SPERRY RAND

UNIVAC 9400 SYSTEM TAPE LIBRARIAN

PROGRAMMER REFERENCE

7667 Rev. 2

CONTENTS

1. INTRODUCTION

1.1. GENERAL

This manual describes the libraries and control statements comprising the UNIVAC 9400 System Tape Librarian. The librarian consists of the Library Services (LIBS) program and the Display and Punch Services (DAPS) program which permit maintenance, service, and copy functions for tape operations. Descriptions of Librarian System formats, the librarian functions, examples of the control stream for LIBS and DAPS, and error types and descriptions are included here. A knowledge of the *UNIVAC 9400 Job Control Programmer Reference, UP-7793* (current version) and of *UNIVAC 9400 Assembler/Central Processor Unit Programmer Reference, UP-7600* (current version) is helpful in using this manual.

1.2. LIBRARY STRUCTURES

Library structures are of two types: the system library structure and the user library structure. The system library structure operates from the system resident volume, and the user library structure operates from separate volumes. Each structure comprises five libraries. The five libraries are referred to as the Load library, the Reserve library, the Proc library, the Copy library, and the Source library.

Each library constitutes a single file in the library structure except the Proc library, which may constitute more than one file.

1.3. GANG OPERATIONS

Some functions of the Tape Librarian may be specified as gang operations; that is, they are performed on groups of modules within a library rather than on an individual module. Gang operation is controlled through use of the leading characters of a module name. The specification of gang operation includes the leading characters of the name, a period, and the word ALL. For example, if the specification SQRT.ALL appears, the operation is performed on all modules with names beginning with SQRT. Gang operations on modules from the object tape or from cards are not allowed due to the unordered sequence.

The functions of the Tape Librarian for which gang operations may be specified are ADD, DEL, DIS, PCH, and PUD. Gang operation is applicable to all libraries. See Section 3 for further explanation and examples of these functions.

1.4. NAMING CONVENTIONS

All names of modules within libraries must conform to the following general rules (specific requirements for certain types of names are described in detail):

- Valid EBCDIC characters for names are the letters A through Z, the numerals 0 through 9, and the special character $ (dollar). No other characters can be used in a name.
- The number of characters in a name may vary from one through eight, except the names of phases comprising modules in the Load library. Phase names are one to six characters followed by a two-character segment number (00-99). If the load module name is less than six characters, it is left justified with padded 0's (EBCDIC) to fill out the six-character field and is followed by the two-character segment number. Object module names are from one to eight characters in length. If the object name is less than eight characters, it is left justified and blank filled.
- A *module* name may not be blank but a control section name (contained within a module in the Reserve library) may be blank.
- The first character of any name may be one of the letters A through Z or the special character $ (dollar).
- Key overlays, which are critical to the operation of the system (such as Job Control and transient routines) may be interspersed in the Load library by constructing their phase names with a leading $ character. This character allows items (with the same name) to appear more than once within the library and to be filed disregarding the normal alphanumeric sequencing. This special handling of $ name modules occurs only in the Load library.
- The following reserved names are utilized by the DAPS and LIBS programs and therefore cannot be used as labels by the user:

DAPS	LIBS
ALL	ADD
DIR	ALL
DIS	COR
PCH	CPY
PUD	DEL
	ENDCARD
	INS
	REP
	TO
	TR

1.5. CONTROL STATEMENT CONVENTIONS

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

- 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 within braces represents necessary entries of which one must be chosen.
- Information within brackets represents optional entries that (depending on program requirements) are included or omitted.
- An ellipsis indicates the presence of a variable number of entries.

1.6. LIBRARIAN CONTROL STATEMENT FORMAT

The librarian control statements are essentially freeform. Information may start in column 1 with the operation field; this field must be separated from the operand field by at least one blank column. The operand field is terminated by the first blank position and cannot extend past column 71. No continuation statements are recognized (column 72 must be blank).

The operation code 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. Thus, an operation code ADDL adds the modules specified in the operand field to the Load library.

Throughout this document when the function designator is stated and a lowercase x follows it, the programmer must supply the appropriate library designator. Tables 1-1 and 1-2 describe the function and library designators for the Tape Librarian.

FUNCTION DESIGNATOR	DESCRIPTION
ADD	Addition
DEL	Deletion
CPY	Copy
COR	Correction
DIS	Display
PCH	Punch
PUD	Punch and Display

Table 1-1. Function Designators

LIBRARY DESIGNATOR	DESCRIPTION
L	Load library
R	Reserve library
C	Copy library
S	Source library
Pn	Proc library; n is the file number: 1 to 251

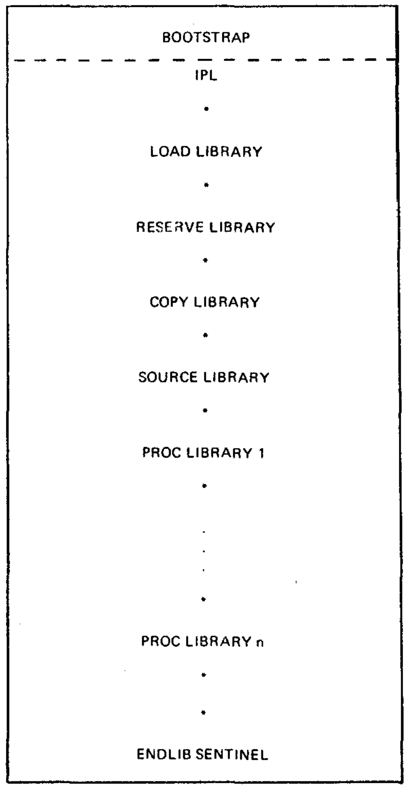
Table 1-2. Library Designators

2. LIBRARIAN SYSTEM FORMATS

2.1. LIBRARY FORMAT

There are two types of library structures in the Tape Librarian: the system library structure and the user library structure. Each structure consists of five distinct libraries. The system library structure occupies the system resident volume, and each user library structure occupies a separate volume. No volume may contain more than one library structure, and no library structure may overflow onto another volume. More than one user library structure may exist for a given job; but only one structure can be updated at any one time.

