Computation Department

M-026

LTSS Livermore Time-Sharing System

Part I: OCTOPUS

Chapter 4: FILES

Pierre Du Bois, et al.

October 14, 1970 Edition - 1

Changed 4/1/71

Lawrence Radiation Laboratory
University of California / Livermore

LTSS-4
Edition 1
Changed: April 1, 1971

LIVERMORE TIME-SHARING SYSTEM

Part I: Octopus
Chapter 4: Files

Pierre Du Bois
John Fletcher
Marilyn Richards

October 14, 1970

Prepared as a service of the Technical Information Department,
Lawrence Radiation Laboratory, University of California, Livermore.

List of Effective Pages

Page No.	*Issue*
*Title	April 1, 1971
*ii	April 1, 1971
iii thru vi	Original
1 thru 70	Original
*71	April 1, 1971
72 thru 75	Original
*76	April 1, 1971
77 thru 84	Original
*85 thru 86	April 1, 1971
87	Original
*88	April 1, 1971
89	Original

Notes:

1. An asterisk (*) indicates pages changed, added, or deleted by the current change.
2. Any portion of the page affected by the current change is indicated by a revision bar in the outer margin of the page.
3. Destroy all superseded pages.

Changes:

1. Replace title page, 71, 76, 85-86, and 88.
2. Insert new page ii.

TABLE OF CONTENTS

ILLUSTRATIONS

P R E F A C E

This chapter of the Livermore Time-Sharing manuals is a current guide to the OCTOPUS File Maintenance and File Transport Systems.

After a study of this chapter, the reader should be able to make full use of all facilities for the saving, retrieving, and intercomputer transmission of files. While casual use of the system requires only a small degree of understanding, as the user's knowledge of the system grows, his use of the system will correspondingly grow in efficiency and convenience.

The PDP-6 processors which presently control the Elephant System will soon be replaced by PDP-10's, and this chapter has been prepared with that change in mind.

The authors hereby acknowledge - with thanks - the work contributed by Paul Lund, who wrote Section 4.2.1., *the CDC 6603 Disc System,* and Section 4.3.1., *The CDC 7638 Disc System.*

J.F.

I N T R O D U C T I O N

A major function of the Octopus network is to provide a central data-storage facility accessible from programs executing in any of the worker computers (e.g., CDC 7600's). This central facility is called the *ELEPHANT System* and is independent of (although connected to) the data storage facilities of the worker computers, each of which has its own filing system and file-management philosophy.

The ELEPHANT System is under the control of a special ELEPHANT computer, a pair of PDP-10 processors and their shared 256K-word memory. (The user may wish to refer to LTSS Chapter 1 for a complete description of the Octopus System hardware.) This computer is directly connected to the worker computers by high-speed (12 megahertz) interfaces for the transport of data between these computers and the ELEPHANT storage media, which currently consist of an IBM Data Cell and and IBM Photodigital store.

The ELEPHANT computer also uses a General Precision Librascope disc as an intermediate store; this is entirely independent of the discs managed by the several worker computers. The ELEPHANT storage devices are discussed in more detail in Section 4.1.13.

Section 4.1: THE ELEPHANT FILE-MAINTENANCE AND TRANSPORT SYSTEM†

4.1.1. File Structure

Proper and complete use of ELEPHANT facilities requires an understanding of certain basic concepts. A *file* is a body of information known by a symbolic *file name* and handled by ELEPHANT as a unit. A file always consists of two parts, *heading* and *text*. The text is the file proper and contains the information which the user primarily intends to preserve; it is very often the only part of the file of which he need be aware. The heading is a small body of information which describes the file to ELEPHANT and which is used by ELEPHANT as the file is manipulated.

There are two basic types of files, *directories* and *simple files*. A simple file is the sort of file with which the user is most familiar. The text of a simple file may be in any form the user desires; it is neither examined nor understood by ELEPHANT. The filing system exists in order to preserve simple files.

---

† Caution: This discussion reflects the current state of the System. The reader is advised to keep informed as to changes which may alter this discussion, or any part of it.

Directories are used in the course of accessing and preserving other files. They have a rigidly formatted text and are accessible to the user only through special routines of the ELEPHANT system. A directory consists of a number of *entries*, from zero to one hundred twelve of them. Each entry associates a symbolic *link name* with a coded pointer which uniquely identifies the file to ELEPHANT; a given link name may occur only once in a directory. The user always refers to a file by means of its symbolic link name in some directory. He may, if he wishes, find out the value of the coded pointer for any of his files, but it is of no use to him in accessing that file. Link names are formed of one to ten characters from the standard 6-bit ASCII subset, excepting the at (@), the colon (:), and the blank ( ).

A user having access to a directory also has access to all the files listed in it. If any file so listed is also a directory, the user in turn has access to the files listed in that directory, and so on; the directories define a tree-like structure. There are certain files, called *base files*, to which a user always has immediate access. Each base file is designated by a link name beginning with a point (e.g., .R). Any ELEPHANT file to which a user has access must be described by a *chain name* consisting of a base file name followed by a number of link names. The names in the chain are separated by colons, and blanks adjacent to the colons are ignored. For example, .R: AARDVARK: MOD4: JAN25 is a chain name. The first link following the base file name (AARDVARK) names a file listed in the base file (which must be a directory); the second link (MOD4) names a file listed in the file (directory) named by the first link; and so on to the last link (JAN25), which names the file to which the entire chain refers.

All links in the chain, except the last, must name directories. Figure 4.1 gives an illustration of how the file structure beginning at a base file (.R) might appear. (Directories are indicated by rectangles and simple files by circles; the arrows indicate the files to which the link names refer.)

A user may create for himself (using the functions described below in sections 4.1.4 to 4.1.6) about a hundred base files; in practice, he would probably have far fewer, and most or all would be directories. Depending upon the name given it, a base file may be *private* or *special*. A user's private base files, and (if any are directories) the files which they list, are accessible only by himself or by users with whom the files have been explicitly shared (as described in section 4.1.7). Since new Octopus facilities may require the designation of new kinds of special base files, it is necessary to define a class of base file names that are guaranteed to always refer to private base files: a base file with a name of at least *three* characters consisting solely of *letters* and *points* (no digits or other characters) is private; so too is the base file .R. (The name .R has been so reserved because it is a common, but not necessary, practice to have most of one's private files accessible from one root directory, and .R is a natural mnemonic for root. As discussed in section 4.1.11, this single root concept is not practical if a user operates at several security levels.)

Special base files exist either for convenience or to implement special Octopus facilities. A few special base files are summarized in Figure 4.2 and the following paragraphs; others are discussed elsewhere in connection with descriptions of the Octopus facilities that use them. Note that the special base files range from being completely private to being almost completely public.

1. Each user may have a private *working file* (.W), which is usually a directory. In a manner described below, the user may select which of his files is to be the working file and later change that selection. ELEPHANT will remember which file is the working file only so long as the user continues to interact with ELEPHANT at a reasonable rate. (The minimum rate is traffic dependent but is never faster than one interaction per minute.) For this reason, the working file should not be relied upon as the only means to access a file. At the beginning of a period of interaction with ELEPHANT a user has no working file. The intended purpose of the working file is to provide a directory as "close" as feasible to the files with which the user intends to be working for a while. For example, if a user has a directory .R: AARDVARK: MOD4 listing all the files relating to the fourth version of the Aardvark program, he would make this his working directory if he intended to work on that version of the program for a while. Then, for example, the output of the program on January 25 might be referred to as .W: JAN25, which is much briefer than .R: AARDVARK: MOD4: JAN25.

In fact, a further convention in regard to the working directory makes matters even simpler. If the first link name following .W does not begin with a point, then .W and the subsequent colon may be omitted entirely; that is, a chain name not beginning with a point is assumed to be preceded by .W:. Thus, for example, JAN25 is the same as .W: JAN25.

2. Each user may have a private *current file* (.C), which is automatically set by most ELEPHANT functions to be one of the files affected by that function; details are given below. Use of .C makes it easier to perform several consecutive operations on one file. The current file is remembered by ELEPHANT on the same basis as is the working file. The definition of .C is unpredictable after an error.

3. Each user may have a *give directory* (.G). Any files given to him by other users will appear in this directory. He may then (using ELEPHANT functions explained below) place these files elsewhere in his file structure. Thus, the user must understand that the directory .G is subject to certain limited alteration by all other users.

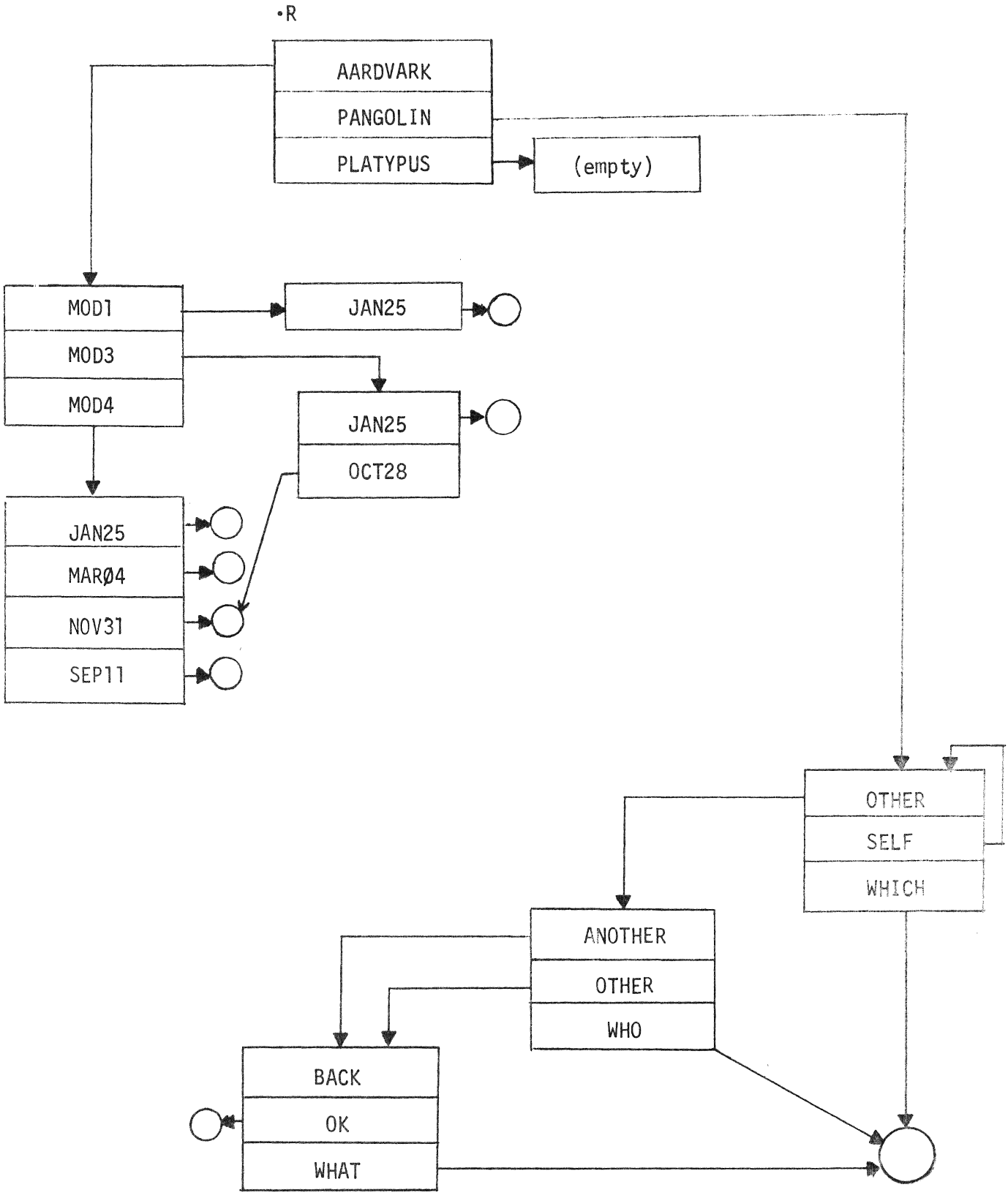


Figure 4.1. A Directory Structure

.W Working file (provides convenient access to a substructure).

.C Current file (a recently accessed file).

.G Give directory (residence of newly received files)

.T Take directory (residence of files placed in public domain)

Figure 4.2. Special Base Files

	Type
D	Directory
F	Simple file
	Lifetime
S	Short
M	Medium
L	Long
	Level
S	Secret
A	Administrative
P	Protected [as restricted data]
U	Unclassified
	Access
Ø	No restrictions
1	Write denied
2	Read denied
3	Execute only
4	Execute denied
5	Read only
6	Write only
7	No access

Figure 4.3 File Property Codes

4. Each user may have a *take directory* (.T). Any files which he wishes to make available to all other users may be listed in this directory, since each user has access to all the files listed in another user's take directory. A take directory is essentially a privately-maintained public directory.

Another user's give or take directory is named by a point (.) followed by that user's number (six decimal digits); the type of operation being performed determines which of the two directories is meant. There are — of course — limitations on the use of such a name; these are given below in section 4.1.7.

