Point Names
10200	GETFD, MAKFIL
10400	COMPRESS, DESTRY
11000	SRCHVTOC, GABUV, OPSCAN, CLW
11200	GETTAP, RELTAP, WVTOC

An alphabetical listing of these subroutines along with their calling sequences is given in Section 5.5.

A listing of the various command modules and the pages they occupy is given below:

Command	Module Name	Location
COMMENT	COMMENT	3342-3343
INDEX	INDEX	3400-3457
DESTROY	DESTROY	3600-3777
SAVE	SAVE	4000-4577
RUN	RUN	5000-5377
RUN	CONSCAN	5400-5777
SIGNOFF	SIG	6000-6177
LOAD	LOAD	6400-6577

5.5 Calling Sequences for Utility Routines

CELMSG

Purpose: to print a string of characters on the teleprinter. Note that the message string format is exactly the same as that of the DEC message routine Digital 8-18U, but that the message string is not in line with the subroutine call.

Calling Sequence:

```
JMS CELMSG   /with address of message in AC
return call + 1
```

CLI

Purpose: to get and interpret an input command. See Section 5.3.2.

Calling Sequence: none to speak of - just jump to CLI.

CLW

Purpose: to perform the analogous function on the PDP-8 as the Compare Logical Characters command on the IBM /360. CLW will compare two lists of words in the same memory bank for equality.

Calling Sequence:

```
JMS CLW   /with length of list in AC
```

Address of the first list here /lists are assumed in the
Address of the second list here /memfld given by:
CDF instruction for list memfld
return call + 4 if unequal
return call + 5 if equal

COMPRESS

Purpose: to compress a string of up to N non-blank 8-bit ASCII characters into their 6-bit form, formed by removing the two high order bits of each character and packing the resultant 6 bit characters two per PDP-8 word. If fewer than N non-blank characters can be found in the ASCII string, the remainder of the output string is padded with blanks.

Calling Sequence:

JMS COMPRESS /with N in AC ; N must be even!
Address of source list
Address of packed sink list
CDF instruction for BOTH lists (Both lists in same bank)
return call + 4

CONSCAN

Purpose: to build a "file or device linkage chain", given the address of a string of filenames. The caller also supplies

the address of the destination area in field one where the chain is to be put, as well as a direction of transfer mask. The direction of transfer mask is used to enforce a restriction that all files whose names are in the name string are either information sources or information sinks. See Section 7.4 for the structure of the linkage chain.

This subroutine is used to build the linkage chain in the I/O communication area in field one when the user issues a RUN command.

Calling Sequence:

JMS CONSCAN

Address of linkage chain in field one

Address of file name string in field one

Direction of transfer mask - 2000 for output
4000 for input

return call + 4

Note: CONSCAN prompts the user for replacement of bad file names.

DESTRY

Purpose: to destroy a file and release the tape space occupied by the file back to the system tape space allocator.

Calling Sequence:

JMS DESTRY /with pointer to FCB of file to be destroyed
/in the AC. The pointer should have been
/gotten via a call to SRCHVTOC or GETFD.

return call + 1

EOL

Purpose: to output the CR-LF combination and the current prefix character for the next line. The current prefix character is found in memory location 100.

Calling Sequence:

JMS EOL

return call + 1 with clear AC

GABUV

Purpose: to Get Address, Buffer and Unit words for a VTOC. The tape unit number must be stored in location 10000 before calling the routine.

Calling Sequence:

JMS GABUV

return call + 1 with clear AC, Address etc. words set on page 10000

GETFD

Purpose: to get a FCB pointer to the file or DCB pointer to the device whose name is stored in packed form in locations 10020 - 10023. All control blocks are stored in the upper memory bank.

Calling Sequence:

JMS GETFD /with file/device name in 10020-23

returns call + 1 if FD exists, with FCB ptr in AC

returns call + 2 if FD does not exist.

GETLINE

Purpose: to get an input line from the user via the keyboard and edit it into compact form for the command language interpreter. The input characters are stored at the buffer address specified on page zero - see Section 5.3.5.

Calling Sequence:

JMS GETLINE

returns call + 1 with number of chars in line in AC

returns call + 2 with clear AC upon end of file (ctl Y struck on 33ASR).

Note: GETLINE allows the user to edit his input lines using much the same conventions as FOCAL. The editing characters and their functions are:

(back arrow) - delete the current line and start a new one.

RUBOUT - delete the last character in the current line.

LINE FEED - echo the contents of the current input line, and then append more text to it.

CARRIAGE RETURN - the user is finished with the current line and wishes to have it returned to the caller.

GETLINE uses IOCS3 and hence can take paper tape input. It also converts all input characters to have value $\geq 200_8$ thus allowing paper tapes to be prepared off-line on a parity-producing teleprinter.

GETTAP

Purpose: to get tape space for a file from the system allocator. Note that the number of blocks needed must be in location 10001 and the tape unit upon which they are needed must be in location 10000 before the subroutine is called.

Calling Sequence:

JMS GETTAP

returns call + 1 with tape blk no in AC if space was allocated

returns call + 2 if not enough room remains on the tape.

MAKFIL

Purpose: to get tape space from the system space allocator and make an appropriate entry into the tape unit VTOC for a new file. The caller must have set up a partial VTOC entry in locns 10020-10027, consisting of the name of the file and the file length and file descriptor words (see Chapter 7). Upon exit from the routine, the VTOC entry at 10020 will be complete, and may be used as a FCB for the newly-created file. The unit upon which the file is to be created should be stored in locn 10000.

Calling Sequence:

JMS MAKFIL

returns call + 1 if all OK

returns call + 2 if no room for the file

MESAGI

Purpose: print a message on the teleprinter in the same manner as the DEC routine Digital 8-18U. Note that the routine uses IOCS3.

Calling Sequence: same as Digital 8-18U.

NDXPRT

Purpose: to print information from the file index of a CPS tape. The caller must set several switches on page zero before calling this routine (see Section 5.3.8).

Calling Sequence:

```
JMS NDXPRT
return call + 1
```

OCTOUT

Purpose: to print a number in octal on the teleprinter.

Calling Sequence:

```
JMS OCTOUT  / with AC containing the number to be
            / printed
return call + 1
```

OPSCAN

Purpose: to advance a pointer to the next operand in the input line buffer. The operand pointer is by convention stored at location 105_8 (see Section 5.3.14).

Calling Sequence:

```
JMS OPSCAN
return call + 1 with updated operand pointer in 00105 and
    clear AC
return call + 2 if end of line mark encountered.
```

PACK

Purpose: to pack-up a tape used for filing purposes, so that there is no wastage of tape space and so that spooling the tape to the starting block of a file is done as fast as possible. Note that in CPS, all file allocation on a tape is done sequentially and thus the pack routine must be called each time a file is destroyed, but after the file entry has been deleted from the VTOC of the tape unit by the routine DESTRY. Upon calling pack, the unit number of the tape unit to be packed must be in loc 10000. Other parameters must also be set in the upper memory field; see the listing in Section 5.6.

Calling Sequence:

```
JMS PACK
return call + 1
```

POPPFX

Purpose: to undo the work of PUSHPFX; i.e., to restore the current prefix character from the stack.

Calling Sequence:

```
JMS POPPFX
return call + 1 with clear AC
```

PRTNAM

Purpose: to print a compressed file name, which is assumed to be in locations 10020 - 10023. Trailing blanks are printed.

Calling Sequence:

```
JMS PRTNAM
return call + 1 with clear AC
```

PUSHPFX

Purpose: to save the current prefix character and substitute the caller-provided character for it. One level of push-down is provided, and if PUSHPFX is called twice in a row, the first prefix character is lost.

Calling Sequence:

```
JMS PUSHPFX  /with new 8 bit pfx char in AC
return call + 1 with clear AC
```

QUIZUSER

Purpose: to ask the user a question, and allow him to answer. Upon return from this routine, the user's answer will be found in memory locations 17600 - 17777.

Calling Sequence:

```
JMS QUIZUSER  /with SYSMSG number of question
              /message in AC
```

return call + 1 with number of chars in answer in AC

if line not empty

return to CLI if line is empty

RELTAP

Purpose: to release file space previously acquired back to the system allocator. Upon entering this routine, loc 10000 must contain the tape unit upon which the storage to be freed was allocated, loc 10001 must contain the number of blocks to be given back, and location 10002 must contain the starting block number of the area to be returned.

Calling Sequence:

JMS RELTAP

return call + 1

SRCHVTOC

Purpose: to search a VTOC for the occurrence of a file name and to return a FCB (file control block) pointer for the file if it is found. The name to be searched for must be stored in compressed form in locns 10020 - 10023 upon entering the routine. The tape unit number whose VTOC is to be searched must be in loc 10000.

Calling Sequence:

JMS SRCHVTOC

returns call + 1 with FCB location in AC if found

returns call + 2 if entry not found for this file in this VTOC.

SYSMSG

Purpose: to print a standard system message on the teleprinter. No carriage return or line feed is put out (unless part of the desired message).

Calling Sequence:

JMS SYSMSG /with number of the message in AC

returns call + 1 with clear AC

SYSTART

Purpose: initial system entry and startup point (see Section 5.2).

Calling Sequence: None - just jump to SYSTART.

TPCOPY

Purpose: general tape copying utility subroutine (see Section 5.3.4).

Calling Sequence:

TPCOPY uses various page zero locns

for parameters:

113 - source tape unit

114 - source blk #

115 - source length in blks

116 - sink tape unit

117 - sink beginning blk #

Note that these locns are not preserved across the call.

JMS TPCOPY /with above locations set

return call + 1

TPERR

Purpose: to process permanent checksum errors on magnetic tape. Called by IOCS3 only.

WVTOC

Purpose: to write out onto the appropriate tape unit an updated VTOC for that tape unit. Upon calling the routine, the tape unit number should be in location 10000.

Calling Sequence:

JMS WVTOC

return call + 1

5.6 Program Listing

A listing of the May 21, 1969 version of the control program follows this page. The listing for field zero precedes the listing for field one.

```
                    /PAGE ZERO DEFNS FOR CPS
                    /
                    *41
0041  3461          TPERR /TAPE ERROR HANDLER ADR FOR IOCS3
                    /
                    *100
0100  0000  PFX, 0 /PREFIX CHAR
0101  1377  ALBUFF, 1377 /ADR OF LINEBUFFER -1 - linebuffer sits in Bank 1
0102  6211  BDF, CDF 10 /LINE BUFFER CDF
0103  7000  NOP /TAKES UP ROOM BECAUSE OF SYSTEM CHANGES 5/20/69
0104  0000  CHAR, 0 /temporary
0105  0000  OPPTR, 0 /OPERAND PTR
0106  0000  LASTCHAR, 0 /temporary
0107  0000  ADDR, 0 /line buffer address
0110  7401  LINLEN, -377 /LAST 5 LOCS USED BY GETLINE
0111  0000  LNSW, 0 /USED BY GETLINE
0112  0000  LPTR, 0 /"  "  "
0113  0000  EEUNIT, 0   source tape unit
0114  0000  EEBLKNO, 0  source block #
0115  0000  ELEN, 0     # of blocks to copy
0116  0000  KKUNIT, 0   sink tape unit
0117  0000  KKBLKNO, 0  sink block #
0120  0000  NBLKS, 0    Temporary
                    } Used by TPCOPY routine
0121  0000  EFCB, 0 /USED BY SAVE & RELEASE
0122  0000  SAVSW, 0 /"  "  "  "  "
0123  0000  SAVTEMP, 0
0124  0000  SW0, 0
0125  0000  SW1, 0   } 2 entry array of switches used by NDXPRT to indicate which VTOC entries are to be printed
0126  0000  PASCNT, 0
0127  0000  OPFLAG, 0
0130  0000  ALLFLG, 0
0131  0000  VTOCAD, 0
0132  0000  VTOCNT, 0   } used by NDXPRT - index printing routine
0133  0000  PFXSTACK, 0 / used by PUSHPFX and POPPFX for last prefix char saving
0134  0000  RBLK, 0
0135  0000  RLEN, 0     / holds unit, blk# & length of file to be run
0136  0000  DUNIT, 0
0137  0000  LUNIT, 0    / used by RUN for logical IO unit scanning
0140  0000  LKAREA, 0
0141  0000  FNAMPTR, 0
0142  0000  PFXTEM, 0 / temp used by PUSHPFX
0143  0000  QTEMP, 0
0144  0000  MSGBAS, 0 /USED BY SYSMSG to index into message pointer table
0145  0000  DETAIL, 0
0146  0000  SADR, 0
0147  0000  KADR, 0   } used by NDXPRT - index printing routine
```

PAGE 02

```
                *2000
                /SYSTEM STARTUP  Come here from BOOT
                /
2000  4442    SYSTART, CALL RESET /set up IOCS3 (see chapter 6)
2001  1377     TAD ("# / set up Control Program prefix character
2002  3100     DCA 100
2003  1376     TAD (MSG-1
2004  3144     DCA MSGBASE/ set up base of message table for system message routine
2005  4447     RDROPN /OPEN READER/KBD
2006  1375     TAD (7773 /CHECK KEY TO SEE
2007  3010     DCA 10 /IF WE CAME IN FROM
2010  6211     CDF 10 /A COLD START
2011  1410     TAD I 10
2012  7001     IAC
2013  7450     SNA /SKP IF NOT CLDSTART
2014  5310     JMP IDY /IDENTIFY- type out CPS(MMODY)
2015  1374     TAD (7401 / otherwise check to see if KEY is present
2016  7640     SZA CLA
2017  5306     JMP PFXOUT /KEY NOT FOUND
2020  7001     IAC
2021  1410     TAD I 10
2022  7640     SZA CLA
2023  5306     JMP PFXOUT /KEY NOT FOUND
2024  7240     STA
2025  1410     TAD I 10
2026  7640     SZA CLA
2027  5306     JMP PFXOUT /KEY NOT FOUND
2030  1373     TAD (-HLT
2031  1410     TAD I 10
2032  7640     SZA CLA
2033  5306     JMP PFXOUT /KEY NOT FOUND
2034  1372     TAD (7764 /KEY FOUND - UPDATE WORKING
2035  3010     DCA 10 /AREA ENTRIES IN VTOCS; check -S first; 7764 is ptr to -S update area
2036  1371     TAD (2013/ 2013 is ptr to -S in VTOC of unit 0
2037  3011     DCA 11
2040  1410     TAD I 10 / get first word - blk # of -S
2041  7450     SNA      / is it zero?
2042  5260     JMP BPAR/ if zero, -S is not to be updated
2043  7104     CLL RAL / otherwise -
2044  7110     CLL RAR /SET  TOP BIT TO 0 to indicate unit 0
2045  3411     DCA I 11 / put into VTOC
2046  1410     TAD I 10 / Get length of -S in blocks & clean it up,
2047  0370     AND (377/
2050  1367     TAD (7000 /IN CASE WE GET GRUNGE FOR UPDATING
2051  3411     DCA I 11 / put into VTOC
2052  1410     TAD I 10 / transfer contents attributes word to VTOC
2053  3411     DCA I 11/
2054  3766     DCA I (0 /WRITE VTOC 0
2055  6201     CDF
2056  6212     CIF 10
2057  4765     JMS WVTOC
2060  1364    BPAR, TAD (4013/ 4013 is ptr to -B in VTOC of unit 1
2061  3011     DCA 11
2062  1363     TAD (7767/ 7761 is ptr to -B update area
2063  3010     DCA 10
```

PAGE 03

```
2064  6211    CDF 10
2065  1410    TAD I 10  / get first word - block #
2066  7450    SNA       / if 0 don't update - B
2067  5306    JMP PFXOUT
2070  7104    CLL RAL   / diddle top bit
2071  7130    STL RAR /TOP BIT OF AC SET
2072  3411    DCA I 11  / deposit into VTOC
2073  1410    TAD I 10  / get file length in blocks
2074  0370    AND (377 /CLEAN UP UPDATING PAR$
2075  1367    TAD (7000/ to remove stray bits
2076  3411    DCA I 11  / deposit into VTOC
2077  1410    TAD I 10  / transfer contents attribute word
2100  3411    DCA I 11  /   to VTOC
2101  7001    IAC
2102  3766    DCA I (0  / write out unit # VTOC
2103  6201    CDF
2104  6212    CIF 10
2105  4765    JMS WVTOC
2106  6201  PFXOUT, CDF / go to command language interpreter
2107  5762    JMP CLI
2110  6201  IDY, CDF
2111  4761    JMS EOL   / identify by typing system message #2 -
2112  1360    TAD (2    /   "CPS(MMDDY)" - on the teleprinter
2113  4757    JMS SYSMSG
2114  5306    JMP PFXOUT / or just plain "JMP CLI" would do.
              /
              /PACK - CALLED BY RELTAP TO PACK UP A
              /FILE TAPE WITH THE FOLLOWING PARAMETER LOCNS:
              /10115 - SINK BLK #
              /10116 - SOURCE LENGTH
              /10117 - SOURCE BLK #
              /10000 - TAPE UNIT
              /
2115  0000  PACK, 0
2116  1756    TAD I (115 / copy destination block #
2117  3117    DCA 117
2120  1755    TAD I (116 / copy source length - # of blocks to copy
2121  3115    DCA 115
2122  1754    TAD I (117 / copy source block #
2123  3114    DCA 114
2154  0117    TAD I (0  / copy tape unit #
2155  0116
2156  0115
2157  2713
2160  0002
2161  3112
2162  3000
2163  7767
2164  4013
2165  1270
2166  0000
2167  7000
2170  0377
2171  2013
2172  7764
```

PAGE 04

```
2173  0376
2174  7401
2175  7773
2176  6577
2177  0243
2124  1753
2125  3113    DCA 113 /
2126  6201    CDF 0   /
2127  1113    TAD 113 / set up sink unit + source unit for TPCOPY
2130  3116    DCA 116 /CALLING PARS FOR TPCOPY SET UP
2131  4752    JMS TPCOPY / do the copy operation
2132  4444    TPWAIT     / wait until high core is restored
2133  6213    CDF CIF 10
2134  5715    EXIT PACK / returns RELTAP in high core
2152  2262  PAGE
2153  0000
```

```
                /PAGE 2200 CPS
                /
                /MESSAGE PRINTING ROUTINE.
                /THIS ROUTINE HAS 2 ENTRY PTS,
                /MESAGI WHICH SIMULATES DEC'S MESAGE ROUTINE,
                /ROUTINE, AND CELMSG WHICH ASSUMES THE MESSAGE ADDRESS
                /IS IN THE AC UPON ENTRY./
                /THIS ROUTINE MUST BE IN THE LOWER MEMFLD;
                /
2200  0000      MESAGI, 0
2201  7240        CLA CMA
2202  1200        TAD MESAGI
2203  3010        DCA 10
2204  3011        DCA RETSW
2205  5215        JMP MSGGET
2206  0000      CELMSG, 0
2207  3200        DCA MESAGI
2210  7040        CMA
2211  1200        TAD MESAGI
2212  3010        DCA 10
2213  7240        CLA CMA
2214  3011        DCA RETSW
2215  1410      MSGGET, TAD I 10
2216  3200        DCA MSGTEMP
2217  1200        TAD MSGTEMP
2220  7012        RTR
2221  7012        RTR
2222  7012        RTR
2223  4227        JMS TYPECH
2224  1200        TAD MSGTEMP
2225  4227        JMS TYPECH
2226  5215        JMP MSGGET
                  MSGTEMP=MESAGI
2227  0000      TYPECH, 0
2230  0377        AND (77
2231  7440        SZA
2232  5237        JMP GO
2233  1011        TAD RETSW
2234  7640        SZA CLA
2235  5606        EXIT CELMSG
2236  5410        EXIT 10
2237  1376      GO, TAD (-40
2240  7500        SMA
2241  5244        JMP .+3
2242  1375        TAD (340
2243  5257        JMP MTP
2244  1374        TAD (-3
2245  7440        SZA
2246  5251        JMP .+3
2247  1373        TAD (212
2250  5257        JMP MTP
2251  1372        TAD (-2
2252  7440        SZA
2253  5256        JMP .+3
2254  1371        TAD (215/
```

PAGE 06

```
2255  5257    JMP MTP
2256  1370    TAD (245
2257  4453   MTP, PTRPUT
2260  7200    CLA
2261  5627    EXIT TYPECH
              RETSW=11
              /
              /TAPE COPIER - USES VARIOUS PAGE ZERO LOCNS
              /FOR PARAMETERS:
              /113 - SOURCE TAPE UNIT
              /114 - SOURCE BLK #
              /115 - SOURCE LENGTH IN BLKS
              /116 - SINK TAPE UNIT
              /117 - SINK BEGINNING BLK #
              /NOTE THAT THESE LOCNS ARE NOT PRESERVED
              /ACROSS THE CALL.  ROUTINE USES HIGH
              /MEMFLD FOR A BUFFER, AND MUST BE CALLED
              /FROM FIELD ZERO.
              /
2262  0000   TPCOPY, 0
2263  4446    WRSTAP /SAVE HICORE IN STACK
2264  4200    UNIT1+200/ on tape drive 1
2265  4040    FLD1+40  / so we can use all of bank 1 for a buffer
2266  0000    0000
2267  1115    TAD 115 /GET LENGTH
2270  1376   COPY, TAD (-40 /IS IT > 40?
2271  7700    SMA CLA /SKIP IF NO
2272  5275    JMP .+3
2273  1115    TAD 115 /DO PARTIAL CORE LOAD
2274  7410    SKP
2275  1367    TAD (40 /DO FULL CORE LOAD
2276  3120    DCA NBLKS /# TO COPY- save for later
2277  1120    TAD NBLKS / set up tape routine calls
2300  1366    TAD (FLD1 /SET FIELD BIT FOR READ
2301  3317    DCA EBLKS
2302  1317    TAD EBLKS
2303  1365    TAD (CHECK /SET CHECK BIT FOR WRITE
2304  3323    DCA KBLKS
2305  1113    TAD 113 /SOURCE UNIT
2306  7112    CLL RTR / unit goes in bit 9
2307  1114    TAD 114 /SOURCE BLK#
2310  3316    DCA EUBLK
2311  1116    TAD 116 /SINK UNIT
2312  7112    CLL RTR / unit goes in bit 9
2313  1117    TAD 117 /SINK BLK#
2314  3322    DCA KUBLK
2315  4445    RDSTAP /INITIATE COPY OF NBLKS- READ into bank 1
2316  0000   EUBLK, 0 /SOURCE UNIT & BLK#
2317  0000   EBLKS, 0 /# OF BLKS+FLD1+CHECK
2320  0000    0000 /LOCATION 0
2321  4446    WRSTAP / write bank 1 onto tape
2322  0000   KUBLK, 0 /SINK UNIT & BLK#
2323  0000   KBLKS, 0 /# OF BLKS+FLD1+CHECK
2324  0000    0000
2325  1114    TAD 114 /UPDATE BLK #'S
```

PAGE 07

```
2326  1120    TAD NBLKS /update source block#
2327  3114    DCA 114
2330  1117    TAD 117
2331  1120    TAD NBLKS / update destination block#
2332  3117    DCA 117 /
2333  1120    TAD NBLKS
2334  7041    CIA /-# OF BLKS COPIED
2335  1115    TAD 115 /+ # LEFT
2336  3115    DCA 115 /= NEW # LEFT
2337  1115    TAD 115 /DONE?
2340  7440    SZA /SKP IF YES - i.e., no more blocks to copy
2341  5270    JMP COPY
2342  4445    RDSTAP /RESTORE HICORE
2343  4200    UNIT1+200
2344  4040    FLD1+40
2345  0000    0000
2346  5662    EXIT TPCOPY
2365  2000  PAGE
2366  4000
2367  0040
2370  0245
2371  0215
2372  7776
2373  0212
2374  7775
2375  0340
2376  7740
2377  0077
```

PAGE 03

```
                /LINE INPUT ROUTINE FOR CPS - SEE
                /ROUTINE WRITEUP FOR CALLING SEQUENCE
                /AND EXTERNAL WORLD PROTOCOL
                /
2400  0000      GETLINE, 0
2401  7200       CLA
2402  1102       TAD BDF /SET BUFFER DATA FLD
2403  3777       DCA GETCDF
2404  1102       TAD BDF /to the field specified in
2405  3215       DCA SETCDF
2406  1102       TAD BDF /BDF on page 0
2407  3327       DCA ECHOCDF/ & CDF to be set
2410  7200      START, CLA / start here to set up routine to receive a new line
2411  1110       TAD 110 /GET MAX BUFF LENGTH- kept as
2412  3011       DCA 11  / a negative # for ISZ purposes
2413  1101       TAD ALBUFF /GET ADR OF BUFF-1
2414  3010       DCA 10 /FOR AUTO INDEX
2415  7402      SETCDF, HLT /REP BY CDF
2416  1376       TAD (240
2417  3410       DCA  I 10
2420  2011       ISZ 11 /SET BUFFER TO BLANKS
2421  5216       JMP .-3
2422  3410       DCA I 10 /END OF BUFFER IS ZERO; i.e., is marked by all zeros word
2423  6201       CDF /RESET DF
2424  3104       DCA CHAR / zero out so that LASTCHAR is zero (see line 2452)
2425  3112       DCA LPTR/ line pointer to 0 ⇒ no characters in line buffer
2426  5230       JMP .+2
2427  5600      GLEXIT, EXIT GETLINE /COMMON EXIT
2430  4450      GETCH, RDRGET / get a character using IOCS3
2431  0375       AND (177 / set 200 bit in case char came from a non-DEC TTY
2432  1374       TAD (200/
2433  3010       DCA 10 / save in temp storage
2434  1111       TAD LNSW / is "literal next" switch set ?
2435  7700       SMA CLA / skip if yes
2436  5243       JMP NORMAL / no- handle char. normally
2437  3111       DCA LNSW / reset switch
2440  1010       TAD 10                  / "literal next" context implies next char from (Command) input device
2441  3104       DCA CHAR                / always goes into line buffer ; BUFPUT deposits char into buffer.
2442  5773       JMP BUFPUT/
2443  1010      NORMAL, TAD 10/ Non-"literal next" context comes here -
2444  1372       TAD (-200       / was input char Backarrow(rubout)?
2445  7640       SZA CLA
2446  5252       JMP .+4 / no
2447  1010       TAD 10
2450  4453       PTRPUT/ yes, echo it, but don't put it into buffer -
2451  5230       JMP GETCH/ instead, get another character
2452  1104       TAD CHAR / save previous char in LASTCHAR - used to check for command line
2453  3106       DCA LASTCHAR/ continuation at SECRET loc 2625
2454  1010       TAD 10
2455  3104       DCA CHAR/ copy latest input into CHAR - it's the acceptable char if come here
2456  1371       TAD (-337 /BACKARROW TEST- check for line delete
2457  1104       TAD CHAR
2460  7640       SZA CLA / skip if CHAR was "←"
2461  5265       JMP ROTEST / if not, go to rubout test
2462  4770       JMS ECHO /echo the "←"
```

```
2463  4767   JMS EOL /start a new line on printer
2464  5210   JMP START / next points to store line in buffer
2465  1366  ROTEST, TAD (-377 /RUBOUT TEST - check for last-character delete
2466  1104   TAD CHAR
2467  7640   SZA CLA / skip if CHAR was a rubout
2470  5307   JMP LFTEST / if not, go to line feed test
2471  7240   STA /IS LINEPTR @ 0?
2472  1112   TAD LPTR
2473  7710   SPA CLA /SKIP IF NOT
2474  5230   JMP GETCH /@0, DO NOTHING
2475  1376   TAD (240 /INSERT BLANK INTO BUFFER
2476  3104   DCA CHAR
2477  4765   JMS PUTBUF
2500  1364   TAD (334 /ECHO A"\" to indicate deletion
2501  3104   DCA CHAR
2502  4770   JMS ECHO
2503  7240   STA /NOW DECREASE LPTR FOR REAL
2504  1112   TAD LPTR
2505  3112   DCA LPTR / lptr now points at the blank we just inserted
2506  5230   JMP GETCH
2507  1363  LFTEST, TAD (-212 /LINE FEED TEST - check for command line ending
2510  1104   TAD CHAR
2511  7640   SZA CLA / skip if CHAR was a line feed
2512  5337   JMP CRTEST / if not, go to carriage return test
2513  1362   TAD (275 / set prefix char to "="
2514  4761   JMS PUSHPFX
2515  4767   JMS EOL / start a new line on teleprinter
2516  4760   JMS POPPFX / recall prefix char to #
2517  7200   CLA /PREPARE TO ECHO LINE
2520  1112   TAD LPTR
2521  7450   SNA
2522  5230   JMP GETCH / nothing to echo - go process more input
2523  7041   CIA /NEGATE FOR CNTR # of chars to echo
2524  3011   DCA ECHOCNT / goes to "ECHOCOUNT"
            ECHOCNT=11
2525  1101   TAD ALBUFF / AC reg 10 points to buffer
2526  3010   DCA 10
2527  7402  ECHOCDF, HLT /REP BY CDF to field that line buffer area is in
2530  1410   TAD I 10 / get char
2531  6201   CDF
2532  4453   PTRPUT / put onto printer
2533  7200   CLA /(this command not needed as AC is clear after PTRPUT.)
2534  2011   ISZ ECHOCNT / done yet?
2535  5327   JMP .-6 /no - do next character
2536  5230   JMP GETCH / yes - go process more input
2537  1357  CRTEST, TAD (-215 /CARR RET TEST - check for logical end of command line
2540  1104   TAD CHAR
2541  7650   SNA CLA / skip if not carriage return
2542  5756   JMP GLRET / handle CR at loc 2625
2543  1104  LNTEST, TAD CHAR /LITERAL-NEXT TEST - are we supposed to establish
2544  1355   TAD (-220 / control P on TTY                literal next context?
2545  7640   SZA CLA  / skip if = CHAR was control P
2546  5754   JMP EOFTST / otherwise go to END of FILE test
2547  7040   CMA  / AC at this -1
2550  3111   DCA LNSW / set literal-next switch
```

PAGE 10

```
2551  4770   JMS ECHO /echo the control P
2552  5230   JMP GETCH / go get the literal character
2554  2600  PAGE
2555  7560
2556  2625
2557  7563
2560  3123
2561  3103
2562  0275
2563  7566
2564  0334
2565  2643
2566  7401
2567  3112
2570  2706
2571  7441
2572  7600
2573  2610
2574  0200
2575  0177
2576  0240
2577  2650
2600  1104  EOFTST, TAD CHAR /EOF TEST- test for user entry of an end of file character
2601  1377   TAD (-231 / control Y on TTY
2602  7640   SZA CLA / skip if CHAR = control Y
2603  5210   JMP BUFPUT / if not EOF, put char into buffer
2604  1376   TAD (22
2605  4313   JMS SYSMSG /TELL USER we recognize his end of file
2606  2775   ISZ GETLINE / return call+2 for end of file
2607  5774   JMP GLEXIT / return to user
2610  2112  BUFPUT, ISZ LPTR /Bump line ptr by 1
2611  1112   TAD LPTR/
2612  1110   TAD 110/ is line longer than specified in LINLEN ?
2613  7710   SPA CLA / skip if too long
2614  5222   JMP LOK / line OK
2615  4773   JMS EOL /LINE TOO LONG - TELL USER
2616  7001   IAC /system message # 1
2617  4313   JMS SYSMSG
2620  4773   JMS EOL/ move ptr to next line
2621  5772   JMP START /start a new line
2622  4243  LOK, JMS PUTBUF /INSERT CHAR INTO BUFFER
2623  4306   JMS ECHO / echo char on printer
2624  5771   JMP GETCH / go process another char
2625  1106  GLRET, TAD LASTCHAR / come here if CHAR is a carriage return
2626  1370   TAD (-255 /CONTINUATION CHECK- was last char a "-" sign?
2627  7640   SZA CLA /skip if yes, & continue command line
2630  5236   JMP GLRTN / otherwise go to return code @2636
2631  7040   CMA
2632  1112   TAD LPTR / set the line pointer to point at the - sign
2633  3112   DCA LPTR/
2634  4773   JMS EOL / start a new line on printer
2635  5771   JMP GETCH / go process another character
2636  1367  GLRTN, TAD (200/ echo a 200₈ code for the carriage return
2637  4453   PTRPUT
2640  7200   CLA / unneeded instruction - PTRPUT returns w/ clear AC
```

perhaps we should just allow him to erase character from end.