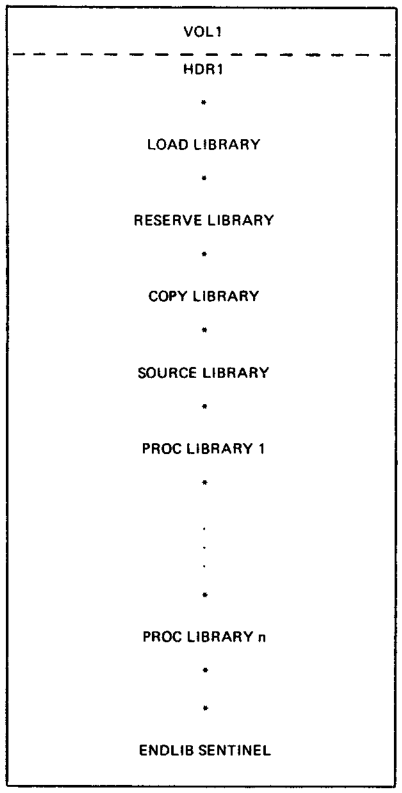
A system library structure begins with a Bootstrap record followed by an Initial Program Load (IPL) record (see Figure 2-1), and a user library structure begins with a VOL1 label followed by a HDR1 label (see Figure 2-2). The user library structure usually contains programs pertinent to a particular user. Thus, no time is wasted in copying the system programs when the user library structure is updated. In a minimum tape system, however, it may be necessary to use the system library structure because of the lack of additional tape units.



Legend:

*signifies the presence of a tape mark

Figure 2-1. System Library Structure



Legend:

*signifies the presence of a tape mark

Figure 2-2. User Library Structure

2.1.1. LOAD LIBRARY

The Load library consists of load modules which are produced by the Linkage Editor. Each load module may be either a single phase or a group of phases. A phase is a single loadable entity. Load modules are processed by the librarian through use of the gang operation option (see 1.3). Single phases of load modules are processed by the librarian by name, one at a time, through use of the OBJFIL or CARD option. Multiphase load modules are processed as a group (thay all have the same prefix) through the use of the ALTLIB option.

The Load library is constructed of phase header blocks and phase blocks. A phase header block describes the phase blocks which constitute a single phase. Each phase header block is immediately followed by one and only one prefix loader block. The phase blocks follow the prefix loader block. Phases are filed in the library in ascending alphanumeric order (A through Z and 0 through 9) according to the phase name in the header block.

Load library updating is done by phase. When library updates involving interspersed load modules (the load module name begins with the $ character) are processed, the following characteristics must be considered:

- Load modules beginning with a $ character are not filed in ascending alphanumeric sequence. Interspersed modules are processed with no recognition of sequence. If a $ name module is to be added (for example, ADDL $TRN0000), it is added to the library output tape in the current tape position. If a $ name module is to be deleted (for example, DELL $TRN0000), the library input tape is copied in a forward direction until the first occurrence of a match ($TRN0000). Any subsequent occurrences of the same module remain unchanged.

- When a library access of a $ name module is performed, alphanumeric sequence is ignored. Accessing an existing $ name module by a DELL $ name or CPYL TO, $ name statement causes a search on equal through the LOAD file with no cutoff when module name of greater alphanumeric order is recognized. An ordinary tape search is performed until the first occurrence of the module is encountered. Subsequent references to the same $ name module located in the LOAD file in several other places requires additional DELL $ name or CPYL TO, $ name statements.

2.1.2. RESERVE LIBRARY

The Reserve library is primarily used for storing object modules created by one of the language processors (Assembler, FORTRAN, COBOL, RPG). These object modules must be processed by the Linkage Editor before they are executable. Therefore, reserve object modules, unlike the phase modules in the Load library, cannot be accessed by a LOAD or FETCH macro instruction.

The Reserve library is constructed of header blocks and associated object blocks. A header block describes the object blocks which comprise a reserve module; the object blocks immediately follow the header block. A reserve module is filed in the library in ascending alphanumeric order according to its module name in the header block.

2.1.3. COPY LIBRARY

The Copy library consists of copy modules. These modules are sets of source statements associated with COBOL COPY statements. The Copy library is constructed of header blocks and source blocks. A header block identifies the source blocks which comprise a copy module; the source blocks immediately follow the header block. The source statements of the copy module are filed in their full 80-character (byte) card image format. A copy module is filed in the Copy library in ascending alphanumeric sequence according to its module name in the header block.

2.1.4. SOURCE LIBRARY

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

The Source library is constructed of header blocks and source blocks. A header block describes the source blocks which comprise a source module; the source blocks immediately follow the header block. The source statements of a source module are filed in their 80-character (byte) card image format. A source module is filed in the Source library in ascending alphanumeric sequence according to its module name in the header block.

