SPERRY RAND

UNIVAC 9400 SYSTEM DISC LIBRARIAN

PROGRAMMER REFERENCE

UP–7745
Rev. 2

PAGE STATUS SUMMARY

ISSUE: UP-7745 Rev. 2

The following table lists the status of each page in this document and indicates the update package (if applicable).

Section	Page Number	Page Status	Update Package	Section	Page Number	Page Status	Update Package
Cover/Disclaimer		Orig.					
PSS		Orig.					
Contents	1, 2	Orig.					
1	1 thru 3	Orig.					
2	1 thru 8	Orig.					
3	1 thru 3	Orig.					
4	1 thru 20	Orig.					
Appendix A	1 thru 4	Orig.					
Appendix B	1 thru 10	Orig.					

CONTENTS

1. INTRODUCTION

1.1. GENERAL

This manual describes the set of programs collectively called the UNIVAC 9400 Disc Librarian. These programs provide a number of functions that may be performed on any one or all of the five types of permanent libraries in the UNIVAC 9400 Disc Operating System. The functions are used to add, delete, copy, or compress a file; to correct source code statements; to rename modules in a library; or to create a library tape. Other functions retrieve information from a specified library and display it as printed output and/or produce a punched card output deck. These functions are performed under the control of the library update services program (LIBUPS).

The functions of the Disc Librarian are not applicable to other libraries specified for the Disc Operating System, such as those created by the UNIVAC 9400 Disc Mapping Program. The permanent libraries operated on by the Disc Librarian reside on the system resident volume or may reside on some other specified disc volume.

A knowledge of the *UNIVAC 9400 System Job Control Programmer Reference, UP-7793*, (current version) and the *UNIVAC 9400 System Assembler/Central Processor Unit Programmer Reference, UP-7600* (current version) is helpful in using this manual.

1.2. LIBRARIES

There are five types of permanent libraries, each of which constitutes a distinct disc file. They are the Load library, Reserve library, Copy library, Source library, and Proc library. Each library consists of a directory and a main body. The structure and format of each of the libraries is discussed in Section 2. Load and Reserve libraries are composed of object modules; Copy, Source, and Proc libraries are composed primarily of source modules.

- Load library

  The Load library consists of load modules, which are the output of the Linkage Editor. (A load module may also be the output of the assembler.) Transient routines are not acceptable as input to the Load library. Each load module may be either a single phase or multiphase. A phase is a single loadable entity. A multiphase load module may be processed by the Disc Librarian through the gang operation option (see 1.4) Updating of a Load library is done by phase.

- Reserve library

  The reserve library is used primarily for storing object modules, which are the output of a language processor (Assembler, FORTRAN, COBOL, RPG). These object modules are in relocatable format and must be processed by the Linkage Editor before they can be executed.

- Copy library

  The Copy library consists of copy modules, which are sets of source statements associated with the COPY statements of COBOL.

- Source library

  The Source library consists of source modules, which are sets of source statements to be processed by a language processor.

- Proc library

  The Proc library consists of Proc groups, which are made up of Proc modules. A Proc module is a set of source statements constituting a procedure definition. When adding Proc modules from a tape library, a Proc module may only be referred to by the module name contained in the header record. When specified on a control statement, the module name need not correspond to any of the names on the NAME directives within the Proc module. When a Proc module is already stored on disc, it is referred to by its first NAME entry in the directory.

1.3. INPUT AND OUTPUT

The Disc Librarian accepts the following as input: librarian control statements; up to five library files to be accessed; object modules and phases from the Module Complex Library (MCL); library modules or phases from an alternate file; or punched cards. Each library file to be accessed or modified is declared by an INP and/or FIL librarian control statement. (See 4.2.2 and 4.2.1.)

The Disc Librarian produces the following as output: an updated library file; a copy of one or more library files; a user library tape in Tape Librarian format; punched cards; or printed listings. The library file into which information is to be copied is declared by an OUT and/or FIL librarian control statement. (See 4.2.3 and 4.2.1.) Library modules are filed chronologically. A module is added to the end of a file in the next available area in both the directory and the main body of the library.

1.4. GANG OPERATIONS

A number of functions of the Disc Librarian may be specified in such a way that they are performed on groups of modules within a library, rather than on an individual module. From one to seven of the leading characters of the module names must be the same. Gang operations are performed on these modules by specifying the leading characters of the module name, followed by a period (.) and the letters ALL. For example, the specification SQROOT.ALL causes a function to be performed on all modules in the library that begin with SQROOT.

The functions of the Disc Librarian for which gang operations may be specified are ADD, DEL, DIS, PCH, and PUD and are applicable to all types of libraries. These functions are explained in Section 4.

1.5. SYSTEM NAMING CONVENTIONS

The name of a program is introduced to the operating system as the symbol in the label field of a statement in a source deck. The symbol may be from one to eight characters in length, the first of which must be alphabetic. The valid characters are the letters A through Z, the numerals 0 through 9, and the dollar sign special character ($). The numerals must not be used as the first character of a name.

Program and module names in all libraries, except for multiphase load modules in a Load library, may be from one to eight characters in length. The names within the Load library are left justified in the field and are zerofilled if fewer than eight characters are specified. The names in the remaining libraries are left justified in the field and are blank-filled if fewer than eight characters are specified.

For multiphase load modules in a Load library, the name of each phase within the module consists of from one to six alphanumeric characters followed by a two-digit phase number (00-99). The phase number is generated by the Linkage Editor and appended to the load module name. If no module name is supplied, the name used by the Linkage Editor for a load module is taken from the one- to six-character name of the first object module in OBJFIL (tape) or MCL (disc). If the module name is composed of eight characters, the Linkage Editor truncates the two rightmost characters and adds the two-digit phase number.

The following names are reserved for use by the Disc Librarian and may not be used as program or module names:

DIR

ALL

CARD

MCL

ALT1

ALT2

1.6. PROGRAMMING CONSIDERATIONS

If the byte labeled SB$CHR in the system information block (SIB) has been set to X'20', the Disc Librarian responds with the following console message after each LIBUPS overlay is loaded:

```
LBXX   nn   name
```

where:

nn	is the LIBUPS overlay number
name	is the name of the module currently being processed

Alter patches, to be used for debugging purposes, may be keyed in at this point. The size of the patch is limited to a maximum of 16 characters.

If no alter patches are required, the standard reply is:

```
nnR AP
```

where:

nn	is the assigned job number

In the UNIVAC 9400 multiprogramming environment, SB$CHR must be reset to X'00' at job termination.

2. LIBRARY STRUCTURE AND FORMAT

2.1. GENERAL

This section describes the structure and format of the various libraries constructed by the Disc Librarian as well as the printer and punch output formats.

2.2. DISC LIBRARIAN OUTPUT STRUCTURE

The Disc Librarian constructs its libraries with unkeyed, fixed-length blocks that are stored sequentially within a partitioned file organization. The physical block size is fixed at 444 bytes; the logical block size may be of variable length.

Each library file, except Proc files, is allocated a single extent consisting of contiguous cylinders. Proc files are allocated two extents. One extent consists of a single track containing the Proc group descriptor block, which contains suballocation information for individual Proc groups. The other extent is subdivided into consecutive partitions each beginning and ending on a cylinder boundary and having a structure similar to other single-extent library files. The structure of the libraries is shown in Figure 2-1 and Figure 2-2.

2.2.1. DIRECTORY

A directory is recorded at the beginning of each of the Load, Reserve, Copy, and Source libraries, as shown in Figure 2-1, and at the beginning of each Proc group, as shown in Figure 2-2. The first 28 bytes in the first block of a directory are reserved for the library file descriptor record (LFDR). This record contains control information for processing library files. A first-level index for each module in the particular library follows the LFDR. Each first-level index entry is 14 bytes in length. Each index entry consists of an eight-byte module name, a five-byte relative starting disc address (CCHHR) of the module, and a one-byte type code. The address is that of the first block of the referenced module in the main body of the particular library. Index entries are arranged in the sequence in which the modules were added to the end of the library. Subsequent directory blocks start with the next sequential index entry; that is, no LFDR is present in subsequent directory blocks for a particular library.

A directory is recorded at the beginning of each proc group, as shown in Figure 2-2. Each proc group directory has essentially the same format, except that the eight-byte name portion of an index entry contains the symbol taken from the label field of a particular NAME statement. Proc modules containing multiple NAME statements have a directory entry for each NAME statement. Proc modules are identified by testing for equal starting disc addresses in consecutive index entries; this address is the address of the first block of the referenced module within the Proc group.