PAGE 11

```
2641  1112   TAD LPTR /get count of # of char in line
2642  5774   JMP GLEXIT/return to addr of call+1
2643  0000  PUTBUF, 0 /ROUTINE TO INSERT "CHAR" INTO LINEBUF
2644  1112   TAD LPTR /GET LINE POINTER
2645  1101   TAD ALBUFF /COMPUTE ADDR OF CHAR
2646  3107   DCA ADDR
2647  1104   TAD CHAR /GET CHAR
2650  7402  GETCDF, HLT /CDF TO LINEBUF FLD
2651  3507   DCA I ADDR /INSERT CHAR INTO BUFFER
2652  6201   CDF
2653  5643   EXIT PUTBUF /RETURN TO CALLER
             /
             /QUIZUSER - SOLICITS A REPLY
             /FROM THE USER TO A QUESTION
             /
2654  0000  QUIZUSER, 0 /Called w/ a message # in AC
2655  3143   DCA QTEMP /save message#
2656  4773   JMS EOL  / start printer on a new line
2657  1143   TAD QTEMP
2660  4313   JMS SYSMSG/ output the message
2661  1366   TAD ("?
2662  4765   JMS PUSHPFX/ set prefix char to "?"
2663  4773   JMS EOL   / start printer on new line
2664  4764   JMS POPPFX / reset prefix char to its former value
2665  1363   TAD (-176
2666  3110   DCA 110 /SET REPLY LENGTH to 120 chars
2667  1362   TAD (REPBUF/"REPLACE" or "REPLY" buffer area at 17400-17577
2670  3101   DCA 101
2671  4775   JMS GETLINE / get a line into the reply buffer
2672  7650   SNA CLA / skip if line had at least one char
2673  5301    JMP CLIRET / go back to command interpreter if END of FILE or 0 chars were entered
2674  1361   TAD (-377/
2675  3110   DCA 110 / reset line length to 255 chars
2676  1360   TAD (1377/
2677  3101   DCA 101 / reset input buffer to 17400
2700  5654   EXIT QUIZUSER / return to caller - we got some input
2701  1361  CLIRET, TAD (-377/ reset line length to 255 chars
2702  3110   DCA 110
2703  1360   TAD (1377/
2704  3101   DCA 101 / reset input buffer to 17400
2705  5757   JMP CLI / go to command line interpreter since null line was input
             /
             /ECHO - PRINT THE INPUT CHAR FOR GETLINE
             /
2706  0000  ECHO, 0
2707  7200   CLA
2710  1104   TAD CHAR/ a little subroutine to print the contents of CHAR on
2711  4453   PTRPUT  / the teleprinter
2712  5706   EXIT ECHO
             /
             /STANDARD SYSTEM MESSAGE ROUTINE
             / Call with message # to be typed in the AC
2713  0000  SYSMSG, 0
2714  1144   TAD MSGBAS /add message address table base to message # to
2715  3010   DCA 10 /construct pointer to message address
```

PAGE 12

```
2716  1410    TAD I 10 / get message address
2717  4756    JMS CELMSG / print message
2720  7200    CLA
2721  5713    EXIT SYSMSG / return to caller
2756  2206  PAGE
2757  3000
2760  1377
2761  7401
2762  7377
2763  7602
2764  3123
2765  3103
2766  0277
2767  0200
2770  7523
2771  2430
2772  2410
2773  3112
2774  2427
2775  2400
2776  0022
2777  7547
```

PAGE 13

```
                *3000
                /
                /COMMAND LANGUAGE INTERPRETER
                /
3000    4312    CLI, JMS EOL / start a new line on printer & print line prefix character
3001    4777     JMS GETLINE / get a line from user
3002    7650     SNA CLA / skip if a non-zero # of chars were input
3003    5200     JMP .-3 / otherwise try again if 0 chars or end of file
3004    1376     TAD (COMTBL
3005    3223     DCA COMADR / set up pointer for search of command table @ 10600
3006    1101     TAD ALBUFF
3007    7001     IAC
3010    3222     DCA LADR / first char of command sets @ C(LADR) in fld 1.
3011    4444     TPWAIT /WAIT UNTIL LAST TAPE OP DONE - just in case high core isn't
                                                        there yet.
3012    6211    NXTCOM, CDF 10
3013    1623     TAD I COMADR / get first word of next entry in command table
3014    6201     CDF 0
3015    7710     SPA CLA
3016    5250     JMP NOTFND / if it's high order bit is set, the end of the table has been found & command
                                                        doesn't exist
3017    6212     CIF 10
3020    1375     TAD (3
3021    4774     JMS CLW
3022    0000    LADR, 0
3023    0000    COMADR, 0
3024    6211     CDF 10
3025    5243     JMP NOTEQL / command & user-typed chars not equal
3026    7200     CLA / command found! get address in low core of
3027    1223     TAD COMADR
3030    1375     TAD (3
3031    3223     DCA COMADR / the command module to handle it by first constructing
                                pointer to module address in COMADR
3032    6211     CDF 10
3033    1623     TAD I COMADR / get command module address from high core
3034    6201     CDF 0
3035    3223     DCA COMADR / save it
3036    1101     TAD ALBUFF
3037    7001     IAC
3040    3105     DCA 105 / set up loc 105 so points to line to be analyzed by
                                command module
3041    4623     JMS I COMADR / jump to command module
3042    5200     JMP CLI / jump back to CLI after command module is finished
3043    7200    NOTEQL, CLA / come here if current command not the one executing
3044    1223     TAD COMADR / the user typed - first increment COMADR
3045    1373     TAD (4
3046    3223     DCA COMADR / to point to the next command table entry
3047    5212     JMP NXTCOM / then try again
3050    4312    NOTFND, JMS EOL / invalid command - or at least its not in command table
3051    1375     TAD (3
3052    4772     JMS SYSMSG / complain to the user
3053    5200     JMP CLI / try again
                /
                /OCTAL NUMBER PRINTING ROUTINE
                / call with # to be typed in AC
3054    0000    OCTOUT, 0
3055    7104     CLL RAL
3056    3006     DCA 6 / rotate AC left & save in loc 6 for loop starting @ 3063
3057    7010     RAR / save link in
```

PAGE 14

```
3060  3007   DCA 7 / location 7
3061  1371   TAD (-4 / set up count - 4 chars to be output
3062  3010   DCA 10
3063  1007 NUMLOO, TAD 7 / i.e. restore
3064  7104   CLL RAL / reset up AC + LINK from last time they were saved
3065  1006   TAD 6
3066  7006   RTL / rotate 3 bits left to develop next octal digit
3067  7004   RAL
3070  3006   DCA 6
3071  7010   RAR / save AC and LINK bit
3072  3007   DCA 7
3073  1006   TAD 6
3074  0370   AND (7 / get lower 3 bits of # after last rotation
3075  1367   TAD (260
3076  4453   PTRPUT / print it
3077  7200   CLA
3100  2010   ISZ 10 / done yet?
3101  5263   JMP NUMLOO / no
3102  5654   EXIT OCTOUT / yes - return to caller
             /
             /PUSHPFX - SAVE CURRENT PFX CHAR AND
             /SUBSTITUTE A NEW ONE.  GNC - "prefix char" refers to LINE prefix char on
             / cf "POPPFX" below                                    teleprinter
3103  0000 PUSHPFX, 0 / called w/ new ASCII prefix char in AC
3104  3142   DCA PFXTEM / save new pfx char in a temp
3105  1100   TAD 100
3106  3133   DCA PFXSTACK / stack old prefix character
3107  1142   TAD PFXTEM
3110  3100   DCA 100 / move temp to prefix char loc'n
3111  5703   EXIT PUSHPFX / return to caller
             /
             /EOL - PUT OUT CR, LF, AND PREFIX CHAR
             /
3112  0000 EOL, 0
3113  4766   JMS MESAGI
3114  4543   TEXT /%# ← this gets decoded as carriage return, line feed
3115  0000   /
3116  7200   CLA
3117  1100   TAD 100 / get prefix char from loc 100 + print it
3120  4453   PTRPUT
3121  7200   CLA  / unneeded instruction
3122  5712   EXIT EOL / return to caller
             /
             /POPPFX - ROUTINE TO RESTORE PREFIX CHAR
             / cf routine "PUSHPFX" above
3123  0000 POPPFX, 0
3124  7200   CLA
3125  1133   TAD PFXSTACK / get previous prefix char + move to loc 100
3126  3100   DCA 100
3127  5723   EXIT POPPFX / return to caller.
             /
             /PRTNAM - ROUTINE TO PRINT THE COMPRESSED ← N.B. There is a
             /FDNAME WHICH APPEARS IN LOCNS 10020 - 10023
             /
3130  0000 PRTNAM, 0
```

N.B. having the link was necessary when CPS used IOCS2 for doing I/O. IOCS2 did not preserve the link across a PTRP.

bug in this routine! See writeup.

PAGE 15

```
3131  7200    CLA
3132  1371    TAD (-4 /SET UP CNT to move the 4-word FDNAME down to this bank.
3133  3010    DCA 10
3134  1365    TAD (17 /AND ADDR REG/ address of source of words in bank /
3135  3011    DCA 11
3136  1364    TAD (PAREA-1 /AND ANOTHER ADDR REG/ address of destination in bank 0
3137  3012    DCA 12
3140  6211    CDF 10 /MOVE THE FDNAME
3141  1411    TAD I 11 / by getting from high core,
3142  6201    CDF 0
3143  3412    DCA I 12/ putting into low core
3144  2010    ISZ 10  / done?
3145  5340    JMP .-5 / no
3146  4766    JMS MESAGI / yes - use MESAGI to print name
3147  0000  PAREA, 0
3150  0000    0
3151  0000    0
3152  0000    0
3153  0000    0 <- This word always stays zero, to mark the end of the name.
3154  7200    CLA
3155  5730    EXIT PRTNAM/ return to caller
3164  3146  PAGE
3165  0017
3166  2200
3167  0260
3170  0007
3171  7774
3172  2713
3173  0004
3174  1124
3175  0003
3176  0600
3177  2400
```

```
                *3200
                /
                /INDEX PRINTING UTILITY ROUTINE
                /USES SWITCHES ON PAGE ZERO
                /TO PRINT VARIOUS FORMS OF INFO
                /ABOUT THE VTOCS
                /  ct volume 1, chapter 7, sect 7.2
3200  0000      NDXPRT, 0
3201  7200       CLA
3202  1377       TAD (-2          / set up
3203  3126       DCA PASCNT       / one pass over each VTOC
3204  1376       TAD (SW1+1       / computes address of current printing switch
3205  1126       TAD PASCNT       /
3206  3007       DCA 7            /
3207  1407       TAD I 7          / get switch
3210  7650       SNA CLA          / if its minus, then print the VTOC's names
3211  5332       JMP PASTST       / otherwise jump to PASTST to see if we're done.
3212  1375       TAD (2           / computes the tape unit whose VTOC we're to print
3213  1126       TAD PASCNT       /
3214  3007       DCA 7            /
3215  1007       TAD 7            / set up address, block #, and unit # words for this
3216  6211       CDF 10           / VTOC in Bank 1
3217  3774       DCA I (0         /
3220  6201       CDF              /
3221  6212       CIF 10           /
3222  4773       JMS GABUV        /
3223  6211       CDF 10           / get Bank 1 address of VTOC
3224  1772       TAD I (AVTOC     /
3225  6201       CDF              /
3226  3131       DCA VTOCAD       / save it
3227  1145       TAD DETAIL       / check detail switch
3230  7700       SMA CLA          / if set, print out the tape unit # as a heading
3231  5240       JMP NAMPRT       / otherwise just print file names
3232  4771       JMS EOL          /
3233  1370       TAD (21          / print "UNIT "
3234  4767       JMS SYSMSG       /
3235  1007       TAD 7            / print unit #
3236  1366       TAD (260         /
3237  4453       PTRPUT           /
3240  7200      NAMPRT, CLA       / set up name printing loop
3241  1131       TAD VTOCAD       /
3242  3010       DCA 10           / get the (negative) # of files catalogued and
3243  6211       CDF 10           / save it in VTOCNT
3244  1410       TAD I 10         /
3245  6201       CDF              /
3246  3132       DCA VTOCNT       /
3247  1010       TAD 10           /
3250  1365       TAD (6           / set up source address - SADR - for first file control block
3251  3146       DCA SADR         /
3252  1364       TAD (240         / set line prefix char to a "Blank"
3253  4763       JMS PUSHPFX      /
3254  7200      NAMLK, CLA        / prepare to move current FCB to location 10020-10027
3255  1362       TAD (-10         /
3256  3127       DCA OPFLAG /USE AS A CNTR
3257  1361       TAD (17  / ptr to destination location in fld 1
```

"File Control Block"

PAGE 17

```
3260  3147   DCA KADR  / "disk address"
3261  6211   CDF 10
3262  2146   ISZ SADR
3263  1546   TAD I SADR
3264  2147   ISZ KADR
3265  3547   DCA I KADR
3266  2127   ISZ OPFLAG
3267  5262   JMP .-5
3270  6201   CDF
3271  1130   TAD ALLFLG / check to see if all file names are to be printed
3272  7710   SPA CLA    / skip if not
3273  5306   JMP PRTFIL / otherwise go print the name
3274  6211   CDF 10
3275  1760   TAD I (20
3276  6201   CDF
3277  0357   AND (7700
3300  1356   TAD (-5200 /* TEST
3301  7450   SNA        / skip if not "*"
3302  5327   JMP VTOTST / is an * - forget about printing system file name.
3303  1355   TAD (-0300 /- TEST
3304  7650   SNA CLA    / skip if not "-"
3305  5327   JMP VTOTST / is a -, forget about printing working area name.
3306  4771   PRTFIL, JMS EOL  / print file name on a new line
3307  4754   JMS PRTNAM
3310  1145   TAD DETAIL / are the VTOC entries for the file to be printed?
3311  7700   SMA CLA    / skip if yes
3312  5327   JMP VTOTST
3354  3130   TAD (-     / set up loop to print 3
3355  7500
3356  2600
3357  7700
3360  0020
3361  0017
3362  7770
3363  3103
3364  0240
3365  0006
3366  0260
3367  2713
3370  0021
3371  3112
3372  0101
3373  1037
3374  0000
3375  0002
3376  0126
3377  7776
3313  1353   3
3314  3127   DCA OPFLAG / file information words from VTOC
3315  1352   TAD (23    / AX 17 points to information words
3316  3017   DCA 17
3317  1351   DOUT, TAD (240 /OUTPUT
3320  4453   PTRPUT /DETAILS ABOUT FILE - separated by a blank
3321  6211   CDF 10
3322  1417   TAD I 17   / get an information word from the copy of the current FCB
                        / which resides in 10020 - 10027
```

Precipitous dumping of literals due to imminent overlap of current page literal table and pushdown stack within MACROS

```
3323  6201    CDF
3324  4750    JMS OCTOUT / print word in octal format
3325  2127    ISZ OPFLAG / done printing info words?
3326  5317    JMP DOUT   / no
3327  2132  VTOTST, ISZ VTOCNT / done printing all files in this VTOC?
3330  5254    JMP NAMLK  / no
3331  4747    JMS POPPFX / yes - restore prefix char
3332  2126  PASTST, ISZ PASCNT / done with both VTOCS?
3333  5204    JMP NDXPRT+4 / no
3334  5600    EXIT NDXPRT  / yes - return to caller
              /
              /NOTYET COMMAND MODULE
              / called when a command is in command table but not yet implemented
3335  0000  NOTYET, 0
3336  4746    JMS EOL
3337  1345    TAD (4
3340  4744    JMS SYSMSG / tell user "command not yet implemented"
3341  5735    EXIT NOTYET
              /
              /COMMENT COMMAND MODULE
              / otherwise known as the command language NOP
3342  0000  COMMENT, 0
3343  5742    EXIT COMMENT
3344  2713  PAGE
3345  0004
3346  3112
3347  3123
3350  3054
3351  0240
3352  0023
3353  7775
```

```
                *3400
                /
                /INDEX COMMAND MODULE
                /SCANS OPERAND STRING AND SETS
                /UP PARAMETERS FOR SUBROUTINE NDXPRT
                /WHICH ACTUALLY PRINTS THE INDEX
                /
3400  0000      INDEX, 0
3401  7200       CLA
3402  3124       DCA SW0     / reset switches which indicate which tape VTOC is to be listed
3403  3125       DCA SW1     /
3404  3130       DCA ALLFLG  / reset "all names to be listed" and "list details!" switches
3405  3145       DCA DETAIL  /
3406  6212       CIF 10
3407  4777       JMS OPSCAN  / set line ptr in loc 00105 to next operand
3410  5217       JMP INDOP   /WE HAVE >0 OPERANDS - scan them
3411  7240       STA         /NO OPERANDS - PRINT BOTH
3412  3124       DCA SW0
3413  7240       STA
3414  3125       DCA SW1     /USER INDEXES
3415  4776      INDPRT, JMS NDXPRT / call NDXPRT to print indexes using switches we
3416  5600       EXIT INDEX  /return to caller.                    have set up.
3417  7240      INDOP, STA   /
3420  1105       TAD 105     / set up AX10 as pointer to this operand.
3421  3010       DCA 10      /
3422  6211      OPLOOP, CDF 10/
3423  1410       TAD I 10    / get a character from the operand string
3424  6201       CDF
3425  1375       TAD (-240
3426  7450       SNA
3427  5215       JMP INDPRT  / if char is a blank, end of operand string has been reached -
3430  1374       TAD (-41 /A TEST                                  go print index
3431  7450       SNA
3432  5253       JMP OPA     / go here if char is an "A"
3433  1373       TAD (-1 /B TEST
3434  7450       SNA
3435  5250       JMP OPONE   / go here if char is a "B"
3436  1372       TAD (-2 /D TEST
3437  7450       SNA
3440  5256       JMP OPD     / here if char is a "D"
3441  1371       TAD (-17 /S CHECK
3442  7650       SNA CLA
3443  5245       JMP OPZERO  / here if char is an "S"
3444  5222       JMP OPLOOP /IGNORE INVALID OPERANDS
3445  7240      OPZERO, STA  / "S" operand ⇒ print symbolic files -
3446  3124       DCA SW0     / set SW0
3447  5222       JMP OPLOOP
3450  7240      OPONE, STA   / "B" operand ⇒ print binary files -
3451  3125       DCA SW1     / set SW1
3452  5222       JMP OPLOOP
3453  7240      OPA, STA     / "A" operand ⇒ print all file names -
3454  3130       DCA ALLFLG  / set ALLFLG
3455  5222       JMP OPLOOP
3456  7240      OPD, STA     / "D" operand ⇒ print details on each file -
3457  3145       DCA DETAIL  / set DETAIL.
```

PAGE 20

```
3460  5222   JMP OPLOOP
             /
             /TAPE ERROR HANDLING ROUTINE
             / Come here after 4 attempts to read tape have failed
3461  3307   TPERR, DCA TPAC /SAVE AC
3462  7010    RAR /AND LINK
3463  3310    DCA TPLK
3464  4770    JMS EOL /OUTPUT CRLF
3465  1367    TAD (31 /AND ERROR MESSAGE
3466  4766    JMS SYSMSG
3467  1040    TAD MFLDS / restore memory extension state
3470  0365    AND (70 /GET INST FLD
3471  1364    TAD (CIF /BUILD INTO INST
3472  3305    DCA TERCIF
3473  1040    TAD MFLDS /NOW DO DATA FIELD
3474  7106    CLL RTL
3475  7004    RAL /GET INTO POSITION FOR MASKING
3476  0365    AND (70
3477  1363    TAD (CDF /BUILD INST
3500  3304    DCA TERCDF
3501  1310    TAD TPLK /RESTORE LINK
3502  7104    CLL RAL
3503  1307    TAD TPAC /RESTORE AC
3504  7402   TERCDF, HLT /REP BY CDF
3505  7402   TERCIF, HLT /REP BY CIF
3506  5437    JMP I PREG /RETURN TO TASK IN PROGRESS
3507  0000   TPAC, 0 /SAVE LOCN FOR AC
3510  0000   TPLK, 0 /SAVE LOCN FOR LINK
3563  6201   PAGE
3564  6202
3565  0070
3566  2713
3567  0031
3570  3112
3571  7761
3572  7776
3573  7777
3574  7737
3575  7540
3576  3200
3577  1067
```

```
                *3600
                /
                /DESTROY COMMAND MODULE - CALLED WHEN
                /USER ISSUES A DESTROY COMMAND TO CLI
                /
3600  0000      DESTROY, 0
3601  6212       CIF 10          / set line pointer in loc 105 to next operand
3602  4777       JMS OPSCAN      /
3603  5210       JMP .+5         / ~~OP NOT THERE~~ / operand found
3604  4776       JMS EOL         /
3605  1375       TAD (11         / operand missing - complain to user and
3606  4774       JMS SYSMSG      / then return to caller
3607  5600      DESEXIT, EXIT DESTROY
3610  1105       TAD 105 /GET LPTR
3611  3123       DCA SAVTEMP / save it
3612  6212       CIF 10
3613  4777       JMS OPSCAN /GET NEXT OPERAND
3614  5217       JMP .+3 / there is one
3615  7200       CLA
3616  5225       JMP NAMCHK      / no second operand - go directly to NAMCHK
3617  6211       CDF 10          /
3620  1505       TAD I 105       / get first char of second operand
3621  6201       CDF 0           /
3622  1373       TAD (-241 / is it a "!"?
3623  7650       SNA CLA  / skip if not
3624  1372       TAD (4000 / otherwise set top bit of SAVSW
3625  3122      NAMCHK, DCA SAVSW / deposit AC as the value of SAVSW
3626  6211       CDF 10          /
3627  1523       TAD I SAVTEMP   / get first char of first operand (which should be a
3630  6201       CDF 0           / file name), and test it
3631  1371       TAD (-252       / is it an "*"?
3632  7450       SNA             /
3633  5243       JMP ASTER / yes
3634  1370       TAD (-3         / is it a "-" sign?
3635  7640       SZA CLA         /
3636  5253       JMP DCOMP / no - go compress file name
3637  4776      NONO, JMS EOL    /
3640  1367       TAD (16         / file name begins with "-" sign & thus designates a
3641  4774       JMS SYSMSG      / working area, which cannot be destroyed - tell user the
3642  5207       JMP DESEXIT     / facts of life & then exit.
3643  7200      ASTER, CLA  / handle asterisk file here
3644  1122       TAD SAVSW / if SAVSW is set, then
3645  7710       SPA CLA         /
3646  5253       JMP DCOMP / allow destruction of the file
3647  4776       JMS EOL         /
3650  1366       TAD (13         / otherwise complain
3651  4774       JMS SYSMSG      /
3652  5207       JMP DESEXIT / and return to caller.
3653  1123      DCOMP, TAD SAVTEMP / set up pointer to file name to be compressed
3654  3260       DCA .+4         /
3655  6212       CIF 10          /
3656  1365       TAD (10         / call COMPRESS in fld 1 to compress the file
3657  4764       JMS COMPRESS    / name to 6-bit form.
3660  0000       0000 /SOURCE - SET ABOVE  (parameter to COMPRESS)
3661  0020       0020 /SINK AREA           (  "      "    "    )
```

PAGE 22

```
3662  6211    CDF 10 /FLD 1 (parameter to COMPRESS routine)
3663  6212    CIF 10 /SEARCH FOR FILE NAME
3664  4763    JMS GETFD/ by calling GETFD in field 1
3665  5272    JMP FFND / file found
3666  4776    JMS EOL
3667  1362    TAD (17      / file not found - complain to user
3670  4774    JMS SYSMSG/
3671  5207    JMP DESEXIT / and then return to caller
3672  3123  FFND, DCA SAVTEMP /"GETFD" found the file and returns in AC either a
3673  1123    TAD SAVTEMP /FCB OR DCB PTR - save it and then construct
3674  1361    TAD (4
3675  3010    DCA 10       / AX w/ pointer to the file length and switches word.
3676  6211    CDF 10
3677  1410    TAD I 10/ get file length and switches word
3700  6201    CDF 0
3701  0360    AND (1000 /
3702  7640    SZA CLA     / if bit 1000 is set, file may not be destroyed
3703  5237    JMP NONO/
3704  1123    TAD SAVTEMP / if bit 1000 not set, call on DESTRY in
3705  6212    CIF 10
3706  4757    JMS DESTRY/ field 1 to destroy the file, and then
3707  5207    JMP DESEXIT / return to caller
3757  0462  PAGE
3760  1000
3761  0004
3762  0017
3763  0200
3764  0400
3765  0010
3766  0013
3767  0016
3770  7775
3771  7526
3772  4000
3773  7537
3774  2713
3775  0011
3776  3112
3777  1067
```