2.1.5. PROC LIBRARY

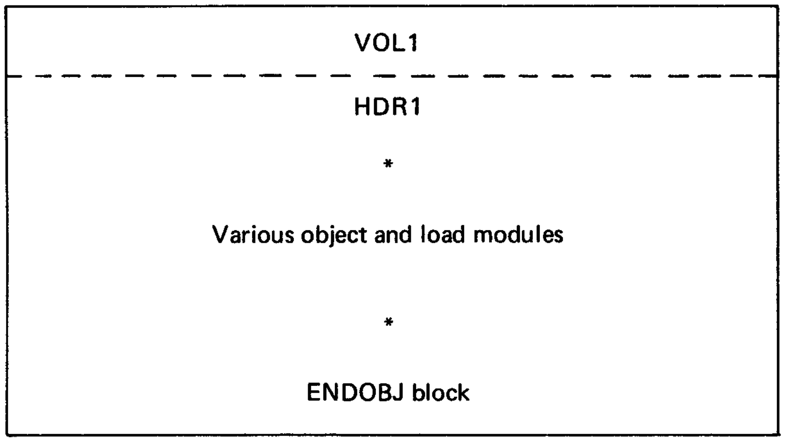
The Proc library consists of Proc groups. Each Proc group is delimited by a tape mark; the last Proc group in the Proc library is delimited by two tape marks.

A Proc group is made up of Proc modules which are a set of source statements that constitute a procedure definition. Within the Assembler, a Proc module may be referred to by any of its names which are designated on the NAME cards within the Proc definition. During a library update, however, a Proc module may only be referred to by the module name which is contained in its header record. This name when included on a control statement (ADDP2 PROCNAME) need not correspond to any of the names on the NAME statements within the Proc module.

The Proc group is constructed of header blocks and source blocks. A header block identifies the source blocks which comprise a Proc module; the source blocks immediately follow the header block. A Proc module is filed in ascending alphanumeric sequence within the Proc group. Proc groups are accessed by group number. The group number follows the library designator P, which follows the desired function code. (An example of this sequence is DELP2 PROC1, which deletes a Proc module identified as PROC1 in Proc group 2.) The source statements which make up a Proc module are filed in the library in their full 80-character card image format (80-byte EBCDIC statement).

2.2. OBJECT TAPE FORMAT

The main difference between the formats of the object tape and the library tapes is that the items on the object tape are not filed in ascending sequence; they are filed on the object tape in the order they were processed. Consequently, an object tape cannot be utilized as a library tape. The format of an object tape is shown in Figure 2-3.



Legend:

*signifies the presence of a tape mark

Figure 2-3. Object Tape Format

2.3. PRINTER OUTPUT FORMAT

The Library Services (LIBS) program and the Display and Punch Services (DAPS) program produce specific printed output formats.

The DVC and LFD job control statements are required for use of the printer. If the statements are omitted, DAPS or LIBS issues a console error message and the job is aborted.

2.3.1. LIBRARY SERVICES (LIBS) PRINTER FORMAT

The printer output format for the LIBS program is a listing of five fields. The purpose of the LIBS program listing is to have a record of the control statements and of the activity that has been performed. An example of the LIBS program printer output is shown in Figure 2-4.

```
ERROR CODE   ERROR FIELD   CONTROL    LINE NUMBER   SOURCE STATEMENT

                           **LIB      IPL,INIT

                           **ADDL     CONV0001,CONV0003,CONV0007,CONV0005

S 3 F        CONV0005

                           **ADDS     COBOLSRC(ALT)

T 2 F

                           **CORS     CONVERT

                             INS      13

                                      0014          LI$TBL DC XL3'003F27'

                           ENDCARD
```

Figure 2-4. LIBS Print Format

- Error Code Field

  This field contains three subfields.

  – The first subfield designates which of the four categories of errors was encountered.

T	=	Tape error
S	=	Control statement error
E	=	Miscellaneous errors
C	=	Card errors

  – The second error code subfield is the error designator. Each category comprises a number of error conditions, which are defined in Appendix B.

  – The third error code subfield designates the error origin within the librarian. This code has no significance to the user.

- Error Field

  The error field contains the expression which was in error or it contains supplemental information as explained in the individual error descriptions in Appendix B.

- Control Field

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

- Line Number Field

  The line number field contains the source statement line number. The line number, which is automatically applied to the source statement by the librarian, is displayed on listings obtained by running the DAPS program of the Tape Librarian. Each source statement is consecutively numbered starting with 1. Any corrections made in the source program on tape must be based on these line numbers.

■ Source Statement Field

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

2.3.2. DISPLAY AND PUNCH SERVICES (DAPS) PRINTER FORMAT

The printer output format for the DAPS program is a listing, which shows, at the option of the user, the contents of the tape, a specified library, or a module in a specified file. The option of printing only header labels is available. Examples of the DAPS program printer output are shown in Figures 2-5 and 2-6.

```
**LOAD FILE (started on new page)

BLK NO. FID    NAME      VER NO. FLAGS  ESID  PHS ADR  LENGTH  PROTECT  MDL ADR  LENGTH

cccccc  cccc   cccccccc  cc-cc   hhhh   hh    hhhhhh   hhhhhh  hhhhhh   hhhhhh   hhhhhh

BYTES   BLOCK NO ccccc            OBJECT STATEMENTS

hhhh hhhhhhhh h ——— 12 bytes/line ————————————————————————————————— h

hhhh hhhhhhhh h ——————————————————————————————————————————————————— h

hhhh hhhhhhhh h ——————————————————————————————————————————————————— h

hhhh hhhhhhhh h ——————————————————————————————————————————————————— h

Legend:    c        signifies alphanumeric characters

           h        signifies hexadecimal notation

           ccccc    represents the decimal block number value.
```