It should be noted that the name of an ELEPHANT file does not reside with the file but only in the chain of directories through which the file is accessed. Consequently, one file may have many names if it is accessible through many directory chains; for example, in Figure 4.1, if the working directory is the same as .R: PANGOLIN, then WHICH, SELF:WHICH, OTHER:WHO, and OTHER:OTHER:WHAT all name the same file. The same link name (e.g., JAN25) may appear in many directories, and chain names referring to different files may involve common link names. Closed loops may occur in the access chains leading from a directory; this implies a hazard to recursive directory searching routines.

It is intended that the user organize his directory structure to reflect the logical arrangement of his work. He thus need only use and examine that portion of the structure relevant to his current activity. The general layout (not detailed content) of a user's structure should be such that he can easily remember it. A user who finds that he must frequently examine his entire structure in order to find the files he needs should give some serious thought to improving his file organization.

4.1.2. File Properties

A file has a number of *properties* recorded in its heading by ELEPHANT. One is the *type*, directory or simple file, as already discussed. In summarizing the properties of a file, it is often convenient to use a few letters and numbers as an abbreviation; for this purpose,

the type is indicated by D (directory) or F (simple file). Other properties of interest are the *lifetime*, the *security level*, and the *access inhibit code*. These are now briefly discussed in turn; letters or numbers used as abbreviations are indicated parenthetically and are summarized in Figure 4.3.

Files may have one of three lifetimes: short (S), medium (M), and long (L). Short lifetime files reside on the ELEPHANT disc and survive only a few hours from last access. Since this is about the same as the survival criterion in worker computer storage, short lifetime files are of no value to the user, and he is not usually permitted to create such files; they are for system use. Medium lifetime files are rewritable but of fixed length, are kept on the Data Cell, and survive one month from last access (in the case of simple files) or indefinitely (in the case of directories). Long lifetime files are not rewritable, are kept on the photostore, and survive indefinitely.

The security level of a file is set when it is created and may not be changed. That is, the highest classification of the information that is to go into a file should be decided at creation, and more highly classified information must never be placed into it; of course the content of the file may be copied into a more highly classified file, which could then accept information that the original could not. Recognized levels are secret (S), administrative (A), protected [as restricted data] (P), and unclassified (U). Security is discussed more fully in section 4.1.11.

Three kinds of access to a file are recognized: execute (4), read (2), and write (1). The access code for a file is the sum of the digits designating those accesses which are inhibited; thus, for example, 5 indicates read only, while Ø indicates all accesses permitted. Access is discussed more fully in section 4.1.1Ø.

The properties of a file are defined for ELEPHANT by the user with a declaration which consists in sequence (no intervening blanks) of the designations for the type and lifetime, a point, and the designations for the level and access. For example,

DM.P1

indicates a directory having medium life, protected level, and write access denied. Since the type and lifetime designations use no common letters, any part of the declaration may be left out without confusing what remains; the point may be omitted only if both the level and access are also. For example,

F.Ø

indicates a simple file with all accesses permitted; lifetime and level are not given.

4.1.3. File Transport System

An Octopus user's program executing in a worker computer (e.g., CDC 7600) has access to files residing on the secondary storage (disc or drum) of that computer. Subject to various restrictions, the program is permitted to create, destroy, edit, execute, output, and input these files. The *ELEPHANT file-transport system* permits the program to output or input files to or from other worker computers or the long-term storage media maintained by the ELEPHANT system. Certain manipulations of files in the long-term ELEPHANT media are also permitted.

The user's program communicates with the ELEPHANT system by means of *message packets* similar to those used for teletypewriter communication. These messages are coded in ASCII. Each message sent to ELEPHANT is a request by the user program for some function to be performed. Upon completion of the function, ELEPHANT sends a reply to the user program. There is thus a one-to-one correspondence between requests and replies. (A special program, ELF†, available on all worker computers, relays messages between ELEPHANT and the user at a teletypewriter, and thus gives the illusion of direct user communication with ELEPHANT.) With two exceptions, explained below, ELEPHANT will work on only one request from a given user at a time. Requests that cannot be acted upon

---

† For a complete description of ELF, please see reference 2 and also the Appendix.

because a previous request is still being processed are rejected and a suitable error reply given. Thus, a program should normally wait for the reply to one request before issuing another. (The program ELF has a stacking facility which enables the user to type requests faster than ELEPHANT can act on them.)

The operating system of the worker computer on which the requesting program is operating (the *requesting computer*) sees to it that each message to ELEPHANT is properly identified with the user's number. This enables ELEPHANT to decide which files it can and should attempt to access.

Included in the messages between a user's program and ELEPHANT are chain names and/or the names of files as they are known on worker computers. Unless otherwise noted, such names, as well as all other items in messages, are blank delimited and include no interior blanks. The name of a file on a worker computer consists of from one to ten six-bit ASCII characters and is understood by ELEPHANT to be a file belonging to the user whose program sent the request; a name may not contain the "at" and may not consist solely of the dash (-). Under some circumstances a name is permitted to refer to a worker computer file belonging to another user. In this case, the name is preceded by a decimal point (.), the other user's number (six decimal digits), and a colon (:); blanks on either side of the colon are ignored.

4.1.4. File Transport Functions

The first item in any request from the user to ELEPHANT is a three-letter operation code defining the kind of function ELEPHANT is to perform. (See Figure 4.4. for a summary listing of all functions.) Five of these functions, which involve actual transport of file content between computers, are discussed in this section. The user's own program takes no part in the actual transport; this is done entirely by ELEPHANT and by worker computer systems working on ELEPHANT's behalf. These systems have the ability to create, read, and write worker files belonging to the user. Note that both the requesting program and the system of the computer in which it is executing (the requesting computer) converse with ELEPHANT, but that they do not converse with one another.

Abort	ABT					
Change name	CHN	Old	Link			
Copy	CPY	Old	New	Prop		
Create directory	CRD	Name	Prop			
Delete	DEL	Name1	Name2	Name3	...	
Declare property	DPR	Name	Prop			
Destroy	DST	Name1	Name2	Name3	...	
Duplicate	DUP	Old	New	Prop		
Fetch from shelf	FFS	Name1	Name2	Name3	...	
How	HOW	Name				
List	LST(*)	Name	First	Last		
Read computer	RDC(*) Comp	Oname	Rname	Ofwa	Rfwa	Count
Read directory	RDD(*)	Ename	Rname	Efwa(=Ø)	Rfwa	
Read storage	RDS(*)	Ename	Rname	Efwa	Rfwa	Count
Report	RPT					
Restore	RST	Name				
Set	SET	New	Old	Prop		
Write computer	WRC(*) Comp	Oname	Rname	Ofwa	Rfwa	Count
Write storage	WRS(*) Prop	Ename	Rname	Efwa(≠Ø)	Rfwa	Count

Figure 4.4. File Function Formats

It is important to realize that a transport is an input-output type of operation. That is, data is copied out of a file (*source*) controlled by one computer and written into a file (*sink*) controlled by another computer. The source file is not destroyed or altered. Once the transport is complete, neither ELEPHANT nor the worker computer systems regard the source and sink as having any special relationship to one another; ELEPHANT files and the files of several worker computers are independent data bases.

Write computer (op code WRC) is used to transport a file from the requesting computer to another computer. Similarly, *read computer* (RDC) is used to transport from the second computer to the requesting computer. ("Write" and "read" in the mnemonics refer to what is done to the other computer.) No "third party" transfers are permitted in which a program executing on one computer initiates a transport between two other computers. The format of the message is:

OP COMP ONAME RNAME OFWA RFWA COUNT

As is the case with all functions, the various arguments (items) in the message are blank delimited (except for optional blanks surrounding colons in chain names). In the present case, OP is of course either WRC or RDC.

The first argument (COMP) is used to specify the other computer involved in the transport; the specification is a single letter (e.g., S). The next two arguments are, respectively, the name of the file as it is (or is to be) known on the other computer (ONAME) and the name of the file on the requesting computer (RNAME). The second of these names may be omitted, in which case it is assumed that the same name is used on both computers. An omitted argument is indicated with a dash; it may also be left out entirely, but in that case all subsequent arguments must also be omitted.

It is normally assumed that the named sink file on the destination computer (ONAME in the case of WRC; RNAME in the case of RDC) does not already exist; it will be newly created and of just sufficient length to hold the incoming information. However, if the op code is immediately followed by an asterisk (*) (e.g., WRC*), then the contrary is assumed, namely that the named sink file already exists and is to be rewritten.

The remaining arguments are all nonnegative octal numbers. They are, respectively, a first-word address on the other computer (OFWA), a first-word address on the requesting computer (RFWA), and a nonzero word count (COUNT). Thus it is possible to transport parts of files into parts of files. The word size used is always that of the requesting computer. Omitted first-word addresses are assumed to be zero, while an omitted count is assumed to extend to the end of the source file. In fact, even when a count is given, it is viewed only as an upper limit; it is not considered an error to transport less than the count because the end of the source file is reached. On the other hand, running over the end of the sink file is an error. The first-word address for the sink file must be zero when it is newly created. (There may be further limitations on first-word addresses and counts as required by the systems of the worker computers involved.)

Write storage (WRS) is used to transport a file from the requesting computer to an ELEPHANT storage medium. (The mnemonic "storage" always refers to ELEPHANT storage.) The message format is:

WRS PROP ENAME RNAME EFWA RFWA COUNT

The first argument (PROP) defines the properties of the file in ELEPHANT storage, using the format given in section 4.1.2. The lifetime must be given, and usually that is all. The type of course is not given, since it must be F; the other properties are discussed below in sections 4.1.10. and 4.1.11. The remaining arguments in the message are interpreted essentially as with WRC and RDC (ENAME and EFWA referring to ELEPHANT storage and RNAME and RFWA to the requesting computer), with the exceptions now noted.

The ELEPHANT file name (ENAME) is, of course, a chain name. If the worker file name (RNAME) is omitted, it is assumed to be the same as the last link of the ELEPHANT name. The size of an ELEPHANT file is limited to 144 x 112 x 1024 = 16,515,072 bits (e.g., 275,251 sixty-bit words).

The option (indicated by an asterisk after the op code) that the ELEPHANT file be rewritten rather than newly created of course does not apply to long life files. When an ELEPHANT file is rewritten, even if only part of it is rewritten, it is not guaranteed that any data previously in the file is preserved; rewriting is merely an opportunity to reuse the same file space. Therefore, no purpose is served in declaring a nonzero ELEPHANT first-word address (EFWA). For this reason it is required that this address be given as zero (either explicitly or through the drop-out convention already discussed). The reason that this unnecessary argument is included in the message format is to keep a high degree of similarity with the formats of the other transport functions and thereby reduce the possibility of confusion. The length of a medium life file is fixed when it is first written.

Read storage (RDS) is used to transport a file to the requesting computer from an ELEPHANT storage medium. The message format is:

RDS ENAME RNAME EFWA RFWA COUNT

A property indication is not needed, since ELEPHANT will know from its own records all necessary information about the ELEPHANT source file (ENAME). The arguments and an asterisk following the op code are interpreted as with WRC and RDC (with the same convention as with WRS for the omission of the worker file name) except as now noted.

The ELEPHANT first-word address (EFWA) must correspond to an integer multiple of 36 x 1024 bits (e.g., must be a multiple of 3 x 1024 words on a 60-bit word computer). This requirement makes for more efficient operation of ELEPHANT storage, and should cause no difficulty, since any unwanted bits can be ignored by the requesting program.

Directories are of use only to the ELEPHANT computer. A function (LST, described in section 4.1.8) is available which should satisfy most requirements for examining the content of directories. However, should a

complete copy of a directory be desired in a worker computer, the function *read directory* (RDD) provides the needed capability. (Read storage does not provide it.) The message format is:

RDD ENAME RNAME EFWA RFWA

The ELEPHANT first-word address (EFWA) must be zero; a count designation is not used. The entire directory (112 entries) is transmitted; unused entries are all zero. Each entry is 144 bits long, and entries are packed without regard to word boundaries. (Thus, 269 words are used on a 60-bit computer.) Other conventions are the same as for RDS.

The cost of performing ELEPHANT operations is borne by the requesting user. Included in this cost are the ordinary charges for the processor time of the program (such as ELF) which issues the requests to ELEPHANT and the ordinary charges for the message packets themselves. In addition, a fixed charge is made for each 36 x 105 x 1024 = 3 870 720 bits (e.g., 64 512, 60-bit words) or portion thereof transported. (That is, a million bits transported by making a thousand 1000-bit requests cost a thousand times what it would cost with one request.) The amount of the fixed charge is determined by the system of the requesting computer.

Transports to or from storage (WRS, RDS, RDD) select the transported file as the current file, except as noted at the end of section 4.1.6; computer to computer transports (WRC, RDC) do not affect the current file.

4.1.5. Creation of a Directory

Any function that causes a new entry to be made into a directory (WRS being the only one thus far described) assumes that the directory already exists. For example,

WRS L .R:AARDVARK:MOD4:OCT28

will lead to an error if there does not exist a directory .R:AARDVARK:MOD4, since an entry (OCT28) is about to be made into it. Therefore, it is necessary to have a means of creating new directories.