PAGE 23

```
                *4000
                /SAVE COMMAND MODULE - CALLED WHEN THE
                /USER WANTS TO SAVE A FILE.  GNC
                /
4000  0000      SAVE, 0
4001  6212       CIF 10          / get pointer in location 105 to first operand
4002  4777       JMS OPSCAN      /
4003  5210       JMP SSAREA-1    / operand found
4004  4776       JMS EOL         /NO OPERANDS
4005  1375       TAD (11
4006  4774       JMS SYSMSG      / complain to user
4007  5600      SAVEXIT, EXIT SAVE / return to caller
4010  1105       TAD 105         / copy first operand pointer
4011  3215      SSAREA, DCA SAREA / deposit AC into source area pointer for first file name
4012  6212       CIF 10          / use high core compress routine to pack
4013  1373       TAD (10         /8 CHARS into a file name for GETFD
4014  4772       JMS COMPRESS
4015  0000      SAREA, 0000      /SOURCE AREA
4016  0020       0020            /SINK AREA
4017  6211       CDF 10          /MEMFLD
4020  6212       CIF 10          call GETFD to search for file name
4021  4771       JMS GETFD
4022  5237       JMP FND         /FOUND
4023  4776       JMS EOL         /FILE DOESN'T EXIST
4024  1370       TAD (6
4025  4774       JMS SYSMSG      / complain to user by printing bad
4026  4767       JMS PRTNAM      / filename and then
4027  1366       TAD (7
4030  4774       JMS SYSMSG      / telling him it doesn't exist
4031  7200      REPSOR, CLA
4032  1373       TAD (10         / Call QUIZUSER to type message # 8, and
4033  4765       JMS QUIZUSER    / get a replacement name
4034  7200       CLA             / replacement name sits at 17400
4035  1364       TAD (7400
4036  5211       JMP SSAREA      / try again
4037  3121      FND, DCA EFCB    /SOURCE FCB - save it
4040  1121       TAD EFCB        / construct pointer to file length and switches word
4041  1363       TAD (4
4042  3010       DCA 10          /CHK IF FILE IS A SOURCE
4043  6211       CDF 10
4044  1410       TAD I 10        /GET SOURCE BIT on file length + switches word
4045  6201       CDF 0
4046  7710       SPA CLA         /TEST IT
4047  5257       JMP SORCE       /OK
4050  4776       JMS EOL         /COMPLAIN by printing
4051  1370       TAD (6
4052  4774       JMS SYSMSG
4053  4767       JMS PRTNAM      / bad filename and then
4054  1362       TAD (5
4055  4774       JMS SYSMSG      / telling user it doesn't name an information source.
4056  5231       JMP REPSOR      / then allow him to replace the bad name
4057  6212      SORCE, CIF 10    / first file name is ok - on to try for a second one.
4060  4777       JMS OPSCAN      /
4061  7610       SKP CLA         / second operand found
4062  5204       JMP SAVE+4      /  "      "    not found - go complain
```

PAGE 24

```
4063  6211   CDF 10 /CHECK FOR SAVEFNAME STARTING
4064  1505   TAD I 105 /WITH A"-"SIGN - GET INITIAL CHAR
4065  6201   CDF
4066  1361   TAD (-255 /SUBTRACT "-"
4067  7650   SNA CLA /SKP IF NOT MINUS
4070  5323   JMP IC /OTHERWISE JUMP TO BAD CHAR ROUTINE
4071  7240   STA /CHECK ALL CHARS FOR VALIDITY-/i.e. has the user tried to use a
4072  1105   TAD 105/set up ptr to source of chars   restricted char in a file name?
4073  3010   DCA 10 /SOURCE OF CHARS
4074  1360   TAD (-10/
4075  3011   DCA 11 / count - check up to 8 chars
4076  6211   CDF 10
4077  1410  ICHK, TAD I 10 /GET A CHAR
4100  0357   AND (77
4101  7450   SNA /IS IT AN "@"SIGN?
4102  5323   JMP IC /YES - BAD
4103  1356   TAD (-50
4104  7450   SNA /IS IT A "(" ?
4105  5323   JMP IC /YES, ILLEGAL CHAR
4106  1355   TAD (-1
4107  7450   SNA /IS IT A ")"
4110  5323   JMP IC /YES - ILLEGAL
4111  1354   TAD (-2 /IS IT A "+" SIGN?
4112  7450   SNA
4154  7776   JMP IC /YES, ILLEGAL
4155  7777
4156  7730
4157  0077
4160  7770
4161  7523
4162  0005
4163  0004
4164  7400
4165  2654
4166  0007
4167  3130
4170  0006
4171  0200
4172  0400
4173  0010
4174  2713
4175  0011
4176  3112
4177  1067
4113  5323
4114  1353   TAD (-22 /IS IT A "="?
4115  7650   SNA CLA
4116  5323   JMP IC /illegal
4117  2011   ISZ 11 /DONE CHKIG?
4120  5277   JMP ICHK/yes
4121  6201   CDF 0   /yes-all ok
4122  5330   JMP COMP
4123  6201  IC, CDF 0/set field to 0
4124  4752   JMS EOL /TELL USER ABOUT ILLEGAL NAME
4125  1351   TAD (12 /SYSTEM MSG # 10
```

PAGE 25

```
4126  4750   JMS SYSMSG
4127  5207   JMP SAVEXIT / return to caller
4130  1105  COMP, TAD 105 / deposit pointer to second file name
4131  3335   DCA .+4      / into COMPRESS calling sequence
4132  6212   CIF 10
4133  1347   TAD (10 / call COMPRESS to compress file name for GETFD
4134  4746   JMS COMPRESS
4135  0000   0000 /SOURCE ADR
4136  0020   0020 /SINK ADR
4137  6211   CDF 10 /MEMFLD
4140  5745   JMP .&7600+200 / page escape
4145  4200  PAGE
4146  0400
4147  0010
4150  2713
4151  0012
4152  3112
4153  7756
4200  6212   CIF 10 /TEST FOR MORE OPERANDS
4201  4777   JMS OPSCAN
4202  5206   JMP GATHER / found another operand
4203  7200   CLA /NO MORE
4204  3122   DCA SAVSW / reset SAVSW to 0
4205  5223   JMP SAVIT
4206  6211  GATHER, CDF 10 / process 3 and successive operands here
4207  1505   TAD I 105 /GET NEXT OP
4210  6201   CDF 0
4211  1376   TAD (-241 / is it a "/"
4212  7640   SZA CLA   / skip if yes
4213  5220   JMP NXTSVOP /IGNORE ILLEGAL OPS
4214  1122   TAD SAVSW
4215  1375   TAD (4000 / set top bit of SAVSW
4216  3122   DCA SAVSW
4217  5220   JMP NXTSVOP / remove this instruction
4220  6212  NXTSVOP, CIF 10 / check next operand - get a pointer to it
4221  4777   JMS OPSCAN
4222  5206   JMP GATHER / found another operand
4223  1121  SAVIT, TAD EFCB / all operands in line have been scanned & SAVSW has been set.
4224  1374   TAD (5
4225  3010   DCA 10 / get pointer to source file contents attribute word
4226  6211   CDF 10
4227  1410   TAD I 10 /GET SOURCE DESCRIPTOR
4230  6201   CDF /RESTORE DF
4231  0373   AND (3400 / check to see if source file is in symbolic or binary equivalent class
4232  7640   SZA CLA /SKP IF ON UNIT 0 - SYMBOLIC
4233  7001   IAC
4234  3113   DCA 113 / word 113 contains the type word on which we'd like to create the
4235  6212   CIF 10 /SEE IF FILE ALREADY EXISTS                          sink file
4236  4772   JMS GETFD /BY TRYING TO GET A FCB PTR FOR IT
4237  7410   SKP /ALREADY EXISTS
4240  5272   JMP MAKVTOC /DOESN'T EXIST - MAK A VTOC ENTRY
4241  3123   DCA SAVTEMP /SAVE FCB POINTER
4242  1123   TAD SAVTEMP /CHECK TO SEE IF A FILE OR DEVICE
4243  1371   TAD (3
4244  3010   DCA 10 /ADR OF FILE BLK# OR DEVICE TYPE word in FCB or DCB
```

```
4245  6211    CDF 10
4246  1410    TAD I 10 /GET FD UNIT and BLOCK # word
4247  6201    CDF
4250  7004    RAL /PUT BIT 1 INTO BIT 0
4251  7710    SPA CLA /SKP IF A FILE
4252  5770    JMP IC /OTHERWISE IS A DEVICE
4253  1122    TAD SAVSW / come here if we have an already existing file.
4254  7710    SPA CLA  / has user specified that file is to be replaced?
4255  5267    JMP KILFIL /USER SAYS REPLACE
4256  1367    TAD (15  / user has not specified whether he wants replacement, therefore
4257  4766    JMS QUIZUSER / ask him now.
4260  7200    CLA
4261  6211    CDF 10
4262  1765    TAD I (REPBUFF+1 / get first character in replacement buffer.
4263  6201    CDF
4264  1376    TAD (-241 /IS IT "!"?
4265  7640    SZA CLA
4266  5764    JMP SAVEXIT /NO- return to caller
4267  6212  KILFIL, CIF 10 / come here if old file is to be destroyed and replaced by new one.
4270  1123    TAD SAVTEMP /DESTROY OLD FILE
4271  4763    JMS DESTRY /
4272  7200  MAKVTOC, CLA /BUILD PARTIAL VTOC FOR "MAKFIL" in locations 10020-10037
4273  1121    TAD EFCB /set up
4274  1371    TAD (3
4275  3010    DCA 10 /SOURCE ADR FOR NEW file information words
4276  1362    TAD (24
4277  3011    DCA 11 /SINK ADR - NEW file info words - NOTE that the name is already
                                                      in loc 10020-10023 incl.
4300  6211    CDF 10
4301  1410    TAD I 10 /GET SOURCE FILE type address
4302  3114    DCA 114 / save for later use by "COPYIT" at loc 4400
4303  1410    TAD I 10 /GET LENGTH WORD of source file
4304  3115    DCA 115
4305  1115    TAD 115 / save in loc 115 for "COPYIT"
4306  0361    AND (4377 /CLEAN IT UP
4307  3411    DCA I 11 / deposit length word into VTOC entry we're building for new file
4310  1410    TAD I 10
4311  3411    DCA I 11 /MOVE OTHER WORDS - this one is creation attribute
4312  1410    TAD I 10  / this one is unused.
4313  3411    DCA I 11 /END OF VTOC ENTRY construction for new file
4314  1113    TAD 113 /GET UNIT # on which we want to construct file,
4315  3760    DCA I (0 / put into loc 10000
4316  6201    CDF / reset data field
4317  6212    CIF 10
4320  4757    JMS MAKFIL /TRY TO MAKE FILE-i.e., to insert 10020-10037 into VTOC
4321  5756    JMP COPYIT /CAME OUT OK
4322  4755    JMS EOL /NO ROOM LEFT
4323  1354    TAD (14
4324  4753    JMS SYSMSG / tell the user that there's no room
4325  5764    JMP SAVEXIT / returns to caller
4353  2713  PAGE
4354  0014
4355  3112
4356  4400
4357  0275
4360  0000
```

PAGE 27

```
4361  4377
4362  0024
4363  0462
4364  4007
4365  7400
4366  2654
4367  0015
4370  4123
4371  0003
4372  0200
4373  3400
4374  0005
4375  4000
4376  7537
4377  1067
4400  7200  COPYIT, CLA / copy from source file to sink file now that sink has been created
4401  1113   TAD 113 /SET UP PARAMETERS
4402  3116   DCA 116 /FOR TPCOPY
4403  6211   CDF 10
4404  1777   TAD I (24 /GET SINK BLK#, which MAKFIL inserted into loc 10024 in the
4405  6201   CDF 0                                  new file VTOC entry
4406  0376   AND (1777
4407  3117   DCA 117 / tape block# on sink unit
4410  1121   TAD EFCB
4411  1375   TAD (3   / construct pointer to source file unit + blk# word
4412  3010   DCA 10
4413  6211   CDF 10
4414  1410   TAD I 10/ get unit + blk # word
4415  6201   CDF
4416  7006   RTL
4417  0374   AND (1/ pick off unit
4420  3113   DCA 113 /SOURCE UNIT to loc 113
4421  1114   TAD 114
4422  0376   AND (1777/ clean up source tape address
4423  3114   DCA 114 /
4424  1115   TAD 115
4425  0373   AND (377/ clean up source length
4426  3115   DCA 115/
4427  4772   JMS TPCOPY / copy from source to sink
4430  4455   SEARCH / after copying,
4431  0434   UNIT0+434 /POSITION UNIT 0 TAPE
4432  5771   JMP SAVEXIT / return to caller
4571  4007  PAGE
4572  2262
4573  0377
4574  0001
4575  0003
4576  1777
4577  0024
```

PAGE 23

```
                *5000
                /RUN COMMAND MODULE
                /
5000  0000  RUN, 0
5001  6212   CIF 10          / get first operand, which should be a coreimage file
5002  4777   JMS OPSCAN
5003  5210   JMP OKROP       / got an operand
5004  4776   JMS EOL         / no operand - complain to the user
5005  1375   TAD (11
5006  4774   JMS SYSMSG
5007  5600  RUNEXIT, JMP I RUN /return to caller
5010  1105  OKROP, TAD 105   / set up call to COMPRESS
5011  3215   DCA RSAREA
5012  6212   CIF 10          / COMPRESS up to 8,10 char file name for
5013  1373   TAD (10         / call on GETFD.
5014  4772   JMS COMPRESS
5015  0000  RSAREA, 0
5016  0020   0020
5017  6211   CDF 10
5020  6212   CIF 10          / Try to find core-image file in index
5021  4771   JMS GETFD
5022  5236   JMP RFND        / found
5023  4776   JMS EOL         / Cannot find file. Therefore tell user
5024  1370   TAD (6          / it doesn't exist.
5025  4774   JMS SYSMSG
5026  4767   JMS PRTNAM
5027  1366   TAD (7
5030  4774   JMS SYSMSG
5031  1373  RREPSOR, TAD (10 / use QUIZUSER to get a replacement core-image file name
5032  4765   JMS QUIZUSER
5033  7200   CLA             / got a replacement name - handle it as if it
5034  1364   TAD (7400       / were right in the original command line
5035  5211   JMP OKROP+1
5036  1363  RFND, TAD (3     / set AX10 as a pointer to the unit + block # word of the FCB.
5037  3010   DCA 10
5040  6211   CDF 10
5041  1410   TAD I 10        / get unit + block # word for coreimage file
5042  3134   DCA RBLK
5043  1410   TAD I 10        / get length + switches word " " "
5044  3135   DCA RLEN
5045  1410   TAD I 10        / get contents attribute word
5046  6201   CDF
5047  0372   AND (400        / is file a core-image file?
5050  7640   SZA CLA
5051  5261   JMP RFOK        / yes
5052  4776   JMS EOL         / NO - complain to user
5053  1370   TAD (6
5054  4774   JMS SYSMSG
5055  4767   JMS PRTNAM
5056  1362   TAD (23
5057  4774   JMS SYSMSG
5060  5207   JMP RUNEXIT     / instead of exiting, should probably allow replacement of load name
5061  4445  RFOK, RDSTAP     /GET FRESH COPY OF
5062  0447   UNIT0+447       /THE I/O COMMUNICATIONS
5063  4001   FLD1+1          /AREA
```

PAGE 29

```
5064  7600   7600
5065  7201   CLA IAC
5066  3136   DCA DUNIT / initialize implicit I/O unit to 1 for scanning rest of line
5067  6212  RSCAN, CIF 10 / check to see if there is another operand on the line
5070  4777   JMS OPSCAN
5071  5304   JMP RCHK-1 /THERE IS AN OPERAND STRING TO SCAN
5072  4445  LOADIN, RDSTAP /GET COR IMAGE
5073  0450   UNITO+450 /LOADER, SYSTEM BOOT,
5074  0002   FLDO+2    /AND READ-ONLY TAPE ROUTINE   } Prepare to run, since all operands have been scanned
5075  7400   7400
5076  4452   RDRCLO /CLOSE READER
5077  4454   PTRCLO /WAIT UNTIL PRINTER IS DONE
5100  4444   TPWAIT /WAIT UNTIL TAPES ARE DONE
5101  6002   IOF /TURN OF INTT SYSTEM
5102  5703   JMP I .+1 / go to Core image loader
5103  7400   7400
5104  4444   TPWAIT /WAIT UNTIL COMMAREA IS INCORE before scanning
5105  1105  RCHK, TAD 105 / get second char in operand into AC ( C(105) points
5106  3010   DCA 10          to first char in operand string.)
5107  6211   CDF 10
5110  1410   TAD I 10
5111  6201   CDF
5112  1361   TAD (-275 / Is second char an "=" sign?
5113  7640   SZA CLA
5114  5760   JMP IMPLCT / no - user desires implicit specification of I/O unit
5115  6211   CDF 10
5116  1505   TAD I 105 / Copy first char of operand to LUNIT - logical I/O unit
5117  6201   CDF
5120  3137   DCA LUNIT
5121  1137   TAD LUNIT
5122  1357   TAD (-261 / is logical unit char < 261 ?
5123  7510   SPA
5124  5330   JMP UNITNG / yes - no good
5125  1356   TAD (-6 / is it less than 260?
5126  7710   SPA CLA
5127  5755   JMP UNITOK / yes - it lies between 1 + 6 inclusive
5130  7200  UNITNG, CLA / logical unit specification is bad
5131  1137   TAD LUNIT
5132  1354   TAD (-320 / see if it may be a "P" for a parameter specification
5133  7650   SNA CLA
5134  5753   JMP PARAM / is a parameter specification
5135  4776   JMS EOL
5153  5200   TAD (2 / not a "  "  - Complain to user
5154  7460
5155  5213
5156  7772
5157  7517
5160  5222
5161  7503
5162  0023
5163  0003
5164  7400
5165  2654
5166  0007
5167  3130
```

PAGE 30

```
5170  0006
5171  0200
5172  0400
5173  0010
5174  2713
5175  0011
5176  3112
5177  1067
5136  1352  4
5137  4751   JMS SYSMSG
5140  5207   JMP RUNEXIT / and go back to CLI when through complaining
5151  2713  PAGE
5152  0024
5200  1377  PARAM, TAD (-40 /
5201  3012   DCA 12        / prepare to move 32 chars from parameter specification
5202  1376   TAD (7707     / operand up to the communication area
5203  3011   DCA 11        /
5204  6211   CDF 10        /
5205  1410   TAD I 10      /
5206  3411   DCA I 11      / move 32 chars on up
5207  2012   ISZ 12        /
5210  5205   JMP .-3       /
5211  6201   CDF           /
5212  5775   JMP LOADIN / and go run the core image.
5213  1137  UNITOK, TAD LUNIT / come here for explicit logical unit specification
5214  0374   AND (7           / get negative of logical unit # into LUNIT by
5215  7041   CIA              / stripping 260g off + CIA'ing
5216  3137   DCA LUNIT        /
5217  1105   TAD 105 / AC contains point to file name string which follows "X="
5220  1373   TAD (2
5221  5236   JMP RMERG / go here to set up scan
5222  1136  IMPLCT, TAD DUNIT / come here for implicit unit specification
5223  1372   TAD (-5 / is implicit unit less than 5 ?
5224  7710   SPA CLA
5225  5232   JMP LCOMPT / yes
5226  4771   JMS EOL
5227  1370   TAD (25       / no - complain and return to caller
5230  4767   JMS SYSMSG    /
5231  5766   JMP RUNEXIT   /
5232  1136  LCOMPT, TAD DUNIT / get negative of implicit block # into LUNIT
5233  7041   CIA               /
5234  3137   DCA LUNIT         /
5235  1105   TAD 105 / AC contains pointer to file name string.
5236  3260  RMERG, DCA FNAME / FNAME contains address of string to scan.
5237  1137   TAD LUNIT / is the logical unit ≤ 3 - an input unit ?
5240  1365   TAD (3    /
5241  7710   SPA CLA
5242  5245   JMP OUTUNIT / no - output unit
5243  1364   TAD (4000 / mask for checking for an input FD
5244  7410   SKP
5245  1363  OUTUNIT, TAD (2000 / output FD mask
5246  3261   DCA RMASK / deposit it into direction-checking mask
5247  1362   TAD (7600-14 / set up initial linkage area address for unit "zero"
5250  3257   DCA LINKAREA /
5251  1257   TAD LINKAREA / now add 14 to this address as
```

PAGE 31

```
5252  1361    TAD (14
5253  3257    DCA LINKAREA
5254  2137    ISZ LUNIT
5255  5251    JMP .-4
5256  4760    JMS CONSCAN
5257  0000  LINKAREA, 0
5260  0000  FNAME, 0
5261  0000  RMASK, 0
5262  1136    TAD DUNIT
5263  1365    TAD (3
5264  3136    DCA DUNIT
5265  5757    JMP RSCAN
5357  5067  PAGE
5360  5401
5361  0014
5362  7564
5363  2000
5364  4000
5365  0003
5366  5007
5367  2713
5370  0025
5371  3112
5372  7773
5373  0002
5374  0007
5375  5072
5376  7707
5377  7740
```

/ many times as the number of the current I/O unit

/ when done, LINKAREA contains address of FD chain for this I/O unit in the communication area

/ call conscan to scan the file name list pointed to by FNAME & set up FD chain in area ptd to by LINKAREA

/ increment implicit I/O unit number by 3

/ go check for another operand

```
                *5400
                /CONCATENATED FILE STRING SCANNING ROUTINE
                / See subroutine writeup section for calling sequence
5400  5601  CONRTN, JMP I CONSCAN
5401  0000  CONSCAN, 0
5402  7200   CLA
5403  1601   TAD I CONSCAN  / get address of where FD chain we're to build is to go
5404  3140   DCA LKAREA     / save on p.0
5405  2201   ISZ CONSCAN
5406  1601   TAD I CONSCAN  / get address of file-device name string (assumed to be in
5407  3141   DCA FNAMPTR    /                                  bank 1- the string that is)
5410  2201   ISZ CONSCAN
5411  1601   TAD I CONSCAN  / get source/sink check mask
5412  3777   DCA CHKMASK    /  and save it
5413  2201   ISZ CONSCAN
5414  1376   TAD (-7        / set file count to -7, => 6 files max can be scanned
5415  3775   DCA FILCNT     /
5416  3774   DCA QUIT       / reset QUIT to "off"
5417  7240   STA
5420  1141   TAD FNAMPTR    / save address of file string -1 for later use with AX regs.
5421  3773   DCA FSTPTR     /
5422  1372  CSCAN, TAD (-10 / set up CHCNT to scan at most 8_10 chars for a file name.
5423  3771   DCA CHCNT      /
5424  6211  CHLK, CDF 10
5425  1541   TAD I FNAMPTR  / get next char in next file name
5426  6201   CDF            /
5427  1370   TAD (-240      /
5430  7440   SZA            / is it a blank, which implies "quit after handling this name"?
5431  5235   JMP PLUSCHK    / no
5432  7240  QUITSET, CLA CMA/
5433  3774   DCA QUIT       / yes - set quit switch to "on"
5434  5257   JMP MOVNAM     / then go process name
5435  1367  PLUSCHK, TAD (-13/ is the char a "+" sign?
5436  7650   SNA CLA        /
5437  5257   JMP MOVNAM     / yes - go process the name accumulated so far
5440  2771   ISZ CHCNT      / no - have we gone for 8 chars?
5441  7410   SKP            / no; < 8
5442  5245   JMP SPACFWD    / yes, = 8 chars; go to spacfwd to move FNAMPTR to next name
5443  2141   ISZ FNAMPTR    /
5444  5224   JMP CHLK       / scan next character.
5445  2141  SPACFWD, ISZ FNAMPTR/
5446  6211   CDF 10         /
5447  1541   TAD I FNAMPTR  / get next char in file name
5450  6201   CDF            /
5451  1370   TAD (-240      /
5452  7450   SNA            / if it equals blank, set quit switch
5453  5232   JMP QUITSET    /
5454  1367   TAD (-13       / does it equal "+" sign
5455  7640   SZA CLA        /
5456  5245   JMP SPACFWD    / no - space forward more
5457  1366  MOVNAM, TAD (10/
5460  1771   TAD CHCNT      / compute negative of # of chars in file name
5461  7041   CIA            /
5462  7450   SNA            / skip if non-zero
5463  5765   JMP CONEND     / otherwise go to postlog since no file name can be the null
                                                                          string
```

PAGE 33

```
5464  3012    DCA 12 / save count in a Temporary for later ISZ'ing
5465  1773    TAD FSTPTR/
5466  3010    DCA 10  / set up AX reg 10 for transferring file name to FAREA (loc 10030)
5467  1364  GETFN, TAD (FAREA-1/
5470  3011    DCA 11          / set AX 11 to point to FAREA
5471  6211    CDF 10
5472  1410    TAD I 10
5473  3411    DCA I 11  / transfer file name to FAREA
5474  2012    ISZ 12
5475  5272    JMP .-3/
5476  1363    TAD (240
5477  3411    DCA I 11/ add a trailing blank for the COMPRESS routine
5500  6201    CDF       /restore data field
5501  1366    TAD (10
5502  6212    CIF 10
5503  4762    JMS COMPRESS  / compress up to 8,10 chars from FAREA to 10020 in
5504  0030    FAREA         / field 1
5505  0020    0020
5506  6211    CDF 10
5507  6212    CIF 10
5510  4761    JMS GETFD  / now try to find an FCB for the file
5511  5760    JMP CONFND / file found in index
5512  4757    JMS EOL
5513  1356    TAD (6
5514  4755    JMS SYSMSG / file not found- complain to user
5515  4754    JMS PRTNAM/
5554  3130    TAD (7
5555  2713
5556  0006
5557  3112
5560  5610
5561  0200
5562  0400
5563  0240
5564  0027
5565  5651
5566  0010
5567  7765
5570  7540
5571  5663
5572  7770
5573  5664
5574  5662
5575  5661
5576  7771
5577  5660
5516  1353
5517  4752    JMS SYSMSG
5520  5751    JMP .&7600+200
5551  5600  PAGE
5552  2713
5553  0007
5600  1377  REPFIL, TAD (10/ use QUIZUSER to ask for a replacement file name
5601  4776    JMS QUIZUSER
5602  7200    CLA
```