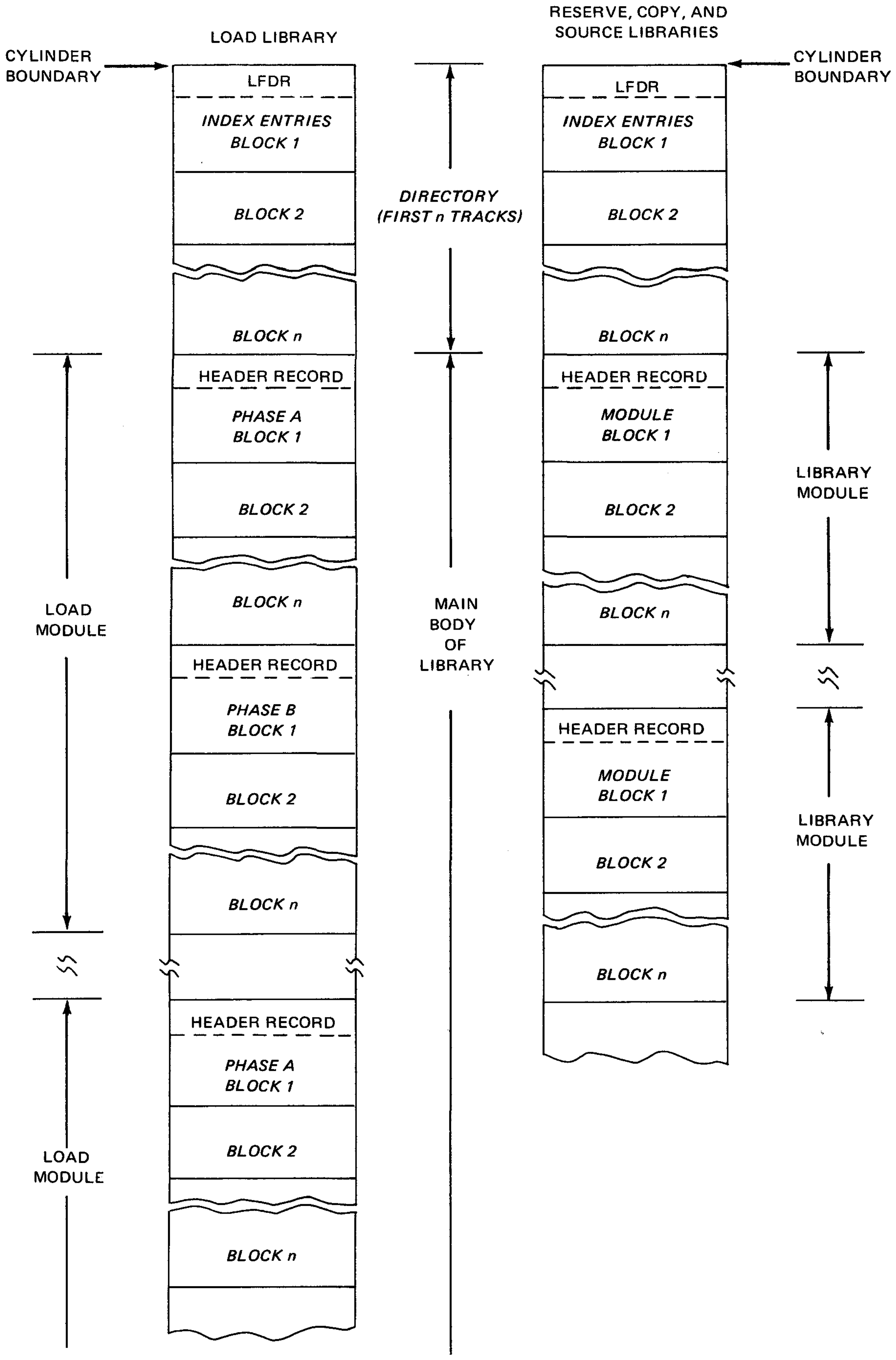


Figure 2–1. Load, Reserve, Copy, and Source Library Structures

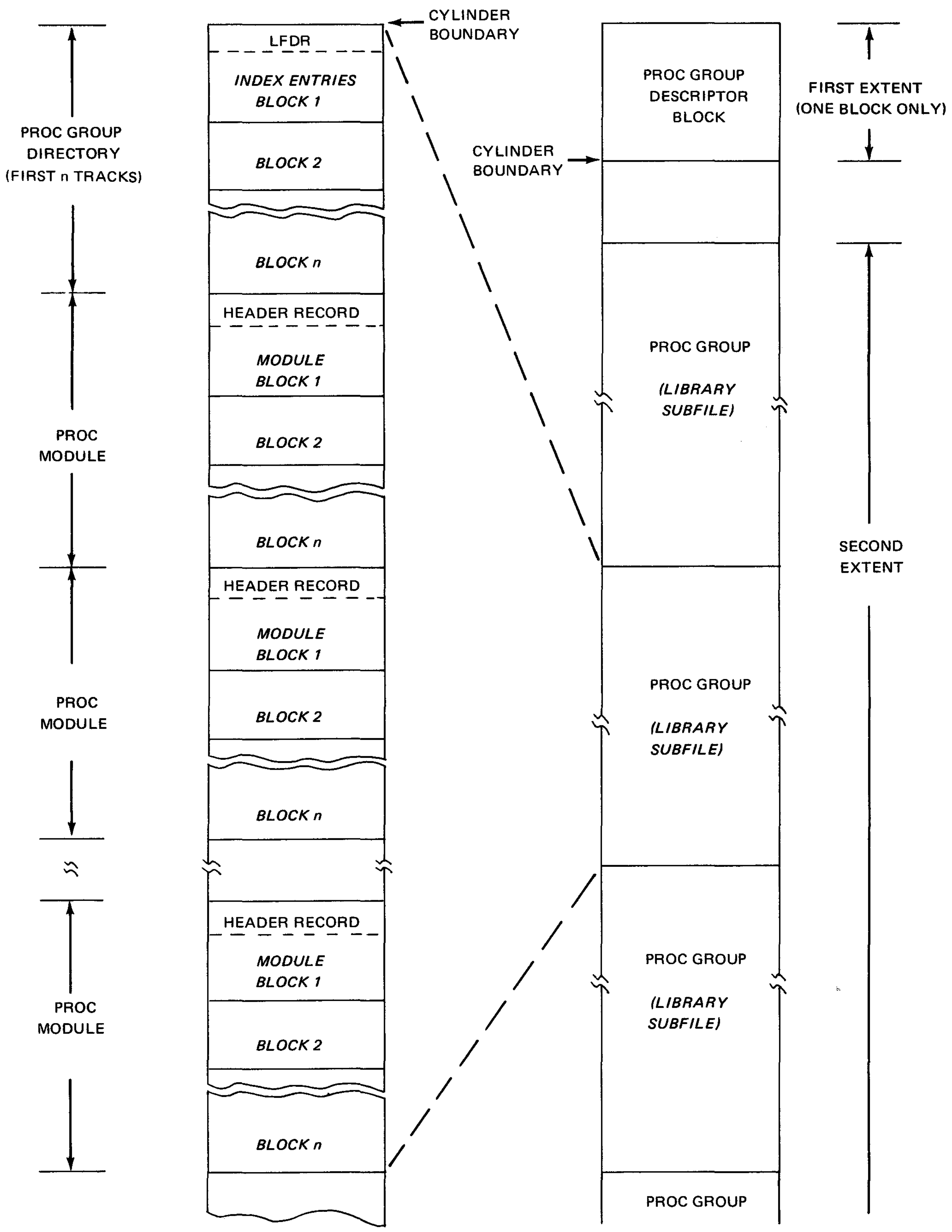


Figure 2–2. Proc Library Structure

Table 2-1 lists the maximum number of entries, bytes, and blocks that may be specified in a directory.

Directory	8411 Disc Unit	8414 Disc Unit
Directory block	444 bytes	444 bytes
	7 blocks/track	13 blocks/track
	2168 entries/cylinder	8058 entries/cylinder
LFDR	28 bytes	28 bytes
Each index entry	14 bytes	14 bytes
First block	29 entries	29 entries
Each subsequent block	31 entries	31 entries
First track	215 entries	401 entries
Each subsequent track	217 entries	403 entries

Table 2-1. Directory Block Statistics

2.2.2. MAIN BODY

The area referred to as the main body of a library is that portion of the file which directly follows the last track of the library directory. The blocks within the main body of a library are 444 bytes in length.

In addition to the first-level index in the directory, a second-level index, or header record, is provided in the main body of the library. A header record appears at the beginning of the first block of each module in the Reserve, Copy, Source, and Proc libraries, and at the beginning of each phase of the load modules in a Load library. The header record supplies additional information regarding a particular module or phase.

When adding or copying from tape to disc, the tape header block is merged with the first data block on the disc file. This factor, combined with the seven blocks per track limit, should be considered when reserving disc space.

2.3. PRINTER OUTPUT

The library update services (LIBUPS) program and the display and punch services functions produce printed output in specific formats.

2.3.1. LIBRARY UPDATE SERVICES (LIBUPS) PRINTER OUTPUT

The printer output format for the LIBUPS program is a listing of five fields. The listing provides a record of the control statements and the activity that was performed. The Disc Librarian lists the declared disc library files immediately following the PARAM statement(s). The logical file names (LIBn) are followed by their format 1 file ID (which corresponds to the file name on the LBL statement) and their volume serial numbers. This listing does not imply an error condition, but is included to aid the user. An example of the LIBUPS program printer output is shown in Figure 2–3.

```
Error Code   Error Field   Control      Line Number                                   Source Statement   LIBUPS Page 0002

                           **// PARAM   LIN=(2,NALT2,NLIB)

                                        LIB1 LOAD$LIB DSP001 }
                                                             } * Declared Library Files
                                        LIB2 PROC$LIB DSP002 }

S 4  M                     **INLL LIB1                       * Misspelled functions –
                                                               invalid control statement
                           **OUTL LIB1

                           **CPYL (ALT1)

                           **INPL LIB1

                           **ADDL JCF.ALL(MCL)               * JCF modules not found

E 0  F       JCF00000

                           **INPP LIB2

                           **RNMP1 GWNC/GWNCJ                * Proc rename illegal

S 2  O       GWNC/GWNCJ

                           ** CORP1     DISCREC

                                0001     DEL      PROC P

                                0002     DISCREC  NAME 0

                            REP         3, 5

                           *****  0003
                                                  MVI DISCY+7,X'F0'
```

Figure 2–3. LIBUPS Print Format

- Error code field

The error code field contains three subfields.

(1) Type of error

D – errors involving disc input

T – errors involving tape input

C – errors involving card input

S – librarian control statement errors

E – miscellaneous errors

(2) Error designator (See Appendix B.)

(3) Error origin – Designates the error origin within the Disc Librarian and is of no importance to the user.

NOTE: If an error does occur, the module that is flagged is incomplete and must be deleted from the library. This is not necessary, however, for errors encountered while processing alternate files (ALT1/ALT2/MCL).

- Error field

  The error field contains the expression in error or other supplemental information.

- Control field

  The control field is used to list librarian and subfunction control statements. Control statements, such as ADDL or CORS, are shown with two leading asterisks. Subfunction control statements, such as INS or REP, are displayed without the asterisks.

- Line number field

  The line number field contains the line number of the source statement. The line number is added automatically to the source statement by the Disc Librarian and is displayed on listings obtained by running the display and punch services of the Disc Librarian. Each source statement is numbered consecutiveiy starting with 1. Any corrections made in the source program must be based on these line numbers.

- Source statement field

  The source statement field reflects the 80-column source statement image and is indented from the control field to distinguish it from control statements.

- Identification and page number

  The program identification (LIBUPS) and the sequential number of the printout page are the final fields in the page headers.

2.3.2. DISPLAY AND PUNCH SERVICES PRINTER OUTPUT

The printer output format for the display and punch services of the Disc Librarian is a listing that shows either the entire contents of a specified library or an individual module in a specified library. The option of printing directories only is also available. Accompanying the directory displays is the calculation of remaining file space. This printout takes the form:

```
RESIDUAL FILE SPACE: xx CYLINDERS AND yy TRACKS
```

where:

xx	is the number of cylinders, in hexadecimal.
yy	is the number of tracks, including partial tracks, in hexadecimal.

The format of displays using the DIS and PUD statements is a listing of five fields for source module libraries and eleven fields for object module libraries. All displays effect the extraction of information from the module header record and display it under the appropriate subheadings in the page header. Examples of both source and object module page headers are shown in Figure 2-4.

Source Header

```
FID     NAME      DISC ADDR   VERS    #BLKS  FLAG        LIBUPS PAGE 0013

P001    NAME1     5B056       01–01   0002   0000

        NAME1     NAME  00

        NAME4     NAME  01
```

Object Header

```
FID     NAME       DISC ADDR   VERS    #BLKS  FLAG   ESID   PHASE  ADDR   LENGTH   PRTCT  MOD ADDR   LENGTH   LIBUPS PAGE 0020

LOAD    DISCBOOT   2F051       01–02   0008   0000   00     000000        000280   0000   000000     000280
```

Figure 2–4. Disc Librarian Page Headers

Both the source and object module page headers contain the following subheadings:

FID	The file identification field indicates the library file type of the displayed file. The file identification is represented as LOAD, RSRV, COPY, SRCE, and Pxxx (where xxx is the Proc group number with a maximum of 109).
NAME	The eight-character module name.
DISC ADDR	The five-character absolute address (CCHHR) indicating the library file (or group) starting disc address.
VERS	The version and update numbers of the module.
#BLKS	The number of blocks in the module (hexadecimal).
FLAG	Information from the flag field in the header record.

In addition to the preceding object module page headers contain the following subheadings:

ESID	External symbol identification field
PHASE ADDR	Six-character hexadecimal string indicating phase address
LENGTH	Six-character hexadecimal string indicating phase length
PRTCT	I/O protect length
MDL ADDR	Six-character hexadecimal string indicating load module address
LENGTH	Six-character hexadecimal string indicating load module length

2.4. PUNCHED CARD OUTPUT

The format for punched card output of a source deck is a duplication of the 80-character disc source record. The format for punched card output of an object deck is a copy of the records as they appear in a block on tape or disc.

Sequence numbers may be inserted in each card in columns 73 through 80. Object code modules are always sequenced. Source modules are sequenced (or resequenced) only if the first source code image has no sequence number (blanks in columns 73 through 80). No resequencing occurs if the card already contains sequence numbers. The 8-character sequence numbers take the following formats:

Object (Load, Reserve)	ppppnnnn

where:

pppp	represents the first and last 2 characters of the module name
and	
nnnn	is the 4-digit sequence number which is incremented by 1's.

Source (Copy,Source,Proc)	pppnnnnn

where:

ppp	represents the first 3 characters at the module name
and	
nnnnn	is the 5-digit sequence number which is incremented by 10's.

When an object module in card format is added to a library, a sequence number check is made to ensure the presence of all cards generated as output by the Disc Librarian.

2.5. USER LIBRARY TAPE OUTPUT

The tape output is in a library format acceptable to the Tape Librarian and Disc Librarian for subsequent processing. There are up to five types of files in Tape Librarian sequence (Load,Reserve,Copy,Source,Proc), and the modules are written in ascending alphanumeric sequence. The tape output is the result of the execution of the DUM function (4.2.10). For a further discussion of the library tape format see *UNIVAC 9400 System Tape Librarian Programmer Reference, UP-7667* (current version).