The required function is *create directory* (CRD), which uses the format

```
CRD NAME PROP
```

NAME is, of course, the chain name by which the directory is to be known. PROP is usually omitted, since the directory will automatically be medium life with enough space for a full one hundred twelve entries; the use of this argument to define level and access is discussed in sections 4.1.10 and 4.1.11. The new directory becomes the current file, except as noted at the end of section 4.1.6. For example, sometime previous to the WRS operation given in the last paragraph, there must have been performed the operation

```
CRD  .R:AARDVARK:MOD4
```

and sometime previous to that the operations

```
CRD  .R:AARDVARK  and
CRD  .R
```

A new user of ELEPHANT (who must be entered into the system by specific request to Computation Department) starts with no files at all. Initially, he will require several CRD operations to build up his file structure. In particular, the operation

```
CRD  .G
```

should be performed in order that he have a give directory; otherwise he will be unable to receive files from other users.

4.1.6. File Structure Manipulation

Seven functions are available for manipulation of directory entries and related purposes in regard to ELEPHANT files. These are discussed in this section.

Duplicate (DUP) generates a new directory entry or base file referring to the same file as another directory entry or base file; the original entry is unaffected. The format is

```
DUP OLD NEW PROP
```

where OLD is a chain name by which the file is already known, and NEW is the name by which it is to be known because of the new entry. PROP is usually omitted, and, in any event, can affect only the access as discussed in 4.1.10. Another function, *set* (SET), is identical in effect, except that the order of the arguments is reversed:

```
SET NEW OLD PROP
```

This is similar to an assignment statement in which the name NEW is set to refer to the value of the name OLD. For example, either

```
DUP  AARDVARK:MOD4  PLATYPUS: MOD1
```

or

```
SET  PLATYPUS: MOD1  AARDVARK: MOD4
```

would result in a new entry MOD1 being made into the directory PLATYPUS; this entry refers to the same file as the entry MOD4 in the directory AARDVARK.

The working file is selected by using .W as the second argument of DUP or as the first argument of SET. For example,

```
DUP  .R  .W
```

makes the base file .R into the working directory.

Copy (CPY) uses the format

```
CPY OLD NEW PROP
```

it copies the file named by OLD and gives the copy the name NEW. The third argument (PROP) gives the properties of the copy. The lifetime must be given. Level and access are discussed in sections 4.1.10 and 4.1.11. Note that DUP (or SET) involves two references to one file, while CPY involves two files (which happen, for the moment anyway, to have the same content). CPY can be used to "freeze" a version of a medium-life file into the photostore.

Delete (DEL) can have any (reasonable) number of arguments:

```
DEL NAME1 NAME2 NAME3 ...
```

It removes the entry defined by each name from the directory in which it appears or discards a base file name. Note that this does not destroy or otherwise affect the file to which the entry refers; it is still accessible if listed somewhere else. On the other hand, *destroy* (DST), which uses a similar format:

```
DST NAME1 NAME2 NAME3 ...
```

not only deletes the reference to the file, but destroys it so that it is inaccessible from all other entries which may refer to it (although the entries remain).

Change name (CHN) is a combination DUP and DEL, and has the format:

```
CHN OLD LINK
```

The entry defined by OLD is changed so as to contain the link name LINK, but it still refers to the same file. For example,

```
CHN  AARDVARK:MOD4 FINAL
```

has essentially the same effect as

```
DUP  AARDVARK:MOD4  AARDVARK:  FINAL
```

followed by

```
DEL  AARDVARK:MOD4
```

An issue which must be considered is: What happens when an entry is made into a directory, displacing an entry of that name which already exists? Whenever this occurs, ELEPHANT so informs the requesting program. For each directory, the entry most recently displaced (or deleted) is remembered by ELEPHANT and may be restored to the same directory by use of the *restore* (RST) function with the format:

```
RST NAME
```

The argument NAME is the chain name by which the file is known after it is restored. The restore will not take place if it introduces another displacement or if the most recently displaced (or deleted) file has already

been restored. RST can also restore the most recently displaced (or deleted) base file, except for .W or .C. (The similar situation in regard to worker files, in which a transport would cause the creation of a worker file of the same name as one which already exists, is considered an error.)

The functions DEL, CHN, and RST do not affect the current file; DST renders it undefined; DUP, SET, and CPY set it to refer to the same file as the new entry which they create, unless that new entry displaces an existing one of the same name. Any time an existing entry is displaced, the file to which it refers becomes the current file.

4.1.7. Giving and Sharing of Files

A file may be given to another user by using a name of the form: "point, user number, colon, single link name." The arguments for which such a name is legal are the sink file names of transport functions which create, rather than rewrite, the sink (ONAME for WRC, ENAME for WRS, and RNAME for RDC, RDS, and RDD) and the new file name of DUP, SET, and CPY. When the file given is an ELEPHANT file, it appears in the recipient's give directory. Note that when DUP or SET is used to effect the give, the giver retains a reference to the same file, thus giving rise to a shared file. If a directory is shared, the various users with access to it may enter new files into it, thus giving rise to additional shared files without the use of the give or take directories. A name of the form "point, user number, colon, single link name" may also be used as an argument of DEL or RST or as the old file name of CHN; this permits cancellation or correction of erroneous gives.

An example of giving is:

```
WRC  X  .ØØØØØ2:SAMPLE
```

User ØØØØØ2 on the X computer receives a file called SAMPLE, which is a copy of a file that the giver also called SAMPLE on the requesting computer. An example of sharing is:

```
DUP  .R:TEST  .ØØØØØ2:  SAMPLE
```

A file which user ØØØØØ2 can reference as .G:SAMPLE is shared with the giver, who calls it .R:TEST. User ØØØØØ2 might then enter the file into a more suitable place by

```
DUP  .G:SAMPLE  AARDVARK:MOD5
```

If it is supposed that the shared file is a directory, user ØØØØØ2 might then request

```
WRS  L  AARDVARK:MOD5:MARØ4  OUT - 1ØØ
```

thus placing a file in the shared directory which the original giver can access under the name .R:TEST:MARØ4.

A chain name beginning "point, user number" (and followed by any number of links) may be used to refer to another user's take directory in any situation which does not result in an alteration of the take directory itself. If the giver of the previous example had performed:

```
DUP  .R:TEST  .T:SAMPLE
```

then any user (including ØØØØØ2) could obtain the file that was given only to ØØØØØ2 in that example. If the giver's number is ØØØØØ1, then ØØØØØ2 could take the file and enter it where he wanted it with

```
DUP  .ØØØØØ1:SAMPLE  AARDVARK:MOD5
```

4.1.8. File Status Functions

The first function considered in this section effects a change in file status. This is *fetch from shelf* (FFS); the format is

```
FFS  NAME1  NAME2  NAME3  ...
```

where there may be any (reasonable) number of chain-name arguments. Each file named, if it has been retired to the photostore shelf (removed from the photostore device because it is several months old), is returned to on-line status in the photostore. The completion of the function means only that a request to the operator has been made; the progress of the operation can be monitored with the HOW function (described below). No operation requiring actual access to the file, except FFS, may be performed on a file on the shelf.

The reply sent by ELEPHANT to the requesting program when a job is done normally consists of just the op code of the function followed by "carriage return, line feed, end." However, this is preceded by a bell and an error comment if the job could not be successfully done, or (if appropriate) by a comment that a new entry displaced an entry of the same name. (See Section 4.1.12 for a discussion of error replies.) The remaining two functions to be considered in this section (and the three to be considered in the next sections) include commentary as part of the normal reply, since they are requests for status information.

List (LST) is used to list a directory; the format is

LST NAME FIRST LAST

The reply is a list of as many entries from the directory named by NAME as can fit into one packet. Further entries from the same directory can be obtained by LST operations with no arguments; ELEPHANT keeps track of where it is in the directory and will produce the complete listing if enough LST requests are made (without any intervening requests, other than RPT, discussed below). The end of the listing is signalled by the appearance of the op code in the reply and by the fact that the last word of the reply packet is zero. (A user of the program ELF should not type LST ops with no arguments, since ELF automatically generates them for him.)

Each item in the listing begins with a declaration of the properties of the file in the format given in section 4.1.2; the letter X or the digit 8 is used to indicate that a defect in the entry prevents proper interpretation. This declaration is followed by the link name, carriage return, and linefeed. The entries are alphabetically ordered.

Thus, the listing

```
FL.U5      DATA
DM.PØ      MAP
```

means that DATA names a long-life unclassified file that can only be read, not written or executed, and that MAP names a medium-life protected directory with all forms of access permitted.

The second and third arguments (FIRST, LAST) are optional. If they are given, the listing will include only those entries which alphabetically do not precede the up-to six-character name FIRST nor follow the up-to six-character name LAST: this enables part of a directory to be listed. If LAST is not given, the listing will extend to the end of the directory, and, of course, if neither FIRST nor LAST is given, the entire directory is listed.

If an asterisk follows the op code (i.e., LST*), then the listing will include, not only the entries in the named directory, but all entries in any directories to which those entries refer. The arguments FIRST and LAST do not apply to the entries in the subdirectories, and only one level of subdirectories is listed. The listing is continued by an LST (not LST*) op without arguments.

A listing of all base files, except .W, .C, and .(user numbers) can be obtained by giving a single point as the argument NAME.

How (HOW) is used to find out a variety of things; the format is:

HOW NAME

If the argument (NAME) is omitted, the reply is a summary of the current status of the ELEPHANT System and its storage devices (up, down, etc.). If the argument names a long-life file, the reply states how far along the file is in being recorded and read-back-checked on the photostore or whether or not it is on the shelf. (Write storage or copy for a long-life file is considered complete and a reply sent when the file has been transported or copied and queued for recording but before it is actually recorded. Otherwise delays of several minutes would be experienced.) If the argument names a medium-life file, the expected expiration date of the file is given in the reply (and this date is not advanced as a result of the HOW operation).

The file referenced by HOW or LST and the last one referenced by FFS become the current file. The remainder of this section describes the replies received from HOW in some detail.

The HOW function applied to a long-life file can elicit several replies.

QUEUED FOR RECORDER.
RECORDING.
DEVELOPING.
WAITING FOR CHECK .

represent successive stages from the completion of the WRS or CPY function which created the file until it is checked for readability and stored away; any of these replies may be followed by a positive decimal integer N and the word RETRIES, which indicates that, because a check could not be obtained, the Nth retry is being made. Each try or retry can take anywhere from a few minutes to many tens of minutes.

ON LINE. N SECTIONS.

where N is a nonnegative decimal integer, indicates that the file is available in the photostore device and is in the section which is Nth in the queue for being retired to the shelf. A newly created file is approximately 200 in this queue. The maximum rate at which sections move to the shelf is four per day, but in practice a file should stay on line for several months.

FILE DAMAGED. RESUBMIT.

means that the file could not be properly recorded on the photostore. This situation is rare and information regarding such files is published. The HOW function elicits this response, however, for only a limited time after recording, since there is limited space for keeping the necessary records; after this initial period has elapsed, the ON LINE response is given.

ON SHELF .

indicates that the file has been retired to the shelf. After an FFS function is applied to the file, the HOW function will elicit the reply

REQUESTED FROM SHELF .

until the file is returned to the photostore, after which

ON LINE TEMPORARILY .

is the reply. Files are fetched from the shelf at scheduled times each day and remain on line for a scheduled period of up to a few hours.

For other than long-life files, the reply to the HOW function is

EXPIRES D

where D is the earliest date or time at which the file will expire.

The reply to the HOW function with no argument either is

ELEPHANT IS UP

or a statement as to the condition of any ELEPHANT storage devices which are below standard. In the extreme case, the reply

ELEPHANT IS DOWN

indicates that the system cannot be used at all.

4.1.9. Request Status Functions

All functions mentioned thus far are only handled one at a time by ELEPHANT, as previously stated. The two functions discussed now are the exceptions. They are intended to be used if the reply to a request seems unreasonably delayed. Neither has any arguments.

Report (RPT) elicits a reply as to the status of the request on which ELEPHANT is currently working. The reply indicates that the request is still being interpreted, indicates that an attempt is being made to abort it (as described next), or (in the case of transport or copy functions), gives its position in the appropriate queue of transport jobs (the top of the queue being zero). If a report is asked for when ELEPHANT is not acting on a request, the reply will be a repetition of the reply to the previous request.

Abort (ABT) requests ELEPHANT to abort the request currently in progress; it also elicits the same response as would RPT, the response usually being that the abort is being attempted. This response should

be soon followed by the reply to the aborted function; usually this reply states that the operation ended because of the request to abort, but sometimes another reply is given, indicating that the request to abort arrived too late to be effective. An ABT request also causes ELEPHANT to discontinue supplying further portions of a listing in response to LST requests with no arguments.

Neither RPT or ABT affects the current file. The remainder of this section describes reports in some detail.

A report as to the status of a request always ends with the op code RPT (even if the report is elicited by an ABT request). If no request is being acted upon, the report message is a repetition of the reply to the most recent request, unless so much time has elapsed (at least one minute) that ELEPHANT has forgotten the working and current files. In this case, no message precedes the RPT. An exception occurs if the most recent request was an LST function. Then the report message is either LST or LST ABORTED. The latter occurs if an ABT request arrives before the listing is complete.

INTERPRETING REQUEST means that ELEPHANT is reading and interpreting a request. The time required for this includes the time required to fetch from storage any directories necessary to the interpretation of chain names, but should seldom exceed a few seconds. Except in the case of an abort, this is the only kind of interim report that can be received regarding a request which does not involve a transport or copy.