PAGE 34

```
5603  1375   TAD (-10
5604  3012   DCA 12      / reset character count in file name to 8
5605  1374   TAD (7377
5606  3010   DCA 10      / let 10 point to QUIZUSER's buffer
5607  5773   JMP GETFN   / and nobody will be the wiser
5610  1372 CONFND, TAD (3    / construct pointer for word 4 of the file index
5611  3010   DCA 10          and use in AX 10.
5612  6211   CDF 10
5613  1410   TAD I 10
5614  3017   DCA 17      / copy file unit & block # to locn 17
5615  1410   TAD I 10
5616  3016   DCA 16      / copy file switches and length to locn 16
5617  6201   CDF
5620  1016   TAD 16
5621  0260   AND CHKMASK / check to see if switch designated by CHKMASK is set.
5622  7640   SZA CLA
5623  5233   JMP CHKOK   / it is
5624  4771   JMS EOL
5625  1370   TAD (6
5626  4767   JMS SYSMSG      / it is not - complain to user and allow him to replace
5627  4766   JMS PRTNAM        the file name
5630  1365   TAD (26
5631  4767   JMS SYSMSG
5632  5200   JMP REPFIL
5633  2261 CHKOK, ISZ FILCNT / check to see if more than 6 filenames have been processed
5634  5241   JMP LKSTUFF / no - go link this file into the link area
5635  4771   JMS EOL
5636  1364   TAD (27
5637  4767   JMS SYSMSG  / too many files - complain & then return to CLI
5640  5763   JMP CLI
5641  6211 LKSTUFF, CDF 10
5642  1017   TAD 17
5643  3540   DCA I LKAREA / put file unit & blk # into LINKAREA
5644  2140   ISZ LKAREA
5645  1016   TAD 16
5646  3540   DCA I LKAREA / put file switches and length word into LINKAREA
5647  6201   CDF 0
5650  2140   ISZ LKAREA
5651  1141 CONEND, TAD FNAMPTR / update File STart PoinTer to point to one less than
5652  3264   DCA FSTPTR          the next file name address in the file name string
5653  2141   ISZ FNAMPTR / update
5654  1262   TAD QUIT
5655  7700   SMA CLA     / is quit switch set?
5656  5762   JMP CSCAN   / no, do another file
5657  5761   JMP CONRTN  / yes - return to caller
               /
5660  0000 CHKMASK, 0 / holds file switch word checking mask
5661  0000 FILCNT, 0  / holds count of # of names processed
5662  0000 QUIT, 0    / -1 if end of name string has been encountered, 0 otherwise
5663  0000 CHCNT, 0   / count of # of chars in a file name
5664  0000 FSTPTR, 0  / points to 1 loc less than the start of the current file name
5761  5400 PAGE
5762  5422
5763  3000
5764  0027
```

PAGE 35

```
5765  0026
5766  3130
5767  2713
5770  0006
5771  3112
5772  0003
5773  5467
5774  7377
5775  7770
5776  2654
5777  0010
```

PAGE 36

```
                *6000
                /
                /SIGNOFF COMMAND MODULE
                /
6000  0000      SIG, 0
6001  4446       WRSTAP /WRITE OUT VTOC 1
6002  4160       UNIT1+SVTOC1
6003  6010       FLD1+CHECK+10
6004  4000       4000
6005  4446       WRSTAP /WRITE OUT VTOC0
6006  0470       UNIT0+SVTOC0
6007  6010       FLD1+CHECK+10
6010  2000       2000
6011  4455       SEARCH /REWIND UNIT1
6012  4002       UNIT1+2
6013  1060       TAD SRCHSW
6014  7640       SZA CLA
6015  5213       JMP .-2
6016  4455       SEARCH /REWIND UNIT0
6017  0002       UNIT0+2
6020  1060       TAD SRCHSW
6021  7640       SZA CLA
6022  5220       JMP .-2
6023  4777       JMS EOL
6024  1376       TAD (30      / output "BYE" message
6025  4775       JMS SYSMSG   /
6026  4452       RDRCLO /SHUT UP SHOP
6027  4454       PTRCLO
6030  6002       IOF
6031  7402       HLT
6032  5774       JMP SYSTART  / just in case user wants to go again
6174  2000      PAGE
6175  2713
6176  0030
6177  3112
```

```
                    *6400
                    /
                    /LOADER INVOCATION COMMAND MODULE
                    /
6400  0000  LOAD,  0
6401  6212   CIF 10
6402  4777   JMS OPSCAN / look for an operand
6403  5210   JMP CKLOP / got one
6404  4776   JMS EOL   / no operand - complain to user
6405  1375   TAD (11   /
6406  4774   JMS SYSMSG/
6407  5600  LEX,  EXIT LOAD / return to caller.
6410  6211  CKLOP,  CDF 10
6411  1505   TAD I 105  / get first char of operand into AC
6412  6201   CDF
6413  1373   TAD (-301  / is it an "A"?
6414  7450   SNA
6415  5225   JMP ABSLOD / yes
6416  1372   TAD (-21   / is it an "R"?
6417  7650   SNA CLA
6420  5240   JMP RELLOD / yes
6421  4776   JMS EOL    / neither A nor R - complain and
6422  1371   TAD (20    / then return to caller
6423  4774   JMS SYSMSG /
6424  5207   JMP LEX
6425  1370  ABSLOD,  TAD (". / set prefix char to "."
6426  4767   JMS PUSHPFX
6427  4445   RDSTAP /GET THE PIECES OF THE ABS LODR
6430  4240   UNIT1+240 /FIRST PIECE SITS HERE
6431  0012   FLD0+12
6432  3400   3400
6433  4445   RDSTAP /GET SECOND PIECE
6434  4252   UNIT1+252
6435  4001   FLD1+1
6436  0600   600 /OVERWRITE COMMAND TABLE
6437  5600   EXIT LOAD /RETURN TO RECONFIGURED SYSTEM
6440  4445  RELLOD,  RDSTAP /GET CORE IMAGE LOADER TO
6441  0450   UNIT0+450 /LOAD RELOCATING LOADER
6442  0002   FLD0+2
6443  7400  RADR,  7400
6444  4452   RDRCL0
6445  4454   PTRCL0
6446  1366   TAD (4260
6447  3134   DCA 134
6450  1365   TAD (5040 /FAKE A CALL TO THE CI LDR
6451  3135   DCA 135
6452  4444   TPWAIT /WAIT UNTIL ALL PENDING I/O IS DONE
6453  6002   IOF /TURN OFF INTT SYS
6454  5643   JMP I RADR /GO TO CORE IMAGE LDR
6565  5040  PAGE
6566  4260
6567  3103
6570  0256
6571  0020
6572  7757
```

PAGE 38

```
6573  7477
6574  2713
6575  0011
6576  3112
6577  1067
```

```
                *6600
                /
                /SYSTEM MESSAGES - LONG FORM
                /
6600  6577      MSG, .-1
6601  6637       M1
6602  6646       M2
6603  6654       M3
6604  6664       M4
6605  6702       M5
6606  6714       M6
6607  6715       M7
6610  6726       M8
6611  6746       M9
6612  6757       M10
6613  6770       M11
6614  7007       M12
6615  7022       M13
6616  7027       M14
6617  7044       M15
6620  7056       M16
6621  7066       M17
6622  7071       M18
6623  7101       M19
6624  7113       M20
6625  7126       M21
6626  7147       M22
6627  7171       M23
6630  7207       M24
6631  7214       M25
6632  7253       E
6633  7253       E
6634  7253       E
6635  7253       E
6636  7253       E /30 - space for up to 30(10) messages in this table
                /
6637  1411      M1, TEXT /LI
6640  1605      NE
6641  4024       T
6642  1717      OO
6643  4014       L
6644  1716      ON
6645  0700      G/
6646  0320      M2, TEXT ;CP
6647  2350      S(
6650  0131      AY
6651  6261      21
6652  7151      9)
6653  0000      ;
6654  1116      M3, TEXT /IN
6655  2601      VA
6656  1411      LI
6657  0440      D
6660  0317      CO
6661  1515      MM
```

PAGE 40

```
6662  0116  AN
6663  0400  D/
6664  0317  M4, TEXT /CO
6665  1515  MM
6666  0116  AN
6667  0440  D
6670  1617  NO
6671  2440  T
6672  3105  YE
6673  2440  T
6674  1115  IM
6675  2014  PL
6676  0515  EM
6677  516   EN
6700  2405  TE
6701  0400  D/
6702  4240  M5, TEXT /"
6703  1617  NO
6704  2440  T
6705  0140  A
6706  2317  SO
6707  2522  UR
6710  0305  CE
6711  4006   F
6712  1114  IL
6713  0500  E/
6714  4200  M6, TEXT /"/
6715  4240  M7, TEXT /"
6716  0417  DO
6717  0523  ES
6720  4016   N
6721  1724  OT
6722  4005   E
6723  3011  XI
6724  2324  ST
6725  0000  /
6726  0516  M8, TEXT /EN
6727  2405  TE
6730  2240  R
6731  2205  RE
6732  2014  PL
6733  0103  AC
6734  0515  EM
6735  0516  EN
6736  2440  T
6737  1601  NA
6740  1505  ME
6741  4017   O
6742  2240  R
6743  7403  <C
6744  2276  R>
6745  0000  /
6746  2417  M9, TEXT /TO
6747  1740  O
6750  0605  FE
```

PAGE 41

```
6751  2740  W
6752  1720  OP
6753  0522  ER
6754  0116  AN
6755  0423  DS
6756  0000  /
6757  1116  M10, TEXT /IN
6760  2601  VA
6761  1411  LI
6762  0440  D
6763  0611  FI
6764  1405  LE
6765  4016   N
6766  0115  AM
6767  0500  E/
6770  0611  M11, TEXT /FI
6771  1405  LE
6772  4023   S
6773  1017  HO
6774  2514  UL
6775  0440  D
6776  1617  NO
6777  2440  T
7000  0205  BE
7001  4004   D
7002  0523  ES
7003  2422  TR
7004  1731  OY
7005  0504  ED
7006  0000  /
7007  1617  M12, TEXT /NO
7010  4022   R
7011  1717  OO
7012  1540  M
7013  1405  LE
7014  0624  FT
7015  4017   O
7016  1640  N
7017  2401  TA
7020  2005  PE
7021  0000  /
7022  2205  M13, TEXT /RE
7023  2014  PL
7024  0103  AC
7025  0577  E?
7026  0000  /
7027  0611  M14, TEXT /FI
7030  1405  LE
7031  4003   C
7032  0116  AN
7033  1617  NO
7034  2440  T
7035  0205  BE
7036  4004   D
7037  0523  ES
```

PAGE 42

```
7040  2422  TR
7041  1731  OY
7042  0504  ED
7043  0000  /
7044  0611  M15, TEXT /FI
7045  1405  LE
7046  4004   D
7047  1705  OE
7050  2340  S
7051  1617  NO
7052  2440  T
7053  0530  EX
7054  1123  IS
7055  2400  T/
7056  1116  M16, TEXT /IN
7057  2601  VA
7060  1411  LI
7061  0440  D
7062  1720  OP
7063  0522  ER
7064  0116  AN
7065  0400  D/
7066  2516  M17, TEXT /UN
7067  1124  IT
7070  4000   /
7071  4543  M18, TEXT /%#
7072  7505  =E
7073  1604  ND
7074  4017   0
7075  0640  F
7076  0611  FI
7077  1405  LE
7100  0000  /
7101  4240  M19, TEXT /"
7102  1617  NO
7103  2440  T
7104  0140  A
7105  0317  CO
7106  2205  RE
7107  4011   I
7110  1501  MA
7111  0705  GE
7112  0000  /
7113  1114  M20, TEXT ;IL
7114  1405  LE
7115  0701  GA
7116  1440  L
7117  1157  I/
7120  1740  0
7121  2516  UN
7122  1124  IT
7123  4016   N
7124  1756  0.
7125  0000  ;
7126  1115  M21, TEXT ;IM
```

PAGE 43

```
7127  2014  PL
7130  1103  IC
7131  1124  IT
7132  4011   I
7133  5717  /O
7134  5520  -P
7135  0122  AR
7136  0115  AM
7137  0524  ET
7140  0522  ER
7141  4023   S
7142  3116  YN
7143  2401  TA
7144  3040  X
7145  0201  BA
7146  0400  D;
7147  4240  M22, TEXT /"
7150  0201  BA
7151  0440  D
7152  5540  -
7153  0411  DI
7154  2205  RE
7155  0324  CT
7156  1117  IO
7157  1640  N
7160  1706  OF
7161  4004   D
7162  0124  AT
7163  0140  A
7164  2422  TR
7165  0116  AN
7166  2306  SF
7167  0522  ER
7170  0000  /
7171  2417  M23, TEXT /TO
7172  1740  O
7173  1501  MA
7174  1631  NY
7175  4003   C
7176  1716  ON
7177  0301  CA
7200  2405  TE
7201  1601  NA
7202  2405  TE
7203  0440  D
7204  0611  FI
7205  1405  LE
7206  2300  S/
7207  0231  M24, TEXT ;BY
7210  0545  E%
7211  4343  ##
7212  4343  ##
7213  0000  ;
7214  2401  M25, TEXT ;TA
7215  2005  PE
```

PAGE 44

```
7216  4005   E
7217  2222  RR
7220  1722  OR
7221  4017   O
7222  0303  CC
7223  2522  UR
7224  2205  RE
7225  0440  D
7226  1116  IN
7227  4001   A
7230  0217  BO
7231  2605  VE
7232  4017   O
7233  2005  PE
7234  2201  RA
7235  2411  TI
7236  1716  ON
7237  5440  ,
7240  2711  WI
7241  1414  LL
7242  4001   A
7243  2424  TT
7244  0515  EM
7245  2024  PT
7246  4024   T
7247  1740  O
7250  0717  GO
7251  4017   O
7252  1600  N;
7253  1116  E,    TEXT /IN
7254  2601  VA
7255  1411  LI
7256  0440  D
7257  2331  SY
7260  2315  SM
7261  2307  SG
7262  0000  /
            PAGE
```

```
                    FIELD 1
                    /PAGE ZERO IN BANK 1 - CPS
                    /NOTE THAT THIS PAGE IS A COMMON DATA
                     /SECTION FOR ALL ROUTINES IN BANK 1
                    /LOCNS 10-17 USED FOR SCRATCH &AX
                     /LOCNS 20-37 USED TO BUILD FILE NAMES,
                     /VTOCS AND THE LIKE
                    /
                    /LOCNS 45 & 46 USED FOR TAPE ROUTINE ADCONS for IOCS3
                    *45
0045  1000           1000 /RDSTAP } these entry points are really in bank 0.
0046  1005           1005 /WRSTAP }
                    /
                    *100
0100  6203  CIFLD,  CIF CDF /USED TO SAVE RETRN FLD
0101  0000  AVTOC,  0
0102  0000  AFFB,   0
0103  0000  ANBLKS, 0
0104  0000  BVTOC,  0
0105  0000  UVTOC,  0
0106  0000  DEVCTR, 0
0107  2000  ATBL,   2000 /VTOC CORE ADDRS / unit 0
0110  4000           4000                / unit 1
0111  0000  MRTEMP, 0
0112  0400  BTBL,   400 /VTOC TAPE ADDRS - unit 0
0113  0170           170 / unit 1
0114  0000  RTEMP,  0
0115  0000  PKDEST, 0
0116  0000  PKBLKS, 0
0117  0000  PKSOUR, 0
0120  1037  AGABUV, GABUV
0121  1230  ARELTAP, RELTAP / addresses for subroutine jumps on account of there wasn't
                              any room left on the page.
0122  0000  NENT,   0
                    PAGE
```

PAGE 02

```
                *200
                /
                /MACRO FOR SAVING RETURN FIELD
                DEFINE SAVRTN SAVDUM
                <TAD CIFLD
                 RDF
                 DCA SAVDUM>
                /
                /GETFD - GET FILE OR DEVICE.  ROUTINE
                /RETURNS A FCB OR DCB POINTER FOR THE
                /FD WHOSE NAME IS IN 10020 - 10023 IN
                /STANDARD FORM.  IF THE FD DOES NOT
                /EXIST, AN APPROPRIATE RETURN IS MADE.
                /SINCE ALL CONTROL BLOCKS ARE IN MEMFLD 1,
                /ONLY THE LOW ORDER 12 BITS OF THE PTR
                /ARE RETURNED.
                /IN ADDITION TO USER FILES, THE FOLLOWING
                /DEVICES AND PSEUDOFILES ARE SUPPORTED:
                /RDR  /33ASR reader
                /PTR  /33ASR printer
                /*DUMMY* /wastebasket
                /
0200  0000      GETFD, 0
0201  7200       CLA
0202  1100       SAVRTN FDRTNO
0203  6214                  /save return memory field
0204  3232
0205  6211       CDF 10
0206  3000       DCA 0 /SRCHVTOC OF UNIT 0
0207  4777       JMS SRCHVTOC
0210  5232       JMP FDRTNO /FILE FOUND
0211  7201       CLA IAC /SRCHVTOC OF UNIT 1
0212  3000       DCA 0
0213  4777       JMS SRCHVTOC
0214  5232       JMP FDRTNO /FILE FOUND
0215  7200       CLA /IF NOT A FILE, THEN SET
0216  1376       TAD (-NDEVS /UP FOR SEARCHING
0217  3106       DCA DEVCTR /DEVICES.
0220  1375       TAD (DEVTBL
0221  3224       DCA .+3
0222  1374      DEVLK, TAD (4
0223  4773       JMS CLW          /compare 4 words
0224  0245      DEVADR, DEVTBL /CHANGES W/ DEVICE
0225  0020       0020 /NAME IS HERE
0226  6211       CDF 10 /ALL IN BANK 1
0227  5234       JMP TESTAGN /NOT FOUND
0230  7200       CLA /FOUND; RETURN PTR
0231  1224       TAD DEVADR
0232  7402      FDRTNO, HLT /REP BY CIDF
0233  5600       EXIT GETFD
0234  2106      TESTAGN, ISZ DEVCTR /TESTED ALL DEVICES?
0235  5240       JMP .+3 /NO
0236  2200       ISZ GETFD /YES. RETURN
0237  5232       JMP FDRTNO
0240  7200       CLA
```

NB: Standard form is stripped ASCII

→ to call +2

Otherwise return is made to call+1 with FCB pointer in AC

to call +2 since name not found.

PAGE 03

```
0241  1224   TAD DEVADR/UPDATE DEVICE ADR
0242  1372   TAD (10
0243  3224   DCA DEVADR
0244  5222   JMP DEVLK
             /
             NDEVS=3 /AS OF 12/14/68
             / Device table for devices OS recognizes follows:
0245  2204   DEVTBL, TEXT /RD
0246  2240   R
0247  4040
0250  4040
0251  0000   /
             *.-1
0251  7775    -3 /READER CODE
0252  5000    5000 /DON'T TOUCH,SOURCE FILE
0253  0200    0200 /ASCII CHARACTERS
0254  0000    0000
0255  2024    TEXT /PT
0256  2240   R
0257  4040
0260  4040
0261  0000   /
             *.-1
0261  7776    -2  /PTR CODE
0262  3000    3000 /DON'T TOUCH, SINK FILE
0263  0200    0200 /ASCII CHARACTERS
0264  0000    0000
0265  5204    TEXT /*D
0266  2515   UM
0267  1531   MY
0270  5240   *
0271  0000   /
             *.-1
0271  7777    -1   /*DUMMY* CODE
0272  7000    7000 /DON'T TOUCH, EITHER SOURCE OR SINK
0273  7700     7700 /CONTAINS ANYTHING
0274  0000    0000
             /
             /
             /MAKFIL - ROUTINE TO CREATE A FILE ENTRY
             /IN A VTOC.  WHEN CALLED, A PARTIAL VTOC
             /MUST BE IN LOCS 10020-10027 (ALL EXCEPT
             /FIRST BLK MUST BE FILLED IN). UNIT UPON
             /WHICH FILE IS TO BE MADE IS IN 10000.
             /
0275  0000   MAKFIL, 0
0276  7200    CLA
0277  1100    SAVRTN MKRTNO/
0300  6214                  / save return field
0301  3353
0302  6211    CDF 10
0303  4771    JMS GABUV/ set up Address, Block, & Unit words for VTOC given in 10000
0304  1101    TAD AVTOC
0305  7001    IAC /PTR TO # OF ENTRIES
0306  3111    DCA MRTEMP
```

PAGE 04

```
0307  1511    TAD I MRTEMP /GET # OF ENTRIES
0310  1370    TAD (176 /ROOM LEFT?
0311  7710    SPA CLA
0312  5352    JMP MKRTN1 /NO ROOM
0313  1025    TAD 25 /GET # OF BLKS NEEDED
0314  0367    AND (377 /MASK OFF SWITCHES
0315  3001    DCA 1 /FOR GETTAP
0316  4766    JMS GETTAP /GET SPACE FOR FILE
0317  7410    SKP /GOT SPACE
0320  5352    JMP MKRTN1 /NO ROOM
0321  3010    DCA 10 /SAVE BEGINNING BLK#
0322  1000    TAD 0 /ENTER IT INTO USERS VTOC
0323  7112    RTR CLL
0324  1010    TAD 10
0325  3024    DCA 24 /USERS VTOC COMPLETE
0326  7240     CLA CMA
0327  1511    TAD I MRTEMP /INCR # OF ENTRIES
0330  3511    DCA I MRTEMP
0331  1511    TAD I MRTEMP /SETUP TO ADD ENTRY
0332  7041     CIA /TO VTOC
0333  7106    RTL CLL
0334  7004    RAL /MULT BY 8
0335  1101    TAD AVTOC /THIS IS WHERE ENTRY GOES
0336  1365     TAD (-1
0337  3010    DCA 10 /SINK ADR
0340  1364    TAD (17
0341  3011    DCA 11
0342  1363    TAD (-10 /COUNT
0343  3012    DCA 12
0344  1411    TAD I 11 /MOVE 8 WORDS
0345  3410    DCA I 10
0346  2012    ISZ 12
0347  5344    JMP .-3
0350  4762    JMS WVTOC /WRITE OUT CHANGED VTOC
0351  7410     SKP
0352  2275  MKRTN1, ISZ MAKFIL
0353  7402  MKRTN0, HLT /REP BY CIDF
0354  5675   EXIT MAKFIL
0362  1270  PAGE
0363  7770
0364  0017
0365  7777
0366  1200
0367  0377
0370  0176
0371  1037
0372  0010
0373  1124
0374  0004
0375  0245
0376  7775
0377  1000
```

from partial VTOC entry

PAGE 05

```
                /COMPRESS - SQUEEZES CHARACTERS FROM 1 TO 2
                /PER WORD.    GNC
                /12/12/68
                /
0400  0000    COMPRESS, 0
0401  7041     CIA /SAVE -N - # of chars to compress was in AC when call was made
0402  3012     DCA N
0403  7240     CLA CMA
0404  1600     TAD I COMPRESS
0405  3010     DCA 10 /SOURCE LIST ADR to AX reg 10.
0406  2200     ISZ COMPRESS
0407  7240     CMA CLA
0410  1600     TAD I COMPRESS
0411  3011     DCA 11 /SINK LIST ADR to AX reg 11.
0412  2200     ISZ COMPRESS
0413  1600     TAD I COMPRESS /GET CDF for the field in which the source & sink of characters
0414  3222     DCA COMCDF                                                          sit.
0415  2200     ISZ COMPRESS
0416  1100     SAVRTN COMRTN/
0417  6214                   / save return field
0420  3235
0421  3013     DCA RSW  / reset RSW to 0 to indicate stuffing in the left half first
0422  7402    COMCDF, HLT /REP BY CDF to character source & sink field.
0423  1410     TAD I 10 /
0424  3014     DCA TEMP/ get a character & save it
0425  1014     TAD TEMP
0426  1377     TAD (-240/ is the char a blank?
0427  7650     SNA CLA /
0430  5237     JMP BLANKS/yes
0431  1014     TAD TEMP/
0432  4244     JMS STUFF/ no- stuff into output string
0433  2012     ISZ N      / have we done N chars yet?
0434  5223     JMP COMCDF+1 / no
0435  7402    COMRTN, HLT /REP BY CIDF to caller's field
0436  5600     EXIT COMPRESS /return to caller
0437  1376    BLANKS, TAD (240/ fill up remainder of output string with blanks
0440  4244     JMS STUFF       /
0441  2012     ISZ N           / have we done N chars yet?
0442  5237     JMP BLANKS      / no- fill with more blanks
0443  5235     JMP COMRTN      / yes- return to caller
0444  0000    STUFF, 0     / subroutine to put stripped char into the output string.
0445  0375     AND (77      / strip char
0446  2013     ISZ RSW      / does it go in the right or the left half of the output string?
0447  5253     JMP LEFT     / left side
0450  1015    RIGHT, TAD HOLD/ add AC to left side handled previously
0451  3411     DCA I 11     / deposit into output string
0452  5644     EXIT STUFF
0453  7106    LEFT, CLL RTL/ put stripped char into left end of AC
0454  7006     RTL
0455  7006     RTL
0456  3015     DCA HOLD  / save in HOLD
0457  7040     CMA       / set RSW to -1
0460  3013     DCA RSW /
0461  5644     EXIT STUFF
                /
```

PAGE 06

```
                    N=12
                    RSW=13
                    TEMP=14
                    HOLD=15
                    /
                    /DESTRY - DESTROY A FILE; CALL W/ FCB PTR
                     /IN AC.
                    /
0462  0000  DESTRY, 0
0463  3114   DCA RTEMP
0464  1100   SAVRTN DYRTN
0465  6214
0466  3363
0467  6211   CDF 10
0470  1114   TAD RTEMP
0471  0374   AND (4000
0472  7640   SZA CLA  / skip if on unit 0
0473  7001   IAC      / IAC if on unit 1.
0474  3000   DCA 0    / save tape unit in loc'n 0 for call on
0475  4520   CALL AGABUV
0476  1101   TAD AVTOC
0477  7001   IAC
0500  3111   DCA MRTEMP
0501  1511   TAD I MRTEMP
0502  7001   IAC
0503  3511   DCA I MRTEMP
 0504  1114   TAD RTEMP
0505  1373   TAD (3
0506  3010   DCA 10
0507  1410   TAD I 10 /GET  FB IN FILE
0510  0372   AND (1777
0511  3002   DCA 2 /SAVE FOR RELTAP CALL
0512  1410   TAD I 10 /# OF BLKS IN FILE
0513  0371   AND (377
0514  3001   DCA 1 /ALSO FOR RELTAP
0515  4521   CALL ARELTAP /FREE TAPE
0516  7240   CLA CMA
0517  1114   TAD RTEMP
0520  3011   DCA 11 /DESTINATION ADR FOR DELETING
 0521  2010   ISZ 10
0522  2010   ISZ 10 /SOURCE ADR
0523  1101   TAD AVTOC
0524  1370   TAD (10
0525  7041   CIA
0526  1114   TAD RTEMP
0527  7112   RTR CLL
0530  7110   RAR CLL
0531  1511   TAD I MRTEMP
0532  3012   DCA 12
0533  1012   TAD 12 /NEG OF # OF ENTRIES TO MOVE
 0534  7650   SNA CLA
0535  5362   JMP DYRTN-1 /NO
0536  1001   TAD 1 /YES
0537  7041   CIA  / FORM THE
0540  3001   DCA 1 /NEG OF # OF BLKS RELEASED
```

/ definitions for COMPRESS

/ save FCB pointer

/ save return field + set data field

/ is the file on unit 1 or 0 - test the location of the FCB to see.

/ GABUV to get info words about this VTOC

/ 1 + address of VTOC = ptr to "# of files" word in VTOC

/ # OF FILES DECREASED BY 1

/ AX 10 points to address of "first block" word in FCB

FB

/ strip off unneeded bits.

/ strip off unneeded bits

/ FREE TAPE space occupied by the file being destroyed.

/ Now pack up the VTOC. Form

/ AX ptr to FCB to be deleted.

/ increment AX 10 twice to get

/ SOURCE ADR for remaining FCB's in VTOC

/ Use nice hairy algorithm to compute the number of VTOC entries - FCB's - to move down over the old FCB to be destroyed, as a negative #.

/ are there any entries to be moved?

PAGE 07

```
0541  1367  MOVTOC, TAD (-4 / use AX13 as a counter
0542  3013   DCA 13
0543  1410   TAD I 10 /MOVE FILENAME, which is 4 words long.
0544  3411   DCA I 11
0545  2013   ISZ 13
0546  5343   JMP .-3
0547  1410   TAD I 10 /OLD FILE ADDR
0550  1001   TAD 1    /- # OF BLKS RELEASED
0551  3411   DCA I 11 /=NEW FILE ADDRESS
0552  1410   TAD I 10
0553  3411   DCA I 11
0554  1410   TAD I 10
0555  3411   DCA I 11
0556  1410   TAD I 10
0557  3411   DCA I 11
0560  2012   ISZ 12   / done?
0561  5341   JMP MOVTOC / no
0562  4766   JMS WVTOC / yes - write the modified VTOC out onto tape
0563  7402  DYRTN, HLT /REP BY CIFD to return field
0564  5662   EXIT DESTRY / return to caller.
0566  1270  PAGE
0567  7774
0570  0010
0571  0377
0572  1777
0573  0003
0574  4000
0575  0077
0576  0240
0577  7540
```

move rest of FCB down

PAGE 08

```
                /COMMAND TABLE - USED BY CLI TO DECODE
                /USER COMMANDS
                /
                COMTBL=.
                /
0600    0323    SAV, "S
0601    0301     "A
0602    0326     "V
0603    4000     SAVE
0604    0304    DES, "D
0605    0305     "E
0606    0323     "S
0607    3600     DESTROY
0610    0311    IND, "I
0611    0316     "N
0612    0304     "D
0613    3400     INDEX
0614    0320    PUNT, "P
0615    0325     "U
0616    0316     "N
0617    3335     NOTYET
0620    0314    LOA, "L
0621    0317     "O
0622    0301     "A
0623    6400     LOAD
0624    0322    RRUN, "R
0625    0325     "U
0626    0316     "N
0627    5000     RUN
0630    0303    COM, "C
0631    0317     "O
0632    0315     "M
0633    3342     COMMENT
0634    0323    SET, "S
0635    0305     "E
0636    0324     "T
0637    3335     NOTYET
0640    0323    SSIG, "S
0641    0311     "I
0642    0307     "G
0643    6000     SIGNOFF
                /
0644    4000     END
                /
                END=4000
                PAGE
```

Format of an entry is 3 words containing the first 3 ASCII characters in the command verb, followed by one word containing the address of the command service module in memory bank Ø.