3. CONTROL STREAM REQUIREMENTS

3.1. GENERAL

The disc librarian is introduced to the UNIVAC 9400 Operating System either as a job or a job step, and its execution is directed by control statements in a control stream. The control stream for the Disc Librarian is made up of job control statements and librarian control statements. Where necessary to avoid ambiguity, the control statements specifically processed by job control are called job control statements; the control statements processed by the Disc Librarian are called librarian control statements. The librarian control statements are incorporated in the control stream between the data delimiters (/$ and /*). An example of a control stream for a Disc Librarian run is given in Appendix A.

The librarian control statements, and the functions they cause to be performed, are described in Section 4.

3.2. JOB CONTROL STATEMENTS

The job control statements required for the proper execution of the Disc Librarian are explained in the following paragraphs. A detailed description of each job control statement is given in *UNIVAC 9400 Job Control Programmer Reference, UP-7793* (current version).

3.2.1. JOB STATEMENT

The JOB statement is normally the first job control statement in the control stream. The name specified on the JOB statement is used to identify the control stream in the job file.

3.2.2. DEVICE ASSIGNMENT SETS

Five library files may be updated during a single execution of the Disc Librarian. Each file must be defined by a job control device assignment set. A device assignment set consists of at least a DVC, VOL, LBL, and an LFD statement or, at most, a DVC, VOL, EXT, LBL, and an LFD statement. Any number of device assignment sets may be in a control stream. The device required for the input and output files must be specified. The printer must be allocated to the job; otherwise, the job is aborted with a console message. The requirements within each device assignment set are:

- DVC statement

  The logical unit number of the device for each file required during the execution of the Disc Librarian must be specified.

- VOL statement

  The volume serial number of each volume to be used as input or output must be specified.

- EXT statement

  The extent requirements for each library must be specified when allocating disc space initially. To expand a Proc file, the disc space required for the larger extent must be increased. This is done by redefining the extent requirements.

- LBL statement

  The file identifier (name/label) or each file used by the Disc Librarian must be specified.

- LFD statement

  The logical name of each file used as input or output by a Disc Librarian function must be specified. The following logical names are required by the Disc Librarian:

File	Logical Name	Remarks
Input	LIBn	Where n may be 1 through 5
Input	SYSPOOL	Required if the module complex library (MCL) is used
Alternate Input	ALT1 ALT2	If tape, it must be in Tape Librarian format
Output	LIBn	Where n may be 1 through 5
Output	LIBOUT	Tape
Alternate Storage	SYSPOOL	Required if a LIBOUT tape is to be created
Printer Output	PRNTR	
Punch Output	PUNCH	

Any extent requirements also must be included on the LFD statement defining the particular file.

3.2.3. EXEC STATEMENT

An EXEC statement is used to identify the program to be executed. The program name must be LIBUPS.

3.2.4. PARAM STATEMENT

The PARAM statements identify the options desired at execution time. PARAM statements may appear in any order or the available options may be listed individually on separate PARAM statements. If any errors are detected, the job terminates upon completion of the PARAM statement processing. The format of the PARAM statement for the Disc Librarian is:

1	10
// PARAM	LIN=(numfil[,NMCL] [,NALT1] [,NALT2] [,NLIB])[,LST= $\left\{ \begin{matrix} D \\ N \end{matrix} \right\}$]

KEYWORD PARAMETER LIN

LIN=	– indicates that the options that follow pertain to an input library; the options may be specified in any sequence.
numfil	– number of files, in decimal; this represents the highest value assigned to the LIBn parameter on the LFD job control statement (for example, 4, if an LFD LIB4 statement exists).
NMCL	– indicates that the MCL is not being used and suppresses any attempts to access it;
	– if blank, MCL is assumed and a check is made for the appropriate control statement.
NALT1 NALT2	– indicates that no alternate library is being used; this results in time saved by not searching for a file definition.
NLIB	– indicates that a LIBOUT tape is not being created in this run and suppresses any attempt to access LIBOUT; this also suppresses any attempt to access the SYSPOOL required to support the DUM function (4.2.10).
	– if blank an alternate library is assumed to be present.

KEYWORD PARAMETER LST

LST=	– indicates that certain printer output options are to be performed.
D	– causes a postmortem dump to be performed at the termination of the Disc Librarian run if any errors were encountered. For a complete storage dump, SYSDUMP must be specified on the OPTION job control statement.
N	– indicates that all librarian printer output is to be suppressed after the PARAM statements are displayed.
if blank	– all librarian output is printed and no postmortem dump is performed.

Example:

1	10
// PARAM	LIN=(2,NMCL,NALT2),LST=D

This example states that the Disc Librarian can expect two disc library files, defined as LIB1 and LIB2; an alternate input source, defined as ALT1; to create a user library tape defined as LIBOUT; and to be able to use SYSPOOL as alternate storage during the operation of LIBOUT; and to provide a postmortem dump in case of any errors within the job. The same information can also be expressed by the following statements:

```
1        10                  20                  30                  40                  50
// PARAM  LIN=(2)
// PARAM  LIN=(NMCL)
// PARAM  LIN=(NALT2)
// PARAM  LST=D
```

3.2.5. DATA DELIMITERS

The start-of-data (/$) and the end-of-data (/*) statements must precede and follow, respectively, the librarian control statements that direct the execution of the Disc Librarian.

4. LIBRARIAN CONTROL STATEMENTS

4.1. GENERAL

The librarian control statements are essentially freeform and need not be defined using fixed starting columns. The operation field may start in column 1 and must be separated from the operand field by one or more blanks. The operand field consists of one or more positional parameters separated by commas, is terminated by a blank, and cannot extend past column 71. Continuation statements are not recognized (column 72 must be blank).

The operation code, written in the operation field, is constructed of a function designator and a library designator. A typical operation code is ADDL; the function designator is ADD, and the library designator is L. The operation code ADDL adds the modules specified in the operand field to the Load Library.

For librarian control statements when the function designator is stated and a lowercase x follows it, the x represents the appropriate library designator, which the programmer must supply. The library designator is compared with the file type of the declared library file; if they differ, an error message is printed. Tables 4-1 and 4-2 describe the function and library designators. A subfunction performed by the Disc Librarian does not require a library designator.

Function Designators	Description
FIL	Create files and declare as input or output
INP	Declare input file
OUT	Declare output file
ADD	Addition
DEL	Deletion
CMP	Compression
RNM	Rename
CPY	Copy
COR	Correction
DUM	Dump Disc-to-Tape
DIS	Display
PCH	Punch
PUD	Punch and Display

Table 4–1. Function Designators

Library File Designators	Description
L	Load library
R	Reserve library
C	Copy library
S	Source library
P	Proc library
Pn	Proc group within a Proc library (n = 1, 2,...,109)

Table 4-2. Library Designators

The conventions used to illustrate the librarian control statements in this manual are:

- Capital letters and punctuation marks (except braces, brackets, and ellipses) are information that must be coded exactly as shown.
- Lowercase letters and terms represent information that must be supplied by the programmer.
- Information contained within braces represents necessary entries of which one must be chosen.
- Information contained within brackets represents optional entries that are included or omitted depending on program requirements; braces within brackets signify that one of the entries must be chosen if that positional parameter is included.
- An ellipsis (a series of three periods) indicates the presence of a variable number of entries.
- Commas are required when positional parameters are omitted except for trailing parameters.

The statements FIL, INP, or OUT within a series of librarian control statements in a control stream must precede any statements requesting modifications to or copying of a library file. Any other restrictions on the placement and sequence of librarian control statements in a control stream are noted under the explanation of those statements.

To preserve file continuity, the same logical file name (LIBn) may not be declared for both input and output files concurrently. If attempted, the file declared first is set inactive during the file declaration process. The LIBUPS error processing provides for updating the LFDR and the Proc Group Descriptor Block when a fatal error is encountered. If an error occurs during updating, the user should consider the control information as being not current; the file may require reconstruction.

The librarian control statements specify and direct which functions and subfunctions of the Disc Librarian are to be executed. These functions and subfunctions may be divided into two types of services:

- Library update services
- Display and punch services

4.2. LIBRARY UPDATE SERVICES

The library update services (LIBUPS) are defined as those functions and subfunctions that alter or update the contents of a library. These services also include the function which creates an output library tape. Each LIBUPS function performed is accompanied by statements output on the line printer reflecting the actions accomplished. The librarian control statements causing the LIBUPS functions and subfunctions to be performed are described in the following paragraphs.

4.2.1. FIL STATEMENT

The FIL statement is used to initialize the specified disc file. This statement defines and formats new library files or initializes an existing library file for reuse. The LFDR information for the file is initialized and the library file directory is preformatted. File initialization on existing files overrides the existing file descriptor information and reformats the file, thus destroying the existing information.

For Proc files, the FIL statement preformats the directory and the Proc Group Descriptor Block and suballocates the necessary disc space. Proc groups must be treated like files, in that groups must be initialized individually. The Proc file must first be declared or initialized (FILP, INPP, or OUTP) before a FILPn statement may be used to initialize a Proc group within the Proc file. Once initialized, individual Proc groups need no further declaration other than declaring the parent Proc file (INPP or OUTP). The OUTP, INPP, or FILP statement causes the Proc Group Descriptor Block for the file to be retrieved or created.

In the format of the FIL statement, the lowercase x represents the appropriate library designators; the format is:

OPERATIONƀ	OPERAND
FILx	$\text{LIBn},\left\{\begin{matrix}\text{I}\\\text{O}\end{matrix}\right\},\left\{\begin{matrix}\text{numtracks}\\\text{SAME}\\\text{numtracks/numcyls}\end{matrix}\right\}$

Positional Parameter 1

LIBn	– specifies the name of the logical disc file on which the function is to be performed; where n is 1, 2, 3, 4, or 5; Proc group referencing (Pn) may be used.

Positional Parameter 2

I	– indicates that the specified file being initialized is to be used as an input library file.
O	– indicates that the specified file being initialized is to be used as the output file into which a library file is to be copied.

Positional Parameter 3

numtracks	– is a decimal number specifying the number of tracks to be allocated for directory space; a maximum of one cylinder can be allocated.
SAME	– indicates that the output file being initialized has the same file attributes as the input library file. If the definition of every Proc group is to be identical to the specified library file, the library designator must be P. The Proc file must have been created using an INPP or FILP statement before attempting to create a Proc group. For all other library files, the space allocated to the directory is made identical. The correct declaration of the library file, through the use of an INP statement, is required before using this option.
numtracks/numcyls	– is used for Proc group definitions only and is composed of two decimal numbers separated by a slash; where numtracks is as previously defined, and numcyls specifies the number of cylinders within the Proc file to be allocated to the Proc group specified by the library designator.

Examples:

1.
```
       FILR      LIB1,I,2
```
– initializes the Reserve library file LIB1 as an input file and allocates the first of two tracks of the file for directory space

2.
```
       FILP      LIB3,O
```
– initializes the Proc library file LIB3 as an output file (copy to be performed into it)

```
       FILP2     LIB3,O,1/5
```
– initializes Proc group 2 within the Proc library file LIB and allocates five contiguous cylinders to Proc group 2, the first track of which is dedicated to the directory

3.
```
       INPR      LIB1
       FILR      LIB2,O,SAME
```
– LIB1 IS THE FILE IN EXAMPLE 1

– initializes the Reserve library file LIB2 as an output file and allocates to it the same amount of directory space as is allocated to the file declared as INP (two tracks of directory space)

4.2.2. INP STATEMENT

The INP statement is used to declare the input library file on which the following functions are to be performed. Each library file must have been previously created through the use of the FIL statement. The INP statement causes LFDR information to be retrieved for the specified file. This statement is utilized when an existing file is to be updated rather than initialized. It is sufficient to declare the file to be modified and/or accessed for all functions except the copy function. For this function, an OUT statement must be defined.