ABORT BEING CONSIDERED means that a request to abort is being acted upon. From when an abort is requested until the abort takes place (or the function involved is otherwise completed) is usually a few seconds, unless a file is currently being moved to or from long term ELEPHANT storage. In this case, the completion of that operation must be awaited. (See the next paragraph for how to estimate the time required for this.)

QUEUE # = N, where N is zero or a positive decimal integer, means that a file is awaiting its turn to access ELEPHANT storage and/or use the file-transport mechanism. Depending upon the function involved, the request is

in one of five independent queues; the integer N indicates how many requests precede the user's request in the same queue. Each queue has a characteristic information rate as follows:

Function		Rate (megahertz)	Time/bit (μsec)
WRS	(long life)	.25	4
WRS	(medium life)	.11 - .16	6 - 9
CPY, RDS, RDD	(long life)	1.25 - 1.50	.7 - .8
CPY, RDS, RDD	(medium life)	.11 - .32	3 - 9
WRC, RDC		.91 - 1.50	.7 - 1.1

Where a range of rates is given, the smallest assumes maximum interference from activity in other queues, and the largest assumes no such interference. (Reading and writing of medium-life files interfere with one another, writing of long-life files interferes with reading them, and all transports to and from storage interfere with intercomputer transports.) The time each request spends at the top of the queue can be found by multiplying the length (in bits) of the file involved by the appropriate time/bit. Thus, the time for a given request to reach the top of a queue is N times the appropriate time/bit times the average file size; the last figure can of course only be estimated on the basis of experience. Even the most careful estimates can be on the order of ten seconds in error because of random interference effects.

4.1.10. Access Control

An access code for a file is recorded, not only in the file heading, but also in each directory entry pointing to the file. The codes so recorded need not be the same. A given kind of access is inhibited if it is indicated as inhibited in the file heading, or if it is indicated as inhibited by the entry through which the file was accessed. Thus it is possible, for example, to be able to write into a file if it is accessed through one directory entry but not if it is accessed through another. One can therefore share a file with others, allowing them only, for example, to read it, while retaining full access for oneself. The inhibitions in the file heading represent inhibitions which apply no matter what path is used to reach the file. When a directory is listed, the access code given with each entry is the code contained in the entry; it must be understood that further inhibitions may be indicated in the file heading.

The access code of a file newly created (by CPY, CRD, or WRS) may be indicated with the argument PROP. This code is placed both in the file heading and the directory entry. If no code is indicated, Ø (all accesses permitted) is assumed. Note that a newly-created write-inhibited directory is useless.

When a new entry is generated pointing to an existing file by means of DUP or SET, the argument PROP may be used to indicate additional access inhibitions that will apply to the new entry. If no code is indicated, the new entry will have the same inhibitions as the old.

The function *declare property* (DPR) may be used to change the access code in a file heading (not in a directory entry). The format is

DPR NAME PROP,

where NAME, of course, names the file and PROP declares the access code. The reply to this function includes a complete declaration of the properties of the file both as given in the directory entry and the file heading (the former in parentheses); therefore, performing a DPR operation with the argument PROP omitted is a way to ascertain the properties of a file. The reply also indicates the file length in words (or, in the case of a directory, the number of entries) as an octal integer. A report on a DPR request may indicate a queue number, as it would for a CPY.

Write access to a file is required for any operation that in any way alters the file. A directory is write accessed if a new entry is placed into it (by CHN, CPY, CRD, DUP, RST, SET, or WRS) or an old one removed (by DEL or DST). A simple file is write accessed if it is rewritten (by WRS*) or destroyed (by DST); in the latter case, the file will be destroyed (if it is write accessible) even if the associated entry deletion cannot be done (because the directory is not write accessible). A file is not considered to be accessed at all as it is newly created, even if (as with CPY and WRS) an initial fund of information is placed into it. The function DPR write accesses the file it references unless the second argument (PROP) is omitted; therefore, once a file is completely write-inhibited (by the access code in its heading), this inhibition can never be removed. All long-life files are write inhibited.

Read access to a file is required for any operation that examines or copies the contents of the file (CPY, DPR, LST, RDD, RDD*, RDS, or RDS*). In the unlikely case of a file that is write but not read accessible, DPR will effect a change in the access code, even if the properties cannot then be returned in the reply.

It is to be noted that the accesses discussed here apply only to ELEPHANT storage. There is no reason, for example, that an ELEPHANT file that is read only cannot be transported to a worker computer and there be rewritten. The worker computers have their own file access codes, which may inhibit transports in some cases. Therefore, RDC and WRC, as well as ABT, RPT, and HOW with no argument, which reference no ELEPHANT files, cannot violate any ELEPHANT access inhibitions.

Execute access to simple files is involved in an attempt to execute the file as a procedure on the ELEPHANT computer; therefore, such access is of no interest to a user on a worker computer. However, execute access has also been given a meaning in the case of directories: a directory is execute accessed if a use is made of any of its entries, either to access the file to which it refers or (with DUP or SET) to replicate the pointer elsewhere. That is, all but the last one or two links of a chain name must refer to executable directories.

A file is not subject to any access inhibitions if it is merely the file referenced by a pointer changed by a directory manipulation (with CHN, DEL, DUP, RST, or SET), is newly created (with CPY, CRD, or WRS), or is a file accessed by FFS or HOW.

4.1.11. Security Levels

At the time that a file is created (with CRD, WRS, or CPY), its security level may be declared by means of the PROP argument. In the case of CRD, if the level is not declared, it is assumed to be protected as restricted data (P). In the case of WRS and CPY, the declared level cannot be less than the source of the information going into the file; if the level is not declared, it is assumed to be the same as that of the source.

Worker computer systems inform ELEPHANT of the level at which a user generating a request is operating and of the level of every worker computer file he asks to be accessed. The following restrictions are enforced: (1) The user may not access any ELEPHANT file of level higher than his own, except that he may create a file of any level (and immediately write into it with a transport function or CPY). (2) The source file (in the case of a transport function or CPY) must be of a level no higher than the sink file. (3) A directory may not list files of level higher than itself. Because of this restriction, a user normally needs at least one private base directory for each level at which he intends to operate, since a base directory of a given level cannot be accessed while operating at a lower level nor can it list files of a higher level. (4) A file of level higher than administrative (A) may not be given or taken; special procedures (not discussed here) are required for transferring higher level files among users.

ELEPHANT records the security level of a file, not only in the file heading, but also in all directory entries pointing to the file. Unlike the situation with access codes, the levels so recorded are all the same.

4.1.12 Error Replies

This section summarizes replies sent by ELEPHANT when some anomalous condition is obtained. Error replies are divided into five categories, depending upon the presumed source of the difficulty. The replies relating to four of these categories are distinguished by being preceded with an identifying letter enclosed in parentheses as follows: elephant (E), transport (X), overload (O), and hardware (H) errors; the replies relating to the last category, user errors, are not preceded by any identifier.

Many errors can be identified with the attempt to access a particular directory or file in ELEPHANT storage. In such cases, the error reply is usually prefaced with the formula,

"@ CM, LN",

where "M" and "N" are positive decimal integers; "M" is the chain name number, and "N" is the link name number. Thus, for example, "@ C2, L3" means the third link of the second chain name in the requesting message. The chain name number or the whole formula may be missing if there can be no confusion as to which file access caused the error. The formula indicates which link name was being interpreted when the difficulty arose.

Elephant (E) errors arise because of failures in the software which manages the ELEPHANT storage media; *transport* (X) errors arise because of failures in the ELEPHANT or worker software which manages the message packets, worker storage media, or actual data transport. The distinction between these two categories is not important to the user; he should report all such errors to Computation Department, except for the UNIMPLEMENTED FEATURE error, which implies that software development is somewhat behind software planning.

Overload (O) errors arise because a resource of the system (storage space, core, system table entries) has been exhausted.

Usually, waiting and trying again later is the proper strategy for the user; complaint to Computation Department should be made only if the condition persists for tens of minutes.

Hardware (H) errors may be divided into three subcategories; one includes those errors which arise from hardware malfunction. The condition should be reported only if it persists after several retries. The replies

PHOTOSTORE TRANSFER ERROR
DATA CELL TRANSFER ERROR
DISC TRANSFER ERROR
WORKER DISC TRANSFER ERROR

indicate a hardware-detected failure in a read or write of the photostore, data cell, ELEPHANT disc, or worker disc, respectively. Persistent failure when reading may imply that the stored data is damaged. The reply

TRANSPORT INCOMPLETE

indicates that a hardware-detected failure in intercomputer data transmission has made it impossible to complete a requested transport.

WORKER INTERFACE IS DOWN and
WORKER INTERFACE STOPPED

indicate that the interface between ELEPHANT and a worker computer is inoperative, the latter reply implying that the failure began during the period ELEPHANT was working on the user's request. Often the inoperative condition is transient, being caused by difficulties in worker computer I/O controllers. Note that no reply discussed here can be taken to imply that the ELEPHANT computer is down; on the contrary, since that computer generates the replies, any reply at all is evidence that it is up.

Another subcategory of hardware errors includes those which arise when an ELEPHANT storage device is shut down for maintenance; the interpretation of these replies should be clear:

PHOTOSTORE RECORDER UNAVAILABLE
PHOTOSTORE READER UNAVAILABLE
PHOTOSTORE UNAVAILABLE
DATA CELL UNAVAILABLE
ELEPHANT NOT AVAILABLE

Normally, reporting these errors is a waste of time.

The remaining hardware errors,

FRAME HEADING ERROR
FILE POINTER ERROR
MISSING CHIP
DISC ERROR DURING RECORDING

arise because of misfiled, lost, or improperly recorded chips in the photostore. These errors should be reported if they persist after retrying.

User errors arise because of mistakes, oversights, or confusion on the part of the user. It is, of course, possible that system errors can lead to false user errors, but a user should be extremely certain of his facts before concluding that an error in this category is not his own fault.

The first subcategory of user errors includes syntactical errors in a request message.

ILLEGAL OP CODE

indicates that the op code does not describe an existing function. Similarly,

ARG N? ,

where N is a positive decimal integer, indicates that the Nth argument of the function is not of the proper form. For example, it may contain an 8, 9, or nondigit when it should be an octal number, or it may be a missing argument for which there is no drop-out convention. If the defective argument is a chain name, more detailed analysis is usually given instead of the ARG N? reply.

ILLEGAL LINK NAME

indicates that a link name or base file name does not conform to the standards given in section 4.1.1.

NULL CHAIN NAME

means that a chain name is entirely missing.

ILLEGAL NAME FOR USER 999999

means that the name selected for a file given to user 999999 on a worker computer does not conform to standards set by the worker system.

Another subcategory of user errors involves missing files.

NO SUCH FILE

means that a worker computer file needed for a transport does not exist.

NO SUCH ENTRY

means that a link name referring to an ELEPHANT file does not occur in the indicated directory. On the other hand,

FILE EXPIRED

means that the link name exists but refers to an ELEPHANT file which has been destroyed, either by specific request or because it has not been accessed for a period of one lifetime.

NO DISPLACED ENTRY

means that there is no file for the RST function to restore.

LONG LIFE FILE ON SHELF

means that the file has been removed from the photostore and placed on the shelf; it may be made available through use of the FFS function (section 4.1.8).

CELL IN USE FOR RECORDING

means that a file on the photostore is temporarily unavailable because it is stored in proximity to files still being recorded. This difficulty clears up of its own accord after several minutes.

The next subcategory of user errors involves ELEPHANT files appearing in the wrong context.

NOT A DIRECTORY

means that an operation appropriate only to a directory was attempted on a file which was not a directory. Conversely,

NOT A SIMPLE FILE

means that an operation appropriate only to a simple file was attempted on a directory. Either of these replies can arise from an attempt to use either the working file or the current file before it has been selected or after it has been forgotten by ELEPHANT because of infrequency of interaction.

WRITE ACCESS DENIED,
READ ACCESS DENIED,
EXECUTE ACCESS DENIED

imply violations of permitted types of access to a file; it should be remembered that all long-life files are not rewritable.

INADEQUATE CLEARANCE

implies that the user is attempting to access a file of level higher than the one at which he is operating or (if appropriate) higher than the threshold (A) for giving or taking.

DECLASSIFICATION NOT PERMITTED

means that the user is attempting to place information into a sink file or make an entry into a directory of security level lower than the source.

ILLEGAL CHAIN NAME

implies use of a base file name in an inappropriate context.

Then there are user errors involving files that exist when perhaps they should not.

FILE ALREADY EXISTS

means that an attempt has been made to create a worker file of the same name as an existing one.

NAME IS PUBLIC

means that an attempt has been made to create a worker file with the same name as a public file; this is not permitted.

ENTRY ALREADY EXISTS

means that an illegal attempt has been made to create a new directory entry with the same link name as an existing one in the same directory. This is not permitted in the case of some functions (CHN and RST). On the other hand,

ENTRY DISPLACED

means that an existing entry of the same name as the new one has been set aside and the new entry made. This is not necessarily an error; it depends upon the user's intent. This displaced entry can be reached through the RST function. This subcategory of errors also includes

DIRECTORY FULL,

which means that an entry cannot be made into a directory because it already contains 112 entries.

