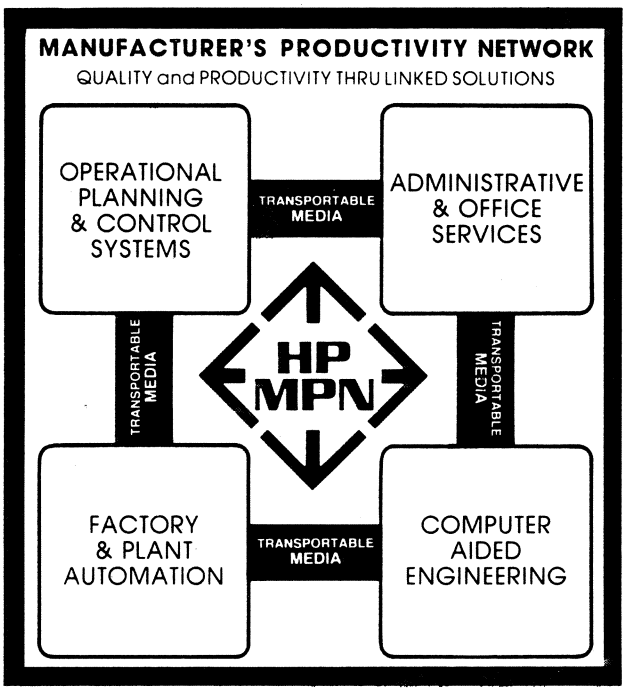
LIF

Logical Interchange Format

Directory Organization & Record Format for Data Interchange



HEWLETT PACKARD

HEWLETT PACKARD

COMPUTER PERIPHERALS GROUP • Post Office Box 39, Boise, Idaho 83707, Telephone 208 376-6000

February 15, 1982

Data compatibility has evolved into a significant issue in the marketplace today. Our major accounts (who are our prime target customers) are demanding the ability to move data across the various HP systems that they own.

A task force has been working on this issue for some time. They have come up with a proposal for a Logical Interchange Format (LIF) to be implemented on all Hewlett-Packard computational systems. The advantage to the driver designers is that this format is media independent and applies equally well to 5¼'' or 8'' flexible discs or ¼'' cartridge tapes.

We have asked Greeley to be the "keeper" of the standard and maintain it, while providing the tools for its implementation. The accompanying manual is the first major tool for all of our driver designers.

Please review this document and take all necessary steps to ensure that our computational systems support this format on a broad basis. Contact Gordon Nuttall at Greeley Division for any necessary clarifications.

Doug Chance
Dick Hackborn
Ed Mc Cracken
Dick Moore
Cyril Yansouni

dhmisc,170

Table of Contents

Chapter 1

The LIF Solution 1
The Problem 1
The Current Status 2
Greeley Division Commitments 2
Host Computer Division Responsibilities 2

Chapter 2

LIF and Compatibility 3
Three Levels of Compatibility 3
Existing LIF Systems 3
Implementation Suggestions 4
Levels of Compliance 5
Minimum Implementation 5
Complete Implementation 6
Sample Catalogs 7
Features and Restrictions 8
Features 8
Restrictions 8

Chapter 3

LIF Implementation 9
Definitions 9
Limitations and References 10
Logical Media Layout 10
Volume Label 11
Directory 12
File Structure 14
Extensions 15
Conformance Verification 16
Support 16
Appendix A 17
Appendix B 18
Appendix C 19
Subject Index 21

Chapter 1
The LIF Solution

Hewlett Packard's new computer strategy, "Manufacturer's Productivity Network" (MPN) "...enables computers to work together, transferring information between application areas quickly and easily..." -Paul Ely.

Transportable media is a very important data link between different HP systems. This is graphically illustrated on the front cover. Transportable media involves written data using a common format. A format each host computer initiates as its operating system or as an additional command to its own mass storage operating system. LIF is a Logical Interchange Format that assures interchangeable data across different operating systems.

The Problem

Hewlett Packard loses an estimated $175 million in computer sales each year because our products are not compatible. Our salespeople spend countless hours trying to explain to customers why two different HP computers cannot read the same disc from the same disc drive. It's embarrassing for the salesperson and for the company as a whole.

Our customers look upon Hewlett Packard as one company and not a collection of divisions. They have the right to expect a unified product spectrum from HP with products that are compatible and that can easily exchange data with each other.

If we can't provide customers with compatible equipment, our competitors will.

HP is in a unique position to provide key accounts with a wide range of computer equipment from hand held calculators to complete business systems. Data transportability across this product spectrum makes it attractive for key accounts to purchase HP equipment solely.

Compatibility will not decrease system performance. In fact, additional performance is gained by being able to upgrade existing data to new state-of-the-art systems.

Divisions can implement any mass storage operating system they choose, to optimize system performance. Data exchange can be accomplished via the initiation of one command within the operating system. THE IMPORTANT CHOICE HERE IS THE DECISION TO IMPLEMENT LIF.