An input file declared by an INP statement remains the active input file unitl another INP statement or a FIL statement creating an input file is encountered in the control stream.

In the format of the INP statement, the lowercase x represents the library designator; the format is:

OPERATIONƀ	OPERAND
INPx	LIBn

Positional Parameter 1

LIBn – specifies the name of the logical disc file on which the function is to be performed, where n is 1, 2, 3, 4, or 5.

NOTE: The library designator (x) is restricted to L, R, C, S, or P. Proc group referencing (Pn) is not permitted. (To process any of the Proc groups contained in a particular Proc file, it is sufficient to declare the Proc file itself by INPP LIBn.) The LFDR for Proc groups is automatically retrieved and set active when the library identifier warrants it (P1 references are changed to P2).

Example:

```
 INPS     LIB3          -FILE DEFINED BY LFD LIB3 IS MADE ACTIVE
 ADDS     MOD1           FOR SUBSEQUENT ACCESSING OR UPDATING
 DISS     MOD1
```

4.2.3. OUT STATEMENT

The OUT statement is used to declare the library file into which a file is to be copied. The file being copied may be either a disc file previously declared by the INP statement or a file on an alternate tape source. Each library file must have been previously initialized through the use of the FIL statement (although not necessarily in the same run). The OUT statement is used when an existing file is to be added rather than initialized.

The OUT statement is used only in conjunction with the CPY statement. Upon completion of the copy process, the file specified in the OUT statement is purged from main storage and set inactive. The LFDR information for the specified file is retrieved and is dumped to disc. Further accessing of this file may be accomplished only after the file has been reactivated through the use of an INP, OUT, or FIL statement.

In the format of the OUT statement the lowercase x represents the library designator; the format is:

OPERATIONb	OPERAND
OUTx	LIBn

Positional Parameter 1

LIBn – specifies the name of the logical disc file on which the function is to be performed; where n is 1, 2, 3, 4, or 5.

NOTE: The library designator (x) is restricted to L, R, C, S, or P; to process any of the Proc groups contained in a particular Proc file, it is sufficient to declare the Proc file itself by OUTP LIBn.)

Example:

```
 OUTP     LIB2          -FILE DEFINED BY LFD LIB2 IS MADE
 CPYP1    (ALT2)         ACTIVE FOR A LATER CPY FUNCTION
```

4.2.4. ADD STATEMENT

The ADD statement is used to add a module or a group of modules to a specified library file. Any number of modules may be added during a single run. If an added module has the same name as a module resident in the library, the resident module is replaced. The directory entry for the resident module is marked obsolete and the new module is added to the end of the file. (Proc modules have an entry for each NAME assembler directive.) If a module is added to a Proc file using an existing name (not a first NAME entry) for a Proc module, an error occurs and the module is not added. This prevents faulty referencing.

When adding a Proc module from an alternate disc library source, the module must be referenced by its first NAME assembler directive. If a module resident in the specified Proc group has the same name as the module being added, the resident module is replaced. Each subsequent NAME entry for the Proc module is marked obsolete in the directory. Because reference is made only to the first NAME directive, care must be taken to ensure that the subsequent NAME directives in the module being added are not already resident in the directory. Otherwise, duplicate name entries cause faulty referencing later. When adding a Proc module, a Proc group must be specified (Pn); the library designator may not specify the Proc file (P) for this function.

When adding a Proc module from an alternate tape library source, the module must be referenced by the name in its tape header record. If the same module already exists on disc and the purpose of the addition is to replace or update it the existing module must be deleted from the file (DEL statement) before it is updated. (The existing module will not be deleted by virtue of the addition even though the new module is entered in the directory with the same NAME directive unless the Proc header name coincides with the first NAME directive. Card input does not require any specific form of referencing; if a Proc module is to be added from card, it is referenced by its first NAME directive. Modules added from card require the ENDCARD delimiter.

In the format of the ADD statement, the lowercase x represents the library designator; the format is:

```
OPERATION␢ |                      OPERAND
-----------+--------------------------------------------------------------------
           |
           |          ⎡⎧ (ALT1)                                  ⎫⎤
           |          ⎢⎪ (ALT2)                                  ⎪⎥
           |          ⎢⎪ (ALT1, group)                           ⎪⎥
           |          ⎢⎪ (ALT2, group)                           ⎪⎥
           |          ⎢⎪ (CARD)                                  ⎪⎥
           |          ⎢⎪ (MCL)                                   ⎪⎥
           |          ⎢⎪                                         ⎪⎥
ADDx       | name-1   ⎢⎨                ⎡⎧(ALT1)⎫⎤               ⎬⎥   [,name-2...n]
           |          ⎢⎪ /new-name-1    ⎢⎨(ALT2)⎬⎥               ⎪⎥
           |          ⎢⎪                ⎣⎩(MCL) ⎭⎦               ⎪⎥
           |          ⎢⎪                ⎡⎧(ALT1)        ⎫⎤       ⎪⎥
           |          ⎢⎪                ⎢⎪(ALT2)        ⎪⎥       ⎪⎥
           |          ⎢⎪   .ALL         ⎢⎨(ALT1, group) ⎬⎥       ⎪⎥
           |          ⎢⎪                ⎢⎪(ALT2, group) ⎪⎥       ⎪⎥
           |          ⎣⎩                ⎣⎩(MCL)         ⎭⎦       ⎭⎦
```

Positional Parameter 1

name-1	– specifies the name of the module to be added.
,ALL	– indicates gang operation; all modules starting with the same one to seven leading characters specified by name-1 are added to the library.
/new-name-1	– specifies the new name for the module when it is added to the library; this option may not be used for a module within a Proc file, or within Source or Copy files if card input is requested.
(ALT1) (ALT2)	– indicates that the modules to be added are in an alternate library, which may be on either tape or disc; this specification may be used for Proc libraries when Proc groups are the input file and alternate source match.
(CARD)	– indicates that the module to be added is on punched cards; an ENDCARD must follow the last punched card (gang operation is not permitted from cards).
(MCL)	– indicates that the module to be added is in the module complex library.
(ALT1,group) (ALT2,group)	– indicates that the module to be added is in an alternate library within the Proc group specified by group, where group is a decimal number and need not correspond with the library designator.

Positional Parameter 2

name-2	– the format of this parameter and any subsequent parameters is an iteration of positional parameter 1.

If the input source is not specified, the following sources are assumed:

Library File Type	Input Source
Load	MCL
Reserve	MCL
Copy	CARD
Source	CARD
Proc	CARD

Example:

```
INPP    LIB1            -DECLARE PROC FILE
ADDP1   CHAR(ALT2,2)    -ADD PROC MODULE WITH FIRST
  •                      NAME DIRECTIVE OF CHAR IN
  •                      PROC GROUP 2 FROM ALTERNATE
  •                      DISC LIBRARY 2 TO PROC GROUP
  •                      1 IN LIB1
INPL    LIB2            -DECLARE LOAD FILE
ADDL    GANG•ALL        -ADD ALL MODULES WITH PREFIX
                         OF GANG FROM THE MCL
```

4.2.4.1. VER SUBFUNCTION STATEMENT

The VER subfunction statement associated with the ADD statement is used to create a version number in the module header record to differentiate between the versions of source code. The VER statement is optional and can be applied only to Copy, Source, and Proc libraries with punched card input. The VER subfunction statement must follow immediately the ADD statement. If more than one module is specified in the ADD statement, a second VER subfunction statement follows the source statement of the module specified by positional parameter 1 in the ADD statement.

The format of the VER subfunction statement is:

OPERATIONƀ	OPERAND
VER	[level-number][,update-number]

Positional Parameter 1

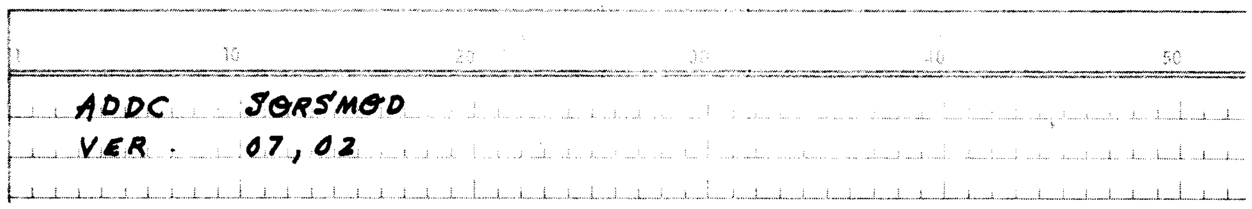
level-number	– specifies the new level number (00-99) in the first byte of the version number; leading zeros must be included.
if blank	– zero is assumed.

Positional Parameter 2

update-number	– specifies the new update number (00-99) in the second byte of the version number; leading zeros must be included.
if blank	– zero is assumed.

NOTE: When a VER subfunction statement is not supplied for Copy or Source libraries and source cards are those of a new file, it is assumed that the version number field should be zero. Blanks in either the level- or update-number field are converted to zeros.

Example:



```
1        10        20        30        40        50
   ADDC     SORSMOD
   VER      07,02
```

4.2.5. DEL STATEMENT

The DEL statement is used to delete a module, a group of modules, a Proc group, or an entire library file. Any number of modules may be deleted during a single run. The deletions are flagged obsolete in the directory, but are not cleared out of the file. Obsolete modules may be eliminated and disc space made maximal through the use of the compress function. (See 4.2.6.) When entire library files are deleted, the disc space is returned to the system for reallocation (i.e., Format 1 label is scratched). For Proc processing, either a particular group may be deleted by specifying Pn as the library designator, or an entire file may be deleted by specifying P as the library designator.

In the format of the DEL statement, the lowercase x represents the library designator; the format is:

OPERATIONƀ	OPERAND
DELx	$\left\{\begin{array}{l}\text{name-1}\\\text{name-1.ALL}\\\text{ALL}\end{array}\right\}\left[,\left\{\begin{array}{l}\text{name-2}\\\text{name-2.ALL}\end{array}\right\}\cdots\right]$

Positional Parameter 1

name-1	– specifies the name of a module to be deleted.
name-1.ALL	– specifies gang operations; all modules starting with the same one to seven leading characters specified by name-1 are deleted from the library.
ALL	– indicates that the entire library file is to be deleted.

Positional Parameter 2

name-2	– specifies the name of a second module to be deleted.
name-2.ALL	– specifies that gang operations are to be applied when deleting the module specified by name-2.

Example:

```
  DELS   A.ALL          DELETES ALL MODULES WITH PREFIX A
                        FROM SOURCE FILE (PREVIOUSLY
                        DECLARED AS INP)
```

4.2.6. CMP STATEMENT

The CMP statement is used to compress a library file (or Proc group) by removing the obsolete modules from the file (or Proc group). The compress function can make available only as much space as is occupied by obsolete modules; compressing a file with no obsolete modules does nothing to the file.

When an obsolete module is removed from the file, the index entry in the directory for that module is also removed. After a module has been removed, the remaining modules in the file are shifted to fill the spaces, thus eliminating any fragmentation in the file. Up to five library files may be compressed any number of times during a single run.

For Proc processing, either a particular Proc group may be compressed by specifying Pn or an entire Proc file may be compressed by specifying P as the library designator.

Because the compress function restructures the file onto itself, any abnormal termination of the compression may yield an incomplete file structure. After displaying the directory of the file, all modules preceding the module listed in the error field may be extracted from the file and the remaining portion of the file must be re-created. Otherwise, the entire file must be restored.

In the format of the CMP statement, the lowercase x represents the library designator; the format is:

OPERATIONƀ	OPERAND
CMPx	

No positional parameters are required for this function.