Several errors relate to various inconsistencies and illegalities in regard to file sizes, first-word addresses, and counts. The restrictions are either obvious or have been detailed previously; so the error replies should be pretty much self-explanatory:

ILLEGAL WORKER FWA
ILLEGAL ELEPHANT FWA
SINK FILE TOO SMALL .

Note again that a source file shorter than the count field implies is not considered anomalous.

The remaining errors are rather miscellaneous.

NOT A USER (REMOTE SYSTEM)

means that the requesting user is not listed as a user of the ELEPHANT system or of another worker computer involved in a transport; to be so listed requires a specific request to Computation Department.

USER NO. IS IN USE

is the reply given if a program running under the requesting user number (on another computer) is using ELEPHANT; ELEPHANT only permits one request per user number at a time.

TOO SOON

is the reply given when the same program (or another program on the same computer) makes one request while its previous one is still being acted upon.

REQUEST ABORTED

indicates that a request was not fully carried out because of a subsequent ABT request.

INSUFFICIENT FUNDS

means that the user is unable to pay for a requested transport.

4.1.13. Storage Hardware

The ELEPHANT (PDP-6) computer is discussed in section 1.1 (reference 1). The present section describes the devices used for file storage in greater detail than is done there. All ELEPHANT devices are oriented to a 36-bit word.

The *General Precision Librascope disc*[3] (officially, the *Librafile 4800 Mass Memory*) consists of two asynchronously rotating disc stacks and the associated electronic controls. It can operate at a transfer rate of either 20.35 megahertz or 10.175 megahertz. In the faster mode, each stack is divided circumferentially into 1216 arcs of 32 words each and radially and axially in 144 bands of two 18-head track sets each; the slower mode differs in that there are 608 arcs and a band consists of four 9-head track sets. Thus, there is a total of 22 413 312 words or 806 879 232 bits on the disc. Rotation time is 70 milliseconds; so average random access is 35 milliseconds, since there is one head per track.

ELEPHANT software treats the disc as consisting of 512-word pages. The pages making up a file need not be physically contiguous; the file heading (one page) contains a map of where the data pages of the file are located. Pages are added to a file as needed, so that the file can grow without allocating any definite space to it at creation. A disc scheduler handles read/write requests, not necessarily in the order received, but so as to have a transfer occur for each arc that includes a page for which a request has been made.

The *IBM Data Cell*[4] consists of a rotatable carousel divided into ten cells, each containing twenty subcells. The carousel may be rotated to bring any subcell to a point at which one of ten strips which it contains may be removed and wrapped around a rotating reel. A strip is

essentially a short, stiff tape; the entire device is somewhat of a tape "juke box." Each strip consists of five bands of twenty tracks each; each track holds 450 words. Thus, there are 90 million words or 3.24 billion bits in the Data Cell. Reading is performed at a rate of 324 kilohertz; writing is half as fast, and a minimum of one track may be written at a time.

Random access averages a few hundred milliseconds, but different tracks on the same strip may be accessed in microseconds. Therefore a file on the Data Cell is (insofar as possible) placed on consecutive tracks. This means that each file kept on the Data Cell has a fixed size. It is possible to remove cells from the Data Cell and replace them with others, but this feature is not exploited in normal operation, since it requires shutting down the entire unit and would seriously degrade random access time.

The *IBM Photo-Digital Store*[5-6] consists of three modules each divided into 75 sections of 30 cells each. A cell is a small box which may be moved pneumatically from its normal location in the device to a recording station or one of two reading stations. Each cell contains 32 chips, which are small pieces of film. A chip is laid out as 32 frames of 4100 words each. Thus, there are 28 339 200 000 words or 1 020 211 200 000 (over one trillion!) bits in the device.

At a reading station, a chip may be selected from a cell and positioned at the desired point; average reading rate, including necessary mechanical motions, is about 1.5 megahertz. A chip may be written only once; for this reason, a file is stored as a contiguous block and is regarded as not rewritable. At the recording station, new (raw film) chips are written, developed, and placed in a previously empty cell; average recording rate is 255 kilohertz (one chip every 18.5 seconds). A minimum of one frame may be written at a time, but an entire chip must be completed before moving on to another. Random access time is about three seconds, but the pneumatic motion times of one cell may be overlapped with the reading and recording of others, permitting essentially continuous operation if each reading operation involves more than some minimum number of words.

Cells may be removed from the photostore and later returned to it. Thus, there is provision for a shelf, a storage area for older cells. The photostore has its own built-in computer which controls and monitors the operation of the device under command of the ELEPHANT computer. The recorder is the most complex part of the machine and includes provision for moving the raw film into a high vacuum, outgassing, recording with an electron gun (which has automatic focusing and filament replacement), and developing by sequential application of chemicals, wash, and dry air; the entire process takes about 3 minutes, 20 seconds.

Figure 4.5, summarizes the characteristics of the ELEPHANT storage devices and compares them with the storage devices of the worker computers.

Device	size (bits)	size (words)	Random Access (ms.)	Read Rate (bits/ms)	Read Rate (words/ms)	Write Rate (bits/ms)	Write Rate (words/ms)
6600 Core Memory*	7.8×10^6	1.3×10^5	.001	6×10^5	1×10^4	6×10^5	1×10^4
6603 Disc*	4.5×10^8	7.5×10^6	230	7.5×10^3	125	7.5×10^3	125
7600 SCM*	3.9×10^6	6.5×10^4	.000275	1.65×10^9	2.75×10^7	1.65×10^9	2.75×10^7
7600 LCM*	3.1×10^7	5.12×10^5	.0017	1.65×10^9	2.75×10^7	1.65×10^9	2.75×10^7
7638 Disc*	5.1×10^9	8.5×10^7	85	3.4×10^4	570	3.4×10^4	570
7600 Drum*	1.6×10^8	2.6×10^6	67	1.7×10^4	292	1.7×10^4	292
PDP-6 Core Memory†	9.44×10^6	2.62×10^5	.0014	2.6×10^4	700	2.6×10^4	700
GPL Disc †	8.07×10^8	2.2×10^7	35	2×10^4	566	2×10^4	566
IBM Data † Cell	3.2×10^9	9.0×10^7	400	324	9.0	162	4.5
IBM Photo † Store	1.0×10^{12}	2.8×10^{10}	3×10^3	1.5×10^3	42	255	7

Figure 4.5. File Storage Media

*One word = 60 bits

†One word = 36 bits

Section 4.2: THE CDC 6600 FILE-MAINTENANCE AND TRANSPORT SYSTEM†

4.2.1. The CDC 6603 Disc System

The CDC disc system[7] consists of a 6603 Disc File Controller and a Bryant Disc file. There are three such systems attached to each of our CDC 6600 computers.

Each disc file consists of two separate banks of twelve parallel disc surfaces. Each disc surface is further divided into zones, tracks, and sectors. There are eight head groups (four in each bank) each of which consists of one read/write head per disc surface. A *ZONE* is associated with a head group and is that portion of the disc surface that can be serviced by one read/write head. A *TRACK* is that position of the heads at a constant radius from the center of the surface along which information is read or written. Each zone has 128 tracks at which the heads can be positioned. All head groups are positioned to a particular track position simultaneously. A *SECTOR* is that portion of a track which is the unit-record size accessible on the disc. There are 128 sectors/track in each of the outer two zones and 100 sectors/track in each of the inner two zones.

The system operates on twelve-bit words; that is, data is read or written by the PPU in groups of twelve bits. Each bit of a twelve-bit word is read/written on a separate disc surface by one of the twelve heads of the selected head group.

Data is written on the disc in sectors, and each read/write must start at the beginning of a sector. Each sector read must be for the exact number of words that were written into that sector, if a valid parity check is desired. Each sector begins with four twelve-bit words of zeros, a series of data words, and a check word. The zero words and the check word are written automatically by the hardware and are not transmitted to the PPU on reading. The hardware regenerates

---

† For a complete description of ELF, please see reference 2 and also the Appendix.

the check word during reading and compares it with the check word recorded on the disc. If the two check words are not the same, a parity flag is set in the status register.

The surface rotates at ∿910 rpm. Hence, one revolution takes ∿66 milliseconds. However, head positioning can take up to 270 ms (up to four revolutions). Once the heads are positioned and the proper sector is found, data may be read or written at 9 microseconds per sixty-bit word from that sector.

Each sector on the LRL discs contains 322 twelve-bit words. The first word contains the TRACK number. The second word contains a *CODE* that is generated by the PPU by taking the two low-order bits of the track number, shifting left three bits, adding the three bits of the head group number, shifting left seven bits, and adding the seven-bits of the sector number . These two words uniquely define every sector on the disc. The remaining 320 twelve-bit words (64 sixty-bit words) contain the valid data that is transmitted to/from central memory.

The table in Figure 4.6 lists some of the disc capacity statistics in usable sixty-bit words. Disc function codes and the status reply word are summarized in Figure 4.7.

	Sixty-bit Words	Revolutions
Words/sector	64	
Sectors/track (inner group)	100	1
Words/track (inner group)	6,400	1
Sectors/track (outer group)	128	1
Words/track (outer group)	8,192	1
Sectors/double track*	228	2
Words/double track	14,592	2
Double tracks/position	4	8
Sectors/position	912	8
Words/position	58,368	8
Positions/disc	128	1024
Sectors/disc	116,736	1024
Words/disc	7,471,104	1024
Discs/machine	3	
Words/machine	22,413,312	

* The LRL system uses tracks alternately from outer and inner groups. Hence, a *double track* is a constant number of words.

Figure 4.6. CDC 6603 Disc Statistics

Function Code	Octal
Read Sector XX (Sectors 00 - 77)	10XX
Read Sector XX (Sectors 100 - 177)	11XX
Write Sector XX (Sectors 00 - 77)	12XX
Write Sector XX (Sectors 100 - 177)	13XX
Select Track XX (Tracks 00 - 77)	14XX
Select Track XX (Tracks 100 - 177)	15XX
Select Head Group X (Y is the read margin Select)	16YX
Status Request	1700

Status Reply Word

Bit	11	10	9	8	7	6						0
	Ø	Ø	G	R	P	S	S	S	S	S	S	S

G	0	Group re-synced
	1	Group not re-synced
R	0	Ready
	1	Not Ready
P	0	No parity error
	1	Parity error (reading only)
S		Sector now under the heads

Figure 4.7 CDC 6603 Function Codes and Status Reply Word

How the PPU uses the disc: Under the operating system at LRL, a user may write or read a variable number of words to or from the disc starting at any location in a sector. If the data does not begin at the first word of the sector, the PPU first reads the whole first sector and, if reading, just transmits to central that portion starting at the word specified by the user. The PPU continues until it has read all of the last sector where only those words are transmitted to complete the word count. If the disc is being written, however, enough data is read from central to overlay the first sector image and the entire first sector is rewritten. The PPU continues to read central and write the disc until the last sector, which it then must read and overlay from central and rewrite. This type of disc write is accomplished at the cost of two extra revolutions of the disc.

In order to assure data transmission to and from the correct positions on the disc, the identifying code in the first two twelve-bit words of the sector are checked before any data is transmitted by the PPU. This means, of course, that a sector must be read. While reading data, every sector is checked. However, if the disc is to be written, a sector must always be read for verification first. In case of a parity error during the read operation, no attempt is made to verify the position but, up to five attempts using three different margins are made to get a parity-free read. In case of a verification failure, the same five attempts are made to recover before returning a parity error to the user. If the disc was to be written, a parity error return is fatal and the user program can not continue.

For disc operations involving more than one sector, two PPU's are used. Each PPU handles alternate sectors so that while one is accessing the disc, the other is accessing central memory. While reading the disc, each PPU checks for parity errors and if it finds one, it does not write the data into central but stores away in a table all information about that sector. When the job is completed, the slave PPU sends all of its parity-error information to the boss PPU

which retries each bad sector up to five times, at three different margins, before finally writing the data in central. If any of the sectors could not be read parity-free, a parity error return is given the user.

4.2.2. The CDC 6600 Filing Subsystem

Each of the Laboratory's three CDC-6600 Computer systems' peripheral storage-equipment complement includes three CDC-6603 disc-storage units. Each unit is capable of storing some 7.5 million 60-bit words.

The discs are used as long-term storage and as auxiliary storage for core memory. Long-term storage is provided for some code files and data files which have general use. These might include general-utility programs, special-purpose codes, and data files which are widely or frequently used. Auxiliary storage is provided as a temporary aid to the programmer and for program swapping by the operating system to facilitate its time-sharing operations.

In these senses, a disc file is defined to be some number of contiguous locations on a particular disc and having a known starting address and length. The file may be privately owned (accessible by only a single user and, sometimes, by the operating system), or publicly owned (accessible by the body of users and the operating system).

The resident-operating system is a completely file-based system. All programs to run under control of this system must originate on disc. During time-shared operations, executing programs will be swapped in and out of central memory from and to disc storage. Many of the operating system's long-term data files are maintained on disc and system accountability is accomplished from some of these data files. Recovery procedures for restarting the system after a "crash" make use of disc-stored data to restore many of the system working tables.

The methods of assigning disc space and maintaining an inventory of disc space is discussed in Section 4.2.4.