The Current Status

Data exchange is supported by management at all levels. The present data exchange efforts and present situation with each division has been reviewed by corporate officers from John Young, Paul Ely and Ed McCracken on down. They enthusiastically support the LIF standard and are supporting the efforts of this document.

The Corporate Solution

1. All HP computers released after 1/82 should have BUILT-IN data compatibility using the Logical Interchange Format (LIF) for all forms of removable media (i.e. no extra binaries or utilities).
2. All existing computers without LIF should be provided with utility programs initiating LIF.
3. The IBM 3740 logical format should be reserved for data interchange with non-HP computers, NOT for internal HP interchange.

Greeley Division Commitments

Greeley Division has taken the responsibility of being the focal point for the LIF effort. We see LIF as a crucial part of making HP a dominant force in the computer business, so we are pushing the idea of data interchangeability across HP computing systems. GLD has assumed the following responsibilities:

- Initiate physical compatibility of address marks, physical sector lengths, etc. on the peripherals we produce.
- Provide media compatibility by selecting appropriate vendors and continued inspection.
- Provide a contact person responsible for all questions concerning the logical format.
- Maintenance of the LIF document and distribution list.
- And we'll provide any conformance verification required to insure data compatibility.

Host Computer Division Responsibilities

What is required from the Host Computer Divisions is the initiation of LIF compatible commands in addition to the mass storage operating systems. So, the message we would like you to get is ... "Think LIF when generating the ERSs for your mass storage operating systems." Together, we can provide one key function to making HP systems the Preferred Computer Systems.

Chapter 2
LIF and Compatibility

Three Levels of Compatibility

Several levels of compatibility must be obtained before information exchange between systems can be realized. The first level is the media itself. You can fold an 8" flexible disc and stuff it in the opening of a 5 1/4" disc drive, but you can't read the data. Likewise, tape cartridges present similar problems. The first level is to have media compatibility. This is obtained via similar drives built in or interfaced to the variety of HP's host computers.

The second level is physical compatibility of address marks, physical sector lengths, etc., written on the medium. Physical compatibility is the responsibility of the peripheral divisions as well as the design and production of state-of-the-art mass storage devices. Much progress has been made in these areas partially due to the separation of the peripherals divisions. Now mass storage devices specify and directly implement physical record schemes.

The third level of compatibility, and the one which is the subject of this standard, is logical compatibility. Logical compatibility is the responsibility of the host computer. It cannot realistically be performed by the peripheral.

A format which specifies logical compatibility allows you to transfer data between different HP computers with a minimal amount of interaction. Two systems that do allow this to happen are the 9826 Desktop Computer and the 2642 Terminal supporting the 13272A Disc Drive. The data communication link between these two systems is the 5 1/4" flexible disc from their mass storage devices. Both systems use complete implementation of the file manager described in this document. Complete and partial implementation are discussed at the latter part of Chapter 2.

Existing LIF Systems

This idea of data interchangeability across systems has been developing for approximately five years and the systems that currently support LIF and have data interchangeability thru transportable media are listed next:

Data Compatibility Using 8" Media	Data Compatibility Using 5¼" Media
HP Series 80 Computers/9895 Disc Drives	HP Series 80 Computers/8290X Disc Drives
HP 250	2642 Terminal/13272A Disc Drive

HP 300	HP 9826 — and with the 8290X
HP 9826/9895 Disc Drives	HP 9836 — and with the 8290X
1000 Model 5/9895 Disc Drives	1000 Model 5