Example:

```
1        10        20        30        40        50
   INPL    LIB1         - DECLARES LOAD FILE LIB1 AS INPUT
   CMPL                 - COMPRESSED LOAD FILE
   INPP    LIB2         - DECLARES PROC FILE LIB2 AS INPUT
   CMPP3                - COMPRESSES PROC GROUP 3 IN LIB2
```

4.2.7. RNM STATEMENT

The RNM statement is used to rename a module. The change is posted in the directory and header record. Any number of modules may be renamed during a single run. Care should be taken not to duplicate directory entries by renaming a module with an existing name in the file, because the Disc Librarian does not check for duplicate names. The allowable library designator (x) is L, R, C, or S. The library designators P and Pn may not be used with this function.

Both the old and the new name for a module may consist of one to eight alphanumeric characters.

In the format of the RNM statement, the lowercase x represents the library designator; the format is:

OPERATIONƀ	OPERAND
RNMx	oldname/newname[,oldname/newname, . . .]

Positional Parameter 1

oldname/	–	is the name of the module to be renamed.
newname	–	is the name to which the module is to be changed.

Positional Parameter 2

This and any subsequent parameters have the same definition as positional parameter 1.

Example:

```
INPS   LIB1                -DECLARES SOURCE FILE LIBI AS INPUT
RNMS   SRCMOD/SOURCEMD     -RENAMES THE SOURCE MODULE FROM
                            SRCMOD TO SOURCEMD
```

4.2.8. CPY STATEMENT

The CPY statement is used to copy an entire library file or Proc groups in separate files. This function automatically provides for file compression. This function may not be used to copy individual modules.

The disc-to-disc copy function requires both INP and OUT statements (or the FIL statement equivalents). The INP statement designates the file which is to be copied into the file specified by the OUT statement. At the termination of the copy processing, LFDR information of the output file is purged from main storage (the input file remains active). If a series of copies into the same library is desired, an OUT statement must be declared preceding the associated CPY statement. If tape input is to be copied into the output file, the tape must be defined as an alternate file instead of as an input file.

For disc input files, a Proc group may be copied by specifying Pn as the library designator, or the entire Proc file may be copied by specifying P as the library designator. Only the active Proc groups are copied; undefined groups are not copied.

For tape input files, only Proc groups (Pn) may be copied; P is an invalid library designator.

In the format of the CPY statement, the lowercase x represents the library designator; the format is:

OPERATIONb	OPERAND
CPYx	$\left\{\begin{matrix}\text{group}\\\text{(ALT1)}\\\text{(ALT2)}\\\text{(ALT1,group)}\\\text{(ALT2,group)}\end{matrix}\right\}$

Positional Parameter 1

group	– is the decimal number specifying the Proc group to be copied from the input file; this specification is used for Proc processing only when the number of the group being copied is different from the number of the group into which it is copied.
(ALT1) (ALT2)	– indicates that the file to be copied is in an alternate library; only tape libraries are acceptable as ALT1 or ALT2.
(ALT1,group) (ALT2,group)	– indicates that the Proc group specified by group is to be copied from the alternate tape library, where group is a decimal number; this option applies only to Proc groups and accepts only tape libraries.

If the input and output group numbers are the same, group need not be specified.

if blank — used for disc-to-disc copies of Load, Reserve, Copy, and Source files, and for Proc files when Proc group numbers coincide.

NOTE: When merging two files through use of the CPY statement, duplicate module names are not detected by the Disc Librarian.

Examples:

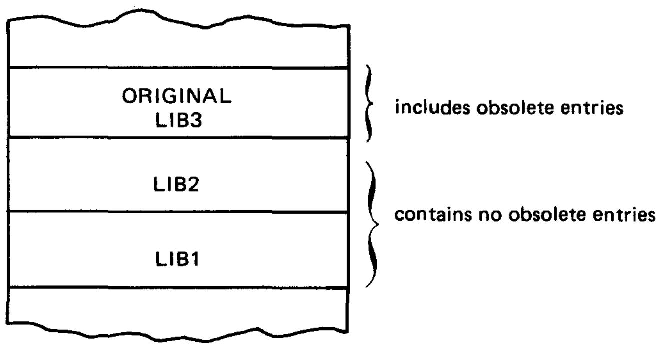
①
```
   INPR  LIB1     -ALL ACTIVE ENTRIES IN LIB1
   OUTR  LIB2      AND LIB2 ARE MERGED TO LIB3
   CPYR
   INPR  LIB2
   OUTR  LIB3
   CPYR
```

②
```
   INPP  LIB1     -THIS EXAMPLE COPIES ALL ACTIVE
   OUTP  LIB2      ENTRIES -- PROC GROUP 1 FROM
   CPYP2 1         LIB1 TO PROC GROUP 2 IN LIB2 --
   OUTP  LIB2      PROC GROUP 3 FROM ALT1 TO PROC
   CPYP2 (ALT1,3)  GROUP 2 IN LIB2
```

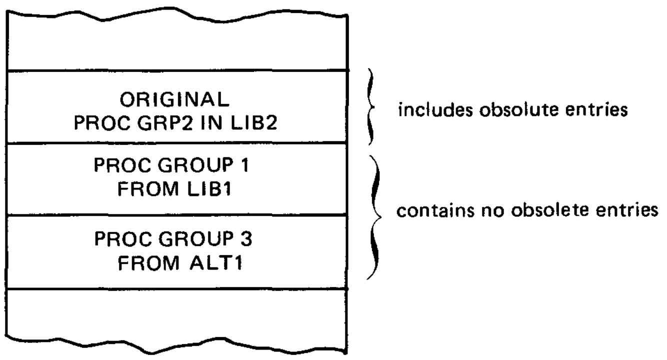
③
```
   OUTL  LIB2     -COPY LOAD MODULES IN LOAD LIBRARY
   CPYL  (ALT1)    ON ALTERNATE TAPE FILE, ALT1,
                   INTO LIB2
```

The resultant library structures produced by the immediately preceding examples are:

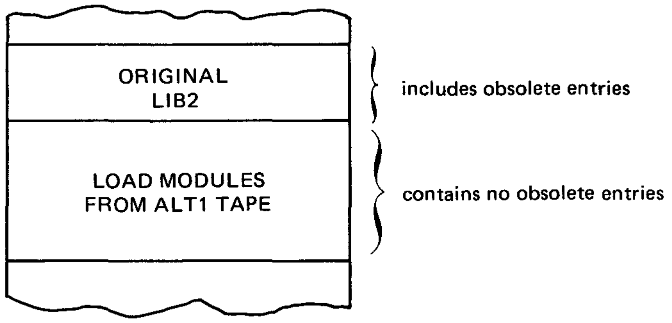
① LIB3



② P2 in LIB2


ORIGINAL PROC GRP2 IN LIB2
PROC GROUP 1 FROM LIB1
PROC GROUP 3 FROM ALT1
includes obsolute entries
contains no obsolete entries

③ LIB2


ORIGINAL LIB2
LOAD MODULES FROM ALT1 TAPE
includes obsolete entries
contains no obsolete entries

4.2.9. COR STATEMENT

The COR statement, with its associated subfunction statements, is used to correct lines of source code within Source, Copy, and Proc libraries and optionally to update their version number by means of a VER subfunction statement. Lines may be added, deleted, or replaced. A modular display of the updated module is provided by the COR function. The updated module is added to the end of the file, a new index entry is added at the end of the directory, and the index entry for the original module is flagged obsolete. Any number of modules may be corrected during a single run. The library designator P is not permitted; reference must be made to a specific Proc group (Pn) for this function. If an alternate library is specified, it must not be the input library.

Caution must be exercised when rerunning a LIBUPS job after an error has occurred; the initial execution of the job (or partial execution) may have caused the source line number references to have been updated.

In the format of the COR statement, the lowercase x represents the library designator; the format is:

OPERATIONb̸	OPERAND
CORx	progname $\left[\begin{Bmatrix}\text{(ALT1)}\\\text{(ALT2)}\\\text{(ALT1,group)}\\\text{(ALT2,group)}\end{Bmatrix}\right]$

Positional Parameter 1

progname – identifies the library module to be corrected.

(ALT1) (ALT2)	– indicates that the module to be corrected is to be found in an alternate Disc library.
(ALT1,group) (ALT2,group)	– indicates that the module to be corrected is to be found in an alternate Disc library within the Proc group specified by group, where group is a decimal number.

NOTE: If the input source is not specified, card input is assumed.

Example:

```
   INPS  LIBS        -DECLARES INPUT SOURCE LIBRARY
   CORS  ASEM(ALT2)  -CORRECTS MODULE ASEM LOCATED IN
                      ALT2, ADDS NEW VERSION AT END
                      OF LIBS. MODULE IN ALT2 IS NOT
                      CHANGED.
```

4.2.9.1. VER SUBFUNCTION STATEMENT

The VER subfunction statement associated with the COR statement is used to create a version number to differentiate between versions of source code. This subfunction applies only to input from punched cards. The VER subfunction statement must be placed immediately following the COR statement.

The format of the VER subfunction statement is:

OPERATIONb	OPERAND
VER	[level-number][,update-number]

Positional Parameter 1

level-number	– specifies the new level number (00-99) in the first byte of the version number; leading zeros must be included.
if blank	– zero is assumed.

Positional Parameter 2

update-number	– specifies the new update number (00-99) in the second byte of the version number; leading zeros must be included.
if blank	– zero is assumed.

NOTE: When a VER subfunction statement is not supplied, the update number is automatically incremented by 1 for card input; blanks in the level or update number field are converted to zeros.

Example:

```
   CORP1 PROCNAME
   VER   03,09       -ASSIGN NEW VERSION NUMBER TO
                      MODULE PROCNAME IN PROC GROUP 1
```

4.2.9.2. INS SUBFUNCTION STATEMENT

The INS subfunction statement is used to insert one or more lines of source statements in a module. The format of the INS subfunction statement is:

OPERATIONƀ	OPERAND
INS	line-number

Positional Parameter 1

line-number — is a decimal number specifying the line number after which the source statements are to be inserted; the source statements must follow the INS statement and are delimited by the next ENDCARD statement.

Example:

```
1        10        20        30        40        50        60
  CØRS   DISC
  INS    3                  -INSERTS SØURCE SØURCE STATEMENTS AFTER
                             LINE NUMBER 3 IN MØDULE DISC
      søurce statements

  ENDCARD
```

4.2.9.3. REP SUBFUNCTION STATEMENT

The REP subfunction statement is used to replace or delete one or more lines of source statements in a module. If no source statements follow the REP statement, the specified lines are deleted. Both positional parameters must be specified. The statements to replace the lines specified must follow the REP subfunction statement and are delimited by the next ENDCARD statement.

The format of the REP subfunction statement is:

OPERATIONƀ	OPERAND
REP	first-line,last-line

Positional Parameter 1

first-line — is a decimal number specifying the first line of source code to be replaced or deleted.

Positional Parameter 2

last-line — is a decimal number specifying the last line of source code to be replaced.

NOTE: first-line and last-line may be equal, thereby replacing or deleting only one line of source code.

Examples of some of the functions that may be performed by the REP subfunction statement are given in Table 4–3.

Task	Sequence of Control Statements
Line A to precede line 1	REP 1,1 Source statement A (to be added) Source statement originally in line 1 ENDCARD
Delete line 3	REP 3,3 ENDCARD
Replace five lines with one line	REP 11,15 MVC WMP,EGP ENDCARD

Table 4-3. Examples of Functions Performed by Use of the REP Statement

4.2.9.4. ENDCARD SUBFUNCTION STATEMENT

The ENDCARD subfunction statement is used to terminate the source statements when adding source cards or is used with the INS or REP subfunction statements. If the ENDCARD subfunction statement immediately follows the REP subfunction statement, the lines specified on the REP subfunction statement are deleted.

The format of the ENDCARD subfunction statement is:

OPERATIONb̸	OPERAND
ENDCARD	

The ENDCARD subfunction statement requires no positional parameters.