In addition, the FROST operating system maintains some selected and less frequently used parts of itself on disc and calls them in as overlays.

4.2.3. File Identification

Each disc file is recognized as having a unique owner who is known by the system. The owner is identified by his user (badge) number and is verified through tables kept by the system. A user creating a file must specify a name for it. This can be any nonzero bit string. However, if the file is to be referenced from a teletype console, the name should be a series of ASCII characters which can be typed. The combination of this name and user number are unique to the newly created file. The previously mentioned publicly owned files are characterized by their absence of a user number, thus allowing access by any user.

The information about this file (including name, user number, disc unit, disc first-word address, length, read/write access, and security information) is kept in a file-index table. This table is capable of cataloging 1088 files. Disc-storage allocation is done through another table, the disc map. The disc map table accounts for void areas on each disc unit.

4.2.4. Disc Mapping

As each request to create private disc space is received by the operating system, it is assigned to a particular disc unit according to the value of a sentinel which is rotated by one each time it is used. This has the effect of assigning sequentially requested file space to alternate disc units. A program written to take advantage of this feature can improve its I/O time through overlapping I/O on different discs. The program may also specify which disc unit is to be used by a system call.[8] Each disc unit is mapped independently; no file may occur partially on one disc unit. Further, a file is assigned as many contiguous sectors of disc (64 words per sector) as are required to contain it. No means exist for linking separated blocks of disc space.

A unit map consists of a word-per-entry table. An entry is made in the table for each void space on the disc. The entry consists of a disc number, a first-word address and a length.

4.2.5. Housekeeping

To make disc space available to a user upon request, it is necessary for the operating system to clean house periodically; that is, to purge itself of long unused disc files. It does this by periodically noting the elapsed time since the file was last referenced and whether or not its owner is currently active in the system. If a file owner is currently working with the system, all his files are secure from purge until he finishes. The maximum age of a file is a system parameter and may be shortened in periods of heavy traffic to try to accommodate all requesting users.

Recently, a scheme for repacking scattered files on a disc unit has been added to the system. This allows unused space to be collected into contiguous locations, thus making as much of the existing disc space as possible available.

4.2.6. File Types

As previously noted, disc files may be accessible by all users (public files, not subject to purge) or by a unique user (private files, purgable). In addition, read/only and read/write access is allowed. A user may direct that any particular file be in either mode. All public program files are normally read/only, while public data files are read/write. A read/only file is one whose information may be read but not modified — no write access to the file will be allowed. A read/write file is one whose contents may be read and altered and rewritten at will.

Program files generated by any of the various compilers, assemblers, and loaders available have a particular format. Two-hundred fifty-six words are added to the code file immediately preceding the code's addressable word zero. These are known as the program's "minus words," and the operating system keeps some of its information regarding the execution of this program here. Most of the information is dynamic and is used as an aid to the swapping scheme. This area is not addressable by the program itself, though its executing length is increased by the length of the area.

Additionally, for every program or data file created, the operating system prefixes one disc sector (64 words) as a system I.D. sector to be used in whatever way the current system might require. The program's executing length does not include these words. This sector is not addressable by any one except the operating system.

4.2.7. Program Minus Words

The term "minus words" refers to a block of 256 words that are added to every problem program preceding the program's relative word zero. These words are not accessible to the program.

Word 0 is usually the name of the program, in ASCII. It is not used by FROST.

Word 1 consists of four fields, one of which is unused.

59 42	41 30	29 12	11 0
FREEP		CUREX	NXJP

where

-NXJP- is the number of exchange packages in use. This is usually one at the beginning of execution.

-CUREX- is a (relative) pointer to the current exchange package within the minus words. It should be zero initially, and will be filled in by FROST. If it is zero, FROST assumes that the first exchange package is located at word two. If it is nonzero, FROST will assume that it is indeed a pointer to the current exchange package.

-FREEP- is a relative pointer to the first free exchange package area. Exchange-package areas are eighteen words long, and up to nine exchange packages can be stored in the minus words. FREEP should be zero at the start of execution and if so, it is filled in by FROST to point to word twenty. The assumption made is only one exchange package at execution startup; therefore, the second exchange package area at word twenty is free. If not zero, FROST assumes this is a pointer to the first free area.

Words 2-277* contain nine exchange package areas - eighteen words each. These words are defined below.

Word 300 is the ASCII word, "STACKED IOD."

Words 301-302 are used by FROST for restacking IOD's.

Words 303-337 are relative pointers to (up to 29) IOD's not yet issued, but already requested by the problem program (stored only when program is suspended to disc).

Words 340-357 contain the ASCII names of (up to 16) tapes and disc files associated with words 360-377 in a one to one correspondence.

Words 360-377, commonly referred to as "minus words 15 through 0," are the minus words pointed to by an IOD. They describe the I/O devide, and are formatted as follows.

For disc IOD's,

-A- determines the access status:
A = 0 for read-only.
A = 1 for read/write.

-B- specifies the channel number.

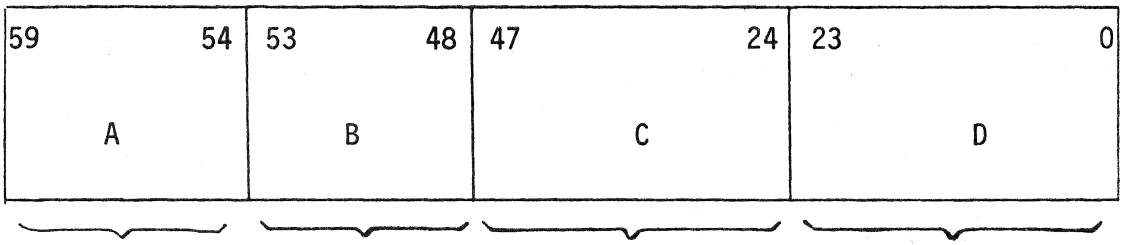
-C- is the absolute first-word disc address of the file.

-D- is the size of the file. (The user is unaware of the fact that it is 64 words longer.)

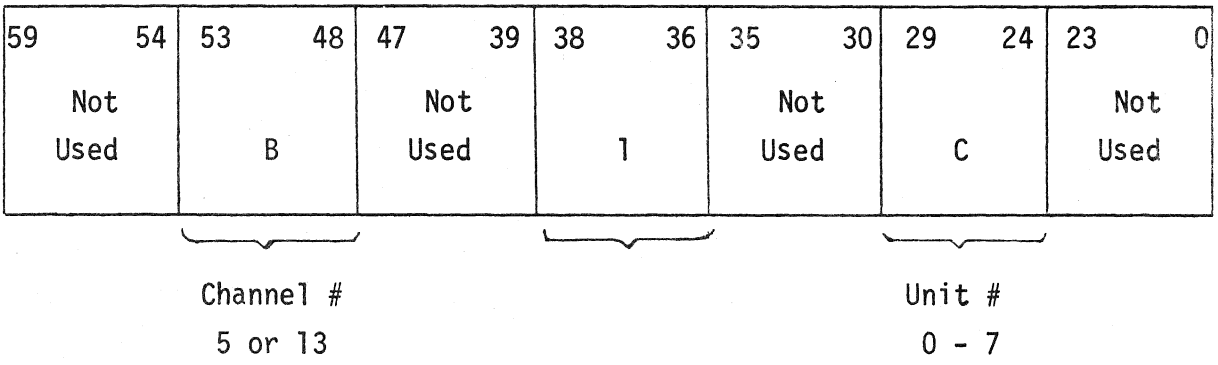
---

*Word numbers are given in octal.

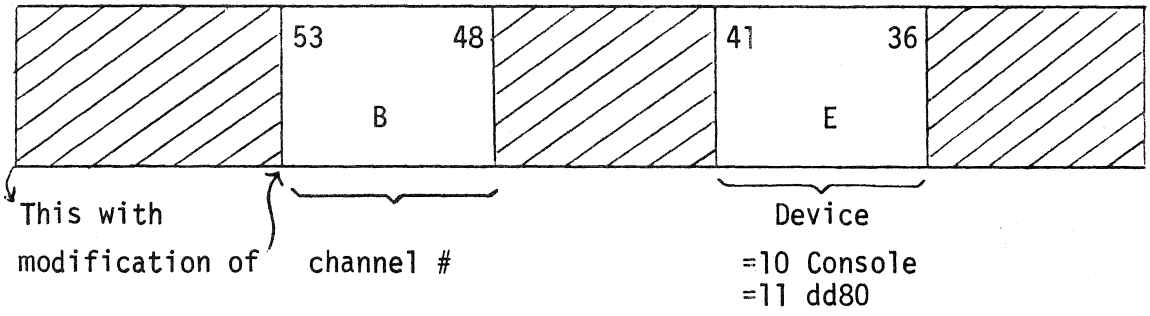
Disc



Tape



dd80 or Console



For tape IOD's,

-A- is unused.
-B- specifies the channel number.
-C- specifies the unit number (0-7).
-D- is unused.

For DD80,

-B- specifies the channel number.
-E- is the number "11" (octal).

For console,

-B- specifies the channel number.
-E- is the number "10" (octal).

Each exchange-package area is eighteen words long; the last sixteen of these contain the exchange-jump package, as described in Reference 11. The exchange package is illustrated in Figure 4.8, and is defined below.

-XJA- is a pointer to either the next exchange package to be executed or to the next free-exchange-package area, depending on whether this one is free or in use.

-IOST- indicates, if on, that this exchange package is I/O stuck.

-GBST- indicates, if on, that this exchange package is GOB-stuck. (It wishes to make a system call, but is prohibited from doing so until I/O has completed.)

-DISS- indicates, if on, that this exchange package is disabled. Usually, an exchange package for an I/O interrupt routine with the disabled bit on says, "do not interrupt this exchange package with a new one." If the bit is off, then a new exchange package (usually the result of another completed I/O operation) can interrupt this one.

-C- is the IOD number (1-17 octal), if any, associated with this exchange package (again apt to be an I/O interrupt exchange package). This IOD number is taken from the C field in Word 1 of an IOD.

-PP word 0-	is the contents of word zero of the problem program the last time it was executing.
-P-	is the contents of the P-counter.
-RA-	is the reference address.
-FL-	is the field length.
-EM-	is the exit mode.
-A0,...,A7-	are the contents of the A-registers.
-B1,...,B7-	are the contents of the B-registers.
-X0,...,X7-	are the contents of the X-registers.

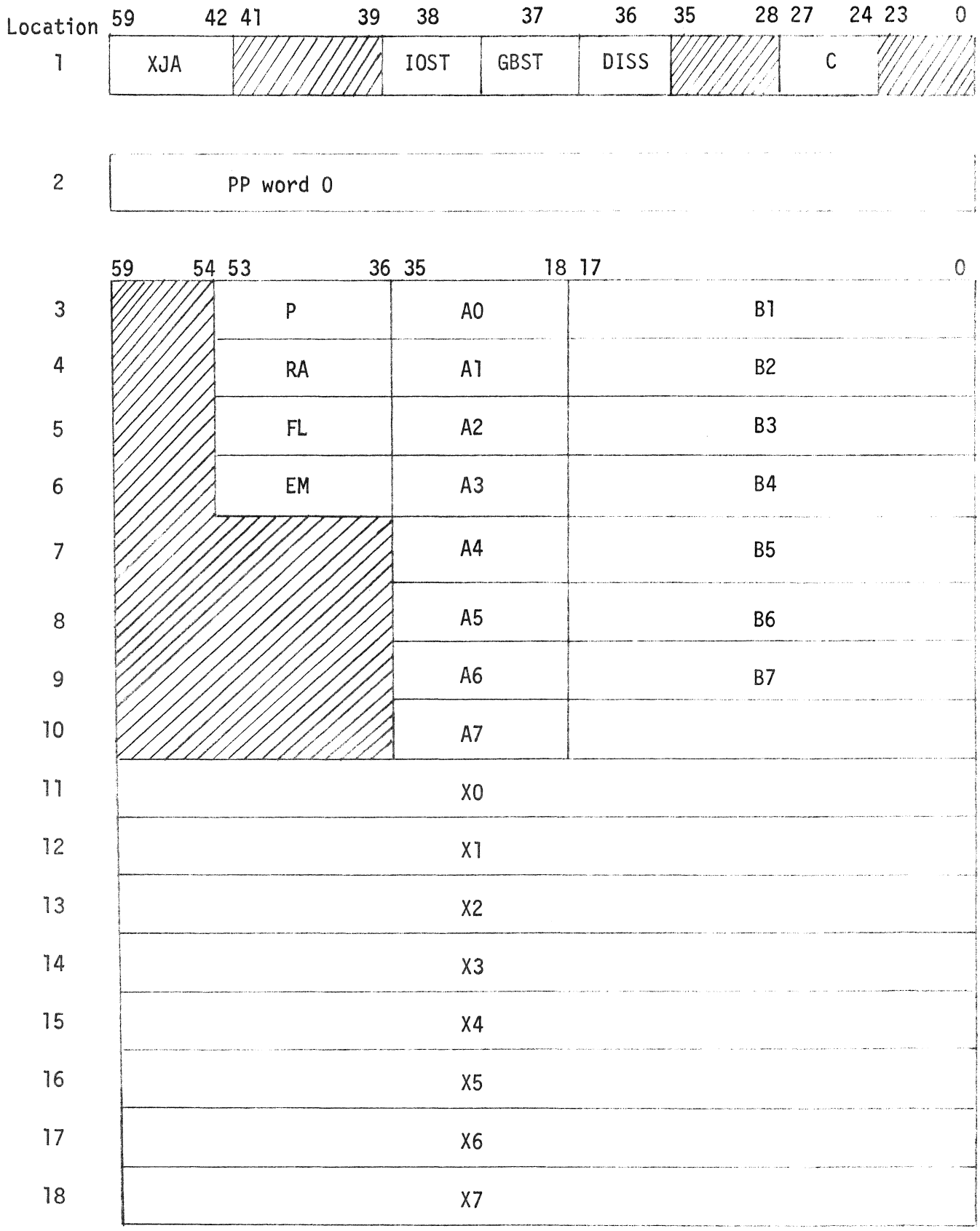


Figure 4.8. CDC 6600 Exchange Package

4.2.8. Data-File Formats

Four data-file formats are used in the CDC 6600 computer system: *ASCII Squoze*, *Monitor*, *Monitor Squoze*, and *Absolute*. These may be generated by a program or by reading cards through the card reader.[9]

Trailing blanks are eliminated in the ASCII Squoze and Monitor Squoze types. Both Monitor types have the same form of sentinel word, but the Monitor type has no characters eliminated from the BCD card image. The Absolute type produces no conversion whatsoever. In all four forms, all eighty columns are read.