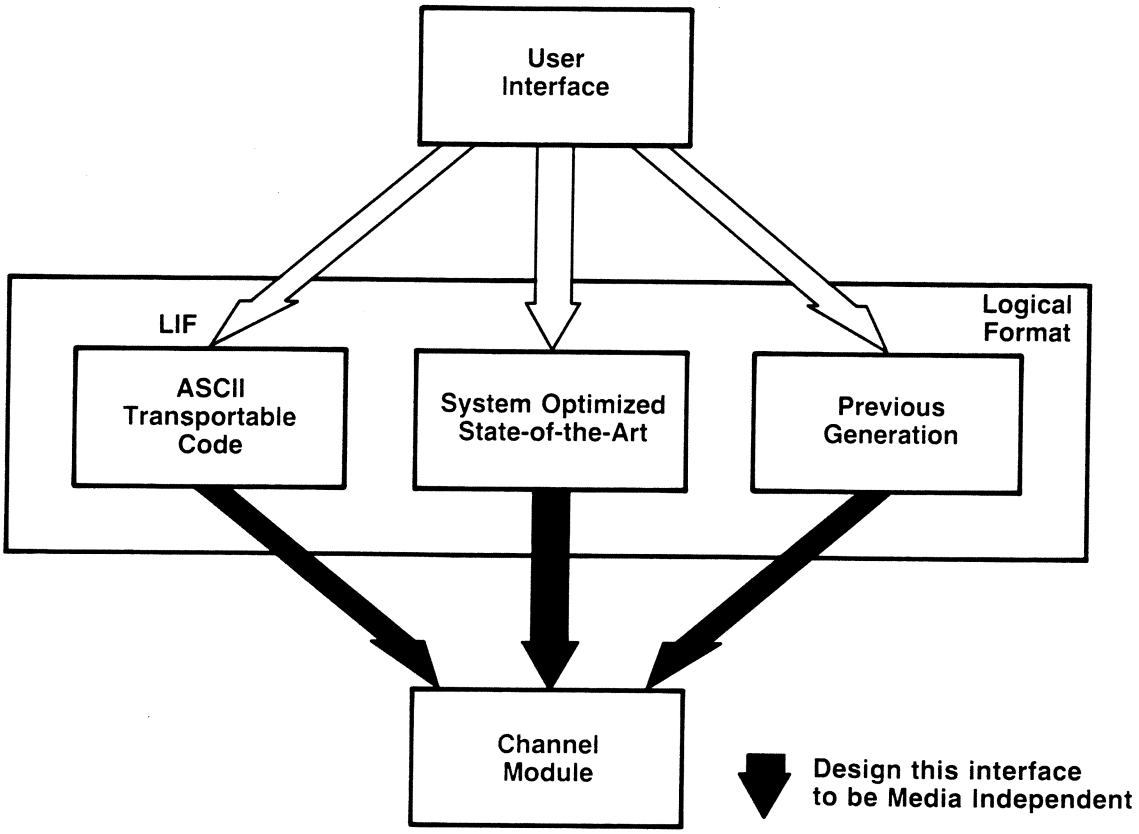
Data Compatibility Using the 9134A/9135A	Data Compatibility Using the 9140 Tape Cartridge Drive (LINUS)
HP Series 80 Computers/ 9134A and 9135A Drives	HP 250/9140 Drive
HP 9826/9134A and 9835A Drives	1000/9140 Drive
1000 Model 5/9134A and 9135A Drives	3000/9140 Drive

Implementation Suggestions

The following figure is a block diagram representation of a file manager. This shows our recommendation of LIF, the ASCII Transportable Code, implementation. The file manager or mass storage operating system shown here is divided into three subsections: the User Interface, Logical Format section, and the Channel Module. As an ideal implementation, the interface between the Logical Formats and the Channel Module should be designed to be media independent.

The outcome of this design is that the User Interface and Logical Format sections will readily interface to a number of different Channel Modules. Channel Modules can and will always assume a variety of different parameters: interface protocols (Amigo, CS/80), interface types (HP-IB, 8-bit parallel, etc.), and unique device requirements (the difference between a tape drive and a hard disc drive). Only in a case where new hardware design necessitates rewriting the code, should the User Interface and Logical Format be rewritten. In the case of using different interfaces, different protocols, or different mass storage devices, a Channel Module dedicated to a particular situation must always be developed.

As you can see, any system optmized code or any previous generation code (downwards compatible for the sake of your customer base) you wish to implement is still left to your discretion.



Incidentally, the sooner LIF is adopted in your host's file management system the sooner the downwards compatible problem concerning data goes away.

Levels Of Compliance

This section discusses the minumum and maximum implementations that are required to be LIF compatible. The minimum level is the ASCII Transportable Code section. Refer to the previous figure. A maximum implementation includes the ASCII Transportable Code as well as the use of the LIF format as the default format (used for your language programs and binaries).

Minimum Implementation

The ASCII Transportable Code section would be the initiation of a "Translate" or "Copy" function. This function would emulate the the follwing steps:

- Format a blank medium with the LIF directory.
- Write
    1. Make a directory entry and check for valid file names.
    2. Read a record from your medium and translate it into LIF ASCII representation.
       a. Numerics translate into X3.42 standard (ASCII representation).
       b. Text translates into straight ASCII.

3. Calculate the record length.
4. Strip out any trailing CR-LF unless they are a meaningful part of the data.
5. Write the record to the LIF formatted medium.
6. Get another record.

- Read

1. Locate a file in the directory.
2. Locate and read a record from a LIF formatted medium.
3. Convert to the working or default format.
4. Get another record.

It is recommended that the above implementation be resident in your host and not as an add on utilitly or routine. This is because of the following user interface considerations:

- Utilities are generally combersome and harder to use than resident code.
- Utilities can become outdated and/or misplaced.
- Utilities provide after introduction forgotten implementations.

Complete Implementation

A complete implementation would include all the facets of a mass storage operating system utilizing the LIF standard. All data records would be capable of being translated into LIF ASCII representation:

- Numerics translate into X3.42 standard.
- Text translates into straight ASCII.

System unique file types (program and binary files) would be designated as per the LIF format and stored on LIF formatted medium. A partial list of functions that might be initiated follows:

- Format a medium with a LIF directory.
- Catalog a LIF volume and check for valid file names.
- Create and find a file entry in the directory.
- Write and read ASCII data records.
- Multi-volume files could be implemented.
- Print volume identifications.
- Repack.
- Time and date.
- Variable length directory.
- Ability to recognize its format type and switch in the appropriate logical format transparent to the user.
- System unique files types on LIF medium.