/ The end of the table is signified by the high order bit of the first word of a table entry being set.

PAGE 09

```
                    /VTOC SEARCHING ROUTINE - UNIT IN LOC
                    /10000, STANDARD FORM FILE NAME IN
                    /10020 - 10023; i.e., stripped ASCII
                    /
1000  0000  SRCHVTOC, 0
1001  7200   CLA
1002  1100   SAVRTN SCRTNO   / save return field and
1003  6214                  / set up data field.
1004  3232
1005  6211   CDF 10
1006  4237   JMS GABUV       / get address info, etc about VTOC
1007  1101   TAD AVTOC       / AX 10 = pointer to "# of files on this tape" word
1010  3010   DCA 10
1011  1410   TAD I 10 /GET # OF FILE ENTRIES
1012  3122   DCA NENT / and save it
1013  1101   TAD AVTOC       / set up ENTADR to point to next FCB to
1014  1377  VTOCLK, TAD (10  / be tested
1015  3220   DCA ENTADR
1016  1376   TAD (4          / call CLW to compare file names
1017  4324   JMS CLW
1020  0000  ENTADR, 0 /ADR OF VTOC ENTRY
1021  0020   0020 /FNAME IS HERE
1022  6211   CDF 10  /BOTH IN FLD 1
1023  5227   JMP TRYAG /UNEQUAL
1024  7200   CLA /FOUND
1025  1220   TAD ENTADR / get FCB pointer into AC
1026  5232   JMP SCRTNO
1027  2122   TRYAG, ISZ NENT/ have we searched all of VTOC?
1030  5234   JMP .+4 /NO
1031  2200   ISZ SRCHVTOC /FILE NOT FOUND
1032  7402  SCRTNO, HLT /REP BY CDF CIF to return field
1033  5600   EXIT SRCHVTOC / return to caller
1034  7200   CLA
1035  1220   TAD ENTADR      / search next VTOC entry
1036  5214   JMP VTOCLK
                    /
                    /GABUV - GET ADR, BLK, AND UNIT OF A VTOC
                    /SUITABLE FOR TAPE WRITING.  GNC
                    / Called w/ tape unit (0 or 1) in loc 10000
1037  0000  GABUV, 0
1040  7200   CLA
1041  1100   SAVRTN GVRTN    / save return field and
1042  6214                  / set up data field
1043  3265
1044  6211   CDF 10
1045  7201   CLA IAC         / set loc 10007 to the tape unit #
1046  0000   AND 0
1047  3007   DCA 7
1050  1007   TAD 7
1051  1375   TAD (ATBL       / add unit # to base of core address table
1052  3101   DCA AVTOC
1053  1501   TAD I AVTOC     / get address of VTOC in core into AVTOC
1054  3101   DCA AVTOC
1055  1007   TAD 7
1056  1374   TAD (BTBL       / add unit # to base of tape block address table
```

PAGE 10

```
1057  3104   DCA BVTOC
1060  1504   TAD I BVTOC   / get tape address of VTOC (block #) into BVTOC
1061  3104   DCA BVTOC
1062  1007   TAD 7
1063  1377   TAD (10 /MEMFLD 1   } This code vestigial from an earlier version
1064  3105   DCA UVTOC           } + should be removed.
1065  7402  GVRTN, HLT /REP BY CDF
1066  5637   EXIT GABUV
             /
             /ROUTINE TO SCAN FOR THE NEXT OPERAND IN
             /AN INPUT STRING.  ASSUMES OPERAND PTR
              /IS IN LOC  0105 in caller's memory field.
             /
1067  0000  OPSCAN, 0
1070  7240   CLA CMA
1071  1773   TAD I (105  / set up AX 10 with
1072  3010   DCA 10 /ADR OF STRING
1073  1100   SAVRTN ONRTNO/
1074  6214                 / save return field
1075  3322
1076  6211   CDF 10       / note - input string is in field 1
1077  1410  BLOOK, TAD I 10 / get next char in string + look for a blank
1100  3014   DCA 14 /save it
1101  1014   TAD 14 / is char all zeros?
1102  7450   SNA
1103  5321   JMP ONRET1 / yes - end of string - return to call+2
1104  1372   TAD (-240  / is char a blank?
1105  7640   SZA CLA
1106  5277   JMP BLOOK / no - go look again.
1107  1410  BFND, TAD I 10 / get next char
1110  7450   SNA
1111  5321   JMP ONRET1     / it was zero
1112  1372   TAD (-240      / is it a non-blank?
1113  7650   SNA CLA
1114  5307   JMP BFND       / no
1115  1010   TAD 10
1116  6201   CDF 00         / yes - set loc 105 in fld 0 to address of operand string
1117  3773   DCA I (105     /
1120  7410   SKP
1121  2267  ONRET1, ISZ OPSCAN /return to call+2
1122  7402  ONRTNO, HLT /REP BY CIDF
1123  5667   EXIT OPSCAN
             /
             /CLW - COMPARE LOGICAL WORDS, IN WHICH WE
              /AGAIN PROVE THAT ONE NEED NOT INCLUDE AN
              /INSTRUCTION IF ONE CAN PROVIDE A SUB-
              /ROUTINE TO DO THE SAME THING.  GNC
             /
1124  0000  CLW, 0 / called w/ # of words to compare in the AC
1125  7041   CIA
1126  3012   DCA LENGTH/ LENGTH ← negative of # of words to compare
1127  7240   CLA CMA
1130  1724   TAD I CLW
1131  3010   DCA 10 /ADR OF LIST 1 to AX Reg 10.
1132  2324   ISZ CLW
```

PAGE 11

```
1133  7240    CLA CMA
1134  1724    TAD I CLW
1135  3011    DCA 11 /ADR OF LIST 2 to AX 11
1136  2324    ISZ CLW
1137  1724    TAD I CLW
1140  3345    DCA CLWCDF /MEMFLD OF LISTS - usually field 1
1141  2324    ISZ CLW    / for return loca
1142  1100    SAVRTN CLWRTN
1143  6214                / save caller's data field for returning
1144  3356
1145  7402  CLWCDF, HLT /REP BY CDF to field where the 2 word lists are
1146  1410    TAD I 10
1147  7041    CIA                / form →10 - →11  (→ ⇒ "pointed to by")
1150  1411    TAD I 11
1151  7640    SZA CLA
1152  5356    JMP CLWRTN /if non-zero, return immediately to call+1 - UNEQUAL
1153  2012    ISZ LENGTH /if zero, are we done
1154  5346    JMP CLWCDF+1 / no
1155  2324    ISZ CLW    / yes - EQUAL - return to call+2
1156  7402  CLWRTN, HLT /REP BY CIDF
1157  5724    EXIT CLW
              /
              LENGTH=12
1172  7540    PAGE
1173  0105
1174  0112
1175  0107
1176  0004
1177  0010
```

PAGE 12

```
                    /GETTAP - ROUTINE TO ACQUIRE TAPE SPACE
                    /FOR A FILE.  CALL WITH UNIT IN LOC 10000 AND # OF
                    /BLOCKS ON 10001.  GNC
                    /
1200  0000          GETTAP, 0
1201  4520           CALL AGABUV /get addresses of VTOC in core & on tape
1202  1101           TAD AVTOC
1203  1377           TAD (2      / Construct
1204  3102           DCA AFFB    /ADR OF "1ST FREE BLK" word in VTOC
1205  1102           TAD AFFB
1206  7001           IAC         / Construct
1207  3103           DCA ANBLKS  /ADR OF "# OF FREE BLKS" word in VTOC
1210  1001           TAD 1
1211  7041           CIA         / form # of free blocks - # of desired blocks
1212  1503           TAD I ANBLKS/   in AC.
1213  7510           SPA         / skip if enough room
1214  5225           JMP GPRET1  / No room - return call+2.
1215  3503           DCA I ANBLKS /GOOD - ENUF ROOM.
1216  1502           TAD I AFFB /SAVE FFB FOR USER
1217  3007           DCA 7      /in loc 7
1220  1007           TAD 7 /FFB#
1221  1001           TAD 1 /+# OF BLKS ALLOCATED
1222  3502           DCA I AFFB /= NEW FFB
1223  1007           TAD 7 /RET BLK# TO USER
1224  5600           EXIT GETTAP
1225  2200          GPRET1, ISZ GETTAP
1226  5600           EXIT GETTAP
                    /
                    /RELEASE TAPE- DEALLOCATE TAPE SPACE AND PACK
                    /UP THE TAPE.  UNIT IN 10000. # OF RETURNED
                    /BLKS IN 10001, BLK #/IN 10002.  GNC
                    /                    of start of returned blocks
                    *.+10&7770
1230  0000          RELTAP, 0
1231  4520           CALL AGABUV /get addresses of VTOC in core & on tape
1232  1101           TAD AVTOC
1233  1377           TAD (2      / construct address of
1234  3102           DCA AFFB/   "first free block" word in VTOC
1235  1102           TAD AFFB
1236  7001           IAC         / Construct addr of "# of free blocks" word in VTOC
1237  3103           DCA ANBLKS /SAME AS ABOVE in GETTAP
1240  1001           TAD 1 /COMPUTE PACKING INFORMATION
1241  1002           TAD 2 /FOR PACK ROUTINE
1242  7041           CIA
1243  1502           TAD I AFFB
1244  3116           DCA PKBLKS /# OF BLKS TO BE MOVED
1245  1002           TAD 2
1246  3115           DCA PKDEST /TO HERE on tape   } These are tape block #'s
1247  1001           TAD 1                         }
1250  1002           TAD 2                         }
1251  3117           DCA PKSOUR /FROM HERE on tape }
1252  1001           TAD 1 /UPDATE VTOC entries
1253  7041           CIA
1254  1502           TAD I AFFB / update the "first free block" word
1255  3502           DCA I AFFB/
```

PAGE 13

```
1256  1001    TAD I
1257  1503    TAD I ANBLKS  / update "# of free blocks" word
1260  3503    DCA I ANBLKS  /
1261  1116    TAD PKBLKS
1262  7650    SNA CLA       / is # of blocks to move = zero?
1263  5630    EXIT RELTAP /NO NEED TO PACK
1264  6202    CIF 00
1265  4776    JMS PACK /OTHERWISE PACK UP TAPE
1266  5630    EXIT RELTAP
              /
              /WVTOC - WRITE OUT A VTOC AFTER MODIFICATION.
               / called w/ tape unit # in 10000
              *.+10&7770
1270  0000    WVTOC, 0
1271  7200     CLA
1272  1100     SAVRTN WVRTN  / save return field & set
1273  6214                   / data field = 1
1274  3313                   /
1275  6211     CDF 10
1276  4520     CALL AGABUV / get core & tape addrs of VTOC
1277  1101     TAD AVTOC
1300  3312     DCA VCORAD /ADR OF VTOC IN CORE
1301  1105     TAD UVTOC /GET UNIT WE'RE SUPPOSED TO WRITE ON
1302  7110      CLL RAR /FOR TAPE CALLING SEQ WRD
1303  7210     CLA RAR /UNIT IN BIT 0
1304  1104     TAD BVTOC / add in VTOC block#
1305  3310     DCA TPUBLK
1306  6202     CIF 00 /CALL TAPE ROUTINE IN FLD 0
1307  4446     WRSTAP
1310  0000    TPUBLK, 0 /VTOC UNIT & BLK#
1311  6010     FLD1+CHECK+10
1312  0000    VCORAD, 0
1313  7402    WVRTN, HLT /REP BY CIFD
1314  5670     EXIT WVTOC
1376  2115    PAGE
1377  0002
```

```
                ./PAGES 12000 AND 14000 - CPS
                 /INITIAL VTOC ENTRIES
                 /
                 *2000
2000  0000      0
2001  7775      -3 /3 INITIAL ENTRIES
2002  0500      500 /FIRST FREE BLK
2003  1300      1300 /# OF BLKS FREE
2004  0000      0
2005  0000      0
2006  0000      0
2007  0000      0
2010  5523      TEXT /-S
2011  4040
2012  4040
2013  4040
2014  0000      /
                  *.-1
2014  0100      100
2015  7010      7010
2016  4000      4000
2017  0000      0
2020  5226      TEXT /*V
2021  2417      TO
2022  0360      CO
2023  4040
2024  0000      /
                  *.-1
2024  0400      400 /LOCN
2025  5010      5010 /LENGTH
2026  0100      100
2027  0000      0
2030  5223      TEXT /*S
2031  2624      VT
2032  1703      OC
2033  6040      0
2034  0000      /
                  *.-1
2034  0470      470
2035  5010      5010
2036  0100      100
2037  0000      0
                  *4000
4000  0000      0
4001  7767      -11 /9 INITIAL ENTRIES
4002  0612      612 /FIRST FREE BLK
4003  1166      1166 /BLKSLEFT
4004  0000      0
4005  0000      0
4006  0000      0
4007  0000      0
4010  5502      TEXT /-B
4011  4040
4012  4040
4013  4040
```

```
4014  0000  /
                *.-1
4014  4100    4100
4015  7004    7004
4016  0400    400
4017  0000    0
4020  5223    TEXT /*S
4021  2624  VT
4022  1703  OC
4023  6140  1
4024  0000  /
                *.-1
4024  4160    4160
4025  5010    5010
4026  0100    100
4027  0000    0
4030  5226    TEXT /*V
4031  2417  TO
4032  0361  C1
4033  4040
4034  0000  /
                *.-1
4034  4170    4170
4035  5010    5010
4036  0100    100
4037  0000    0
4040  5201    TEXT /*A
4041  1417  LO
4042  0104  AD
4043  0522  ER
4044  0000  /
                *.-1
4044  4240    4240
4045  5020    5020
4046  0100    100
4047  0000    0
4050  5222    TEXT /*R
4051  1417  LO
4052  0104  AD
4053  0522  ER
4054  0000  /
                *.-1
4054  4260    4260
4055  5040    5040
4056  0100    100
4057  0000    0
4060  5205    TEXT /*E
4061  0440  D
4062  4040
4063  4040
4064  0000  /
                *.-1
4064  4320    4320
4065  4014    4014
4066  0400    400
4067  0000    0
```

```
4070   5201    TEXT /*A
4071   2315   SM
4072   4040
4073   4040
4074   0000   /
                *.-1
4074   4334    4334
4075   4060    4060
4076   0400    0400
4077   0000    0
4100   5206    TEXT /*F
4101   1722   OR
4102   2422   TR
4103   0116   AN
4104   0000   /
                *.-1
4104   4414    4414
4105   4077    4077
4106   0400    0400
4107   0000    0
4110   5223    TEXT /*S
4111   0102   AB
4112   2240   R
4113   4040
4114   0000   /
                *.-1
4114   4513    4513
4115   4077    4077
4116   0400    0400
4117   0000    0
```

CHAPTER 6

IOCS3 - THE I/O CONTROL SYSTEM FOR CPS

CONTENTS

Preface

This chapter is intended for use as a reference for IOCS3 as well as an introduction to its use as a standard I/O package on the Cooley Electronics Laboratory LINC-8. IOCS3 is the most complex in a line of standardized I/O controllers written for use on the CEL LINC-8, but is easy to use from the user's viewpoint. The program specifications given herein are subject to change at any time as ways are found to improve the package.

IOCS3 is the third version of a comprehensive I/O system for the LINC-8 to be written at CEL, and is the first to do all operations under interrupts. The teleprinter routines are largely lifted in toto from an earlier controller written by Kurt Metzger (who also commented on the system design). The magnetic tape routines were done largely by the author.

IOCS3 does all I/O operations associated with the CPS Control Program. Originally written expressly for this purpose, it has found use in other systems and user programs.

6.1 Introduction

IOCS3 is an input/output control system program written for the Cooley Electronics Laboratory LINC-8. It provides fully buffered teleprinter and reader/keyboard I/O and automatic magnetic tape operation in a convenient, easy to use package. Using IOCS3, a programmer is freed from the burdens of providing his own I/O routines with their attendant bookeeping and addressing problems, and is free to concentrate on the algorithmic aspects of his program. In addition, he gains the benefits of I/O processing under interrupts.

As an example of the ease of use of IOCS3, the program in Fig. 6.1 will read 128 characters from the keyboard and echo them; when the 128th character is read, all 128 characters will be written onto tape drive 1, block 400; the program will halt after all I/O operations are finished.

6.2 Concepts and Facilities

IOCS3 is a general purpose I/O system for use with PDP-8 programs run on the LINC-8 computer. It handles the teleprinter, reader/keyboard, and LINC tapes, which comprise the set of I/O devices necessary for most programs. All data transfers are done concurrently with the execution of the user's program utilizing the program interrupt facility on the PDP-8 processor in the LINC-8. Facilities are provided the user to inquire about the status of various data transfers occurring within the system.

```
        *2000
        CALL RESET      / Initialize I/O system
        RDROPN          / Initialize reader/keyboard
        TAD (5777       / Set up buffer address
        DCA 10
        TAD (-200       / Set up loop count
        DCA 7
LOOP,   RDRGET          / Get character
        DCA 6           / Save it
        TAD 6
        PTRPUT          / Echo character to printer
        TAD 6
        DCA I 10        / Store char into buffer
        ISZ 7           / Done yet?
        JMP LOOP        / No
        WRSTAP          / Yes - write tape
        UNIT1+400       / Unit and blk number
        FLD0+1          / Mem bank and # of blks
        6000            / Memory location
        PTRCLO          / Wait for printer to finish
        TPWAIT          / Wait for tape to finish
        HLT             / All done
```

FIG. 6.1. EXAMPLE PROGRAM

Both the teleprinter and reader/keyboard are buffered to allow significant overlap of computation and reading/printing. Up to 32 characters may be queued for printing; up to 32 characters may be read in from the keyboard/reader and held for the user's program until it calls for them.

The system is capable of reading or writing up to 1777 (octal) blocks of tape in one operation. Transfers may be into or out of any memory bank and to or from either tape unit. In addition the user may specify that write checking or reading without transmitting is to be done. Provision has been made to operate on tapes marked at either 128 or 256 words per block. 256 word per block tapes are PROGOFOP compatible.

A transfer vector of system entry point addresses is provided on page zero as a convenience to the user, and a set of IOCS3-related mnemonics and pseudo-ops has been defined in an I/O symbol table which the programmer may use in assembly code programs to facilitate the use of the system.

6.3 Requirements

IOCS3 resides in the following memory locations: 0 through 2, 20 through 77, and 200 through 1377 octal. Memory locations 3 through 17, 100 through 177, and 1400 octal upward are available for the user's programs.

Equipment required is a LINC-8 computer with appropriate

reader selection modifications which allow the reader flag to be cleared without advancing the reader. (See DECUSCOPE, vol. 6, #5, 1967, p. 24 and vol. 6, #6, 1967, p. 10.) The CEL 8K LINC-8 has been so modified, and memos from Wayne Von Wald and Kurt Metzger documenting this change are available separately.

6.4 Use of IOCS3

6.4.1 Conventions. There is no provision for memory protection on the PDP-8, so the user must be careful not to store into the memory locations occupied by IOCS3. In addition, IOCS3 makes the assumption that it has full control over all I/O operations and that the user will not do any I/O himself. Users may turn off interrupts for short periods (less than 70 microseconds or so) in order to do delicate diddling. Any interrupt coming from a source other than the 33ASR or the LINC tapes is (hopefully) ignored. The user is responsible for turning off the interrupt system after he is finished using IOCS3.

In order to invoke IOCS3 routines, the user may issue a JMS Indirect through page zero. However, remembering which locations are which is a difficult job at best, so a set of definitions for the assembler, called the I/O SYMBOL TABLE, has been created which allows each specification of the desired I/O operation. For example, a call to RDRGET could be coded as "JMS I 50"; using the I/O SYMBOL TABLE allows the programmer to substitute the string

"RDRGET" which is defined inside the symbol table to be a JMS I 50. The calling conventions for the routines are given using the symbols from the I/O SYMBOL TABLE.

6.4.2 System Routines.

6.4.2.1 Protocol. The system makes extensive use of switches and flags to interlock and keep track of the state of I/O transfer operations. Before using the system to transfer data, these must be set to certain initial values by invoking the system routine RESET. Additionally, since both the reader/keyboard and teleprinter are buffered, it is necessary to reset certain switches and zero out character buffers before doing any I/O on the 33ASR. This operation is referred to as the "opening" of the device in question. The teleprinter is opened when RESET is called.

In particular, the RESET routine does the following:

1. Clears all device flags.

2. Cleans all characters from the printer buffer and opens the printer.

3. Sets the maximum number of times a tape operation may be attempted to the system default value of 4.

4. Resets the tape routine "IGNORE ERRORS" flag to OFF.

5. Closes the reader.

6. Stops any tape operation in progress and reinitializes the tape routines.

7. Clears LINC interrupts and then selects (enables interrupts from) the LINC processor.

8. Sets the tape reading and writing mode to 128 words per block.

9. Turns on the interrupt system.

A routine called MANUAL is provided to allow calling of RESET from the PDP-8 console switches. This capability is useful for debugging sessions.

6.4.2.2 Calling Sequences.

TO RESET THE I/O SYSTEM

CALL RESET

return call+1 w/ clear AC and LINK, interrupt system enables.

The routine MANUAL may be invoked by starting the PDP-8 at location 00200 (octal) from the console switches. After the system has been reset, the CPU will stop at location 00201.

6.4.3 33ASR Routines.

6.4.3.1 Protocol. With the thought that a particular application might require shutting off the reader for a while and then restarting it without losing any characters, the reader open routine has been split off as a separate entry point to the system. Thus if any reader/keyboard input is to be done, the reader open routine, RDROPN, must be called after the system reset routine has been called.

Fetching of input characters from the 33ASR may be done by either of two routines. RDRTAG, reader test and get, is used when an immediate return from the reader routine is desired whether a character from the 33ASR is present or not. RDRGET, reader get, is used in case the user wishes to have the reader routines wait until a character is available from the 33ASR before returning. With both routines, the character fetched from the input buffer will be in the PDP-8 accumulator upon return. IOCS3 will shut down the paper tape reader when its internal buffer of 32 characters is full. If characters are input from the keyboard when the input buffer is full, they will be ignored. At any time, the user may shut down the reader and hold in abeyance any typed character by calling the routine to close the reader, RDRCLO. Note that after the reader has been closed by calling RDRCLO, it can only be used again by reopening it - calling RDROPN.

Output to the teleprinter is done by loading the character to be typed into the PDP-8 accumulator and calling upon the printer put routine, PTRPUT. This routine will not return to the caller until that character has been inserted into the output buffer. At any time the user may wait for all characters in the buffer to be typed by calling PTRCLO, printer close. Note that the name PTRCLO is somewhat misleading since it is unnecessary to reopen the printer after calling PTRCLO. The name was chosen to retain both compatability with the rest of the computing world and symmetry with the reader routines.

6.4.3.2 Calling Sequences.

TO OPEN THE READER

RDROPN / resets reader switches and flags, starts
/ reading.

return call+1 with clear AC; LINK unchanged across the call.

TO GET A CHARACTER FROM THE 33ASR (Two Routines)

RDRGET / get a char and don't return until you do.

return call+1 with character in AC4-11, AC0-3 clear, and LINK bit unchanged across the call.

or

RDRTAG / check to see if a char from 33ASR is available.

return call+1 with clear AC, LINK unchanged, if no char available.

return call+2 with char in AC4-11, AC0-3 clear, LINK unchanged if char was available.

TO CLOSE THE READER

RDRCLO / shutdown reader, accept no more chars for
/ input buffer.

return call+1 with clear AC, LINK bit unchanged across the call.

TO WAIT WHILE TELEPRINTER BUFFER IS CLEARED

PTRCLO /wait until all chars in printer buffer have been
/typed.

return call+1 with clear AC, LINK bit unchanged across the call.

TO TYPE A CHARACTER ON THE TELEPRINTER

PTRPUT /call with character to be typed in AC4-11.
/AC0-3 ignored.

return call+1 with clear AC, LINK bit unchanged across the call.

6.4.4 Magnetic Tape Routines.

6.4.4.1 Protocol. Transfer of data to and from LINC magnetic tapes may be accomplished by calling the read and write tape select routines, RDSTAP and WRSTAP respectively. The user must provide several parameters to the tape routines such as tape drive desired, block number desired, and the like. (See Calling Sequences, section 6.4.4.2.) The tape routine will make copies of the calling parameters, set the "LINC TAPE ON" flag cell to -1 indicating that a tape operation is in progress, and start the proper tape drive. The routine then returns to the caller. The tape data transfer proceeds in interrupt time, at times stealing up to 45 out of every 100 memory cycles from the task time process. When the tape operation is

completed, the "LINC TAPE ON" flag cell is reset to zero. A call to the tape routines before a priori operation is completed will automatically be delayed until the prior tape operation is complete.

IOCS3 contains a routine, TPWAIT, which will wait until a pending read or write tape operation is completed and then return to the caller. A tape-mode changing routine, TPMODE, is provided so that PROGOFOP compatible tapes marked at 256 words per block may be handled. A block searching routine, SEARCH, allows tape block searching operations to be overlapped with computation. A flag, SRCHSW, indicating whether a block searching operation is in progress is provided to allow the user to control the real time behavior of his program. Any tape read or write operation will override an executing block search, but a requested block search operation will wait until all pending tape read or write operations are completed before executing.

Normal system action in the event of a checksum error while reading magnetic tape is to retry the pending operation up to three times. (The number of times the operation may be tried can be changed by the user.) If the operation is successful within the maximum allowed try count, a normal return is made to the user. If the retry is unsuccessful (checksum errors are still present), the system will check to see if the user has specified that checksum errors be ignored. (The default is that checksum errors are <u>not</u> ignored.) If

the user has specified that errors be ignored, a normal return to the user is made. If errors are not to be ignored, the tape routines make a task time exit to the tape error routine. (Note that all checking for errors is done in interrupt time.) This error routine is defaulted to a single halt instruction within IOCS3, but the user may provide his own error handling routine. In the event of a tape error, the number of checksum errors as well as the location to return to after the error has been handled are available to the user's program.

To recap, the tape system follows the flowchart in Fig. 6.2 in the event of a tape error (see next page).

FLOWCHART SYMBOL DEFINITIONS (All symbols are on page zero and thus are available to the user.)

CSCNT	contains the number of checksum errors encountered in a read operation.
TRYCNT	initially contains the negative of the total number of times a tape operation may be tried.
IGNORE	set to -1 if checksum errors are to be ignored, 0 otherwise.
PREG	holds the location where the user program was interrupted when a task time tape error exit was made.
MFLDS	holds the contents of the Interrupt Buffer (memory field settings when the user program was interrupted) when a task time error exit is made.