4.2.10. DUM Statement

The DUM statement provides the capability of dumping disc library files to tape in a format that is acceptable to both Disc and Tape Librarians for subsequent processing. All five types of library files may be dumped during a single library run, but they must be dumped in the correct sequence of Load, Reserve, Copy, Source, and Proc libraries. All active modules in the disc library directory are dumped on the output tape in ascending alphanumeric sequence. Nullfiles are generated as required. Entire Proc files or Proc groups may be dumped. Different Proc groups may be dumped from discrete Proc files onto a common output tape. If individual elements are required, they must be placed in a temporary library file by adding from an alternate file. These elements can then be dumped from the temporary library file.

In the format of the DUM statement, the lowercase x represents the library designator; the format is:

OPERATIONb̸	OPERAND
DUMx	

Processing time for the dump function is reduced significantly if additional storage is allocated in excess of the required X'3C00' bytes. The number of sort/merge passes on the data is thereby reduced.

Table 4–4 reflects the maximum disc library file size which may be dumped onto a standard 2400 foot tape reel at the indicated recording densities. The maximum number of tape blocks are approximate and include a 5% tolerance for wasted tape space.

Number of Tape Channels	Recording Density*	Maximum Number		
		Tape Blocks	Cylinders	
			8411	8414
7	200	7300	104	28
7	556	15000	All	57
7	800	18000	All	69
9	800	23600	All	90
9	1600	31000	All	119

*frames per inch

Table 4–4. Tape Capacity in Terms of Disc Cylinders

4.3. DISPLAY AND PUNCH SERVICES

The display and punch services are defined as those functions of the Disc Librarian used to retrieve information from a given library and to display it as printed output and/or convert a module into a punched card output deck. The display and punch services are executed under the control of LIBUPS and may be interspersed with the functions of the Library Update Services.

The control stream for the display and punch services requires an INP librarian control statement and a PARAM job control statement which specifies the maximum number of files. (The number of files must be the same as the higher number specified by the LIBn parameter on the LFD job control statement.)

If an entire Proc file is to be displayed and/or punched, the library designator is P, the option ALL is specified as positional parameter 1, and no other parameters are required.

The librarian control statements causing the display and punch functions to be performed are described in the following paragraphs.

4.3.1. DIS STATEMENT

The DIS statement is used to display (print out) a module, a group of modules, the directory, or the entire library file. Any number of modules may be displayed in a single run. The display function always effects the extraction of information from the module header record to be displayed under appropriate fields in the page headers. (See Figure 2-4.) For module displays, this information is followed by the module in the main body of the library.

The format of the DIS statement is:

OPERATIONƀ	OPERAND
DISx	$\left\{\begin{array}{l}\text{name-1}\\ \text{name-1.ALL}\\ \text{DIR}\\ \text{ALL}\end{array}\right\}\left[,\left\{\begin{array}{l}\text{name-2}\\ \text{name-2.ALL}\\ \text{DIR}\end{array}\right\}\right],\ldots$

Positional Parameter 1

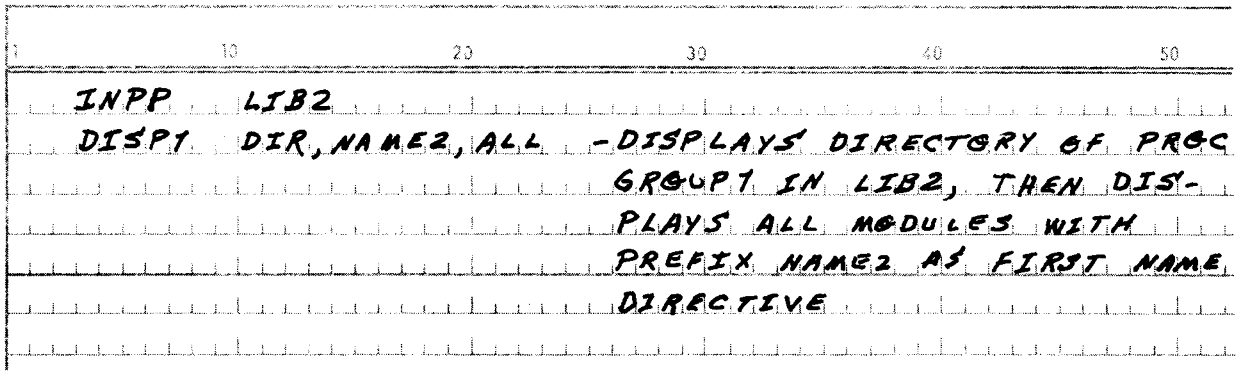
name-1	– specifies the name of the module to be displayed.
name-1.ALL	– specifies gang operation; all modules starting with the same characters specified by name-1 are displayed.
DIR	– indicates that the library directory is to be displayed.
ALL	– indicates that the entire library is to be displayed; Proc groups or the entire Proc file may be specified.

Positional Parameter 2

name-2	– specifies the name of another module to be displayed.
name-2.ALL	– specifies that gang operations are to be applied when displaying the module specified by name-2.
DIR	– indicates that the library directory for the module specified by name-2 is to be displayed.

NOTE: ALL may be specified only as positional parameter 1.

Example:



```
1        10        20        30        40        50
  INPP   LIB2
  DISP1  DIR,NAME2,ALL  -DISPLAYS DIRECTORY OF PROC
                          GROUP1 IN LIB2, THEN DIS-
                          PLAYS ALL MODULES WITH
                          PREFIX NAME2 AS FIRST NAME
                          DIRECTIVE
```

When the DIR option is selected, the remaining file space is calculated and printed out at the end of the directory display. The format is:

RESIDUAL FILE SPACE: xx CYLINDERS AND yy TRACKS

where:

xx is the number of cylinders, in hexadecimal.

yy is the number of tracks, including partial tracks, in hexadecimal.

4.3.2. PCH STATEMENT

The PCH statement is used to convert a module in a library into a punch card output deck. This statement may be used to punch a module, a group of modules, or an entire library file. Any number of modules may be punched in a single run.

The format of the PCH statement is:

OPERATIONb	OPERAND
PCHx	$\left\{\begin{array}{l}\text{name-1}\\ \text{name-1.ALL}\\ \text{ALL}\end{array}\right\}\left[,\left\{\begin{array}{l}\text{name-2}\\ \text{name-2.ALL}\end{array}\right\},\ldots\right]$

Positional Parameter 1

name-1	– specifies the name of the module to be punched.
name-1.ALL	– specifies gang operation; all modules starting with the same characters specified by name-1 are punched.
ALL	– indicates that the entire library is to be punched; Proc groups or the entire Proc file may be specified.

Positional Parameter 2

name-2	– specifies the name of a second module to be punched.
name-2.ALL	– specifies that gang operations are to be applied when punching the module specified by name-2.

NOTE: ALL may be specified only as positional parameter 1.

Example:

```
 INPL    LIB4
 PCHL    LIBRY001       -PUNCHES MODULE LIBRY001 FROM
                          LOAD LIBRARY LIB4
```

4.3.3. PUD STATEMENT

The PUD statement is used to display a module as printed output and to convert the module into a punched card output deck. The display and punch function may be used to display and punch a module, a group of modules, or an entire library file. Any number of modules may be displayed and punched in a single run.

The format of the PUD statement is:

OPERATION	OPERAND
PUDx	$\left\{\begin{array}{l}\text{name-1}\\ \text{name-1.ALL}\\ \text{ALL}\end{array}\right\}\left[,\left\{\begin{array}{l}\text{name-2}\\ \text{name-2.ALL}\end{array}\right\},\ldots\right]$

Positional Parameter 1

name-1	– specifies the name of the module to be displayed and punched.
name-1.ALL	– specifies gang operation; all modules starting with the same characters specified by name-1 are displayed and punched.
ALL	– indicates that the entire library is to be displayed and punched; Proc groups or the entire Proc file may be specified.

Positional Parameter 2

name-2	– specifies the name of a second module to be displayed and punched.
name-2.ALL	– specifies that gang operations are to be applied when displaying and punching the module specified by name-2.

Example:

```
   INPS   LIB3
   PUDS   ASM.ALL,BASM.ALL   -PUNCHES AND DISPLAYS
                              FROM LIB3 ALL MODULES
                              WITH PREFIXES OF ASM
                              AND BASM
```

APPENDIX A. CONTROL STREAM EXAMPLE

This appendix gives an example of a Disc Librarian run using five input library files and performing a number of functions. The control stream in the example defines two alternate input sources (one is disc; the other, tape); an output tape; and a punch. The example is well commented as to what is being performed. Where lowercase letters are used in the coding, the user must supply the appropriate logical unit numbers or identifiers.

```
LABEL    OPERATION OPERAND                                        COMMENTS
1        10    16                                                                  72      80
// JOB   DISCLIB,,,6000                                           STORAGE IN EXCESS OF 3C00 IS UTILIZED BY DUM FUNCTION
// DVC   u1 // LFD PUNCH
// DVC   u2 // LFD PRNTR
// DVC   t1 // LFD ALT1                                           ALTERNATE TAPE INPUT SOURCE
// DVC   t2 // VOL SPTttt // LFD LIBOUT                           OUTPUT TAPE
// DVC   d1 // VOL DSPxxx                                         ALTERNATE DISC INPUT SOURCE
// LBL   PROCLIB1,DSPxxx
// LFD   ALT2
// DVC   d1 // VOL DSPxxx                                         DISC LIBRARY FILE #1
// LBL   SOURCLIB,DSPxxx,1,76330,71066
// LFD   LIB1
// DVC   d2 // VOL DSPyyy                                         DISC LIBRARY FILE #2
// LBL   PROCLIB1,DSPyyy
// LFD   LIB2
// DVC   d3 // VOL DSPzzz                                         DISC LIBRARY FILE #3
// LBL   RESVLIB1,DSPzzz
// LFD   LIB3
// DVC   d3 // VOL DSPzzz                                         DISC LIBRARY FILE #4
// LBL   LOADLIB1,DSPzzz
// LFD   LIB4
// DVC   d1 // VOL DSPxxx                                         DISC LIBRARY FILE #5
// EXT   C,,TRK,1,C,,CYL,20                                       ALLOCATES NEW FILE
// LBL   PROCLIB2,DSPxxx,1,76330,72217                               NOTE: THIS FILE MAY BE USED ONLY AS A PROC FILE (2 EXTENTS).
// LFD   LIB5,,,NEW
// DVC   d3 // VOL DSPzzz // LFD SYSPOOL                          SYSPOOL REQUIRED FOR DUM FUNCTION
// EXEC  LIBUPS,LOAD$LIB,,REL
// PARAM LIN=(5,NMCL),LST=D                                       SPECIFIES 5 LIBRARY FILES, 2 ALTERNATE INPUTS, TAPE OUTPUT, PM-DUMP IF ERROR
/$
INPP     LIB2                                                     DECLARES PROC FILE, LIB2, AS BASE FILE
FILP     LIB5,0,SAME                                              INIT LIB5 WITH SAME CHARACTERISTICS LIB2
```