Sample Catalogs

For both implementations (minimum and complete) when you catalog a LIF formatted medium, all the file types should be shown. File types that are not recognized by your system should have the decimal equivalent printed out in the catalog. Following are sample catalogs from the 9826 (Basic) and the HP 85.

:INTERNAL
Volume Label: B9826

File Name	Pro Type	Rec/File	Byte/Rec	Address
REVID	ASCII	1	256	16
CBACKUP	PROG	50	256	17
FBACKUP	PROG	86	256	67
LISTER	PROG	82	256	153
XREF	PROG	74	256	235
PHYREC	BIN	9	256	309
CAT	PROG	29	256	318

:INTERNAL
Volume Label: HP 85

File Name	Pro Type	Rec/File	Byte/Rec	Address
CATTY	ASCII	33	256	16
CATTY-2	ASCII	23	256	46
C	-8184	6	256	72
DGTSAV	-8184	7	256	113
LIF	-8160	36	256	120
COPY	-8160	2	256	156
PROG	-8176	2	256	175

As you can see, the above catalogs came from a 9826. All ASCII files are transportable. System unique programs are properly labeled if there are understood, or they are shown with a decimal representation if they belong to another system. The catalogs shown below are of the same two media but cataloged on the HP 85. The system unique files appear in the opposite catalog.

LIFVOL: B9826

Name	Type	Bytes	Recs
REVID	asci	256	1
CBACKUP	-5808	256	1
FBACKUP	-5808	256	1
LISTER	-5808	256	1
XREF	-5808	256	1
PHYREC	-5775	256	1
CAT	-5808	256	1

LIFVOL: HP 85

Name	Type	Bytes	Recs
CATTY	asci	256	33
CATTY-2	asci	256	23
C	BPGM	256	6
DGTSAV	BPGM	256	7
LIF	PROG	256	36
COPY	PROG	256	2
PROG	DATA	256	2

Features and Restrictions

This section lists the features and restrictions accompaning this logical format. In cases where the restrictions seem overly restrictive possible solutions are given.

Features

- Transportable Media
  This is by far the best feature of the LIF implementation. Imagine that in a couple of years data compatibility exists from CVD's Personal Computers thru the Desktops and Terminals to HP's Business Computers.
- Allows system unique files types.
  This feature allows this format the capability of specifing your system unique file types (program files and binary files).
- Time and Date
  This is an internal method of recording file revisions and/or file creations.
- Multi-volume
  Allows file data to span more than a single medium.
- Simple
  This format can span a wide range of systems.
- Media type Independent
  Implement LIF once and design the channel module (drivers) for protocol, interface, and mass storage device.

Restrictions

- File name cannot exceed 10 characters.
- File name must consists of uppercase alphanumeric characters (A to Z and 0 to 9) and the underscore character.
- 1st character of the file name must be an alpha character.
- Linear search directory.
  In some cases, listing to CRTs supported by smaller memories the catalog could overflow the memory. The implementation of a selective search function that searches on a single first character (e.g., catalog all files beginning with the letter C) might be a valid solution.
- The physical location of files sequentially on the medium can cause the pack function to be overused in cases where files are continually updated.
- Data conversion to ASCII may cause the tape device to stop. This may necessitate a large buffer design or the ability to que commands to the tape device.

LIF does not identify the type of data to be transferred and all 8 bits are significant. Therefore, the sender and receiver of the medium must have an agreed upon convention for the data via an external method. For example, a desktop controller instrumentation system is taking volt-meter readings and storing them on the LIF disc to be later transferred to a production control system which is to manipulate the data to report long term trends. The program on the receiving system must know that the data is stored as 4 byte values. The LIF directory entry does not by itself convey information as to the nature of the data in the file.

Chapter 3
LIF Implementation

This chapter is a detailed description of LIF. It is for the individual that is implementing the "Standard".

Definitions

This section is presented as an aid to understanding some of terms used within this document.

Interchangeability — The ability to read information from a medium on a host computer other than the one which originally wrote the information.

Integer — A 16-bit signed binary number.

Double Word Integer — A 32-bit signed binary integer. For this application, a double word integer is considered to be packed into two 16-bit words such that the most significant bits are in the first word, and the least significant bits are in the second word.

Sector — Used interchangeably with PHYSICAL RECORD. A grouping of contiguously recorded bits. This includes a header which is a series of bits that recognizes and synchronizes the start of the sector and the body of the sector which contains the user information and certain device applications such as checksum and error correcting code (ECC). All references to sector location are in the linear list mode.

Unit of Addressing — This is a blocking factor which establishes valid start addresses. The LIF addressing unit is always 256 bytes.

File — A user defined collection of logically contiguous records written onto the data portion of each sector and containing a set of user defined data.

Directory — A file which is found in a known location, and contains a list of user files on the medium. The directory also contains the following information about about each file: starting location of the file, length of the file, and type of data in the file.