Figure 2–5. DAPS Print Format (Load File)

```
**COPY FILE (started on new page)

BLK NO.   FID      NAME      VER NO.   FLAGS

cccccc    cccc     cccccccc  cc-cc     hhhh

LINE NO.  SOURCE STATEMENT

ccccc     c ————————————————————————————————————————————————————— c

ccccc     c ————————————————————————————————————————————————————— c

ccccc     c ————————————————————————————————————————————————————— c

ccccc     c ————————————————————————————————————————————————————— c

Legend:   c        signifies alphanumeric characters

          h        signifies hexadecimal notation
```

Figure 2–6. DAPS Print Format (Copy, Source, and Proc Files)

2.4. PUNCH OUTPUT FORMAT

The DAPS program also produces output in punched card form. The format for source card output is an exact duplication of the 80-character tape source record. The format for object punched card output is a copy of the records as they appear in a block on tape. Seventy-two columns are reserved for object data; additional identification, such as program identification and sequence number, is inserted into each card in columns 73 through 80. However, it is necessary to have several cards in object format before an entire tape block can be composed. When object output in card format is added to a library, a sequence number check is made to ensure the presence of all cards that have been output by the DAPS program.

When a hole count error occurs in the DAPS PCH or PUD function, eight attempts are made to repunch the card. The cards in error are stacker selected into the error stacker.

3. LIBRARIAN FUNCTIONS

3.1. GENERAL

The Tape Librarian consists of two programs:

- Library Services (LIBS) program
- Display and Punch Services (DAPS) program

Library services are defined as those library routines which alter the contents of a library. These functions are used to add, delete, copy, and correct components of the libraries through use of the LIBS program. The LIBS program is initiated through use of the EXEC LIBS job control statement. Each LIBS function performed is accompanied by printed statements reflecting the actions accomplished. The programmer may suppress this printing by using the NOPRNT option as described in 3.2. A diagram of the functions of the LIBS program is shown in Figure 3-1.

Display and punch services are routines which do not alter the contents of a library. These functions are used to retrieve information from a given library, display it as printed output, and/or convert a given module into a punched card output deck. These functions are accomplished through use of the DAPS program. The DAPS program is initiated by an EXEC DAPS job control statement. A diagram of the functions of the DAPS program is shown in Figure 3-2.

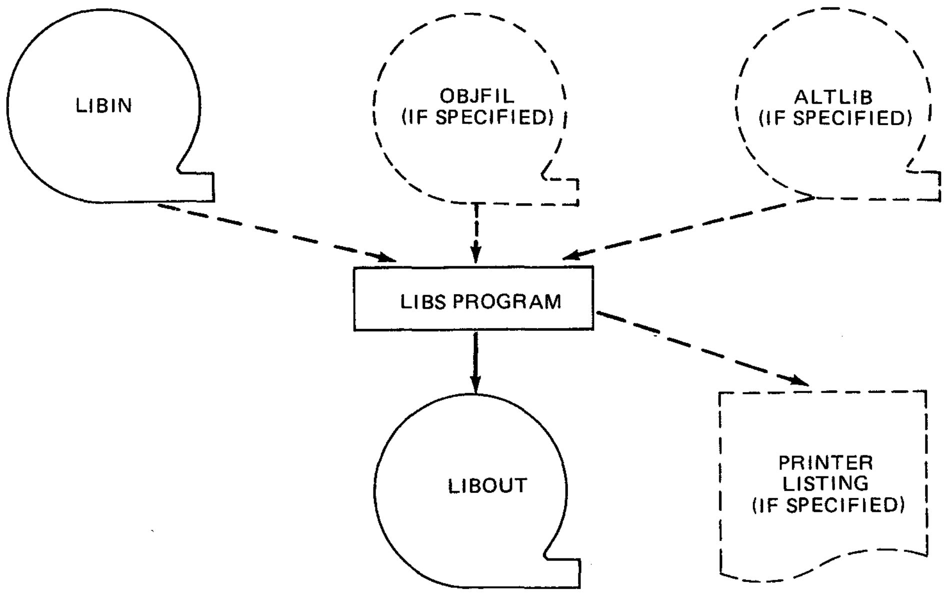


Figure 3–1. Functional Diagram of the Library Services (LIBS) Program

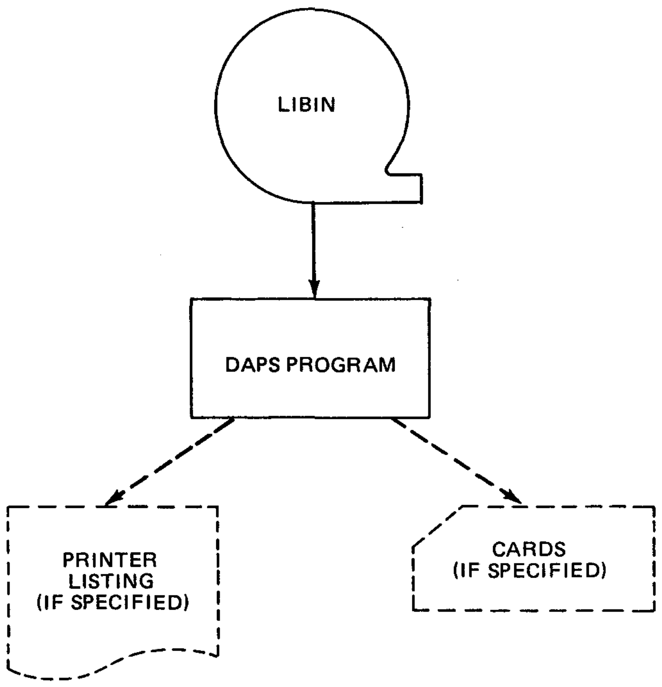


Figure 3–2. Functional Diagram of the Display and Punch Services (DAPS) Program

3.1.1. LOGICAL FILE DESIGNATIONS

Logical file names specified in the LFD statements defining the four tapes which can be involved in a library update are discussed in the following table.