LABEL 1	OPERATION 10	OPERAND 16	COMMENTS
CPYP3	1		COPIES PROC GROUP 1(LIB2) INTO GROUP 3(LIB5)
INPP	LIB5		DECLARES PROC FILE, LIB5, AS BASE FILE
ADDP3	GUNC(ALT2,2)		* ADDS PROC FROM GROUP 2 IN ALT2(LIB2) TO GROUP 3
CORP3	CANCEL(ALT2,3)		* ADDS CORRECTED VERSION TO GROUP 3 FROM GROUP 2(ALT2)
VER	09,11		NEW VERSION IS LEVEL 9, UPDATE 11
REP	7,7		REPLACES LINE 7
ENDCARD			DELIMITER FOR CORRECT FUNCTION
INPL	LIB4		DECLARES LOAD FILE, LIB4, AS BASE FILE
ADDL	SQROOT·ALL(ALT1)		ADDS MODULES (PREFIX=SQROOT) FROM ALT1 (TAPE)
DELL	CONVRT07		DELETES MODULE NAMED CONVRT07
RNML	RDTAPE00/TAPE1000		RENAMES RDTAPE00 – NEW NAME IS TAPE1000
CMPL			COMPRESSES LOAD LIBRARY
DUML			DUMPS LOAD LIBRARY ON LIBOUT IN TAPE LIBRARY FORMAT
INPR	LIB3		DECLARES RESERVE FILE, LIB3, AS BASE FILE
DUMR			DUMPS RESERVE FILE ON LIBOUT IN TAPE LIBRARY FORMAT
INPS	LIB1		DECLARES SOURCE FILE, LIB1, AS BASE FILE
ADDS	BINTOHEX		ADDS SOURCE MODULE, BINTOHEX, FROM CARDS
VER	,07		VERSION IS LEVEL 0, UPDATE 7
SOURCE STATEMENTS			
ENDCARD			DELIMITER FOR ADD FUNCTION
ADDS	CONV·ALL(ALT1)		ADDS MODULES (PREFIX=CONV) FROM ALT1 (TAPE)
CORS	CARDREAD		CORRECTS SOURCE MODULE, CARDREAD. UPDATE INCRE BY 1
REP	23,23		REPLACES LINE 23 WITH FOLLOWING:
	BNE	*+12	
ENDCARD			DELIMITER FOR CORRECT FUNCTION
REP	31,33		REPLACES LINES 31 THROUGH 33
ENDCARD			DELIMITER FOR CORRECT FUNCTION
INS	35		INSERTS, AFTER LINE 35, THE FOLLOWING:
	AI	COUNT,1	
ENDCARD			DELIMITER FOR CORRECT FUNCTION

```
LABEL     ␢ OPERATION ␢     OPERAND                       ␢      COMMENTS
1           10        16                                                          72        80
DELS        IO·ALL                           DELETES MODULES (PREFIX=IO) FROM SOURCE FILE
CMPS                                         COMPRESSES SOURCE LIBRARY
DISS        DIR                              DISPLAYS SOURCE DIRECTORY & SPACE REMAINING INFO
PUDS        ALL                              PUNCHES/DISPLAYS ALL SOURCE MODULES IN FILE
INPP        LIB5                             DECLARES PROC FILE, LIB5, AS BASE FILE
DISP        DIR                              DISPLAYS DIRECTORIES OF ALL PROC GROUPS IN LIB5
DUMP                                         DUMPS ALL PROC GROUPS TO LIBOUT TAPE. THOSE PROC GROUPS
/*                                           WHICH ARE EMPTY WILL APPEAR AS NULLFILES ON TAPE
// DELETE
/&
                                             * THE PROC MODULES ARE REFERENCED BY THEIR FIRST
                                               NAME ASSEMBLER DIRECTIVES BECAUSE THEY ARE BEING
                                               ADDED/CORRECTED FROM DISC
```

APPENDIX B. ERROR CODE DISCRIPTIONS

This appendix lists the error codes that may be output on the line printer during the execution of the Disc Librarian. The error codes are listed and described by type and designator. The suggested recovery procedures also are given for each error code.

D 0 Description: File accessed by the preceding control statement is locked out because of a previous file error. The entire control statement is being ignored.

Suggested Recovery Procedure: Refer to previously reported error.

D 1 Description: An insufficient number of files (INP, OUT, ALT1, ALT2, MCL, LIBOUT, or SYSPOOL), have been declared to support the current operation. The expression in the error field is being ignored.

This error code is also used to mean the library designator on the control statement does not match the file type.

Suggested Recovery Procedure: Check control stream for file declaration statement and PARAM statements for incorrect parameters or check for mispunching of library designator on control statement.

D 2 Description: Proc file was not properly defined by means of the OUTP, INPP, or FILP statement before a Proc group was referenced. It is also used when an error or an INP, OUT, or FIL statement is encountered.

This error code is also used to mean LIBn declared for Proc file does not match LIBn declared for Proc group.

Suggested Recovery Procedure: Check librarian control stream, paying special attention to the correct usage of INP, OUT, and FIL statements.

D 3 Description: File size is too small to contain the file or group being constructed.

Suggested Recovery Procedure: Reinitialize file by means of the FIL statement.

Copy file. (For the Proc files, this removes obsolete Proc groups from the file.)

Reallocate file (enlarging it) and initialize it.

D 4 Description: Invalid SAME option. The INP file is not active or has not been declared.

Suggested Recovery Procedure: Check control stream; INP file must be active and declared.

D 5 Description: File listed in the error field was not found; file is locked out from further referencing for the remainder of the job step.

The file name appears in the error field, along with a further breakdown of the error which may be:

01	Volume serial number not found in PUB or wrong volume mounted.
02	File identification not found in VTOC; label field in error or file not on volume.
10	FCB not found, file not defined using LFD job control statement.
20	Error encountered while searching for MCL. Either no MCL was found within SYSPOOL or an error in GIVE, TAKE, or QUERY processing was encountered. (See *UNIVAC 9400 Supervisor Programmer Reference, UP-7689* (current version) under "Dynamic Allocation of Direct Access Storage".)
30	LBL job control statement not provided; file identification is indeterminate.
40	File serial number of LBL job control statement does not match format 1 label.
50	Creation date on LBL job control statement does not match format 1 label.

Suggested Recovery Procedure: Check job control statements for accuracy or verify the volume which is mounted.

D 6 Description: Format error was encountered in file listed in error field; file is locked out from further referencing for the remainder of the job step. Current operation listed in control field is incomplete. Possible causes are:

- Proc file does not consist of two extents.
- The extent containing the Proc group descriptor block exceeds one track. The file must be reallocated so that the extent for the Proc group descriptor block consists of a single track.
- Accessing a file not initialized by means of the FIL statement.
- At least part of a file has been destroyed, causing the format to change; to recover, initialize file (after extracting all possible modules outside of the affected track).
- No-record-found condition as a result of a hardware seek failure.
- Load, Reserve, Copy, or Source library file has more than one extent.
- A file which does not begin on a cylinder boundary. (This does not apply to the first extent of a Proc file.)

Suggested Recovery Procedure: Check control stream for FIL statement. Make certain that a Proc file consists of two extents and that the second extent begins on a cylinder boundary. All other file types consist of only one extent which must begin on a cylinder boundary. Extract all possible modules outside of affected track and initialize file.

D 7 Description: Unrecoverable error was encountered while processing file listed in the error field; the file is locked out from further referencing for the remainder of the job step. Current operation listed in control field is incomplete.

Suggested Recovery Procedure: Module being processed at time of error is incomplete. File control information has not been updated. Retry job. If error recurs, it may be necessary to re-initialize the file.

D 8 Description: Unable to process file listed in error field due to previously detected error.

This error also occurs when an attempt has been made to access a library file. LIB2–LIB5 (by means of an INP, OUT, or FIL statement) which has not been included in the numfil field of the LIN parameter.

Suggested Recovery Procedure: Refer to the previously reported error or check LIN parameter for the correct numfil field.

D 9 Description: Parity error encountered on file listed in error field; the file is locked out from further processing for the remainder of the job step. Current operation listed in control field is incomplete.

Suggested Recovery Procedure: Extract data outside of affected track. If error recurs, it may be necessary to re-prep the affected track.

D A Description: The alternate input source specified is not a disc file as is required by the current operation. Tape alternate files are not acceptable.

Suggested Recovery Procedure: Check control stream for correct usage of alternate sources and correct file definitions for mispunches.

D B Description: Directory of file listed in error field has overflowed its limits; current operation listed in control field is incomplete. All following ADD, COR, and CPY operations will be suspended for this file or Proc group.

Suggested Recovery Procedure: Directory space must be initialized with a larger number of tracks specified or file must be compressed (CMP). File control information has been updated.

D C Description: Library file listed in error field has overflowed; current operation listed in control field is incomplete. All following ADD, COR, and CPY operations will be suspended for this file or Proc group. File must be enlarged in size or compressed.

Suggested Recovery Procedure: All modules causing file overflow must be deleted from the file, they are incomplete. Compress the file. If the file overflows during an ADD, CPY, or COR operation, following the compression, the file must be enlarged. File control information has been updated.

D D Description: No active library files remain; LIBUPS execution is terminated.

Suggested Recovery Procedure: All files have been locked out due to previously detected errors. Correct errors and rerun job.

D E Description: An attempt was made to access the undefined Proc group listed in the error field. The control statement is being ignored.

Suggested Recovery Procedure: Specify the intended Proc group. Define the Proc group (FILPn).

D F	Description: An attempt was made to access the undefined Proc group listed in the error field. The current expression is being ignored. Suggested Recovery Procedure: Specify the intended Proc group. Define the Proc group (FILPn).
D G	Description: The alternate library (ALT1 or ALT2) specified is not a magnetic tape file. Disc alternate files are not acceptable. The expression listed in the error field is being ignored. Suggested Recovery Procedure: Check control stream for correct usage of alternate sources and correct file definitions for mispunches.
D H	Description: The current library designator (L, R, S, C, P) does not correspond to the file type of the library file specified for LIBn. This error code is also used to indicate that an improper number of extents exist for this type of file. Suggested Recovery Procedure: Check control stream for possible mispunches of library designator or check for correct number of extents for file (two for Proc, one for others).
D I	Description: Directory space requested exceeds the limit; one cylinder of directory space has been allocated instead of the invalid number requested. Suggested Recovery Procedure: Check parameter 3 on FIL statement (it must not exceed one cylinder) or check for mispunch of parameter 3. If less than one cylinder is desired for directory space, it will be necessary to rerun the job.
D J	Description: Copy attempted to the undefined Proc group listed in the error field. Suggested Recovery Procedure: Initialize listed Proc group through use of the FIL statement.
D K	Description: RDFCB error. File was not scratched because the file definition was not found. Suggested Recovery Procedure: Confirm correct LFD statement; re-try.
D L	Description: Proc not referenced by first NAME line. Check Proc names for correct reference. Can also occur when adding a Proc, if the name on the ADD statement is the same as a NAME line of a module already resident on disc, but is not the first NAME line of the resident Proc. Suggested Recovery Procedure: Check directory display for proper Proc name referencing, making certain that the name on the ADD statement does not match any subsequent names of a Proc module in the file.
D M	Description: Unable to retrieve the tape loader routine for object card punching or for dumping the load library out on LIBOUT. Suggested Recovery Procedure: Re-try making sure librarian resides with standard system library. Could also be caused by hardware failure during search for loader.
D N	Description: SCRATCH error during a DELn ALL operation. Probable hardware failure. Suggested Recovery Procedure: Re-try.
D P	Description: Input and output files are of different types; copy not possible. Suggested Recovery Procedure: Check control stream for possible mispunches of file designator on INP or OUT statements (or FIL equivalents).

D Q

Description: File control information (LFDR) unable to be updated due to consecutive I/O errors on the file listed in the error field. File must be considered not updated during this run.

Suggested Recovery Procedure: Operation flagged is incomplete. Control information has not been updated. (it may be necessary to re-create the entire file.) An attempt may be made to extract as much data as possible and re-create the rest of the file.

D R

Description: MCL format is invalid; file type does not correspond.

Suggested Recovery Procedure: Re-try. The MCL file was either over- written or a hardware failure may have been experienced.

D S

Description: The module listed in error field caused file overflow, is incomplete, and should be replaced. The file has been compressed or dumped out to tape successfully; this is strictly a warning. If more than one module caused overflow (each listed in the directory), compress will cut off processing after it processes the first of these modules.

Suggested Recovery Procedure: Delete the module listed. It is incomplete and must be replaced.

D T

Description: An I/O error was encountered while attempting to update the LFDR or the Proc Group Descriptor Block associated with the file listed in the error field. The flagged operation must be considered incomplete. The file control information (LFDR) and Proc Group Descriptor Block must be considered not updated.

Suggested Recovery Procedure: Control information has not been updated. The operation flagged is incomplete. An attempt should be made to extract as much data from the file as is possible and re-create the rest.