Volume Label — Also refered to as "Directory Header". This is a file that is created upon initialization of the medium at a known location (usually the first sector), and that contains information about the medium. Such information includes such things as type of media and start and length of the directory.

Track — The area of the medium that can be read by a single detector (head) without movement of that detector.

Cylinder — A Cylinder number denotes the position of an actuator when more than one detector is positioned by the same actuator mechanism.

Cylinder Mode Addressing — This is used where there are several surfaces on a particular medium. It would normally require three parameters to find a particular piece of data in this case, namely: cylinder number, surface (side) number, and sector number. Cylinder mode addressing sequentiallizes the medium so that any location can be defined by a single parameter. This mode is defined as follows: the absolute sector following the last sector on cylinder 0, surface 0, is taken to be the first sector on cylinder 0, surface 1, rather than the first sector on cylinder 1 surface 0.

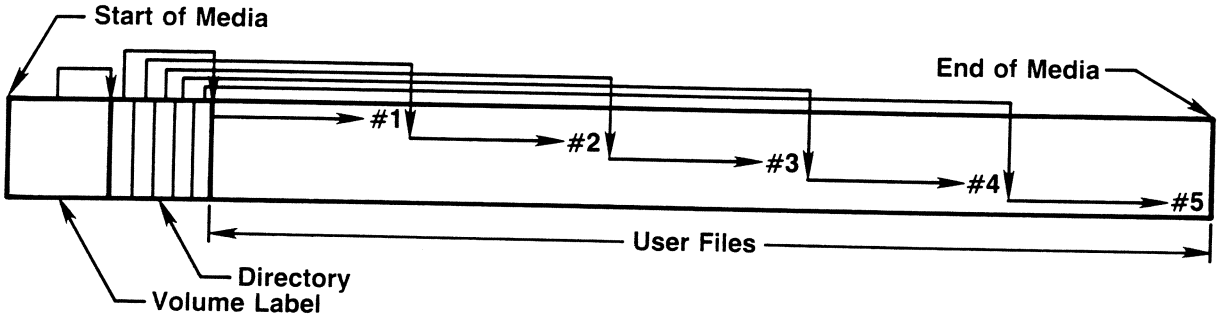
Volume Number — In certain instances, a file may be placed on several different media. This usually happens because the size of the file is greater than the capacity of the medium. In this instance the volume number is an indication of which portion of a file is on this medium.

Limitations and References

This standard can be implemented on all disc and preformatted tape media, but specifies only logical standards. Physical compatibility of sectoring, track spacing, bad track masking, etc. is not specified for this reason. It is suggested that the reader familiarize himself with the ERS for his particular drive for such device specific information. Appendix A contains some of the most used information for your convenience.

Logical Media Layout

The following drawing represents the linear layout of all disc and preformatted tape media.



The volume label references the directory and each directory entry references a user file.

Volume Label

The purpose of the volume label is to allow easy identification of media type and to find file information (directory). The format of the volume label is as follows:

WORD	(16 Bits) CONTENTS
0	100000 OCTAL (LIF ID)
1 2 3	VOLUME LABEL (0-6 ASCII CHARACTERS)
4 5	DIRECTORY START ADDRESS
6	OCTAL 10000 NEEDED FOR SYSTEM 3000
7	DUMMY (SET TO 0)
8 9	LENGTH OF DIRECTORY (FIXED AT INITIALIZATION)
10	VERSION NUMBER (0 FOR MEDIA WITH NO EXTENSIONS)
11	SET TO 0
12-20	LEVEL 1 EXTENSIONS — SEE EXTENSIONS SECTION
21-126	RESERVED FOR EXTENSIONS AND FUTURE USE (SET TO 0)
127	RESERVED BY SYSTEM 250 FOR MEDIA MAINTENANCE WORD

Location — The location of the volume label is defined to be the first unit of addressing available on the medium. All words in the second addressing block are to be initialized to 0 for compatibility with the System 3000.

LIF ID — This identifies media written in the Logical Interchange Format specified by this standard.

Volume Label — (0-6 ASCII characters) Characters are packed with the first character of a pair in the high order byte. Trailing characters are spaces. To provide maximum interchange, characters are limited to upper case letters (A-Z) and digits (0-9). The first character (if any) must be a letter. The purpose of this field is to identify which particular volume is in a drive. Default volume label is 6 spaces.

Directory Start Address — This is a double word integer showing the address for the start of the directory in number of addressing units. The first word contains the high order bits and the second word contains the low order bits. The value of this field is 2 at the present time.

Word 6 — This word is included to eliminate console messages on the SYSTEM 3000. It must be written as shown.

Length of Directory — This double word integer entry is included to stop the directory from overwriting user storage space. This entry is decided upon at disc initialization and thereby fixing directory size at that time. This entry contains the maximum allowable length of the directory in number of addressing units.

Version Number — This field is used to allow for implementation of extensions to the standard. It should be set to 0 for a base level implementation. For a list of capabilities added with each version, see the section on extensions.