LOGICAL FILE NAME	MEANING	FUNCTION OR DESCRIPTION
LIBIN	Input library tape	This input tape may be assigned as a system resident (SYSRES) volume or as another tape volume.
LIBOUT	Output library tape	The input library is updated and copied onto this tape. This tape has ordered sequence, with the exception of the interspersed modules.
OBJFIL	Object tape	Output of the Linkage Editor and the language processors is "stacked" onto this tape. This tape has a random sequence.
ALTLIB	Alternate library tape	A library structure may be assigned as ALTLIB to enable selected components to be merged with LIBIN. This tape has an ordered sequence, with the exception of the interspersed modules.
PRNTR	Printer	Required; if omitted, a console error message is produced and the job step is terminated.
PUNCH	Punch	Required in DAPS only for punch options. If omitted, a console error message is produced and the job step is terminated.

Table 3-1. Logical File Name Descriptions

3.1.2. JOB CONTROL STATEMENTS

The librarian functions are performed through use of librarian control statements. The job control statements used for a complete Tape Librarian run are:

- JOB statement
- DVC, VOL, LFD, LBL,..., device allocation statements (allocation of printer is required)
- EXEC statement with the applicable librarian program designated
- PARAM statements (for DAPS only)
- /$ statement which indicates start of librarian control statements
- Librarian control statements pertaining to the function to be accomplished
- /* statement which delimits the librarian control statements
- /& statement

The device allocation control statements used for LIBS functions should follow these conventions. For each library tape that is required, a logical file name (LFD) statement must be included for those logical file names listed in 3.1.1 (for example, // LFD OBJFIL). In addition, VOL and LBL job control statements allow the librarian to check input tapes (LIBIN, OBJFIL, and ALTLIB with VOL1 and HDR1 labels) to ensure that the correct volumes are mounted. The library file identification and volume serial number may be confirmed to prevent the merging or copying of an incorrect tape volume. The system library structure may be updated from the master tape or from an alternate tape. The DVC statement associated with the LFD LIBIN determines the input source.

The device allocation control statements used for the DAPS functions are similar to the requirements previously described. If a library display and/or punch is performed, LIBIN must be assigned as the logical file name for the input tape. Back-to-back library operations may be accomplished by reallocating a specific device associated with the previous LIBOUT to LIBIN.

The PARAM statements in a control stream for a DAPS operation allow the user the following options:

- a check for the maximum block size of specified LIBIN modules
- a sequence number check of tape source image statements.

The format of the PARAM statement for requesting a block size check is:

1	10
// PARAM	BLKSZERR

The DAPS routine checks the number of bytes in the specified modules to determine whether the maximum system standard block size has been exceeded. This check can be used for all DIS and PUD options except the DIR option.

The format of the PARAM statement for requesting a sequence number check is:

1	10
// PARAM	SEQCHKER

DAPS checks for an ascending order of the sequence numbers.

3.2. LIBS PROGRAM

The LIBS functions are performed by updating LIBIN in the order of its libraries (files). The order of the libraries is shown in Figures 2-1 and 2-2. If a particular library is not affected by an update, the entire library is automatically copied from the input library tape.

Within a given library the modules must be updated according to their order on tape. In all libraries the modules are filed in ascending alphanumeric sequence by module name. In the Proc library, modules are sequenced within each Proc group, but no sequencing relationship is found from one Proc group to another.

Before any of the LIBS functions (add, delete, copy, and correct) may be accomplished, certain parameters must be submitted. The parameters are introduced through use of the LIB librarian control statement.

A tape file used in a librarian run can have only one device assignment and logical file name associated with it within a job step. For example, in a given job step, a file specified as LIBIN cannot also be specified as an ALTLIB or a LIBOUT file.

3.2.1. LIB STATEMENT

The format for the LIB librarian control statement is:

OPERATIONb	OPERAND
LIB	[IPL] [,INIT] [,NOPRNT] [,NOBJ] [,NALT]

OPTIONS

IPL	– specifies that the Bootstrap and IPL blocks and a tape mark are to be written as the first items on the output library tape (LIBOUT) in the system library structure.
if blank	– causes VOL1 and HDR1 blocks and a tape mark to appear as items on LIBOUT signifying a user library structure. The user has the normal option of specifying labeling information through use of VOL and LBL statements.
INIT	– specifies that a new library structure is to be initiated (on LIBOUT). During this operation, there is no input library structure (LIBIN); LIBOUT is constructed from only the items which are added during this library update. The input items are obtained from OBJFIL or punched cards as defined in the following sections.
if blank	– the librarian assumes a LIBIN exists. The user should define the input library to the LIBS program by means of the appropriate job control statements.
NOPRNT	– causes the suppression of all printing of the librarian control statements and their related messages.
if blank	– printing of the librarian control statements is performed.
NOBJ	– specifies that no OBJFIL is required for this particular library update; no LFD or DVC job control statements will have been provided for OBJFIL.
if blank	– the librarian assumes that OBJFIL is present; the run is terminated if the conditions are found to be otherwise.
NALT	– specifies that no ALTLIB is required for this library update; no LFD or DVC job control statements will have been provided for ALTLIB.

if blank — the librarian assumes that ALTLIB is present; the run is terminated if the conditions are found to be otherwise.