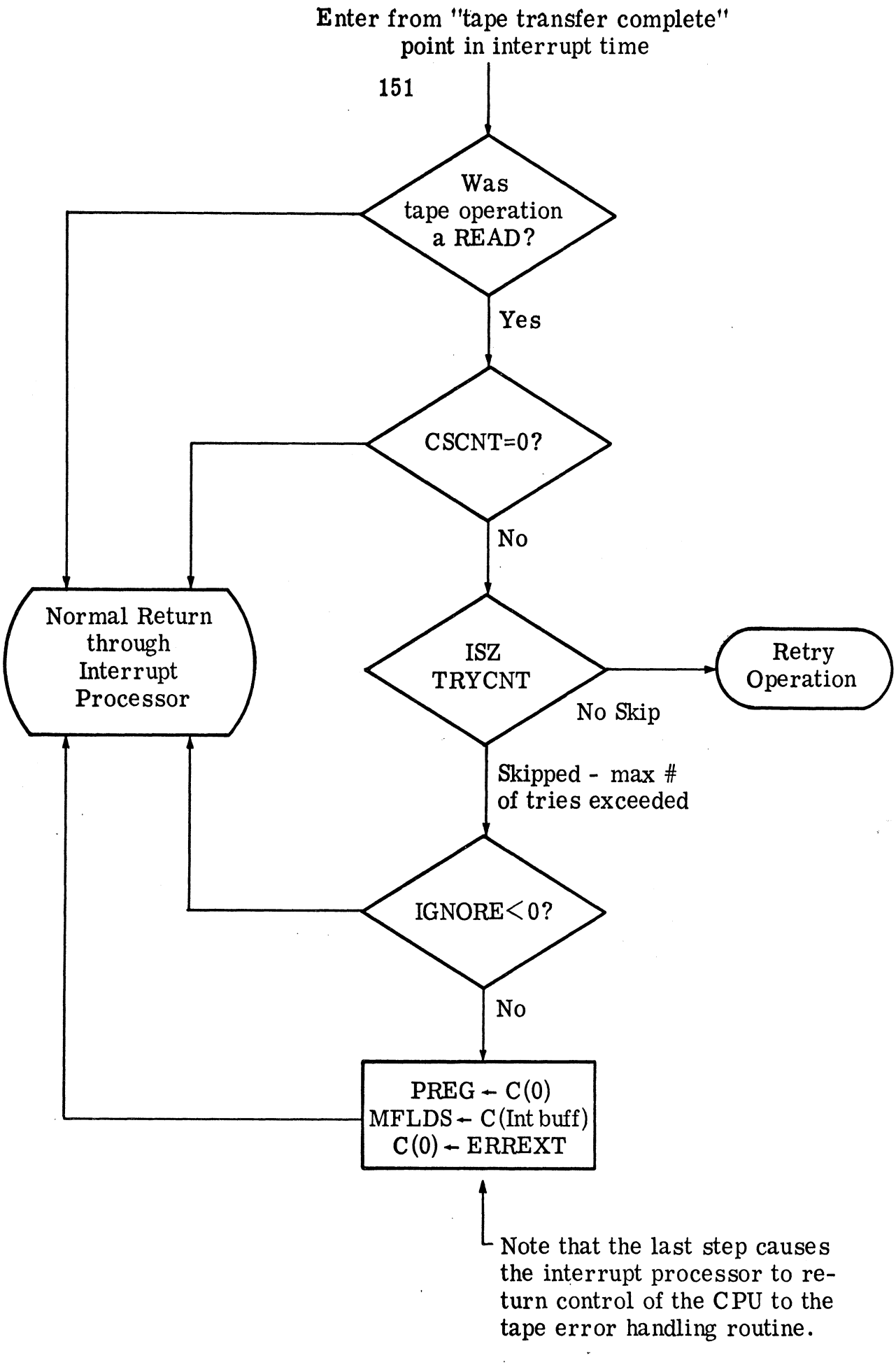


FIG. 6.2. MAGNETIC TAPE ERROR HANDLING PROCEDURE

ERREXT holds the address of the tape error handling routine to which a task time tape error exit is made.

From the flowchart and the table of symbols, user options for tape error handling should be fairly apparent. For example, if the user wishes to read tapes with bad checksums, he may deposit a -1 into IGNORE. After the tape read is complete, the number of errors may be found in CSCNT. If he wishes to change the number of tries the tape routines may make to "N", he need only deposit -N into TRYCNT. If he wishes to handle tape errors in a separate routine of his own, he may do so by depositing the address of his routine into ERREXT; in his routine, he should save the AC and the LINK bit before handling the error. After processing the error, his routine should restore the AC and LINK, restore the memory fields from MFLD, and then jump indirect through PREG to get back to the interrupted program. N.B. The user may change default tape error options only after RESET has been called.

6.4.2.2 Calling Sequences.

TO CHANGE MAGNETIC TAPE MODE

```
CALL TPMODE     /for 128 word per block tapes, call
                /with LINK bit cleared to zero.
                /for 256 word per block tapes, call
                /with LINK bit set to one.
```

return call+1 with clear AC and LINK.

TO READ OR WRITE MAGNETIC TAPE

```
RDSTAP or WRSTAP    / RDSTAP to read, or WRSTAP to write.
4000*UNIT+BLKNO     / UNIT is 0 or 1, BLKNO is tape block
                    / number.
4000*FLD+2000*CHECK+#BLKS
                    / FLD is memory bank, either 0 or 1.
                    / CHECK is 0 for no checking and 1 for
                    / checking. If reading and check is set,
                    / the read will be done without trans-
                    / mitting words to core - i.e., the
                    / operation is simply a checksum check.
                    / If writing and check is set, a read-
                    / without-transmitting of the just-writ-
                    / ten blocks will be done to insure the
                    / tape is readable.
                    / #BLKS is the number of tape blocks
                    / to be involved in the transfer, and
                    / ranges from 0 through 1777 octal.
CORLOC              / beginning core location within a bank
                    / to be involved in the transfer.
```

return call+4 with clear AC and LINK bit.

TO WAIT UNTIL A TAPE READ OR WRITE OPERATION HAS BEEN COMPLETED

```
TPWAIT          / wait until "LINC TAPE ON" flag is
                / reset to zero.
```

return call+1 with clear AC, LINK bit unchanged across the call.

TO MOVE A TAPE ONTO A DESIRED BLOCK

```
SEARCH          / hunt for the block specified below.
4000*UNIT+BLKNO / see corresponding parameter on
                / RDSTAP.
```

return call+2 with clear AC and LINK.

6.4.4.3 Useful Tape Mnemonics in the I/O Symbol Table.

Five mnemonics useful for specifying tape operations are defined in the I/O SYMBOL TABLE:

```
FLD0=0000       for specifying the memory field
FLD1=4000       on tape operations

CHECK=2000      for specifying checking on tape operations

UNIT1=4000      for specifying tape unit
UNIT0=0000      on tape operations
```

An example of the use of the 5 mnemonics is in order. Suppose we want to write locations 10200-10577 in memory onto tape drive 1 starting at block 375, and check the operation to be sure that the tape we produce can be read properly. Then the following may be used:

```
WRSTAP
UNIT1+375          / unit and blk no.
FLD1+CHECK+2       / memory field, check bit, and # of
                   / blks.
0200               / location in memory bank 1 where
                   / transfer starts.
return here
```

Note the readability of this code as opposed to the following (equivalent) code:

```
JMS I 46
4375
6002
0200
```

6.4.5 Useful Locations on Page Zero. Many useful flags as well as system entry point addresses are provided on the page zero portion of IOCS3. The relevant parts of page zero are structured as follows. Symbolic names are taken from the I/O SYMBOL TABLE.

LOCN	SYMBOLIC NAME	CONTENTS
23	ARFLAG	Address of reader buffer flag. The reader buffer flag is negative if there is at least one character present in the input buffer, and is positive if there is no character present in the buffer.
24	CIFD	A CDF CIF instruction (6203 octal). This is useful for constructing returns for off bank subroutine calls.
25	LTON	LINC TAPE ON flag. This flag has value -1 if a read or write tape operation is in progress, and is zero otherwise.
26	APFLAG	Address of printer buffer flag. The printer buffer flag is positive if there is room left in the printer buffer for at least one character, and is negative otherwise.
31	CSCNT	Number of checksum errors encountered in the last LINC magnetic tape operation.

LOCN	SYMBOLIC NAME	CONTENTS
34	IGNORE	LINC-tape checksum error ignore switch. This switch is set to zero if checksum errors are not to be ignored, and is set to -1 if checksum errors are to be ignored.
35	TRYCNT	The negative of the maximum number of times a tape operation may be tried. This count is used for tape error processing. It is set to -4 (4 trys allowed) by the system reset routine, RESET.
37	PREG	The program counter value to which the LINC-tape error handling routine should return after the error has been handled.
40	MFLDS	The memory field settings which the LINC-tape error handling routine should set up before returning through locn 37. These settings were obtained by reading the interrupt buffer via a RIB instruction.
41	ERREXT	Address of the LINC tape error exit routine, which is assumed to be in memory field zero.

LOCN	SYMBOLIC NAME	CONTENTS
60	SRCHSW	LINC tape block searching switch. This switch has value -1 if a block search is being conducted, and has value Ø otherwise.

Locations 42 through 55 contain system entry point addresses:

LOCN	CONTAINS ADDRESS OF	WHOSE FUNCTION IS
42	RESET	system reset routine
43	TPMODE	change tape mode
44	TPWAIT	wait for tapes to finish
45	RDSTAP	read tape
46	WRSTAP	write tape
47	RDROPN	open reader
50	RDRGET	get a char from reader
51	RDRTAG	get a char from reader
52	RDRCLO	close the reader
53	PTRPUT	type a character
54	PTRCLO	wait for printer to finish typing
55	SEARCH	search for a certain tape block

6.4.6 I/O Symbol Table Listing.

```
/IOCS3 SYMBOL TABLE FOR THE VERSION OF 19 MAY, 1969
/
/SYSTEM TRANSFER VECTOR DEF'NS
RESET=42
TPMODE=43
TPWAIT=4444
RDSTAP=4445
WRSTAP=4446
RDROPN=4447
RDRGET=4450
RDRTAG=4451
RDRCLO=4452
PTRPUT=4453
PTRCLO=4454
SEARCH=4455
/USEFUL CONSTANTS & FLAGS
ARFLAG=23
CIFD=24
LTON=25
APFLAG=26
CSCNT=31
IGNORE=34
TRYCNT=35
PREG=37
MFLDS=40
ERREXT=41
SRCHSW=60
/TAPE DEFINITIONS
FLD1=4000
FLD0=0000
CHECK=2000
UNIT0=0
UNIT1=4000
/
/"CALL" is defined within the CEL assembler as "JMS I"
```

6.5 Program Listing

A listing of IOCS3 can be found overleaf.

```
                /IOCS III - AN I/O CONTROL SYSTEM FOR THE
                /COOLEY ELECTRONICS LAB LINC-8.
                /THIS SYSTEM IS RUN TOTALLY UNDER
                /INTERRUPTS, AND SUPPORTS THE 33ASR
                /READER AND PRINTER, AS WELL AS LINC TAPES
                /MARKED AT 128 WORDS PER BLOCK.
                /
                /A MACRO FOR SAVING RETURN MEMORY FIELDS
                DEFINE SAVRTN SAVDUM
                <TAD CIFD
                 RDF
                 DCA SAVDUM>
                /
                /TRAP PROCESSOR JUMP
                *1
0001   5402      JMP I .+1
0002   0245      TRPROC
                /
                /PAGE ZERO CONSTANTS
                *20
0020   1251     ATAPE, TAPINT
0021   0475     ARDR, RDRIE /INTT HANDLING
0022   0357     APTR, TTYIE /ROUTINE ADDRESSES
0023   0000     GBPTR, 0 /GET BUFFER-POINTER
0024   6203     CIFD, CIF CDF /FOR OFF-BANK RETURNS
0025   0000     LTON, 0  /IS LINC-TAPE RUNNING? SW
0026   0000     PBPTR, 0 /PUT ROUTINE BUFFER-POINTER
0027   0000     SWITCH, 0 /TAPE ROUTINE SWITCH WORD
0030   0000     LADR, 0   /ADR OF APPROPRIATE ENTRY IN BLKTBL
0031   0000     BADCS, 0 /BAD CHECKSUM FLAG, 0 IF OK, NO N-ZERO IF BAD
0032   7600     BLKSIZ, -200 /TAPE BLOCK SIZE
0033   0037     BMASK, 37 /ALLOWS 32DEC LOCN BUFFER
0034   0000     IGNORE, 0 /-1 IF CKSUM ERRORS ARE TO BE IGNORED
0035   7774     RETRY, -4 /TAPE OPERATION MAX RETRY CNT
0036   0000     RETCNT, 0 /NUMBER OF RETRIES DONE
0037   0000     PREG, 0 /PROG CNTR FOR TAPE ERRORS
0040   0000     MFLDS, 0 /MEMFLDS FOR  "
0041   0747     ERREXT, SYSHLT /TAPE ERROR ROUTINE ADR
0042   0203     ARESET, RESET /USER TRANSFER VECTOR
0043   1332     ATPMOD, TPMODE
0044   0541     ATPWAT, TPWAIT
0045   1000     ARDTAP, RDSTAP
0046   1005     AWRTAP, WRSTAP
0047   0317     ARDOPN, RDROPN
0050   0345     ARDGET, GET
0051   0440     ARDTAG, GETCH
0052   0524     ARDCLO, RDRCLO
0053   0640     ATTPUT, PUT
0054   0706     ATTCLO, TTYCLO
0055   1056     ASERCH, SEARCH
0056   0000     PTRTEM, 0 /TEMP LOC FOR PTR ROUTINES
0057   0000     TTYON, 0 /IS PTR RUNNING? SW
0060   0000     SRCHSW, 0 /-1 IF SERCHING FOR TAPBLK, 0 FOR RD OR WR
                PAGE
```

PAGE 02

```
                    /ENTRY FOR MANUAL SYSTEM RESET
                    /
0200  4203          MANUAL, JMS RESET
0201  7402           HLT
0202  5201           JMP .-1
                    /
                    /SYSTEM I/O RESET ROUTINE
                    /HAS SAME EFFECT AS SYSTEM RESET
                    /BUTTON ON A /360
                    /
0203  0000          RESET, 0
0204  6002           IOF /TURN INTTS OFF
0205  6132           RCC /CLEAR ALL FLAGS
0206  6042           TCF
0207  6312           IBCF
0210  6302           ADCC
0211  6356           ECF EOF /EXTERNAL INTT OFF
0212  1024           SAVRTN RTRTN
0213  6214
0214  3242
0215  6201           CDF /just in case
0216  1377           TAD (WBUFF /OPEN PRINTER
0217  4303           JMS ZERO   /SINCE IT'S SUCH A NICE DEVICE
0220  1377           TAD (WBUFF /AND TAKES NO SCREWY HANDLING
0221  3026           DCA PBPTR  /LIKE THE READER
0222  1026           TAD PBPTR  /OPEN PTR BY ZEROING BUFFER
0223  3776           DCA TBPTR  /AND SETTING UP POINTERS
0224  3057           DCA TTYON  /PTR  NOT RUNNING
0225  3775           DCA RDRMOD /RDR IS CLOSED
0226  3774           DCA RFLAG  /NO PENDING RDR INTTS
0227  3034           DCA IGNORE /ALL TAPE ERRORS PRODUCE HLT
0230  1373           TAD (-4 /SET MAX RETRY CNT
0231  3035           DCA RETRY
0232  3025           DCA LTON   /TAPE NOT RUNNING
0233  6141           ICON /STOP TAPES
0234  1372           TAD (7
0235  6141           ICON /CLEAR LINC INTTS
0236  7001           IAC
0237  6141           ICON /SELECT LINC processor
0240  7300           CLA CLL /RETURN W/ CLEAR AC&LINK
0241  4771           JMS TPMODE /SET TAPES TO 128 WRDS /BLK. RET W/ CLR AC
0242  7402          RTRTN, HLT /REP BY CIFD
0243  6001           ION /TURN ON INTTS
0244  5603           EXIT RESET
                    /
                    /SYSTEM TRAP PROCESSOR
                    /ALL I/O INTERRUPTS COME HERE
                    /
0245  3301          TRPROC, DCA SAVEAC
0246  7010           RAR /SAVE AC AND LINK
0247  3302           DCA SAVELK
0250  6147           INTS /GET LINC INTT STATUS
0251  7710           SPA CLA /SKIP IF NOT TAPE
                            /NOTE THAT IT MIGHT BE THE CONSOLE SWITCHES
0252  5420           JMP I ATAPE /GO TO TAPE INTT TIME ROUTINE
```

PAGE 03

```
0253  6131    RSF /TEST RDR FLAG
0254  7410    SKP /NOT A RDR INTT
0255  5421    JMP I ARDR /WAS A RDR INTT - GO TO INTT TIME ROUTINE
0256  6041     TSF /TEST PTR FLAG
0257  7410    SKP /NOT A PTR INTT
0260  5422    JMP I APTR /WAS A PTR INTT - GO TO INTT TIME ROUTINE
0261  6312    IBCF /CLEAR ALL REMAINING FLAGS
0262  6356    ECF EOF /EXTERNAL INTT OFF
0263  6302    ADCC
0264  1025    TAD LTON /IS LINC TAPE RUNNING?
0265  7710    SPA CLA  /SKIP IF NOT
0266  5271    JMP TRPRET
0267  1372    TAD (7
0270  6141    ICON /CLEAR LINC INTTS
0271  6244   TRPRET, RMF /RESTORE PREVIOUS FLDS
0272  7630    SZL CLA /LINK IS 1 IF THERE IS A TASK-
0273  6203    CIF CDF 0 /TIME ERROR RETURN FROM TAPE ROUTINES
0274  1302     TAD SAVELK /RESTORE LINK BIT
0275  7104    CLL RAL
0276  1301    TAD SAVEAC /RESTORE AC
0277  6001    ION
0300  5400    EXIT 0
              /
0301  0000   SAVEAC, 0 /AC SAVE LOCN
0302  0000   SAVELK, 0 /LINK SAVE WRD
              /
              /A ROUTINE TO ZERO-OUT THE READER
              /AND PRINTER BUFFERS
              /
0303  0000   ZERO, 0    /called with address of buffer to be zeroed in AC
0304  3315    DCA ZADR /SAVE ADDRESS OF BUFFER
0305  1033    TAD BMASK
0306  7040    CMA       / compute
0307  3316    DCA ZCNT /# OF BUFFER LOCNS
0310  3715    DCA I ZADR
0311  2315    ISZ ZADR   / actual loop to zero buffer
0312  2316    ISZ ZCNT
0313  5310    JMP .-3
0314  5703    EXIT ZERO
              /
0315  0000   ZADR, 0
0316  0000   ZCNT, 0
              /
              /READER OPEN ROUTINE
              /
0317  0000   RDROPN, 0
0320  7600   CLA2, 7600 /CLR AC W/ GROUP 2 CLA TO SAVE MEMFLD (vestigial from an
0321  1024    SAVRTN RNRET                                    earlier version of
0322  6214                                                    IOCS3)
0323  3343
0324  6201    CDF
0325  3775    DCA RDRMOD /CLOSE READER TO BE SURE
0326  1370    TAD (RBUFF /SETUP RDR BUFFER
0327  4303    JMS ZERO
0330  1370    TAD (RBUFF /SET PTRS
```

PAGE 04

```
0331  3767    DCA RBPTR
0332  1767    TAD RBPTR
0333  3023    DCA GBPTR
0334  6002    IOF /TURN OFF INTTS FOR DELICATE DIDDLING
0335  1774    TAD RFLAG /INTT PENDING?
0336  7640    SZA CLA
0337  4766    JMS RDRINP /YES
0340  7240    STA /SET SWITCH TO RDROPN
0341  3775    DCA RDRMOD
0342  6001    ION
0343  7402  RNRET, HLT /REP BY CIFD
0344  5717    EXIT RDROPN
              /
              /READER CHARACTER FETCHING ROUTINE
              /WILL WAIT UNTIL A CHARACTER COMES IN
              /
0345  0000  GET, 0
0346  7200    CLA
0347  1024    SAVRTN GTRTN /save return field
0350  6214
0351  3355
0352  6201    CDF
0353  4765    JMS GETCH/ try for a character
0354  5353    JMP .-1 /DIDN'T GET CHAR, TRY AGAIN
0355  7402  GTRTN, HLT /REP BY CIFD
0356  5745    EXIT GET
              /
              /TELEPRINTER INTT EXIT ROUTINE
              /
0357  6042  TTYIE, TCF /clear flag
0360  4764    JMS TYPER / do buffer management
0361  5271    JMP TRPRET/ return to trap processor
0364  0667  PAGE
0365  0440
0366  0502
0367  0540
0370  0400
0371  1332
0372  0007
0373  7774
0374  0535
0375  0536
0376  0721
0377  0600
```

PAGE 05

```
                    /IOCS III
                    /READER ROUTINES
                    /
                    RBUFF=.
                    /RDR BUFFER OF 32 CHARS GOES HERE
                    *RBUFF+40
                    /
                    /GET-A-CHARACTER ROUTINE - RETURNS IMMEDIATELY
                    /IF A CHARACTER IS NOT AVAILABLE
                    /
0440  0000  GETCH, 0
0441  7200   CLA /SAVE CALLER'S MEMORY FIELD
0442  1024   SAVRTN GETRET
0443  6214
0444  3273
0445  6201   CDF 00
0446  1423   TAD I GBPTR /DO WE HAVE A CHAR TO RETURN?
0447  7700   SMA CLA /SKP IF YES-high order bit is set
0450  5273   JMP GETRET /OTHERWISE RETURN W/ CLR AC
0451  1423   TAD I GBPTR /GET CHAR
0452  0377   AND (377 /STRIP FLAGS
0453  3337   DCA RDRTEM /SAVE CHAR
0454  3423   DCA I GBPTR /RESET BUFFER location-i.e, zero high order bit
0455  7201   CLA IAC /UPDATE GET BUFFER
0456  1023   TAD GBPTR /POINTER
0457  0033   AND BMASK /RBUFF MUST BE ON THE PROPER BOUNDARY
0460  1376    TAD (RBUFF /FOR THIS TO WORK
0461  3023   DCA GBPTR
0462  6002   IOF /TURN OFF INTTS - DELICATE DIDDLING GOING ON
0463  1336   TAD RDRMOD
0464  7640   SZA CLA
0465  1335   TAD RFLAG
0466  7640   SZA CLA
0467  4302   JMS RDRINP
0470  6001   ION /TURN INTTS BACK ON
0471  1337   TAD RDRTEM /GET CHAR
0472  2240   ISZ GETCH /BUMP RETURN ADR
0473  7402  GETRET, HLT /REP BY CIFD
0474  5640   EXIT GETCH
                    /
                    /READER INTT EXIT ROUTINE
                    /
0475  1336  RDRIE, TAD RDRMOD /IS READER OPEN?
0476  7700   SMA CLA /SKP IF YES
0477  5320   JMP RCLSD
0500  4302   JMS RDRINP /GET THE CHAR
0501  5775   JMP TRPRET /RETURN TO TSK TIME
                    /
                    /READER INPUT ROUTINE - GETS
                    /PENDING CHAR INTO BUFFER IF
                    /POSSIBLE
                    /
0502  0000  RDRINP, 0
0503  1740   TAD I RBPTR /IS THERE ROOM IN BUFFER?
0504  7710   SPA CLA /SKP IF YES
```

PAGE 06

```
0505  5320    JMP RCLSD
0506  6036    KRB /GET CHAR FROM RDR BUFFER
0507  1374    TAD (4000 /set high order bit to indicate core buffer soon in use.
0510  3740    DCA I RBPTR
0511  7201    CLA IAC /UPDATE BUFFER POINTER
0512  1340    TAD RBPTR
0513  0033    AND BMASK
0514  1376    TAD (RBUFF
0515  3340    DCA RBPTR
0516  3335    DCA RFLAG /NO CHAR PENDING
0517  5702    EXIT RDRINP
              /
              /COME HERE IF RDR IS CLOSED AND WE
              /GET AN INTT, OR NO ROOM IN BUFFER
              /
0520  7240  RCLSD, STA /SET CHAR PENDING FLAG
0521  3335    DCA RFLAG
0522  6132    RCC /CLR FLAG, DON'T ADVANCE READER
0523  5775    JMP TRPRET
              /
              /READER CLOSE ROUTINE
              /
0524  0000  RDRCLO, 0
0525  7200    CLA /SAVE CALLER'S MEMFLD
0526  1024    SAVRTN RORET
0527  6214
0530  3333
0531  6201    CDF 00
0532  3336    DCA RDRMOD
0533  7402  RORET, HLT /REP BY CIFD
0534  5724    EXIT RDRCLO
              /
0535  0000  RFLAG, 0 /-1 IF RDR INTT PENDING, 0 IF NOT
0536  0000  RDRMOD, 0 /-1 IF RDR IS OPEN, 0 IF CLOSED
0537  0000  RDRTEM, 0 /HOLDS CHAR FOR GETCH
0540  0000  RBPTR, 0 /RDR INTT TIME BUFFER PTR
              /
              /MAGNETIC TAPE WAITING ROUTINE
              /
0541  0000  TPWAIT, 0
0542  7200    CLA
0543  1024    SAVRTN TPWRTN
0544  6214
0545  3351
0546  1025    TAD LTON
0547  7710    SPA CLA
0550  5346    JMP .-2
0551  7402  TPWRTN, HLT /REP BY CIFD
0552  5741    EXIT TPWAIT
              /
              /COMMON "AFTER-BLOCK" ROUTINE FOR BOTH
              /MAG TAPE READ AND WRITE
              /
0553  1773  CLEANUP, TAD SSUWRD
0554  6141    ICON /SET SEARCH ON TAPE UNIT
```

PAGE 07

```
0555  7300    CLA CLL /LINK TO ZERO FOR NORMAL RETURN
0556  2772    ISZ DESIRED /UPDATE DESIRED BLKNO
0557  2364    ISZ BLKSLEFT /ANY MORE BLKS LEFT
0560  7410    SKP /YES
0561  5771    JMP POSTLOG /NO MORE - STOP
0562  4770    JMS WAIT /WAIT FOR BLK INTT
0563  5767    JMP SERCH /AND THEN GO TO SERCH ROUTINE
0564  0000  BLKSLEFT, 0 /CONTAINS NEGATIVE OF # OF BLKS
0567  1200  PAGE
0570  1246
0571  0722
0572  1361
0573  1152
0574  4000
0575  0271
0576  0400
0577  0377
```

PAGE 08

```
                    /IOCS III
                    /TELEPRINTER ROUTINES - NOTE THAT A COPY
                    /OF THESE ROUTINES COULD BE USED FOR HANDLING
                    /A HIGH SPEED PRINTER LIKE THE INKTRONIC.
                    /
                    WBUFF=.
                    /PTR BUFFER OF 32 CHARS GOES HERE
                    *WBUFF+40
                    /
                    /PUT CHARACTER ROUTINE
                    /NOTE THAT A RETURN TO THE CALLER
                    /IS NOT MADE UNTIL THE CHARACTER
                    /HAS BEEN INSERTED INTO THE BUFFER
                    /
0640  0000  PUT, 0
0641  0377   AND (377 /IN CASE HE GAVE US GARBAGE
0642  3056   DCA PTRTEM /SAV CHAR
0643  1024   SAVRTN PUTRET
0644  6214
0645  3265
0646  6201   CDF
0647  1426   TAD I PBPTR /SNIFF AT BUFFER - ROOM?
0650  7710   SPA CLA /SKP IF THERE IS ROOM
0651  5247   JMP .-2 /OTHERWISE WAIT
0652  1376   TAD (4000
0653  1056   TAD PTRTEM /CONSTRUCT CHAR + SWITCH
0654  3426   DCA I PBPTR /AND INSERT INTO BUFFER
0655  7201   CLA IAC /UPDATE PUT BUFFER POINTER
0656  1026   TAD PBPTR /USING MASKS
0657  0033   AND BMASK
0660  1375   TAD (WBUFF
0661  3026   DCA PBPTR
0662  1057   TAD TTYON
0663  7700   SMA CLA
0664  4267   JMS TYPER
0665  7402  PUTRET, HLT /REP BY CIFD
0666  5640   EXIT PUT
                    /
                    /PRINTER TYPING ROUTINE
                    /
0667  0000  TYPER, 0
0670  1721   TAD I TBPTR /IS THERE A CHAR TO TYPE?
0671  7500   SMA /SKP IF YES
0672  5304   JMP TR1
0673  6046   TLS
0674  7300   CLA CLL
0675  3721   DCA I TBPTR /RESET BUFFER
0676  7201   CLA IAC /UPDATE TELEPRINTER BUFFER
0677  1321   TAD TBPTR /POINTER USING MASKS
0700  0033   AND BMASK
0701  1375   TAD (WBUFF
0702  3321   DCA TBPTR
0703  7240   STA /SET TTYON SW
0704  3057  TR1, DCA TTYON /AC TO TTYON
0705  5667   EXIT TYPER
```