The four formats may be illustrated by examining the following card:

```
col. 1     7              21                  80
     LABEL_FORCE(X)=_2_.35_____________________
```

1) ASCII Squoze File Format. The card will be translated to the following ASCII Six-bit BCD form.

Word	Octal	
1	5441424554	0046576243
2	4510701135	0022001623
3	2501010101	0101010101

Note that "01" is used as a sentinel for "end of record." The next card will go into the next word. The end of the file is signalled by the word

0001000000 0000000000.

2) Monitor-File Format. The card will be translated as indicated, and will be preceded and followed by a record sentinel word. This has the form

000000ZZYY YYYYXXXXXX

where

YYYYYY is the length of the preceding record.
XXXXXX is the length of the following record.

ZZ is the mode of the following record.
ZZ = 00 for BCD.
ZZ = 01 for binary.

Word	Octal
1 (sent.)	0000000000 0000000010
2	5441424554 0046576243
3	4510701135 0022001623
4	2500000000 0000000000
5-9	0000000000 0000000000
10 (sent.)	0000000000 0010000010

The-end-of-file sentinel is preceded by a record-sentinel word with XX = 0. The end-of-file has YY = 0 and ZZ = 0.

3) Monitor-Squoze-File Format. The form is similar to the Monitor format. The same record sentinel and end-of-file format is used. However, trailing blanks are suppressed.

Word	Octal
1 (sent.)	0000000000 0000000010
2	5441424554 0046576243
3	4510701135 0022001623
4	2500000000 0000000000
5 (sent.)	0000000000 0010000010

4) Absolute-File Format. Each column of the card is translated to a twelve-bit byte. There is no record sentinel. The end-of-file sentinel is the same as the ASCII squoze end-of-file.

Word	Octal				
	L	A	B	E	L
1	2100	4400	4200	4020	2100
		F	0	R	C
2	0000	4040	2010	2001	4100
	E	(	X	)	=
3	4020	1042	1004	4042	0102
		2		.	3
4	0000	0002	0000	4102	0003
	5				
5	0005	0000	0000	0000	0000
6-16	0000	0000	0000	0000	0000

The ASCII-Squoze form cannot handle binary cards. The other three forms can, and binary and BCD cards may be intermixed. Binary cards in Monitor and Monitor squoze files are in the same format as in absolute files.

Detailed discussions of the file format used by the various I/O equipment are available in other chapters. (See reference 9 for card reader and card punch, and reference 11 for printers and dd80.) The particular format used by the various system and utility programs is mentioned with the program write-ups.

Section 4.3: THE CDC 7600 FILE-MAINTENANCE AND TRANSPORT SYSTEM†

4.3.1. The CDC 7638 Disc System

The CDC 7638 Disc-File Subsystem[10] consists of two control units and one CDC 817 Disc File. Each CDC 7600 computer system at LRL has two such subsystems attached. The disc file is organized into two units, each of which is controlled by one control unit. Two 7600 PPU's can operate each control unit on a time-shared basis. Each PPU interface contains two twelve-bit input/output channels (one for data and one for control) to operate the system.

The disc file consists of two double-ended disc drive spindles (stacks) which rotate asynchronously at 1,800 rpm. Each spindle is divided into two half-spindles (upper and lower) containing 18 discs (32 recording surfaces). There are two hydraulic positioning access assemblies (upper and lower), each one servicing a pair of half-spindles. The upper assembly is denoted as unit 0 and the lower as unit 1. Each access assembly contains two separate horizontally opposed groups of 16 head arms, one group for each half-spindle. Mounted on the end of each arm are two head pads, each of which contains one read/write head.

Thus, there is one head for each of the 32 surfaces on a half-spindle. Each unit is divided into two head-groups of 16 heads each on each stack. Each access assembly can be moved to any of 512 data positions.

Data is transferred in twelve-bit bytes between the PPU and the control unit and in sixteen-bit bytes between the control unit and the storage units. The data is recorded on 16 parallel tracks. Each track is divided into sectors. Each sector is divided into a preamble section, a data section, and a postamble section. The controller generates a sixteen-bit cyclic parity word for each sector.

---

†For a complete description of ELF, please see reference 2 and also the Appendix.

The PPU must write the preamble followed by the data. The controller writes the postamble and the cyclic-parity word. Neither the preamble nor the postamble are transmitted to the PPU on a read operation. The controller regenerates and compares the cyclic-parity word on a read operation and sets the parity-error status bit if they do not check.

STORAGE UNIT TIMING

Latency	35.5	ms
Recovery Time (write to read)	60	µs
Disc Positioning Times: (approximate)		
Head-group switching	60	µs (average in same stack)
Stack for Stack	20	ms
Adjacent Position	30	ms
Random Average	85	ms
Max Position Movement	145	ms

Disc Storage Capacity: A disc-file-sector record contains 516 sixty-bit words (a four-word header and 512 words of data). There are 40 sectors per head-group per position. At LRL we have defined the two halves of a stack, i.e., one spindle, as one *logical unit*. There is a separate disc map for each of the four logical units comprising the two disc subsystems on each CDC 7600. Data is stored in sequential sectors using one head-group and then the other for a given position on Unit 0, then, in sequential sectors using each of the head-groups for the same position on Unit 1, then, repeating the procedure for the next position, and so on. In this way, head positioning on one unit can be overlapped by data transmission on the other unit.

The table in Figure 4.9 lists some of the disc capacity statistics in usable sixty-bit words. Function codes and the status reply word are summarized in Figure 4.10.

	Sixty-bit Words
Words/Sector	512
Sectors/Track	40
Words/Track	20,480
Head-groups/Position/Unit	2
Words/Position/Unit	40,960
Units/Stack	2
Words/Position/Stack	81,920
Positions/Stack	512
Words/Stack	41,943,040 (one logical unit)
Stacks/Disc	2
Words/Disc	83,886,080 (two logical units)
Discs/System	2
Words/System	167,772,160 (four logical units)

Figure 4.9. CDC 7638 Disc Statistics

Function Codes

	11 10 9	8 7 6	5 4 3	2 1 0
Position Select	1 0 X	X X X	X X X	X X Y
Head-group Select	0 1 0	- - -	- - -	- - X
Write Select	0 0 1	- - -	X X X	X X X
Read Select	0 0 0	- - A	X X X	X X X

(-) bit not examined

(X) bit can be 0 or 1

(Y) stack 0 or 1

(A) if 1, means read next sector

STATUS REPLY WORD

11 6	5 4	3	2	1	0
S S S S S S	--	C	A	P	R

S - sector count + 1 of sector under heads

C=1 means not on cylinder

A=1 means abnormal condition exists

P=1 means parity error

R=1 means not ready

Figure 4.10. CDC 7638 Function Codes and Status Reply Word

4.3.2. The CDC 7600 Filing Subsystem

Each person capable of accessing a 7600 through the FLOE operating system must have an entry in the *system user directory*. This is a profile of the user, containing such information as the user identification, user security combination, user's allowed charge time, a pointer to the user's file index, etc. An entry in this table does not, however, assure the user of drum space to catalog his files - no drum space is allocated until it is required.

A list of files having no identification number is kept in large core at all times. These are known as *public files* and are accessible by all users, usually as read/execute files. They are, largely, code files of a general utility nature.

Included in the list of files having no user number is a subset known as *political files*. Within this subset are many smaller sets, each having a boundary representing limits of access to the files in the group. Currently, LRL division codes are used to bound groups of files. Any user in a division may access any file within that user's domain. These files are not accessible to users outside the boundary.

A master tape of the public files is maintained by the Systems Operation Section. The disc images of public files are updated as required by S.O.S. The political files are copied from disc to tape once each day and saved for a short time to allow restoration after a disc failure. Both public and political files are considered to have infinite lifetimes and the operating system will not exercise the file-purge routine against either of these two file types.

There are three identification numbers which are exceptions. These numbers are recognized by the system as *privileged users*. These users perform tasks of benefit to the system, such as 999999 (called User-1). As such, these numbers are considered to always be active, and their file catalogs are always available in large core. Their private files are not generally subject to purge.

Disc files are cataloged by name and user identification number in a *file-index table*. Files are recognized as being active or inactive according to these criteria:

1) User is logged into the system through some terminal - *active*.
2) User is not logged in, but has logged out while a job of his is executing or is queued for execution - *active*.
3) User is logged out and has no job entered into the system - *inactive*.

If a user is determined to be active, his file-index entries are maintained in a table in the large core-store for ready access. If a user is known to be inactive, his file index is maintained on the system drum in a list of linked blocks of 21 file indices each.

Current definitions of the file-index table as designed for the drum subsystem are:

1) Large-core file-index table: 8448 words
   6 words/entry
   1399 active files
   No limit on the number of files one user may own.

2) Drum-file index: 256,000 words
   42,000 files cataloged
   128 words/sector
   21 indices/sector
   No limit on the number of sectors a user may own.

These definitions are readily adjustable as statistics indicate other requirements.

At the time a user logs into the system, his file-index entries are read from drum to a large-core-memory buffer, and from there are scattered through the file-index table according to a hashing technique which attempts to minimize collisions and optimize table fill. The file-index entries for a particular user are chained together throughout the table for rapid access. A pointer to the head of the file chain for each user is kept in a *user activity table*.

At the time a user logs out and has no job active within the system, his chain of file-index entries is gathered into a large-core memory buffer and, from there, written into drum-file-index blocks with appropriate linkage.

4.3.3. Disc Mapping

Each logical disc unit is mapped independently, and no file may be split among units. Further, a file is assigned as many contiguous sectors of disc as are required to contain it. No means exist for linking separated blocks of disc. This is done to minimize accesses and head motion, even while recognizing that some optimization of assignment of available disc space is lost. Though the time-shared system is also recognized to be a resource-shared system, the burden of effort is placed on minimizing time delays at the expense of some lost resource availability.

A disc-unit map consists of a word-per-entry table which is always resident in the large-core store. An entry is made in the table for each filled block on the disc. A filled block may contain many distinct files of various lengths. The information regarding location and length of each file is kept in the file-index table. An empty block is simply a number of contiguous sectors of disc available for total or partial assignment. An empty block is implied by a discontinuity in addresses between two consecutive map entries. A disc-map entry consists of the logical disc-unit number, first-word address of the block and the number of sectors in that block.

Note that although there are four logical disc units, there are only two channels for disc input/output. The operating system assigns disc file space one file at a time on alternating channels and then by alternating units. The logical units 1 and 2 are on channel 1, the logical units 3 and 4 are on channel 2. The assignment sequence by logical unit number is 1, 3, 2, 4, 1, 3, 2, 4, etc. A user program may specify the unit for his file or accept the system schedule and may take advantage of the assignment alternation to optimize disc input/output.

4.3.4. Disc Flaws

As delivered from the factory, CDC 817 disc files contain some number of known surface flaws. As the file unit is connected to the operating system for initial debugging, more permanent bad spots are discovered. Each such flaw requires the removal of the sector containing it from assignable space. The FLOE operating system contains a FLAW TABLE in which these permanently unassignable sectors are recorded. It is easily possible for flaws to be so positioned on a disc as to allow for the assignment of no single large block of space.

4.3.5. File Retrieval From Long-Term Mass-Storage Devices

The FLOE operating system, in conjunction with the aforementioned privileged user, and the PDP-6 ELEPHANT System, can initiate transportation of a file of information from its own disc memory to mass storage (the IBM photo store or data cell) or retrieve a file from such a medium. These jobs are requested specifically by a user at a teletype console. Included in the overall capability of this file-transport system is the ability for a user to send a copy of a file from one 7600 to another.

Ultimately, those files in the public domain (that is, those of general utility to all users) will reside permanently in long-term storage and be retrieved automatically by the 7600 operating system upon demand.

4.3.6. File Types

Five types of files are cataloged by the FLOE operating system. Any of the types may appear in any of the three domains: private, political, or public.