Words 21-126 — These words are reserved for extension fields and for implementation dependent fields. Any usage of this area MUST be co-ordinated with your primary support contact. The reason for this is to avoid different implementations from using the same field for different purposes. When implemented, additional extension fields will be allocated up from word 21, and user dependent fields allocated down from word 126. See the section on extensions for the currently implemented extensions.

Word 127 — This word is reserved for System 250's use as a maintenance word for the LINUS tape cartridge. Tapes received from other machines must have this word set to 0.

Directory

The purpose of the directory is to allow data to be easily located. All information necessary for reading a file is contained in the directory. The directory is a linear list of directory entries each of which is organized as follows:

WORD	CONTENTS	
0 1 2 3 4	FILE NAME (1-10 ASCII CHARS, TRAILING BLANKS)	
5	FILE TYPE	
6 7	STARTING ADDRESS	
8 9	LENGTH OF FILE	
10 11 12	TIME OF CREATION (12 BCD DIGITS)	
13	L	VOLUME NUMBER
14 15	IMPLEMENTATION	

Location — The directory is located starting on an addressing block boundry defined by the Directory Starting Location field of the volume label. Directory location is further limited to the logical front of the medium. Thus, user space cannot span the directory.

File Name — (1-10 ASCII characters) Characters are packed with the first byte of a pair in the most significant byte of the word. Characters are limited to digits (0-9) and upper case letters (A-Z). The first character must be a letter. All non-purged files must have at least one character in their file name and implementations must be capable of distinguishing all ten characters. Padding is with trailing blanks.

File Type — This is a 16-bit signed integer. The presently defined file types are shown next:

0 — Purged file. This standard makes no statement on directory names of purged files. However the purged file name must be capable of being identical with a valid file. Implementation of this standard must not give duplicate file name errors for purged files.

1 — ASCII data file. This is the interchange file type. The associated file consists of 8-bit ASCII characters. See "File Structure" for a complete description.

-2 — Binary data file. See "File Structure" for a complete description.

Other negative file types are reserved for system dependent applications. It is not necessary for any implementation to be able to read these file types for conformance. They are merely for flexibility and convenience and to be used by the experienced user. These types are broken into the following fields:

RANGE (OCTAL) (2's complement form)	APPLICATION
2 TO 77777 -3 TO -2000	Reserved for future use by the standard. Special interdivisional file types commonly used for interchange, i.e., data bases. These shall be well documented and available as standards supplements.
-2001 TO -4000	GSD (SYSTEM 300)
-4001 TO -6000	CSY (SYSTEM 3000)
-6001 TO -10000	DSD (SYSTEM 1000)
-10001 TO -12000	GSD (SYSTEM 250)
-12001 TO -14000	DCD (9845/9835)
-14001 TO -16000	DTD
-16001 TO -20000	CVD
-20001 TO -22000	CSD
-22001 TO -77777	Subject to further definition.

Starting Address — The starting unit-of-addressing number for a file is a double word integer. The most significant bits are in word 6 and the least significant bits are in word 7.

Length of File — A double word integer showing the allocated space for the file (not current length) on this volume. Again the most significant word comes first. The length is given as the number of 256-byte blocks. Lengths less than 0 are not allowed. The usage of free space is implementation dependent. Therefore, the length and start address fields of a purged file are

not guaranteed to be accurate. Free space is computed from the start address and length fields of the 2 nearest valid files which surround the free space.

Time of Creation — 12 BCD digits of the form YYMMDDHHMMSS. These words will be packed with the first digit in the most significant bits of word 9 and the last digit in the least significant bits of word 11. This field may be used for a version number on systems not using a real time clock. If the year and month fields are 0, the other fields will be a version number. However, all digits must be valid BCD numbers. Also, if a particular file spans several volumes, the directory entries for each volume of this file must have the same time stamp or version number.

L — Last volume flag. If L = 0, this is not last volume of the file. If L = 1, this is the last volume of file.

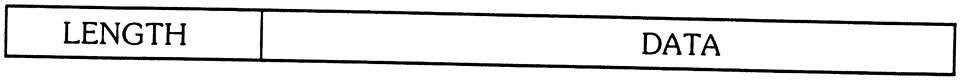
Volume Number — This is a 14-bit unsigned integer containing the volume number of this file on this medium. Volume number = 0 is NOT a valid condition. All files must be contained on at least one volume. Volumes start with number 1 and are incremented by one for each subsequent volume.

Implementation — Words 14 and 15 are available for implementation dependent uses by file types -2000 to -77777. These fields MUST be set to 0 for all interchange file types and for file types -2 to -177.

Directory Organization — The directory is defined to be a linear list of directory entries. The directory has no fixed length, except as fixed at the time of initialization. A directory entry which has a file type of -1 is defined to be the end of logical directory. A logical end of directory mark is written unless the directory is filled. The physical end of directory is determined by adding the start of directory and length of directory fields. This address is considered an absolute end of directory indicator and thus precludes the need for a logical end of directory marker on full directories. Free space begins at Address # = (Starting Address # of last file entry + last file length). Implementations must also be aware of the absolute end of directory found in the volume label. Methods of packing and re-allocating free space are not defined. However, entries must be stored so that they are in order of strictly increasing starting addresses. Directory entries are undefined after the logical end of directory. Thus, when a file is appended, the next directory entry must be set to the logical end of the directory.