Example:

```
1        10        20        30        40        50
LIB IPL, INIT, NOPRINT
```

According to this example, the Bootstrap and IPL blocks and a tape mark are written as the first items on the output library tape, and a new library structure is initiated on LIBOUT. There is no printing of library control statements and their related messages. However, if a librarian control statement is out of order, it is ignored and a message stating this action is printed.

3.2.2. ADD FUNCTION

The ADD function adds a module or group of modules to the library which is specified by the library designator. Any number of modules may be added during a single run. Modules, which are inputs from various sources, must be added in their proper order. If a module to be added to a particular library has the same name as a module which is already resident in the library, the resident module is replaced. Thus, the delete function is automatically provided. An exception to the automatic delete function takes place when interspersed modules are added ($name modules, see 2.1.1) to the load file. An interspersed module is added immediately. No checking of sequence or position is done; therefore, an explicit delete statement must be added before the ADD statement for $ name modules. If an advanced position is desired, the existing library tape must be explicitly positioned by the user. The copy function (CPY) or the delete function (DEL) is useful for this purpose.

The ADD statement has three basic formats. Choice of a format depends on the library being added to and the input source. The first format is used for adding to the Load or Reserve library, the second for adding to the Copy or Source library, and the third for adding to a Proc library. Because of duplication of positional parameters among formats, each parameter is described only once following the formats. The formats are:

OPERATIONƀ	OPERAND
ADD $\begin{Bmatrix}\text{L}\\\text{R}\end{Bmatrix}$	name-1 $\left[\begin{Bmatrix}\text{(CARD)}\\\text{(OBJ)}\\\text{(ALT)}\end{Bmatrix}\right]$ [,name-2...]
ADD $\begin{Bmatrix}\text{C}\\\text{S}\end{Bmatrix}$	name-1 $\left[\begin{Bmatrix}\text{(CARD)}\\\text{(ALT)}\end{Bmatrix}\right]$ [,name-2...]
ADDPn	name-1 $\left[\begin{Bmatrix}\text{(CARD)}\\\text{(ALT)}\\\text{(ALT,group)}\end{Bmatrix}\right]$ [,name-2...]

Positional Parameter 1

name-1 — may be up to eight characters in length, representing the modules to be added. The gang operation (see 1.3) causes all modules having the same leading characters to be added to the library.

(CARD) — indicates that input is from the card reader.

(OBJ) – indicates that input is from the OBJFIL tape. The OBJFIL tape contains only load modules produced by the Linkage Editor, an assembly, or a compilation. These modules can be transferred to the librarian load file; however, a language processor is not acceptable for inclusion in the load file.

(ALT) – indicates that input is from the ALTLIB tape. The modules in the ALTLIB file are in alphanumeric sequence. The tape remains positioned after the last module specified.

(ALT,group) – indicates that input is from the ALTLIB tape and specifies the group number which contains the Proc. If group is not present, the library designator is used to identify the alternate library file. If the module name specified is greater in value than the name associated with the ALTLIB tape position, the tape is rewound to the beginning of the file.

NOTE: If no input source is indicated, the following input sources are assumed:

LIBRARY	SOURCE
Load	OBJFIL
Reserve	OBJFIL
Copy	Card
Source	Card
Proc	Card

Positional Parameter 2

name-2... – if present, positional parameter 2 has the same form as positional parameter 1.

Examples:

```
1        10        20        30        40        50
ADDL CONV0088,INPT0000
ADDR FORT.ALL(ALT),TWEST(OBJ)
ADDC INT0000(CARD)
ADDS FORT.ALL(ALT)
ADDP6 COS(ALT,2),SQRT(ALT,12)
```

3.2.2.1. TRANSIENT ROUTINES

The ADD function may be used to create transient routines from existing phase or object modules and add the transient routines to the Load or Reserve library. The input phase must be self-relocating and must not exceed 1536 bytes.

An input module may be on an ALTLIB or an OBJFIL tape volume. An attempt to construct a transient routine from an existing transient routine produces an error and the control statement related to the transient routine is ignored.

The format of the ADD statement for creation of transient routines is:

OPERATIONb̸	OPERAND
ADD $\left\{\begin{matrix}L\\R\end{matrix}\right\}$	TR,name-1 $\left[\left\{\begin{matrix}\text{(OBJ)}\\\text{(ALT)}\end{matrix}\right\}\right]$ [,TR,name-2...]

Positional Parameter 1

TR – specifies that a transient routine is to be created. This parameter must be present.

Positional Parameter 2

name-1 – represents the name of the phase which is to be transformed into a transient routine. This parameter may be eight characters in length; if multiple appearances occur within the Load library, the first character must be $ (see 2.1.1).

(OBJ) – indicates that the input is from the OBJFIL tape.

(ALT) – indicates that the input is from the ALTLIB tape.

Positional Parameter 3

TR – specifies that a transient routine is to be created. This parameter must be present if positional parameter 4 is present.

Positional Parameter 4

name-2 – if present, positional parameter 4 has the same form as positional parameter 2.