DU

Description: Insufficient space on SYSPOOL to support the disc-to-tape dump function, or an error in GIVE processing was encountered (status byte printed – see *UNIVAC 9400 System Supervisor Programmer Reference, UP-7689* (current version) under Dynamic Allocation of Direct Access Storage).

Suggested Recovery Procedure: Use another SYSPOOL volume or remap SYSPOOL and retry, after taking corrective measures where applicable. That is, if error encountered is an input/output operation, disc volume may require reprepping.

DV

Description: Attempt was made to dump a void disc file. A nullfile has been generated on the LIBOUT tape.

Suggested Recovery Procedure: Check control cards; perhaps the library identifier was incorrectly specified. If so, correct and retry the job.

DW

Description: Invalid block size or record size was detected while attempting to dump the module listed in the error field. The module has not been dumped onto tape.

Along with the error code is printed a subcode

01	signifies a block size error
02	signifies a record size error

Suggested Recovery Procedure: Replace source type (Source, Copy, Proc) modules on disc by adding from cards. For object code, module must be reassembled and replaced in the library file.

T 0 Description: Parity error encountered while processing the item listed in the error field; *output may be unreliable.* Processing is continuing.

Suggested Recovery Procedure: Check integrity of item listed in error field. It may be necessary to replace it using a different tape unit.

T 1 Description: File listed in error field is locked out from further processing because of repeated parity errors.

Suggested Recovery Procedure: Check tape input. Re-try job with tape on another tape unit. If error recurs, it may be necessary to re-create the tape.

T 2 Description: Block missing from alternate tape file while processing name listed in error field.

Suggested Recovery Procedure: Re-try using different tape drive for recovery.

T 3 Description: Item listed in error field was not found because of a tape I/O error; file is locked out from further processing; current expression is being ignored.

Suggested Recovery Procedure: Re-try job with tape on another tape unit.

T 4 Description: Unrecoverable I/O error was experienced on file listed in error field; current operation is incomplete; file is locked out from further processing.

Suggested Recovery Procedure: Re-try job with tape on another tape unit.

NOTE:

The following error subcodes are printed in the error field, together with the file name, for certain errors encountered during OPEN initialization processing. These errors are T4, T5, T6, and T8; the error subcodes are:

01 File control block for the specified file is missing. Check appropriate LFD statement.

02 Device type specified is not tape and is not acceptable.

03 Device type specified is disc and is not acceptable.

04 Specified output tape does not have a write enable ring, and the operator responded with U to the OPR message. The test for the ring would be retried if the operator inserts the ring and types in an R.

05 The output file specified as OBJFIL is missing the ENDOBJ block. The OPEN transient routine searches for this block if the file is not at load point.

06 The first block on the specified tape file does not conform to the standard VOL1 label or BOOT record and cannot be processed.

07 The volume serial number in the VOL1 label does not match the volume serial number specified in the control stream. (Either the wrong VOL statement was submitted, or the wrong tape was mounted for the specified file.)

08 The HDR1 label was not found for the specified tape file during the OPEN phase.

09 The file-id in the file control block does not match the file-id in the HDR1 label. (Either the wrong file-id was specified on the LBL job control statement, or the wrong tape was mounted.)

10 The creation date in the file control block and the HDR1 label do not match. (Either the creation date on the LBL job control statement is incorrect, or the wrong tape was mounted.)

11 The expiration date for the specified output tape file does not match the expiration date (Julian format) in the job preamble, and the operator responded with U to the OPR message. (Either the expiration date on the LBL job control statement is incorrect, the wrong tape was mounted, or the Julian date was not keyed in with the SET DATE command at the start of the run.)

T 5

Description: Invalid tape specified for file listed in error field; this is a label processing error (LBL statement), an invalid tape format (all alternate tapes must be in tape librarian format), a NULLFILE record was accessed, or there was a tape positioning failure (caused by hardware).

Suggested Recovery Procedure: Check tape format; tape must be in library tape format – not OBJFIL. If format is correct, and there are no label processing errors and the file being accessed is not a NULLFILE, re-try job using a different tape unit. See Note following T4 error code.

T6

Description: File definition error. The first two tape blocks are not the required VOL1/HDR1 or BOOT records.

Suggested Recovery Procedure: Re-try job with tape on another tape unit. If the error persists, reprep the tape volume or execute specifying OPTION NOVOL. See Note following T4 error code.

T7

Description: Invalid block size (greater than 444) detected while adding/copying from tape the module listed in the error field. This is strictly a warning. The module will not be dumped to tape if the disc-to-tape dump function is used.

Suggested Recovery Procedure: Replace the module (either with a new assembled version or, for source modules, add from cards).

T 8

Description: File was not defined properly through use of the DVC and LFD job control statements.

Suggested Recovery Procedure: Check job control stream for tape (ALT1, ALT2) file definitions. See Note following T4 error code.

T 9

Description: The alternate input source is not a tape file as required by the current operation.

Suggested Recovery Procedure: Check control stream for valid use of alternate tape input sources.

T A

Description: The tape file listed in the error field is locked out because of previously detected errors.

Suggested Recovery Procedure: Refer to previously detected error.

T B

Description: End-of-reel detected on output library tape, LIBOUT. The tape has been closed and locked out from further processing. All information processed, up to the point of the end-of-reel detection, can be accessed.

Suggested Recovery Procedure: Diminish the size of the disc library file or substitute a larger tape reel and re-try. If larger reel doesn't satisfy, observe tape drive for possible write failures which cause tape wastage.

TC	Description: An attempt was made to access a transient routine from an alternate tape file; transient routines cannot be added to disc library files. Suggested Recovery Procedure: Transient routines are not acceptable as input; confirm that module name is correct.
S 0	Description: This type of error indicates the invalid use of delimiter. Suggested Recovery Procedure: Check control statement for mispunches of delimiters.
S 1	Description: This type of error indicates the program name exceeds eight characters; the erroneous expression listed in the error field is ignored. Suggested Recovery Procedure: Check proper name of module.
S 2	Description: This type of error indicates the illegal use of librarian options; the erroneous expression listed in the error field is ignored. Suggested Recovery Procedure: Check control stream for misuse of options ((attempt to rename a Proc module using the RNM function 4).
S 3	Description: Invalid sequence on DUM command. The correct sequence is L, R, C, S, and P. Any other order of processing is invalid. Suggested Recovery Procedure. Check control stream for cards out of place. Re-try.
S 4	Description: This type of error indicates an invalid librarian control statement; the entire statement is ignored. Suggested Recovery Procedure: Check control stream for misspelled librarian control functions; comment lines are flagged in this manner.
S 5	Description: This type of error indicates an invalid librarian control statement; the erroneous expression listed in the error field is ignored. Suggested Recovery Procedure: Check control stream for misspelled librarian control functions; comment lines are flagged in this manner.
S 6	Description: The name in the operand field is blank; the expression is ignored. Suggested Recovery Procedure: Check control statement; this function requires operands.
E 0	Description: Disc Librarian was unable to locate the module listed in the error field. Suggested Recovery Procedure: Check spelling of module name; if correct, check directory display for actual presence of module; may have to add module.
E 1	Description: The ENDCARD is missing for module listed in error field; the end-of-data (/*) statement was read; Disc Librarian run is terminated. The module has been added/corrected but may contain extra statements (source modules) if other librarian control statements followed the ADD/COR. Suggested Recovery Procedure: Re-try with ENDCARD in proper place (ADD) or, if extra statements appear, correct the new module and re-try the remainder of the job.

E 2 Description: The ENDCARD is out of sequence for module listed in the error field; Disc Librarian continues with next expression.

Suggested Recovery Procedure: Re-try with ENDCARD in proper place.

E 3 Description: A NAME or PROC statement is missing; Disc Librarian skips to next ENDCARD and continues with next expression; a partial output may exist.

Suggested Recovery Procedure: Re-try with missing NAME or PROC statement included.

E 4 Description: Proc name exceeds eight characters; the Proc name is truncated on the right to eight characters and processing continues.

Suggested Recovery Procedure: Check control stream for proper name of Proc. Proc must be referenced by eight characters.

E 5 Description: Header record is missing for object card input; Disc Librarian skips to next ENDCARD and continues with next expression.

Suggested Recovery Procedure: Re-try addition of card input (card reader may have skipped over header card); if error recurs, repunch the card deck.

E 6 Description: Object cards are out of sequence, Disc Librarian skips to next ENDCARD and continues with next expression.

Suggested Recovery Procedure: Check object deck; reorder the deck according to sequence numbers in columns 73-80, if necessary and re-try.

E 7 Description: The object block just constructed exceeds the maximum block size; Disc Librarian skips to next ENDCARD and continues with next expression (tape transient routines punched into cards are not acceptable input).

Suggested Recovery Procedure: Re-try; punch deck again, if necessary.

E 8 Description: Checksum error is detected while processing (from alternate tape and disc sources) the module listed in the error field; operation continues; this is only a warning.

Suggested Recovery Procedure: Check module flagged; it may be necessary to replace it, if it appears to have been truncated.

E 9 Description: Checksum error is detected while processing (from card input) the module listed in the error field; operation is not completed; Disc Librarian skips to next ENDCARD and continues with next expression.

Suggested Recovery Procedure: Repunch deck if retrial fails.

E A Description: PARAM statement is missing; Disc Librarian makes the following assumptions:

- Five LIB files.
- MCL, ALT1, ALT2, LIBOUT, and SYSPOOL (to support DUM function) are present.
- Printing is not to be suppressed.
- No postmortem dump will be performed.

Suggested Recovery Procedure: If any of the files or alternate input sources are not defined in the control stream, label processing errors result; retry with PARAM statement.

E B	Description: The Load or Reserve module listed in the error field contains a block larger than the permissible 442 bytes. This module cannot be dumped to tape using the disc-to-tape function. The module may be a very old version.
	Suggested Recovery Procedure: Replace with a new assembled/compiled version.
E C	Description: Invalid parameter in a PARAM statement is listed in the error field. Disc Librarian run is terminated.
	Suggested Recovery Procedure: Invalid parameter must be removed from stream. (See PARAM discussion.) Re-try.
E D	Description: Disc Librarian is unable to find the file listed in the error field; check the PARAM, DVC, or LFD for job control statements.
	Suggested Recovery Procedure: Check job control file definitions; re-try with all necessary definitions.
E E	Description: A line number on the INS or REP card is out of sequence.
	Suggested Recovery Procedure: Check for mispunches of line numbers or, for REP, reversal of limits; also, check for proper sequence of line corrections.
E F	Description: PROC statement does not lead off the Proc module.
	Suggested Recovery Procedure: Re-try with PROC statement added as first statement.
E G	Description: Punch not allocated; unable to punch cards.
	Suggested Recovery Procedure: Re-try with punch allocated.
C 0	Description: COR function is terminated because of detection of the /* statement.
	Suggested Recovery Procedure: Re-try with ENDCARD as delimiter for source line corrections.
C 1	Description: COR function is terminated because of missing INS or REP statement.
	Suggested Recovery Procedure: Check control stream for missing INS or REP statement or duplicate ENDCARDs and re-try.
C 2	Description: Excessive number of hole count errors; suspending current operation; possible causes are hardware, poorly punched cards, or previously punched cards in input hopper.
	Suggested Recovery Procedure: Re-try making sure that input hopper contains blank cards, and so forth.

CUT

Comments concerning this manual may be made in the space provided below. Please fill in the requested information.

System: ______________________________

Manual Title: ______________________________

UP No: ____________ Revision No: ____________ Update: ____________

Name of User: ______________________________

Address of User: ______________________________

Comments:

FOLD

FIRST CLASS
PERMIT NO. 21
BLUE BELL, PA.

BUSINESS REPLY MAIL NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY

UNIVAC

P.O. BOX 500

BLUE BELL, PA. 19422

ATTN: SYSTEMS PUBLICATIONS DEPT.

FOLD

CUT