File Structure

A file is a list of records containing data. Defining the structure for file types 1 and -2 provides a mechanism for transferring files containing virtually any type of data. The file does not attempt to convey the meaning of the data as part of the file. This is best done by external means (naming conventions, attached hard-copy description etc.). A file consists of a linear list of records, each of which is as follows:



Location — A file location is determined by the value of the starting address field within the directory entry. A file always starts on an addressing block boundary. Further, a logical record will not traverse a volume. Thus, all volumes will start with a complete logical record making it possible to recover partial data.

Length — This is a 16-bit integer showing the length of the record in bytes, but not including the length field in the count. A length of -1 will denote logical end of file. A file will have an end of file marker unless its length is at the maximum length defined in the directory. Also, the length field of the last record may exceed the physical space remaining; in this case the file is terminated by physical length of file. If the last volume flag is set for the file, this is the end of file mark, otherwise it is the end of the volume. A "0" length record is valid. In this case, the length field is followed by the length field for the next record. A record of length less than -1 is not allowed. Finally, if the length is odd the record is rounded to the next word boundry and the extra byte is ignored.

Data — Defined as follows:

File type 1: The record consists exclusively of 8-bit ASCII codes. All 8-bits are defined (i.e., foreign character sets) and no parity is checked or generated. Codes 0 thru 127 have US ASCII meaning. Codes 128 to 255 are open but the SYSTEM 300 definition in Appendix B is recommended. In addition, no FORTRAN carriage control, escape sequences, etc. are defined. Numeric data are formatted as per ANSI standard X3.42, which is available from the contact person or ANSI. Data records should be stored with 1 item per record. Thus an n element array would require n logical records for storage. Programs are stored with one source line per record. Records must not contain a trailing CR-LF unless it is a meaningful portion of the data item. Logical records are allowed to span physical record boundries.

File type -2: The data format within the record is undefined and must be known to the user.

Extensions

An extension field is one that is defined to be of general interest to a large set of implementors, but is not required for implementation of the standard. No extension is allowed that would prohibit an implementation from reading, writing or creating a lower level implementation. Levels are hierarchical, that is, level N includes levels 0 to N-1.

Level 0 is defined to be the level with no extensions. This level must be supported for all implementations.

Level 1 contains the following extensions:

Words 12-17
Physical Attributes Data. Words 12 and 13 contain the number of tracks per surface with the most significant word being word 12. Words 14 and 15 contain the number of surfaces per medium, with the most significant word being word 14. Words 16 and 17 contain the number of sectors per track with the most significant word being word 16.

Words 18-20
Volume Stamp. This field is identical to the time stamp field of the directory listing, except that this field reflects volume creation. All comments about valid BCD digits, version numbers etc. still apply.

Conformance Verification

Correct implementation of this standard can be verified by sending a sample of the medium to the contact person listed below. It will be compared byte-by-byte to a "standard" medium which is maintained to the latest revision of LIF. The sample will also be tested on the other systems supporting a LIF file system to verify the sample's interchangeability. Results of these tests will then be reported to the originator.

Support

The contact person will provide the on-going support to the systems divisions implementing LIF on their mainframes. This includes answering technical questions, maintaining a current distribution list, and making revisions to this document when needed. Also, the contact person could initiate another conference if later discrepancies or enhancements are found.

Copies of the standard can be obtained through the specs group of the peripherals division manufacturing the mechanism.

The Greeley contact person for LIF appears next:

Gordon Nuttall
(Greeley Division)
3404 Harmony Road
Fort Collins, Colo. 80525
303-226-3800

APPENDIX A

This appendix lists some of the more commonly needed bits of information that are device-specific and do not belong in the body of the LIF standard but are frequently referred to by designers of mass storage drivers.

Number of Tracks Per Disc Medium

The number of tracks per disc surface is a software function due to bad track sparing at initialization. The number of good tracks is fixed for interchange format discs as follows:

DRIVE	#TRACKS
9895/7902 double-sided	150 (2 surfaces)
single-sided	73
9885	63
9130K (Mini-Floppy)	66 (2 surfaces)

It should be noted that for discs which have both fixed and removable platters such as the 7905/7906 that surface mode addressing scheme is used to prevent part of the data from becoming non-removable.

DC600 Cartridge Tape Drive (LINUS) Device Notes:

1) Sector (physical block) = 1024 bytes
2) The number of usable sectors per tape is readable using the DESCRIBE command (16,320 for initialized short tape, 65,408 for initialized long tape.)
3) For writing, use skip sparing, SPARE BLOCK command S = 0. (Auto skip sparing is recommended, device specific options A = 1, S = 1).
4) On used tapes, convert jump spares to skip spares - INITIALIZE TAPE, E = 1.
5) It is also recommended that a file mark be written at the third address block to prevent the disc from accidentally being IMAGE restored by a LIF tape.