PAGE 09

```
                /.
                /TELEPRINTER CLOSE ROUTINE
                /
0706  0000    TTYCLO, 0
0707  7200     CLA /SAVE CALLER'S MEMFLD
0710  1024     SAVRTN TORET
0711  6214
0712  3317
0713  6201     CDF 00
0714  1057     TAD TTYON
0715  7640     SZA CLA
0716  5314     JMP .-2
0717  7402    TORET, HLT /REP BY CIFD
0720  5706     EXIT TTYCLO
                /
0721  0000    TBPTR, 0 /PTR INIT TIME BUFFER POINTER
                /
                /POSTLOG ROUTINE - CLEANUP AFTER A
                /MAG TAPE OPERATION
                /
0722  1374    POSTLOG, TAD (4
0723  6141     ICON /RESET TAPE SEARCH MODE
0724  7200     CLA
0725  6141     ICON /STOP TAPE MOTION
0726  2430     ISZ I LADR /UPDATE LAST BLKNO
0727  7000     NOP /COULD SKP
0730  1773     TAD TAPEOP /WHAT WERE WE DOING?
0731  7700     SMA CLA /SKP IF A WRITE
0732  5341     JMP CHKCS /WAS A RD - CHECK CHKSUM
0733  1027     TAD SWITCH /MUST WE CHECK?
0734  7700     SMA CLA /SKP IF YES
0735  5344     JMP RETURN /OTHERWISE GO BACK
0736  3773     DCA TAPEOP /SET OPCODE TO READ
0737  4772    GO, JMS STARTUP /START TAPE DRIVE
0740  5771     JMP WAIT+1 /EXIT BACK TO TASK TIME
0741  1031    CHKCS, TAD BADCS /DID WE HAVE ANY CS ERRORS?
0742  7640     SZA CLA /SKP IF NOT
0743  5350     JMP SYSHLT+1 /OTHERWISE CHK FOR RETRY
0744  3025    RETURN, DCA LTON
0745  3060     DCA SRCHSW /RESET FLAGS FOR USER
0746  4770     JMS WAIT /RETURN TO TASK TIME
0747  7402    SYSHLT, HLT /COMPLETE HLT FOR TAPE ERRORS, ETC.
0750  2036     ISZ RETCNT /MAY WE RETRY THE OPERATION?
0751  5365     JMP AGAIN /YES
0752  1034     TAD IGNORE /NO - CAN WE IGNORE BAD CS
0753  7710     SPA CLA /SKP IF NOT
0754  5344     JMP RETURN /RETURN TO TASK TIME
0755  1000     TAD 0 /DIDDLE TASK TIME RETRN ADR
0756  3037     DCA PREG /SAVE OLD ADR
0757  1041     TAD ERREXT /AND INSERT TAPE
0760  3000     DCA 0 /ERROR ROUTINE ADR
0761  6234     RIB /SAVE MEMFLDS
0762  3040     DCA MFLDS
0763  7320     STL CLA /INDICATE ERROR RETURN TO TRAPPER
0764  5344     JMP RETURN
```

PAGE 10

```
0765  4767  AGAIN, JMS SETUP
0766  5337   JMP GO /TRY OPERATION AGAIN
0767  1113  PAGE
0770  1246
0771  1247
0772  1125
0773  1365
0774  0004
0775  0600
0776  4000
0777  0377
```

PAGE 11

```
                /IOCS III
                /MAGNETIC TAPE ROUTINES
                /
1000  0000      RDSTAP, 0
1001  7200       CLA
1002  1200       TAD RDSTAP
1003  3205       DCA WRSTAP /DIDDLE RETURN ADDRESS
1004  5210       JMP .+4
1005  0000      WRSTAP, 0
1006  7200       CLA
1007  1377       TAD (4002 /WRITE FUNCTION
1010  3200       DCA RDSTAP /HOLD FUNCTION HERE TEMPORARILY
1011  1025       TAD LTON /SEE IF THE OPERATION CAN PROCEED
1012  7710       SPA CLA  /SKP IF IT CAN
1013  5211       JMP .-2  /OTHERWISE WAIT
1014  1200       TAD RDSTAP
1015  3353       DCA FUNCT
1016  4301       JMS GETUBK /GET UNIT # AND BLK #
1017  1605       TAD I WRSTAP /GET SWITCHES AND # OF BLKS
1020  3027       DCA SWITCH
1021  2205       ISZ WRSTAP
1022  1605       TAD I WRSTAP /GET MEMORY LOCN
1023  3355       DCA LOC
1024  2205       ISZ WRSTAP /POINTS TO RETURN LOCN
1025  1024       SAVRTN GOBAK
1026  6214
1027  3277
1030  6201      TPCDF, CDF
1031  1027       TAD SWITCH
1032  0376       AND (1777
1033  7450       SNA
1034  5277       JMP GOBAK /DON'T BOTHER W/ 0 BLKS
1035  3354       DCA BLKS
1036  1027       TAD SWITCH
1037  7710       SPA CLA
1040  1375       TAD (10
1041  1230       TAD TPCDF
1042  3774       DCA WCDF
1043  1774       TAD WCDF
1044  3773       DCA RCDF /SET UP CDF'S FOR DATA TRANSFER
1045  3060       DCA SRCHSW /RESET SEARCHING SW
1046  7240       STA
1047  3025       DCA LTON /SET MAGTAPE ON SW
1050  1027       TAD SWITCH /GET CHECKING BIT FOR
1051  7004       RAL /EASIER TESTS
1052  3027       DCA SWITCH
1053  1035       TAD RETRY /SET UP RETRY COUNT
1054  3036       DCA RETCNT
1055  5275       JMP TMERGE
1056  0000      SEARCH, 0
1057  7200       CLA
1060  1025       TAD LTON /MAY OPERATION PROCEED?
1061  7710       SPA CLA /SKP IF YES
1062  5260       JMP .-2 /OTHERWISE WAIT
1063  1024       SAVRTN GOBAK
```

```
PAGE 12

1064  6214
1065  3277
1066  6201    CDF
1067  1256    TAD SEARCH /DIDDLE RETURN ADR
1070  3205    DCA WRSTAP
1071  4301    JMS GETUBK /GET UNIT# AND BLK#
1072  7240    STA /SET SEARCHING SW
1073  3060    DCA SRCHSW
1074  3353    DCA FUNCT
1075  4313  TMERGE, JMS SETUP /SET UP PARAMETERS FOR TRAP TIME COROUTINE
1 076  4325    JMS STARTUP /STARTUP THE PROPER TAPE DRIVE
1077  7402  GOBAK, HLT /REP BY CIFD
1100  5605    EXIT WRSTAP
              /
              /A ROUTINE TO GET THE TAPE UNIT #
              /AND BLKNO.
              /
1101  0000  GETUBK, 0
1102  1605    TAD I WRSTAP / get parameter word, which is 4000*UNIT +BLKNO
1103  0372    AND (4000
1104  1371    TAD (2 /FORM SET-SEARCH-AND-UNIT WRD
1105  3352    DCA SSUWRD
1106  1605    TAD I WRSTAP
1107  0370    AND (3777 /GET BLKNO
1110  3356    DCA BLKNO
1111  2205    ISZ WRSTAP /ADVANCE PTR
1112  5701    EXIT GETUBK
              /
              /PARAMETER SETUP ROUTINE
              /
1113  0000  SETUP, 0
1114  1352    TAD SSUWRD / use tape unit to determine
1115  7710    SPA CLA
1116  7001    IAC
1117  1367    TAD (BLKTBL / address from BLKTBL to be used to
1120  3030    DCA LADR /  point to word containing position of tape unit in question
1121  3031    DCA BADCS / reset BADCS
1122  1353    TAD FUNCT
1123  3766    DCA TAPEOP / set up "tape operation" word
1124  5713    EXIT SETUP
              /
              /TAPE UNIT STARTING ROUTINE
              /
1125  0000  STARTUP, 0
1126  1365    TAD (SERCH
1127  3764    DCA WAIT / set up WAIT routine
1130  1356    TAD BLKNO
1131  7001    IAC                / set up desired tape blk#
1132  3763    DCA DESIRED/
1133  1354    TAD BLKS
1134  7041    CIA /FOR ISZ LOOP
1135  3762    DCA BLKSLEFT
1136  1355    TAD LOC
1137  3761    DCA CORLOC /PTR TO CORE LOCATION
1140  1352    TAD SSUWRD /SELECT TAPE UNIT
```

PAGE 13

```
1141  6141    ICON
1142  7300    CLA CLL
1143  1356    TAD BLKNO /COMPUTE
1144  1430    TAD I LADR /INITIAL
1145  7210    CLA RAR   /DIRECTION
1146  7001    IAC      /OF TAPE MOTION
1147  6141    ICON
1150  7300    CLA CLL
1151  5725    EXIT STARTUP
1152  0000   SSUWRD, 0 /SET-SEARCH-AND-UNIT WRD FOR TAPE UNIT
1153  0000   FUNCT, 0  /TAPE TRANSFER FUNCTION - 0 OR 4005 FOR R OR W
1154  0000   BLKS, 0   /(POSITIVE) # OF BLKS TO TRANSFER
1155  0000   LOC, 0    /LOWEST LOCN IN CORE TO TRANSFER
1156  0000   BLKNO, 0  /STARTING BLK # ON MAG TAPE
1161  1364   PAGE
1162  0564
1163  1361
1164  1246
1165  1200
1166  1365
1167  1366
1170  3777
1171  0002
1172  4000
1173  1270
1174  1311
1175  0010
1176  1777
1177  4002
```

PAGE 14

```
                    /IOCS III
                    /MAGNETIC TAPE ROUTINES - CONTINUED
                    /
                    /BLOCK SEARCHING ROUTINE - OPERATES
                    /IN INTERRUPT TIME
                    /
1200  6171  SERCH, IAAC /GET BLKNO FROM TAPE
1201  3430   DCA I LADR /SAVE IN BLKTBL
1202  6171   IAAC /GET IT AGAIN
1203  7500   SMA  /SEARCH ALGORITHM COURTESY
1204  7120   STL  /OF KURT METZGER
1205  1361   TAD DESIRED
1206  7650   SNA CLA
1207  5225   JMP THERE /PROPER BLKNO IF ZERO
1210  6147   INTS /CHECK MOTION
1211  7010   RAR
1212  0377   AND (4000
1213  7460   SZA SNL
1214  5223   JMP SMERGE /PROPER DIRECTION
1215  7020   CML
1216  7520   SNL SMA
1217  5223   JMP SMERGE /PROPER DIRECTION
1220  6141   ICON  /CHANGE DIRECTION
1221  7001   IAC
1222  6141   ICON  /BY STOPPING AND STARTING
1223  4246  SMERGE, JMS WAIT /RETURN TO TASK TIME
1224  5200   JMP SERCH /GO THRU SERCH AGAIN
1225  1060  THERE, TAD SRCHSW /ARE WE SEARCHING FOR A BLOCK?
1226  7710   SPA CLA /SKP IF NOT
1227  5776   JMP POSTLOG /SEARCH IS DONE - QUIT
1230  6147   INTS /GET DIRECTION INFO
1231  7012   RTR /GET MOTION BITS
1232  7620   SNL CLA
1233  5223   JMP SMERGE
1234  1032   TAD BLKSIZ /GET BLK SIZE
1235  3362   DCA TAPCNT /AND PUT INTO WRDCOUNT
1236  3363   DCA DATASUM /INITIALIZE TO 0
1237  1375   TAD (3 /SET BLOCK MODE ON TAPE UNIT
1240  6141   ICON
1241  1365   TAD TAPEOP /GET OPERATION CODE
1242  7500   SMA /SKIP IF A WRITE
1243  5255   JMP IREAD /GO TO READ ROUTINE
1244  6141   ICON /SET WRITERS ON MODE
1245  5310   JMP WRITE /GO TO WRITE ROUTINE
                    /
                    /WAIT ROUTINE - ALL INTT TIME ROUTINES
                    /RETURN HERE TO GO BACK TO TASK TIME
                    /
1246  0000  WAIT, 0
1247  7300   CLA CLL /RESET LINK FOR NORMAL RETURN TO TRAPPER
1250  5774   JMP TRPRET
1251  1373  TAPINT, TAD (7 /COME HERE FOR TAPE INTT
1252  6141   ICON /CLR LINC INTTS
1253  7300   CLA CLL /RETURN W/ CLR AC & LINK
1254  5646   EXIT WAIT
```

PAGE 15

```
                /
                /INTT TIME TAPE READING ROUTINE
                /
1255   4246     IREAD, JMS WAIT /WAIT FOR GUARD MARK
1256   4246     READ, JMS WAIT /READ FOR REAL
1257   6171      IAAC /GET WRD FROM LINC AC
1260   3360      DCA TAPTEM /SAVE IT
1261   1363      TAD DATASUM
1262   1360      TAD TAPTEM /UPDATE DATASUM
1263   3363      DCA DATASUM
1264   1027      TAD SWITCH /SHOULD DATA BE TRANSFERRED TO CORE?
1265   7710      SPA CLA /SKIP IF YES
1266   5275      JMP RTEST /OTHERWISE JMP - CHECKING TAPE
1267   1360      TAD TAPTEM /GET WORD
1270   7402     RCDF, HLT /CDF TO PROPER BANK WILL GO HERE
1271   3764      DCA I CORLOC /WORD TO CORE
1272   6201      CDF /RESTORE DATA FLD
1273   2364      ISZ CORLOC /UPDATE CORE LOCN PTR
1274   7000      NOP /LOCN 7777 SHOULD WORK TOO
1275   2362     RTEST, ISZ TAPCNT /HAVE WE DONE A BLK YET?
1276   5256      JMP READ /NO, GOT TO DO AGAIN
1277   4246      JMS WAIT /YES - WAIT FOR CKSUM
1300   6171      IAAC /GET CKSUM FROM LINC AC
1301   7041     RDCS, CIA /THIS WILL BE A NOP FOR LINC TAPE
                /ABOVE INSTRUCTION IS CHANGED BY ROUTINE "TPMODE"
1302   1363      TAD DATASUM
1303   7640      SZA CLA /SKP IF ALL OK
1304   7001      IAC /OTHERWISE SET AC TO 1
1305   1031      TAD BADCS /ADD C(AC) TO BADCS
1306   3031      DCA BADCS
1307   5772      JMP CLEANUP /COMMON "AFTER-BLOCK" ROUTINE
                /
                /INTT TIME TAPE WRITING ROUTINE
                /
1310   7200     WRITE, CLA
1311   7402     WCDF, HLT /REP BY CDF FOR PROPER FLD
1312   1764      TAD I CORLOC /GET WORD FROM CORE
1313   6161      IACB /TRANSFER IT TO LINC B-REG
1314   6201      CDF /RESTORE DATA FIELD
1315   1363      TAD DATASUM /UPDATE DATASUM
1316   3363      DCA DATASUM
1317   2364      ISZ CORLOC /UPDATE COR LOCN PTR
1320   7000      NOP
1321   4246      JMS WAIT /GO BACK TO TASK TIME
1322   2362      ISZ TAPCNT /COME HERE FROM TASK TIME
1323   5311      JMP WCDF /DO THIS IF < 1 BLK WRITTEN TO DATE
1324   1363      TAD DATASUM /1 BLK WRITTEN - WRITE CHKSUM NOW
1325   7000     WRCS, NOP /CIA FOR LINC TAPES
                /ABOVE INSTRUCTION CHANGED BY ROUTINE "TPMODE"
1326   6161      IACB /TRANSFER CHKSUM TO LINC B-REG
1327   4246      JMS WAIT /WRITE IT
1330   4246      JMS WAIT /WAIT FOR GUARD MARK
1331   5772      JMP CLEANUP /COMMON "AFTER-BLOCK" ROUTINE
                /
                /USER-CALLABLE ROUTINE TO CHANGE TAPE MODE
```

PAGE 16

```
                /FROM LINC-TAPES TO PDP-8 TAPES AND VICE-VERSA.
                /LINC TAPES ARE PROGOFOP COMPATIBLE AND ARE
                /WRITTEN AT 256 WORDS/BLK  PDP-8 TAPES ARE
                /DEC-TAPE COMPATIBLE, AND ARE WRITTEN AT 128 WORDS
                /PER BLK.  CALLING SEQUENCE - SET LINK BIT IF LINC
                /TAPES ARE DESIRED, OTHERWISE CLR LINK BIT FOR
                /PDP-8 TAPES.
                /
1332  0000      TPMODE, 0
1333  7600      C7600, 7600 /GROUP 2 CLA INSTR
1334  1024       SAVRTN TMRTN
1335  6214
1336  3356
1337  6201       CDF
1340  1371       TAD (CIA
1341  7430       SZL
1342  3325       DCA WRCS
1343  7420       SNL
1344  3301       DCA RDCS
1345  1370       TAD (NOP
1346  7420       SNL
1347  3325       DCA WRCS
1350  7430       SZL
1351  3301       DCA RDCS
1352  1333       TAD C7600 /SET BLKSIZE
1353  7430       SZL
1354  1333       TAD C7600 /CLRS LINK BIT TOO
1355  3032       DCA BLKSIZ
1356  7402       TMRTN, HLT /REP BY CIFD
1357  5732       EXIT TPMODE /W/ CLR AC AND LINK
1360  0000      TAPTEM, 0 /USED FOR TEMP BY RD AND WR
1361  0000      DESIRED, 0 /1+BLKNO FOR WHICH WE'RE SEARCHING
1362  0000      TAPCNT, 0 /CNTR FOR # OF WRDS TRANSFERRED IN THIS BLK
1363  0000      DATASUM, 0 /SUM OF ALL DATA WRDS TRANSMITTED TO DATE
1364  0000      CORLOC, 0 /PTR TO WRD  IN CORE
1365  0000      TAPEOP, 0 /ACTUAL WRD USED TO ESTABLISH R OR W
1366  0000      BLKTBL, 0 /LAST BN FOR TAPE 0
1367  0000              0 /LAST BN FOR TAPE 1
1370  7000      PAGE
1371  7041
1372  0553
1373  0007
1374  0271
1375  0003
1376  0722
1377  4000
```

CHAPTER 7

SUMMARY OF SYSTEM CONVENTIONS

7.1 CPS Detailed Magnetic Tape Organization

Unit Ø tape:

Blk #	Length in Blks	Contents
0	1	cold start loader (CLDSTART)
1	377	symbolic working area (-S)
400	10	unit Ø VTOC (file index)
410	37	bank Ø portion of CPS Control Program
447	1	fresh copy of communication area
450	1	core-image loader (CILDR)
451	1	system bootstrap loader (BOOT)
452	16	unused
470	10	copy of unit Ø VTOC as of last signoff
500	1300	user file area

Unit 1 tape:

Blk #	Length in Blks	Contents
0	1	cold start error program (UNITBAD)
1	147	binary working area (-B)
150	10	bank 1 portion of CPS Control Program

Unit 1 tape (cont.):

Blk #	Length in Blks	Contents
160	10	copy of unit 1 VTOC as of last signoff
170	10	unit 1 VTOC (file index)
200	40	scratch area used while saving and destroying files
240	20	absolute loader (only 12 of the 20 blocks are used)
260	40	relocating loader (not all 40 blocks are used)
320	14	+text editor (*ED)
334	60	+MACRO8 assembler (*ASM)
414	77	+FORTRAN compiler (*FORTRAN) (not all blocks used)
513	77	+@SABR assembler (*SABR) (not all blocks used)
612	1166	user files

+The system does not assume this file is at the location shown.

+@SABR MUST follow *FORTRAN on the tape where *FORTRAN is stored.

7.2 File Tape Table of Contents Structure

The volume table of contents (VTOC) for a CPS file tape consists of 2000_8 words kept in 10_8 contiguous blocks on tape. Each file stored on the tape has an entry called a "file control block" in the VTOC for that tape. The file control block includes the file name and various items of information such as location of the file on tape, file length, and so on. The VTOC is structured as a 1-dimensional array, and each file control block consists of eight contiguous words in this array. The VTOC has the capability to catalog 127 files.

The first 8 words in every VTOC are reserved for tape storage allocation information, and may not be deleted from the VTOC under any circumstances. This pseudo file control block is structured as follows.

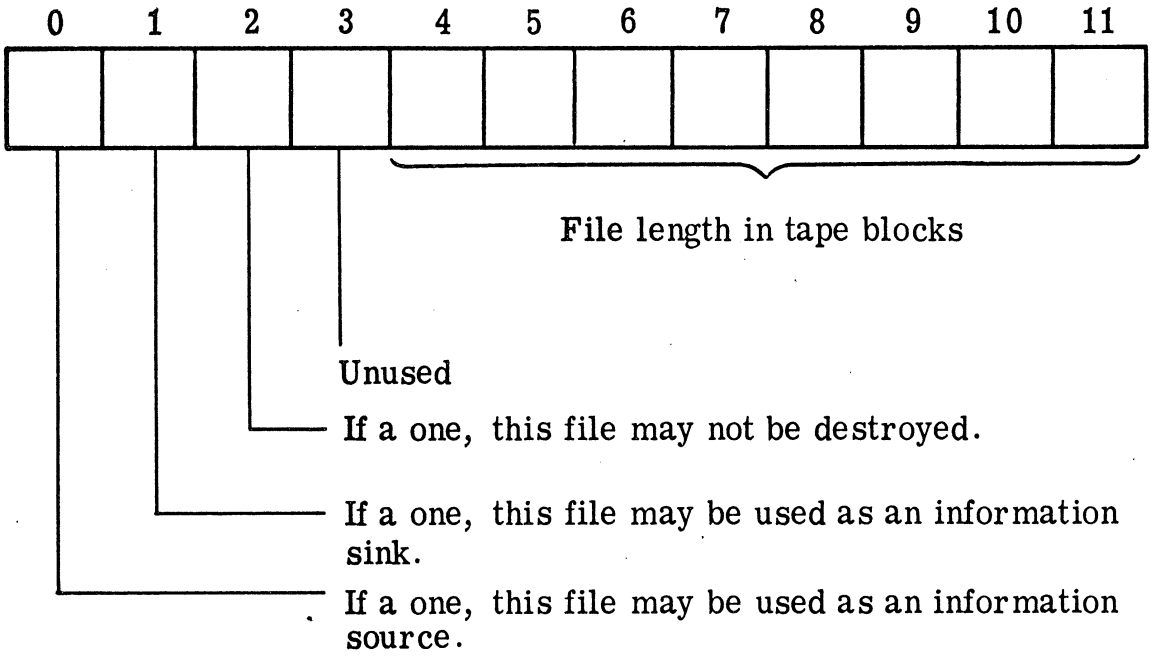
Word No.	Contents
1	Unused
2	negative (i.e., 2's complement) of the number of files catalogued in this VTOC
3	block number of the first free block on the tape
4	number of remaining free blocks on the tape
5	Unused
6	
7	
8	

Files are a sequential contiguous collection of tape blocks stored on tape starting from the beginning of the file storage area and extending toward the end (i.e., toward higher block addresses) of the tape. Files are always packed - there can be no unallocated space between 2 adjacent files on tape. Tape storage for newly-created files is allocated at the end of the last file on the tape by removing the desired number of tape blocks from the set of free blocks and assigning them to the new file. Whenever a file is destroyed, all files further down the tape are moved toward the beginning of the tape, overwriting the destroyed file in the process; a number of blocks equal to the length of the destroyed file becomes free after the last file and is subsequently added to the free storage pool of that file tape.

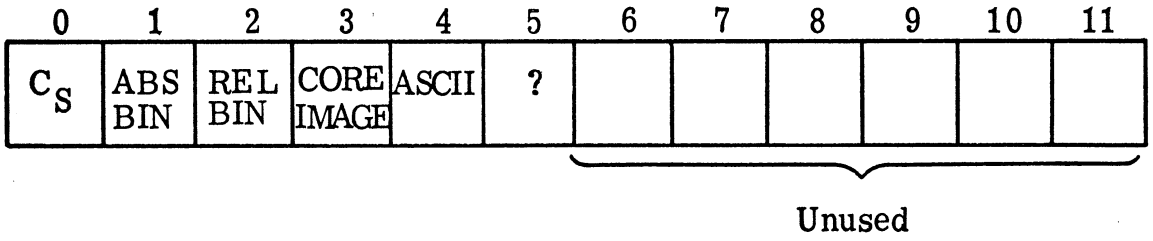
Each file control block has the following format:

Word No.	Contents
1	8 character file name, left justified,
2	packed 2 stripped-ASCII characters
3	per word, padded on the right with
4	trailing blanks.
5	4000* tape unit + tape block number of the first tape block in the file
6	File length and switches word - see below
7	File contents attribute word - see below
8	Unused

Word 6 - File length and switches word



Word 7 - File contents attribute word



Bit	If Set, File Contains
0	Text in compressed symbolic code
1	Absolute assembler object module
2	Relocatable assembler object module (SABR)
3	Core image
4	USASCII characters, one per word
5	Unspecified contents

Management of the VTOC is completely analogous to the management of the file storage it describes. Entries for new files are inserted at the end of the current set of entries, and if an entry is deleted, the remaining entries (if any) are packed to fill the space formerly occupied by the deleted entry.

7.3 Conventions Regarding Storage of Information in Files

7.3.1 Compressed Symbolic Files. The text editor *ED produces line-oriented files of text in compressed symbolic code. This comma-free code trades CPU time for about twice the information filing capability per tape block as straight one character per word techniques would have. All systems programs which allow text input from files assume that the text is coded in compressed symbolic form.

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 2 per computer word. The only exception to this last rule involves the line terminator, the carriage return. By convention each new line starts at the left-end of a computer word.

The following encoding scheme is used:

ASCII	6-BIT
240-276 300-337	the lower 6 bits in the ASCII representation
200-237 340-376 277	a 77 group followed by the lower 6 bits in the ASCII representation

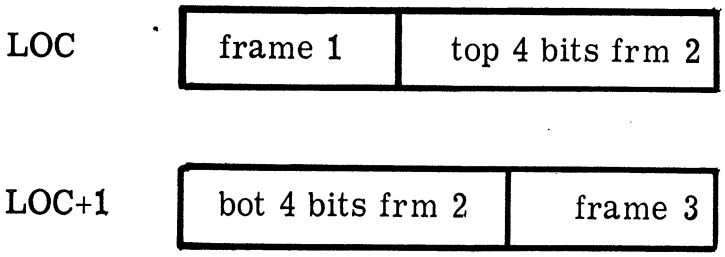
The RUBOUT character is not included in the above scheme.

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

```
1177
1500
0417
7715
```

By convention, files encoded in compressed symbolic form are terminated by a line of text containing the USASCII ETX character (control C on a Teletype). *ED automatically provides this terminator in the text files it produces.

7.3.2 Language Translator Binary Output Files. Both absolute and relocatable 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.

7.3.3 Core-Image Files. Core-image files are produced by the system loaders, and are used to hold programs (both user and systems) in a form suitable for execution via the CPS RUN command. By convention, they contain in a specially-structured form all machine instructions necessary for the proper execution of a program; that is, the code within a core-image file may not contain an external reference to code not contained within the file. The machine instructions within

a core-image file are placed into core memory by the core-image loader when the user RUNs the program. The program in the core-image file may be composed of code which sits in any memory bank, although only banks 0 and 1 are supported in the system loaders.

Each core-image file contains 2 distinct components, a Program Code Area (PCA) and a Descriptor Table (DT). Each 128-word tape block in the PCA contains a piece of the program in the form of a copy of one memory page from the program. For example, if code for a program sits in pages 0, 200, and 600, then the Code Area for the core-image file containing this program would be 3 blocks long on tape; 2 contiguous blocks would contain the binary code from pages 0 and 200, followed immediately on tape by a third block for the code on page 600. If a program has code resident in a number of contiguous core pages, then the corresponding tape blocks in the PCA which contain the code will be contiguous also, running from lowest to highest contiguous page in that memory bank. These contiguous tape blocks in the PCA have acquired the name "subimage" since they constitute part of an image of core. In the PCA, the subimages belonging to the highest memory bank come first on the tape, followed by the subimages for the next highest bank, and so forth down to bank 0. Within the block of subimages destined for a particular memory bank, the subimage to be loaded at the lowest address must come first, followed by subimages for successively higher addresses.