Examples:

```
1        10        20        30        40        50
ADDL  TR,$OPN004(ALT)
ADDL  TR,IPUT0012,TR,OTTT0003
```

3.2.2.2. VER SUBFUNCTION

The VER subfunction of the ADD function creates a version number for differentiation between the versions of source code when adding to the Copy, Source, or Proc libraries. The VER statement is used only when input is from card. The VER statement must immediately follow the ADD statement. If positional parameter 2 is used in the ADD statement, the VER statement follows the ENDCARD for the source statements of the module named in positional parameter 1 of the ADD statement. The format of the VER statement is:

OPERATIONb̸	OPERAND
VER	level-number,update-number

Positional Parameter 1

level-number	– specifies the new two-digit level number. This number is the first part (or byte) of the version number. The range is 00-99. Two digits must always be specified.

Positional Parameter 2

update-number	– specifies the new two-digit update number. This number is the second part (or byte) of the version number. The range is 00-99. Two digits must always be specified.

NOTE: When a VER statement is not supplied, it is assumed that 0's are inserted for the version number field. For an ADD function that replaces information, no automatic updating of the version number occurs.

Example:

```
1        10        20        30        40        50
ADDP8 DTCBONVS
VER 06,08
```

3.2.3. DEL FUNCTION

The DEL function deletes a module or group of modules from the library specified by the library designator. Any number of modules may be deleted during a single run; modules to be deleted must be in proper order and module names must be separated by commas if more than one operand appears on a librarian control statement. The DEL function is also used when updating $ name modules (interspersed modules) in the load file to position the tape to the correct location or to replace a $ name module of the same name.

The DEL statement has four formats. The first is used to delete the entire remaining portion of the library structure (therefore, no library designator appears); the second, to delete the entire remaining portion of a specific library. The third is used to delete a group of modules. The fourth is used to delete a specific module or a number of specific modules. In the DEL statement formats, the lowercase x represents the library designator. Because of the duplication of positional parameters among formats, each parameter is described only once following the formats. The formats are:

OPERATION	OPERAND
DEL	ALL
DELx	ALL
DELx	TO,module
DELx	name-1[,name-2]

Positional Parameter 1

ALL	– causes all or the remaining portion of a library or the entire library structure to be deleted, beginning at the current tape position.

NOTE: If LIBIN is positioned in the Reserve library and DEL ALL is specified, the remaining reserve modules are deleted along with the Source, Copy, and Proc libraries, and two tape marks are generated on LIBOUT followed by the ENDLIB sentinel.

TO	– specifies that the deletion process is to begin with the module in the current tape position.
name-1	– specifies the particular module or group of modules to be deleted. The gang operation option (see 1.3) causes all modules with the same specified leading characters to be deleted from the library.

Positional Parameter 2

module	– specifies the first module which is not to be deleted.
name-2	– if present, positional parameter 2 has the same form as positional parameter 1.

Examples:

```
1        10        20        30        40        50
DEL  ALL
DELC ALL
DELS TO,COBOLSRC
DELR ASM.ALL
```

3.2.4. CPY FUNCTION

The CPY function is used to copy LIBIN selectively or entirely. A complete copy consists of generating the Bootstrap and IPI (system library structure) or checking and updating the VOL1 and HDR1 label (user library structure) and copying all of the individual libraries (files). During a library update, the lack of reference to a particular library causes the entire file to be copied into LIBOUT. Therefore, if the library is to be skipped, a DELx ALL statement is required.

The CPY statement has three formats. The first format is used to copy an entire library structure; therefore, no library designator appears. The second format is used to copy groups of modules; the third, to copy a specific module or a number of specific modules. In the CPY statement formats, the lowercase x represents the library designator. The formats are:

OPERATIONƀ	OPERAND
CPY	ALL
CPYx	TO,module
CPYx	name-1[,name-2...]

Positional Parameter 1

ALL	– causes the entire library structure to be copied. The information on the LIB statement causes the appropriate library tape records or labels to be generated on LIBOUT. Thus, either the Bootstrap and IPL records are generated, or the VOL1 and HDR1 labels are generated.
TO	– designates that the copy process is to begin with the module in the current tape position.
name-1	– specifies a module to be copied.

Positional Parameter 2

module	– specifies the first module which is not to be copied.
name-2	– specifies an additional module to be copied.

Examples:

```
1        10        20        30        40        50
CPY  ALL
CPYL TO,GENR0000
CPYL TRIL0000
```

3.2.5. COR FUNCTION

The COR function with its subfunctions is used to correct lines of source code within the Source, Copy, and Proc libraries and optionally to update the version number. Permissible source corrections include adding, deleting, and replacing source lines. Any number of modules may be corrected in a single run; modules must be corrected in their proper order. The librarian control statements used to correct modules are as follows:

OPERATIONƀ	OPERAND
CORx	name
VER	level-number,update-number
INS	n_1
ENDCARD	
REP	n_1,n_2
ENDCARD	

The COR statement is used to specify the name of the module that is to be corrected. Only the Source, Copy, and Proc libraries may be corrected.

OPERATIONb̸	OPERAND
$\text{COR}\begin{Bmatrix}\text{C}\\ \text{S}\\ \text{P}\end{Bmatrix}$	name

Positional Parameter 1

name	– specifies the module to be corrected; name has a maximum of eight characters.

3.2.5.1. VER SUBFUNCTION

The VER subfunction creates a version number for the user to differentiate between the versions of his source code. The VER statement is used only when input is from card. It must immediately follow the COR card.

OPERATIONb̸	OPERAND
VER	level-number,update-number

Positional Parameter 1

level-number	– specifies the two-digit new level number. This number is the first part (or byte) of the version number. The range is 00-99. Two digits must always be specified.