APPENDIX B

System 300 Extended Character Code Assignments

NOTE

The SYSTEM 300 codes are not completed at the time of this printing. They are allowed for by this standard, but do not directly involve the implementation of this standard.

APPENDIX C

This appendix lists the logical format identifier word by HP product or format. This method by which the medium can identify its logical format, or directory and file structure, was agreed upon by attendees at a meeting at DSD on June 10, 1981. The first two bytes of the first record on the medium are now reserved for this identifier, and assignments are listed in the following table. The HP Logical Interchange Format (LIF) contact person will assign codes for this identifier for as-yet-undefined formats and otherwise maintain the list.

Media Logical Format Identifier

LIF	8000 HEX
DCD 9825/35/45	0500 HEX
9826 PASCAL (UCSD)	LIF, volume label = "P9826"
DAWN	0700 HEX
C.Spgs. ORION	Minifloppies — 0 to 117 hex
	other — not constant
DTD BOBCAT	2000 HEX
GSD 125	LIF, volume label = "HP125 "
250	0600 HEX
DSD RTE	Not constant (data area)
CSY systems II and III	
System discs	4000 — 4FFF HEX
Series 33/44	5359 HEX
Private volumes	
& series discs	0000 HEX
HPE	5400 HEX *
HP UNIX	3000 HEX *
IBM 3740 series	not constant (system boot area)
	(also proposed F.I.P.S)

There are a couple of "holes" in this method with some existing implementations. The first is DSD's RTE-formatted media. They have the directory at the logical end of the medium rather than the more typical beginning allowing data to extend to record 0. Another is non-HP floppies, which in general follow the precedent set by the IBM 3740 series of data entry stations, and is the basis for the proposed Federal Information Processing Standards series which reserves some sectors for a system boot area.

Also, some systems (HP125, PASCAL 9826) reserve the first 3-4 sectors to write a LIF volume label and directory, but the working directory co-exists in another area of the medium. Creating a LIF file on a 125 medium will garbage the CPM data. The PASCAL 9826 creates a fake file that fills the medium to guard against this.

These two logical format identifier bytes provide a simple method for the system file driver to determine that the medium is alien, but these are not necessarily sufficient to insure that the medium is a particular format. Each format can have other constants which can be inspected to provide a higher confidence of correct identification.

Subject Index

a

Allowed Extensions 15

b

Bad Track Masking 10
Binary Files 6

c

Catalogs Sample 7
Channel Module 4
Commitments 2
Compatibility 1
Compatibility Using 5 1/4" 3
Compatibility Using 8" 3
Compatibility Using 9134A 4
Compatibility Using Linus 4
Complete Implementation 6
Conformance Verification 16
Contact Person 16,2
Cylinder 10
Cylinder Mode Addressing 10

d

Data Defined 15
Defined Data 15
Definitions 9
Directory 12,8,9
Directory Location 13
Directory Organization 14
Directory Start Address Location 11
Double Word Integer 9

e

ERSs 2
Existing Systems 3
Extensions 15

f

Features LIF 8
File 9
File Manager Diagram 4
File Name 13
File Structure 14
File Type 13
Files Layout 10
Fixed and Removable Platters 17
Format Identifiers 19

g

GLD Support 16

h

HP 85 Catalog 7
HP 9826A Catalog 7

i

IBM 3740 2
ID Location 11
Identifiers Media 19
Implementation 1
Integer 9
Interchangeability 9

l

Label Volume 11
Length of Directoy 12
Levels of Compatibility 3
Levels of Compliance 5
Levels of Extensions 15
LIF Features 8
LIF ID Location 11
LIF Restrictions 8
LIF Standard 2

Linear Files 10
Location Directory 13
Location of LIF ID 11
Location of Volume Label 11
Logial Format 4
Logical Compatibility 3
Logical Format Identifiers 19
Lost Sales 1

m

Manual Definitions 9
Masking Bad Tracks 10
Media Identifiers 19
Medium/Tracks 17
Minimum Implementation 5
Minimum Level 5
MPN 1

n

Number Version 12

o

Orion1 Catalog 7

p

Physical Compatibility 3
Physical Sectoring 10
Program Files 6

r

Read Implementation 6
Responsibilities 2
Restrictions LIF 8

s

Sample Catalogs 7
Sector 9
Sectoring Physical 10
Solution 2
Spacing Tracks 10
Stamp Volume 16
Support GLD 16
Supported 2
System 300 Character Codes 18
System Optimized Code 4
System Unique Files 6

t

Time of Creation 14
Track 10
Track Spacing 10
Tracks/Medium 17

u

Unit of Addressing 9
User Interface 4

v

Verification 16
Version Number 12
Volume Label 11,9
Volume Number 10
Volume Stamp 16

w

Write Implementation 5