Type one files are considered to have *read/write access*. They may be read from, written into, or executed at will.

Type two files are considered to have *read only access*. They may be read from or executed at will, but cannot be written into.

Type three files are considered to be *data files*. These may be read or written, but not executed. Further, they are exempt from the file purge so long as the owning user is active within the system.

Type four files are considered to be *execute/only files*. They may not be read or written but may be executed. These are usually public utility-code files updated only through a system procedure.

Type five files are considered to be *qualified execute/only* files. The qualification is the ability of the operating system to write into them. These are, unconditionally, program drop files which have been created by the executing program itself.

4.3.7. File Purging

In order to maintain reasonable space in the file-index catalogs, unused files are purged from the systems memory after some lifetime. In every case, the lifetime is restarted by simply referencing the file. The lifetimes differ by file type and are so constantly being modified as to defy definition.

Within the operating system, the file purge routine is executed once each 15 minutes for all files in the active file index and once each two hours for the drum file index. Additionally, a purge scan is made at log-in time and log-off time as a user's file-index entries pass through the drum-core buffer.

4.3.8. Program Minus Words

The term "minus words" refers to a block of 152 words that are added to every problem program preceding the program's relative word zero. These words are not accessible to the program. All 152 minus words are kept in LCM; the last 29 are also loaded into SCM with the program.

Words 0- 3*	contain a system IOD.
Words 4- 24	contain the number of unprocessed IOD's.
Words 25- 44	contain the number of issued IOD's and the number of C-field IOD's.
Words 45-124	contain sixteen input-output connectors, three words per IOC. (See Figure 4.11).
Words 125-134	contain an exchange-jump package pool.
Words 135-155	contain the word zero first interrupt.
Word 156	contains the problem-program fatal error in the high order 12 bits, and the location of the program at the time the error was detected in the low order 48 bits.
Word 157	contains the problem-program input/output charge, given in number of machine cycles.
Word 160	contains the number of input/output cycles used.
Word 161	contains the amount of PPU input/output time charged to the program, given in integer microseconds.
Word 162	contains the total amount of SCM time charged, given in microseconds.
Word 163	contains a "trusted-program" flag.
Word 164	contains the classification of the program.
Word 165	unused.
Word 166	I/O accounting flag.
Words 167-174	are unused by by FLOE.
Word 175	(word 2 of the SCM portion of the minus words) contains the number of load-block copies used in this LCM slot.

*Word numbers are given in octal.

Word 176 contains the LCM slot time.

Word 177 contains the accumulating system call time.

Word 200 contains the accumulating LCM time.

Word 201 contains the SCM slot time.

Word 202 contains the accumulating system call time.

Word 203 contains the accumulating SCM time.

Word 204 contains a "static-memory" flag.

A problem program may be organized such that a portion of its image may be unchanged by computation. This read-only portion does not need to be copied from SCM to LCM, and therefore saves block-copy time. The minus word format is

OOOOX XXXXX OOOOY YYYYY

where

-X- is the beginning address of the read-only portion.

-Y- is the beginning address of the second section of the read/write portion.

Both X and Y are checked for bounds fault on an end-of-load from disc; if either is out-of-bounds, a fatal error 231 results. If X=Y=0, then the entire program image is copied from SCM to LCM.

Word 205 contains the date last accounting.

Words 206-207 contain the "seventeenth" and "eighteenth" words.

Words 210-227 contain the executing exchange package. The format of this package is given in reference 1.

An input/output connector consists of three words, as illustrated in Figure 4.11. The ten fields in these words are described here.

-A- is the (right-adjusted) name, for tape and disc IOD's. The name may have up to ten characters for a disc file, and up to six characters for a tape.

-B- is the first-word address of a disc file. (Includes the 512 word ID sector.) It is blank for nondisc IOD's.

-C- is the size of a disc file, in integer number of words. (Includes 512 words for the ID sector.) It is blank for nondisc IOD's.

-D- is the public-file bit.

D = 1 if the file is a public or political file.

D = 0 if the file is a private file.

-E- is the logical unit number, as given by the following table.

Device	EEEEE
Disc	00001
	00010
	00011
	00100
Drum	01001
Tape	01100
	01110
	01111
Card Reader	10100
Printer	10101

-F- is the type of unit, as given by the following table.

Device	FFFFFF
Disc	000000
Tape	000001
Drum	000111

-G- is the tape unit, in the form XXXNNN, where

-XXX- is tape bank 0 or 1.

-NNN- is tape number 0 - 7.

It is unused for nontape IOD's.

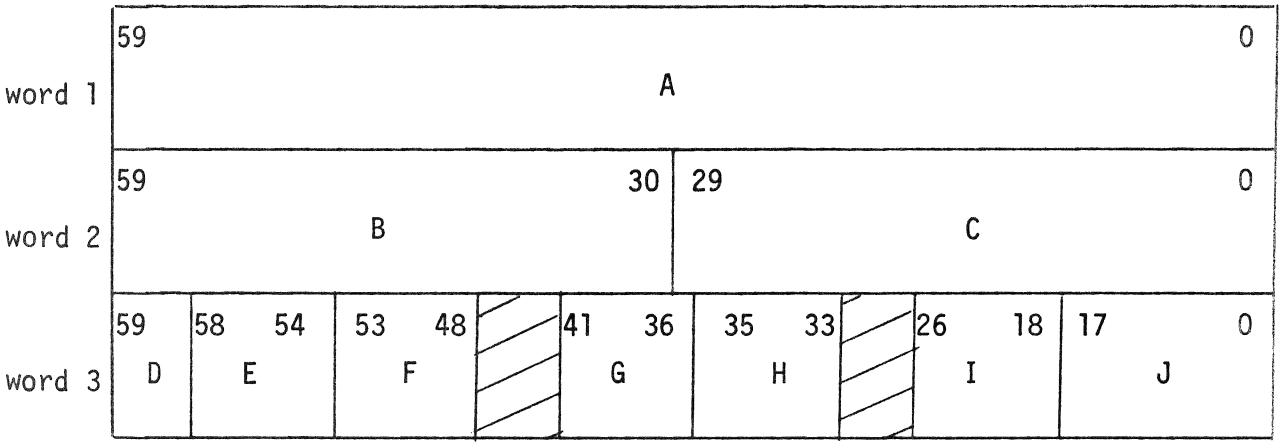


Figure 4.11. Input/Output Connector

-H- indicates the type of disc file. It is used only for disc IOD's.

H = 0001 for a read/write file.
H = 0010 for a read only file.
H = 0011 for a read/write only file (data file).
H = 0100 for an execute only file.
H = 0101 for a qualified execute/only file.

-I- is a descriptor-block pointer.

-J- is a count of I/O references. Each input or output request adds one to this field.

An IOC package is cleared by the system on a close or destroy system call.

APPENDIX
Utility Routine ELF

(Extracted from UR-202)

General Description

ELF relays messages between the ELEPHANT file transport system on the PDP-10 and a Teletype (or other controller) on a CDC 7600 or CDC 6600. ELF accepts from the TTY (or controller) any sequence of valid requests for jobs to be done by the ELEPHANT system; sends them on, one at a time, to the PDP-10; receives replies back from the PDP-10; and relays the replies to the TTY (or controller). Jobs may involve transporting files between worker computers, transporting files between your worker computer and the long-term storage media connected to the ELEPHANT system, or the manipulation of files and directories in the system.

Generally, a new request may not be sent to the PDP-10 until the job initiated by the last request has been completed. However, ELF does offer a stacking facility which permits you to issue job requests as rapidly as desired. Each request is saved by ELF until the request's turn to be relayed to the PDP-10 occurs. (However, two exceptions are requests RPT (report) and ABT (abort) which are relayed immediately.) Another feature in ELF allows it to respond to LST (list directory) with the listing of the entire directory no matter how long it is. Also, ELF contains four special macro-requests which allow a group of files to be destroyed or transported.

This write-up describes in detail only the ELF routine. It does not give details as to the operations, request language, message interpretation, or file-directory system of the ELEPHANT system on the PDP-10. These details are in LTSS Chapter 4. However, a summary of the request language forms available through ELF and the ELEPHANT filing system are given in Table 1.

Definition of Terms

request(s) A request for ELF to perform or initiate some action. It may be (1) a job request to be sent directly to the PDP-10; (2) NEXT; or (3) one of the macro-requests starting with RDSALL,

WRSALL, WRCALL, or DSTALL. See LTSS Chapter 4 for detailed information concerning *requests* which are sent directly to the PDP-10.

reply — A message returned by the PDP-10 system through ELF or by ELF itself in response to a *request*. The *reply* consists of the opcode which was sent by the user, preceded by any information requested or error comments. The exception is the *reply* to ABT which ends with RPT rather than ABT.

Dname — The chain name of a directory in the ELEPHANT storage system.

Life — The desired lifetime of a file determines the storage medium to be used. Use either for the Data Cell or for the photostore system.

Comp — The letter designator of a worker computer (e.g., R, S, M).

Options Available

NEXT — Indicates to ELF that the job running is to be aborted, and that the next request in the stack is then to be initiated. The *reply* returned will end with RPT.

END — Indicates to ELF that it is to terminate itself as soon as all requests made so far have been fulfilled or an error is detected. A message to ELF consisting of only a "linefeed" is equivalent to the END option.

DSTALL *Dname* — Indicates that all files in the directory *Dname* are to be destroyed. If *Dname* contains any directories, then these contained directories and their contents are not disturbed.

RDSALL *Dname* — Indicates that all files in directory *Dname* are to be read from ELEPHANT storage into the worker computer being used. Directories in *Dname* will be ignored.

WRSALL *Life Dname* Indicates that all of one's private files in the worker computer being used are to be written into ELEPHANT storage in directory *Dname* with the indicated *Life*.

WRCALL *Comp* Indicates that all of one's private files on the worker computer being used are to be written into the worker computer indicated by *Comp*.

Summary of Usage Forms

Form 1. Single Job Request

```
User:      ELF request / t v
Routine:   reply
           ALL DONE
```

Form 2. Multiple Job Requests

```
User:      ELF / t v
Routine:   •
User:      request
Routine:   •reply
User:      request
Routine:   •reply
   .
   .
   .
User:      request
Routine:   •reply
User:      END
        or
           (linefeed)
Routine:   ALL DONE
```

Usage Form 1: Single Job Request

Use this form to issue a single job request to the ELEPHANT system. After the execute line is typed, ELF replies by relaying the message sent

back by the PDP-10 and then terminates. While waiting for the message from the PDP-10, either RPT or ABT may be typed if desired. If any *request* other than RPT or ABT is typed during this period, it will be ignored.

Usage Form 2: Multiple Job Requests

Use this form if more than one job is to be requested. Two methods of operation are available with Form 2: method (1) is to wait for the *reply* to the last *request* before typing another request, method (2) is to use the stacking facility. There are no restrictions for going back and forth between these methods while using ELF.

REFERENCES

1. J.G. Fletcher and J.D. Lawrence, *Livermore Time-Sharing System: Chapter 1, Hardware*, Lawrence Radiation Laboratory, Livermore, California, Computer Information Center, CIC LTSS-1 (March 10, 1970).

2. J. Engle, *Utility Routine ELF*, Lawrence Radiation Laboratory, Livermore, California, Computer Information Center, CIC UR-202 (June 1, 1970).

3. *Librafile 4800: Mass Memory Operation and Maintenance Instructions, Volume I*, General Precision Systems, Inc., Librascope Group, Glendale, California (September 1, 1967).

4. *IBM 2321 Data Cell Drive: Original Equipment Manufacturers' Information*, International Business Machines Corp., San Jose, California, IBM Product Pub. Dept., Publication No. A26-3574-0 (May, 1965).

5. *IBM 1360 Photo-Digital Storage System: System Summary*, International Business Machines Corp., San Jose, California, IBM Systems Dev. Division Publication (no number or date).

6. G. Boer, *Some Remarks About a Photo-Digital Storage Device*, Lawrence Radiation Laboratory, Livermore, California, UCRL-71859 (July 17, 1969).

7. *Control Data 6000 Series Computer Systems: Peripheral Equipment Reference Manual*, Control Data Corporation, St. Paul, Minn., Tech. Pub. Dept., Publication No. 60156100 (rev. January 6, 1969).

8. P. Du Bois and J.D. Lawrence, *Livermore Time-Sharing System: Chapter 9, 6600 System Calls and I/O Requests*, Lawrence Radiation Laboratory, Livermore, California, Computer Information Center, CIC LTSS-9 (May 27, 1969).

9. J.D. Lawrence, *Livermore Time-Sharing System: Chapter 5, Card Reader/ Card Punch*, Lawrence Radiation Laboratory, Livermore, California, Computer Information Center, CIC LTSS-5 (March 12, 1970).

10. *Control Data 7638 Disc Storage Subsystem*, Control Data Corporation, St. Paul, Minn., CDC Publication No. 6026550 (December 30, 1969).

11. J.D. Lawrence, P.E. Lund and L.O. Tennant, *Livermore Time-Sharing System: Chapter 7, Visual Recording Equipment*, Lawrence Radiation Laboratory, Livermore, California, Computer Information Center, CIC LTSS-7, Ed. 1 (in preparation).

INDEX

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

102664992