The Descriptor Table (DT) is an encodification of the memory map of the program stored in the Program Code Area. It is used by the core-image loader (CILDR, Chapter 4) to distribute the various subimages comprising the program to the core pages for which they were assembled (or to which they were relocated in the case of the SABR relocating loader). One tape block is allocated to the Descriptor Table, giving the capability to handle 42_{10} subimages within the core-image. The Descriptor Table block precedes the Program Code Area on tape.

The Descriptor Table is structured as a simple 1 dimensional array whose entries describe subimages in the PCA. Normally, each subimage in the Program Code Area has a three word entry in the Descriptor Table. However, the last subimage to be loaded has a five word entry in the table, 3 words of which are a regular subimage entry, and 2 words of which contain program starting address information.

There is a one to one correspondence between Descriptor Table entries and successive subimages on tape in the PCA, i.e., the first Descriptor Table entry describes the first subimage in the PCA, the next Descriptor Table entry describes the next subimage in the PCA, and so forth. The core-image loader utilizes this correspondence to properly situate subimages in core memory. The regular 3 word Descriptor Table entry for a subimage is structured as follows:

Word No.	Contents
1	This word contains the number of tape blocks in the Program Code Area comprising this subimage. This number will usually be less than 40_8 since the last page in each memory bank is reserved for system use in the present 8K version of CPS. The top 6 bits of this word must be zero with one exception: the leftmost bit of this word will be set to a 1 as a flag to indicate that this Table entry is describing the last subimage to be loaded.
2	Bits 6-8 of this word contain the memory field into which the subimage is to be loaded. All other bits in this word must be zero since it will be added to a CDF instruction. Thus the only valid contents of this word in the present version of CPS are either 0000 or 0010_8.*
3	This word contains the full 12 bit address (within the memory field specified by word 2) where loading into core of the subimage should start. Both of the system loaders force this address to be page aligned; however, the core image loader does not assume any

*In other words, one will usually not desire to load into core memory which does not exist!

special alignment of this address. (Thus there is a possibility of playing some games by loading across page boundaries; this has not yet been exploited.)

Bits 6-8 in word 4 in the last subimage Descriptor entry contain the memory field where the program is to be started; all other bits in this word must be zero. Word 5 in the last entry contains the full 12 bits starting address in the field specified by word 4.

7.4 Structure of the CPS/User Program Interface

All user program/CPS intercommunication is achieved by depositing information (in a sense, messages) into the communication area which sits in page 17600. In this page, we have 128 words available. If we allow up to 6 files to be concatenated and assigned to a logical I/O unit, transmission of this information will consume 72 words. We can make the additional reservation of 32 words to hold the contents of the parameter field on the RUN command, and still have 24 words left over for future expansion and kludges. With this in mind, the organization of the area is as follows.

7.4.1 I/O Communication Area.

The structure of the portion of the communications area which passes FD linkages to an executing program is critical to the implementation chosen for concatenated files. Each file or device (FD) chain for a logical I/O unit will consist of 12 consecutive memory locations (organized in 6 entries of 2 words each). The file system will assign to these locations a two word entry for each FD in the logical I/O unit specification given in the RUN command. Up to 6 files may be accommodated on each logical I/O unit, and any number of files over 6 will be flagged as an error. The two-word entry to be assigned for files will include in the first word of the entry, the beginning block number of the file along with the tape unit on which the file resides, and in the second word of the entry, the length of the file in blocks. The information supplied will be copied

from words 5 and 6 in the VTOC of the file tape. To transmit other types of information in the FD chain, the following code will be used:

0000 - an all-zero word as the first word in an entry implies the logical end of the FD chain for the referenced I/O unit. (Note that if 6 files are assigned to a logical I/O unit, no all-zero word will appear. Thus the file access procedure will have to bookkeep the number of files referenced.)

7777 - -1 in the first word of an entry implies the unit is assigned to *DUMMY*, the infinite wastebasket and non-producing source (returns only end-of-file when read). The second word may contain anything.

7776 - -2 in the first word of an entry implies the unit is assigned to the printer. There may be anything in the second word.

7775 - -3 in the first word of an entry implies the unit is assigned to the keyboard/reader. The second word is meaningless.

The RUN command module will check to see if logical I/O units are assigned properly. At the current time, units 1 - 3 are reserved for use as input units, and units 4 - 6 are reserved for use as output units. Also at the current time, only a working area or the printer

may be assigned to an output unit.

To assign a unit, one may use the construct UNIT=FDNAME, where UNIT is an integer between 1 and 6 inclusive, and FDNAME is the name of any currently existing file. To speed up typing, the RUN command module will also accept the following syntax:

RUN COREIMAGEFILENAME UNIT1NAME UNIT4NAME ....

where the COREIMAGEFILENAME is the name of the file containing the core-image to be loaded and run, UNIT1NAME is the name of an information source FD to be attached to logical I/O unit 1, and UNIT4NAME is the name of an information sink FD to be attached to logical I/O unit 4. The "...." is meant to signify the further assignment of I/O units using the UNIT= construct as well as a parameter specification if such are necessary. Any set of up to 6 FD names may be effectively formed into a name for a file having the contents of all files named by the use of the + concatenation operator. If unit 1 or unit 4 names appear, specification of unit 1 or 4 in the following UNIT= construct will override the names given in the free-standing context.

The default FDNAMES for the logical I/O units are:

UNIT	FDNAME
1	RDR (reader/keyboard
2	*DUMMY*
3	Unassigned
4	PTR (teleprinter)
5	*DUMMY*
6	Unassigned

7.4.2 Parameter String Area. If the user supplies a parameter string in his RUN command via the "P=..." construct, the first 32 characters he supplies are copied in full 8-bit ASCII form into the parameter string area in the communication area. If the user supplies fewer than 32 characters, the string is padded on the right with trailing blanks. If the user does not supply a parameter string, the parameter string area is not disturbed and remains all zeros (as established by the RUN command module before input line scanning is initiated).

The user program is responsible for scanning the string and decoding its contents in light of what options are available in the program.

7.4.3 Update Key and Update Parameters. User programs (including systems-type programs) often produce files of information in the working areas. It is necessary to update the VTOC of the appropriate file tape if the working areas are thus changed. To do this, the

user program must deposit information describing the changed file (either -S or -B or both) into the communication area, and then set the update key in the area to certain values to tell the Control Program that working area descriptions must be updated. The locations and values of these parameters are given in the next section.

7.4.4 Memory Map of the Communications Area.

LOCATIONS	CONTENTS
17600 - 17613	Logical I/O unit 1 FD (file device) chain
17614 - 17627	Logical I/O unit 2 FD chain
17630 - 17643	Logical I/O unit 3 FD chain
17644 - 17657	Logical I/O unit 4 FD chain
17660 - 17673	Logical I/O unit 5 FD chain
17674 - 17707	Logical I/O unit 6 FD chain
17710 - 17747	Parameter string from user's RUN command (32 8-bit ASCII characters), or all zeros if no parameters specified.
17750 - 17764	Unused
17765 - 17767	-S updating parameters; starting block number, length in blocks, and contents attribute word respectively.
17770 - 17772	-B updating parameters; starting block number, length in blocks, and contents attribute word respectively.

LOCATIONS	CONTENTS
17773	Tape unit number and tape block number of the next file on the tape after the core-image loaded by the last RUN command. Used for overlays in system programs.
17774 - 17777	Working area updating key. The key area is set to zeros by the RUN command, and must be reset by the user program if it desires to have the entries for either -S or -B updated. The proper key is: -7402, -1, 1, 7402 in that order. Note that the Cold Start Bootstrap Loader sets location 17774 to -1 in order to indicate a cold start; this in turn causes the identifying comment "CPS(XXXXX)" to be typed out on the teleprinter. (See Cold Start Bootstrap Loader writeup, Chapter 2.)

7.5 System Messages

Message No.	Message
1	LINE TOO LONG
2	CPS (AY219)
3	INVALID COMMAND
4	COMMAND NOT YET IMPLEMENTED
5	" NOT A SOURCE FILE
6	"
7	" DOES NOT EXIST
8	ENTER REPLACEMENT NAME OR <CR>
9	TOO FEW OPERANDS
10	INVALID FILE NAME
11	FILE SHOULD NOT BE DESTROYED
12	NO ROOM LEFT ON TAPE
13	REPLACE?
14	FILE CANNOT BE DESTROYED
15	FILE DOES NOT EXIST
16	INVALID OPERAND
17	UNIT
18	% # =END OF FILE
19	" NOT A CORE IMAGE
20	ILLEGAL I/O UNIT NO.

Message No.	Message
21	IMPLICIT I/O-PARAMETER SYNTAX BAD
22	" BAD DIRECTION OF DATA TRANSFER
23	TOO MANY CONCATENATED FILES
24	BYE %####
25	** TAPE ERROR OCCURRED IN ABOVE OPERATION, WILL ATTEMPT TO GO ON

CHAPTER 8

SYSTEM GENERATION

8.1 Introduction

The following step by step procedure describes how to generate a working version of CPS on LINC tapes marked at 128 words/block. A slight modification of this procedure will allow CPS to be generated for DECtape. The instructions assume that a MACRO8-like assembly system will be available to assemble the source code for CPS.

It is also assumed that a program of the following sort is available: a utility program which will read binary paper tape into either core bank (thus encompassing the functions of the binary loader) and which will allow writing onto magnetic tape of selected portions of either core bank in the ratio of one page per tape block. An additional useful capability of this program would be to allow reading of tapes, thus allowing patching of information on the tape. We will call this program "PAPMAG".

8.1.1 Selection and Marking of Tapes. Obtain 2 reels of magnetic tape for the tape drives which are connected to the target machine. These should be formatted with at least 2000_8 blocks of 128_{10} words/block. While it is theoretically possible for CPS to utilize up to 4000_8 block tapes, we are not certain that we have indeed kept the necessary

conventions throughout the system to allow this to be done, so only 2000_8 blocks will be used.

Mark the 2 tapes with paper labels; one should read "CPS tape 0" while the other should read "CPS tape 1".

8.1.2 Preliminary Remarks on Magnetic Tape. CPS uses 3 different tape routines for general purpose applications as well as a read-only routine for certain system functions. For operation on a LINC-8, the routines IOCS3, F1TAPE, and PAGER are supplied. The DECtape user must generate his own versions of these routines, and moreover they must preserve the calling sequence of the LINC-8 routines if CPS is to be used without modification. IOCS3 is described in detail in Chapter 6 of this volume, and F1TAPE and PAGER are described in Chapter 5 of volume 4.

8.1.3 Preliminary Remarks on System Modules. If all system modules are assembled into binary paper tape form before attempting to build the system, it should be possible to generate CPS on a pair of tapes in half a day. Refer to the individual module sections in this chapter for instructions on how to assemble the source code for the various modules.

The source code for the system modules is available in machine-readable form. The distribution of CPS (AY219) to Digital Equipment Corporation was done via the 4K CPS System, while the modules are available in-house at CEL on the CPS System Library Tape. Some of

the modules will also soon be available on paper tape.

An alphabetical listing of the various source modules as they are named on the System Library Tape (names are the same for the DEC distribution) is given in Fig. 8.1. The entries are ordered via the standard ASCII collating sequence.

8.2 Cold Start Programs

The two cold start programs are the first to go onto tape. They form a basis for building the rest of the system. The cold start boot loader will be put onto unit 0 after appropriate modification such that invoking it will cause PAPMAG to be read into core. The cold start error program will be put onto unit 1.

8.2.1 Cold Start Boot Loader. Read in PAPMAG on the target machine using the standard DEC binary loader. Using PAPMAG, read in the binary tape produced by the assembly of the file CLDSTART. (DECtape users must provide their own version of this program.) Using PAPMAG, write PAPMAG out onto the unit 0 tape starting say at block 2. Using the console switches, patch the call to the read-only tape routine within CLDSTART so that when invoked it will read in the copy of PAPMAG which was just written onto the tape and then halt. Using PAPMAG, write this modified CLDSTART program onto block 0 of the unit 0 tape. From now on, when PAPMAG is needed it may be loaded simply by lifting the load switch. (DECtape users see next section.)

FILE NAME	PART OF	FILE NAME	PART OF
$	UT	E2	ED
AL1	AL	E3	ED
AL2	AL	E4	ED
AL3	AL	E5	ED
AL4	AL	EDBUFFNG	ED
BNK0POOL	CP	EDFIX1	ED
BNK1POOL	CP	EDFIX2	ED
BOOT	CP	EDFLD1	ED
C1	FORT	F1TAPE	UT
C2	FORT	FILCPYSY	FILECOPY
C3	FORT	INDEX	CP
C4	FORT	IOCS30	IOCS3
C5	FORT	IOCS31	IOCS3
C6	FORT	IOCS32	IOCS3
C7	FORT	IOCS33	IOCS3
CILDR	CP	IOCS34	IOCS3
CLDSTART	CP	IOSYMTAB	IOCS3
CONSCAN	CP	L1	RL
DESTROY	CP	L2	RL
E0	ED	L3	RL
E1	ED	L4	RL

FIG. 8.1. LISTING OF FILES COMPRISING SYSTEM COMPONENTS

FILE NAME	PART OF	FILE NAME	PART OF
L5	RL	NUC10000	CP
L6	RL	NUC10200	CP
L7	RL	NUC10400	CP
L8	RL	NUC10600	CP
L9	RL	NUC11000	CP
L10	RL	NUC11200	CP
LCOMTBL	AL	NUC2000	CP
LCOMPOOL	AL	NUC2200	CP
LDRTAPE	RL	NUC2400	CP
LOAD	CP	NUC3000	CP
M00	ASM	NUC3200	CP
M02	ASM	P0000	CP
M10	ASM	SPAPRBIN	PAPERBIN
M12	ASM	RUN	CP
M14	ASM	S1	SABR
M16	ASM	S2	SABR
M20	ASM	S3	SABR
M22	ASM	S4	SABR
M8PATCH	ASM	S5	SABR
MESSAGES	CP	S6	SABR

FIG. 8.1. (CONT.)

FILE NAME	PART OF
SAVE	CP
SIGNOFF	CP
SMOCT10	UT
SPAGER	UT
SPUNA	UT
STRWZ	UT
UNITBAD	CP
VTOCS	CP

FIG. 8.1. (CONT.)

UT = UTILITY
AL = ABSOLUTE LOADER
CP = CONTROL PROGRAM
FORT = FORTRAN COMPILER
ED = TEXT EDITOR
RL = RELOCATING LOADER
ASM = ABSOLUTE ASSEMBLER

8.2.2 Suggested Cold Start Provisions for Machines without IPL Circuitry. The basis for the CLDSTART program is the LINC-8 initial program load (IPL) circuitry. Most pieces of 8 equipment do not have such circuitry, and must rely on the rim loader/binary loader route for IPL. We suggest that a very small skeleton tape routine be written for such machines, and punched in RIM format. This skeleton tape routine can simulate the IPL hardware by reading in block 0 of the tape mounted on drive 0, and then transferring to location 0 in memory. The HELP loader system may also prove useful for loading this skeleton routine.

In any event, there should be a well-defined method by which the control program can tell it has been invoked from a cold start. This is so that the system may configure itself, make certain it does not try to update working area descriptions using definitions left over from the last user, and the like. This problem will certainly require some thought in machines without IPL hardware as it would be desirable to leave a system boot loader in core between successive uses of the machine just as the binary loader remains in core on more basic systems. Perhaps the SIGNOFF command could leave a special cold start boot loader in page 7600.

8.2.3 Cold Start Error Program. Assemble the file UNITBAD and feed the resulting binary tape into the target machine using PAPMAG. Write the page of code just loaded onto block 0 of the

unit 1 tape.

8.3 Control Program Bootstrap Loader

The control program bootstrap loader is stored in the file named BOOT; assemble this file and load the resulting binary paper tape into core using PAPMAG. Write the core page just loaded onto block 451 of the unit 0 tape.

Note: DECtape users must supply their own version of BOOT; see Chapters 3 and 4 of this volume.

8.4 Core Image Loader

The core image loader is assembled from the file CILDR. Read the resulting binary paper tape into core using PAPMAG, and write the core page occupied by CILDR out onto block 450 of the unit 0 tape.

DECtape users may have to rewrite portions of this module as it interacts strongly with BOOT; see Chapters 3 and 4 of this volume.

8.5 Prototype Communication Area

The Control Program requires a prototype of the Communication Area on block 447 of the unit 0 tape. This area may be built from the console switches as follows.

Deposit zeros in all locations on page 200 - i.e., in locations

200 - 377 incl. Deposit the following numbers:

Locn	Contents	Function
200	7775	defaults unit 1 to RDR
214	7777	defaults unit 2 to *DUMMY*
244	7776	defaults unit 4 to PTR
260	7777	defaults unit 5 to *DUMMY*

Using PAPMAG write page 200 onto block 447 of the unit 0 tape.

This module is independent of whether the target machine uses LINCtapes or DECtapes.

8.6 Establishment of Initial VTOCs

The VTOC (file index) area on each reel of tape must be initialized. The file VTOCS contains the code which initializes these areas; it creates entries as follows:

Unit 0 - -S, *VTOC0, *SVTOC0

Unit 1 - -B, *SVTOC1, *VTOC1, *ALOADER, *RLOADER, *ED, *ASM, *FORTRAN, *SABR

Assemble the code in VTOCS and read it into core using PAPMAG; then write page 2000 out onto block 400 of the unit 0 tape, and page 4000 out onto block 170 of the unit 1 tape.

This operation is independent of the type of tape used on the target machine.

An assembly listing of the initial VTOC entries is given at the end of Chapter 5 of this volume.

8.7 Control Program

The Control Program occupies both memory banks; because of current memory space limitations and lack of an appropriate assembler, each bank is assembled separately. Two files containing pools of definitions for symbols defined in one bank and used in the other are provided to allow the 2 banks of program to link properly.

8.7.1 Bank 1. Assemble the file BNK0POOL+IOSYMTAB +NUC10000+NUC10200+NUC10400+NUC10600+NUC11000+NUC11200+$. The resulting binary paper tape should be loaded into core using PAPMAG. Finally locations 10000 - 11377 (6 pages) should be written onto blocks 150 - 155 of the unit 1 tape.

This portion of the control program does not vary with the target machine.

8.7.2 Bank 0. Assemble the file IOCS30+IOCS31+IOCS32 +IOCS33+IOCS34+$. The resulting binary paper tape should be labeled "IOCS3" and saved. It will be used to build PAPERBIN (Section 8.9) as well as here. The DECtape user should substitute an equivalent controller here.

Assemble the file IOSYMTAB+BNK1POOL+P0000+NUC2000+ NUC2200+NUC2400+NUC3000+NUC3200+INDEX+DESTROY+SAVE+RUN

+CONSCAN+SIGNOFF+LOAD+MESSAGES+$. Label the resulting binary tape "CP" for "control program."

Load IOCS3 into core (using PAPMAG); then load CP into core. IOCS3 must precede CP. Then write 37_8 blocks starting at core location 00000 onto the unit 0 tape starting at block 410. (Obviously PAPMAG must either sit in an unused area in the bank 0 portion of the Control Program, or else in bank 1.)

The tape unit dependence for bank 0 of the Control Program is taken care of by having the user supply either IOCS3 for LINC-8 or his own controller for other pieces of 8 equipment.

8.8 A First Check

A check will be made of the system as it stands on the tapes at this point. Using PAPMAG, transfer the copy of PAPMAG which is stored on the unit 0 tape over to block #2 of the unit 1 tape. Starting at location 200, toggle into core via the console switches a Descriptor Table for PAPMAG. For instance, if PAPMAG sits in a single contiguous set of pages starting, say, at 16000_8 and extending through 16777_8, and if it has starting address 16030, the Descriptor Table would read:

Loc	Contents	Remarks
200	4004	Last subimage to be loaded - 4 blocks long
201	0010	It goes in field 1
202	6000	Starting at location 6000
203	0010	Start execution in field 1
204	6030	Start execution at 6030 in above field

All other locations in page 200 may contain anything - they are not used. Using PAPMAG, write out page 200 onto block 1 of the unit 1 tape. You have now constructed a CPS core image file containing PAPMAG in a runable form.

The next step will consist of invoking the Control Program. Deposit the following words into the Communication Area to tell the control program that PAPMAG is in -B .

Loc	Contents
17765	0000
17766	0000
17767	0000
17770	0001
17771	length in blocks of PAPMAG core-image file; for the above example, this would be $4 + 1 = \underline{5}$
17772	0400
17774	0376

Loc	Contents
17775	7777
17776	0001
17777	7402

Mount the unit 0 tape on drive 0, and read block 451 into page 07600; mount the unit 1 tape on unit 1. Start the computer at location 7600 - the BOOT loader which you just read in should load and start the Control Program. Issue the command

SAVE -B PAPMAG

When this command has been completed, you should be able to run PAPMAG via

RUN PAPMAG

Run PAPMAG and use it to restore the locations patched on the cold start loader, block 0, unit 0 tape. After this is completed, lifting the IPL (LOAD) switch on the LINC console should load the Control Program directly.

8.9 Absolute Loader

Assemble the file IOSYMTAB+LCOMPOOL+L1+L2+L3+L4+$. Load the binary tape into core and write from locations 3400 - 5777 (12_8 blocks) to block 240, unit 1 tape. Assemble the file LCOMPOOL+

LCOMTBL+$. Load the resulting binary tape into core and write the page just loaded (10600) out onto block 252 of the unit 1 tape.

The absolute loader is an overlay on the Control Program, and thus its tape routine problems are taken care of by IOCS3 (or its equivalent).

8.10 PAPERBIN

PAPERBIN will allow binary paper tape to be read into -B and subsequently to be saved in a CPS file. In conjunction with the absolute loader which was activated above, it will allow construction of the text editor and absolute assembler core-image files.

Assemble the file IOSYMTAB+SPAPRBIN+$. Invoke PAPMAG using CPS, and load first the IOCS3 binary tape and then the binary paper tape just produced by the assembly of SPAPRBIN. Write core locations 0 - 1377 (6 pages) onto the unit 1 tape starting at block 2. Write core locations 2000 - 2577 (3 pages) onto the unit 1 tape starting at block 10_8.

Toggle the following Descriptor Table into core memory using the console switches:

Loc'n	Contents
200	0006
201	0000
202	0000
203	4003
204	0000
205	2000
206	0000
207	2000

Write page 200 out onto the unit 1 tape, block 1.

Now using the console switches, alter the following locations in the Communication Area to the contents shown:

Loc'n	Contents
17765	0000
17766	0000
17767	0000
17770	0001
17771	0012
17772	0400
17774	0376
17775	7777
17776	0001
17777	7402

Using the console switches, start the computer at 7600. BOOT will load the Control Program which will then update the description of

-B to indicate that PAPERBIN is stored there. Execute the CPS command

SAVE -B PAPERBIN

After PAPERBIN is thus filed, it will be used to build the text editor and the assembler.

The DECtape user will of course supply his own version of IOCS3.

8.11 Text Editor

Assemble the file E0+E1+E2+E3+E4+E5+$. Cycle up CPS and run PAPERBIN; read in the binary paper tape produced by the assembly and save -B into the file "ED1". Similarly, assemble EDBUFFMG and save the binary code in the file "ED2". Assemble EDFLD1 and save in "ED3"; assemble EDFIX1+$ and save in "ED4"; assemble EDFIX2+$ and save in "ED5". Assemble SPAGER and save in "PAGER".

Invoke the absolute loader and build the editor by issuing the following CPS commands:

```
LOAD ABS
GET ED1
GET ED2
GET ED3
```

```
GET ED4
GET ED5
GET PAGER@10400
REP 10503 6211
REP 10505 6211
REP 10536 6211
REP 10540 6211
BUILD
supply starting address of 10200
SAVE -B *ED !
```

The DECtape user will have to supply his own tape routine in place of SPAGER. This routine must be set up to read into and write from memory bank 1.

8.12 Absolute Assembler

Assemble the file M8PATCH and using PAPERBIN read the resulting binary tape into CPS and save it in the file "ASMPATCH". Similarly assemble M00+M02+M10+M12+M14+M16+M20+M22+$ and save it in a file named "ASMBNK1". Assemble the DEC MESAGE routine, DIGITAL-8-18-U at location 1600 and read the binary paper tape into a CPS file named MESAGE. Read the paper tape produced by an assembly of F1TAPE into a CPS file named "F1TAPE". Finally, feed in the MACRO8 assembler binary tape, DEC-08-CMA1-PB, using

PAPERBIN, and save -B into ASMBNK0.

Now invoke the absolute loader and build the assembler by issuing the following CPS commands.

```
LOAD ABS
GET ASMBNK0
GET MESAGE @11600
GET ASMBNK1
GET ASMPATCH
GET F1TAPE @10600
BUILD
supply starting address of 10200
SAVE -B *ASM !
```

The DECtape user will have to supply his own routine in place of F1TAPE. This routine must be set up to read into and write from memory bank 1.

8.13 Final Cleanup

Issue the following CPS commands to remove nonexistent files from the file index and to move *ED and *ASM as close to the beginning of the binary file area as possible. This last is done so that *ED and *ASM can be run as quickly as possible since they are the most-used programs in the system.

```
DESTROY *SABR !
DESTROY *FORTRAN !
SAVE PAPMAG DUMMY
DESTROY PAPMAG
SAVE DUMMY PAPMAG
SAVE PAPERBIN DUMMY !
DESTROY PAPERBIN
SAVE DUMMY PAPERBIN
DESTROY DUMMY
```

8.14 Remarks on Mass Storage Device Independence

CPS was written to take advantage of the mass storage capability of the LINC tapes attached to the LINC-8. Thus mass storage device addresses for files are given in terms of "unit zero" and "unit one", following LINC conventions, and this is reflected in the fact that one and only one bit (for two devices) is allocated in each file control block entry for a mass storage device address. If one is attempting to set up CPS to use 2 DECtapes where device addresses are 3 bits wide, note that the DECtape routine can by convention interpret the single unit address bit in a file control block to be either of two 3-bit wide, predetermined DECtape unit addresses.

A more general solution to the problem of specifying device addresses would be to dispose of the "source", "sink", and "do not destroy" bits in the sixth word of the file control block, and use some

of the bits in this sixth word in conjunction with the extant device address bit to specify type of device, i.e., disk or tape or drum, and device address. These bits could then be interpreted by the file access routines so that actual I/O operations could be delegated to the proper device support routine.

Security Classification

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

CPS Program Logic Manual
Volume I -- CPS SYSTEM ARCHITECTURE AND CONVENTIONS

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Technical Memorandum No. 102-I - 03674-23-M February 1970

5. AUTHOR(S) *(Last name, first name, initial)*

Cederquist, G. N. and Metzger, K.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
February 1970	223	

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
Nonr-1224(36) b. PROJECT NO. NR187-200 c. d.	TM102-I 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* 03674-23-M

10. AVAILABILITY/LIMITATION NOTICES

Reproduction in whole or in part is permitted for any purpose of the U. S. Government.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Office of Naval Research Department of the Navy Washington, D. C. 20360

13. ABSTRACT CPS is a generalized programming and file management system written for use on the PDP-8 processor of Digital Equipment Corporation's LINC-8 computer. A minimum memory size of 8192 words is required. Extensive use is made of the two tape units present on every LINC-8 for both file storage and system residence. The four volumes entitled CPS System Architecture and Conventions, CPS Basic Programming Package, CPS 8-K FORTRAN Package, and CPS System Utility Programs were written in order to take a snapshot of CPS at one point in its continuing development. This version of CPS is considered to be a first generation system; successive versions are on the drawing boards and internally resemble their parent less and less every day.

DD FORM 1 JAN 64 1473

Security Classification

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.

2*a*. 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.

2*b*. 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.

7*a*. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7*b*. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8*a*. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8*b*, 8*c*, & 8*d*. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9*a*. 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.

9*b*. 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.