Positional Parameter 2

update-number	– specifies the two digit new update number. This number is the second part (or byte) of the version number. The range is 00-99. Two digits must always be specified.

NOTE: When a VER statement is not supplied, the update number is automatically incremented by 1.

3.2.5.2. INS SUBFUNCTION

The INS subfunction is used to insert source statements in a module. The source statements to be inserted must immediately follow the INS statement and terminate with an ENDCARD statement.

OPERATIONb̸	OPERAND
INS	n_1

Positional Parameter 1

n_1	– specifies which line the inserted source statements are to follow. The line numbers may be obtained through use of the DAPS program (see 3.3).

3.2.5.3. REP SUBFUNCTION

The REP subfunction is used to replace or delete source statements in a module. The source statements which are to replace the existing lines (n_1 to n_2 inclusive) must immediately follow the REP statement and terminate with the ENDCARD statement. Table 3-2 lists some of the subfunctions which may be accomplished by the REP statement.

OPERATION	OPERAND
REP	n_1,n_2

Positional Parameter 1

n_1 – specifies the first line to be replaced.

Positional Parameter 2

n_2 – specifies the last line to be replaced.

NOTE: n_1 and n_2 may be equal.

TASK	SEQUENCE OF CONTROL STATEMENTS TO ACCOMPLISH TASK
LINE A PRECEDING LINE 1	REP 1,1 Source statement A to be added Source statement formerly 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 3-2. Examples of Functions Accomplished by Use of the REP Statement

3.2.5.4. ENDCARD STATEMENT

The last card to be included after the source statements associated with an INS or REP statement is the ENDCARD statement. This statement is used by the LIBS program as a sentinel for terminating an INS or REP operation. If the ENDCARD immediately follows the REP statement, the positional parameters of the REP statement are omitted.

The format of the ENDCARD statement is:

OPERATION	OPERAND
ENDCARD	

No positional parameters are required.

3.3. DAPS PROGRAM

The DAPS functions (display, punch, and display and punch) must be performed on a library structure in the same order as defined previously for the LIBS functions. The order of the libraries is given in Figures 2-1 and 2-2. The modules in the libraries are filed in ascending alphanumeric sequence by module name.

Additional checking is provided to ensure the integrity of the multifile librarian tape. Validity checks are included to detect:

- Missing or incorrect file identification field.
- Missing prefix loaders for all modules in the Load library except transients.
- Block size exceeding the maximum allowed for a specified file.
- When modules are punched, ascending order of sequence numbers, which appear in columns 73 through 80 for all Copy, Source, and Proc files. For source formatted modules, the number consists of a five-character sequence number, incremented by 10, and a three-character program identification number. For object formatted files, the number consists of a four-character sequence number, incremented by 1, and a four-character program identification number. For sequence number checking, the parameter statement

  ```
  // PARAM SEQCHKER
  ```

  must be included between the // EXEC and the /$ statements in the control stream. If the user has specified no sequence number on the first statement of the module, DAPS creates its own sequence numbers.

3.3.1. DIS FUNCTION

The DIS function displays a module, a group of modules, an entire library, or the header records of an entire library. Any number of modules may be displayed during a single run.

The DIS statement has three formats. The first format is used to display the entire library structure; therefore, no library designator appears; the second is used to display a particular library, and the third, to display a specific module or a number of specific modules. In the DIS statement formats, the lowercase x represents the library designator. Because of duplication of positional parameters among formats, each parameter is described only once following the formats. The formats are:

OPERATIONb	OPERAND
DIS	$\left\{\begin{matrix}\text{DIR}\\ \text{ALL}\end{matrix}\right\}$
DISx	$\left\{\begin{matrix}\text{DIR}\\ \text{ALL}\end{matrix}\right\}$
DISx	name-1 [,name-2...]

Positional Parameter 1

DIR	– causes all of the header records in the library structure or the designated library to be displayed. The output is in EBCDIC or hexadecimal notation, depending on the type of information.
ALL	– causes the display of each module in the library structure or designated library to be printed with its header record and associated records. The information is displayed in EBCDIC or hexadecimal notation according to the type of information.
name-1	– specifies the module to be displayed and contains a maximum of eight characters. The gang operation option (see 1.3) causes all modules having the same leading characters to be displayed as printed output. The printed output is in EBCDIC or hexadecimal notation according to the type of information.

Positional Parameter 2

name-2	– if present, positional parameter 2 has the same form as positional parameter 1.

Examples:

```
1        10        20        30        40        50
DIS ALL
DISL CORMFM,WMPEGP
DISC DIR
```

3.3.2. PCH FUNCTION

The PCH function is used to convert a module in a library into a punched card output deck. This function may be used to punch an entire library, a group of modules, or a specific module. Any number of modules may be punched during a single run. When object modules are converted to cards using this function, the header block is punched; however, when source modules are converted to cards, the header block is ignored.

The PCH statement has two formats. The first is used to produce a punched card deck of an entire library with header records; the second, to produce a punched card output of specific modules with header records. In the PCH format, the lowercase x represents the library designator. The formats are:

OPERATION	OPERAND
PCHx	ALL
PCHx	name-1 [,name-2]

Positional Parameter 1

ALL	– causes each module in the designated library to be punched with its header record and associated records.

name-1 – specifies the module to be punched. Each module is punched with its header record and associated records. The gang operation option (see 1.3) causes all modules having the same specified leading characters to be punched.

Positional Parameter 2

name-2 – if present, positional parameter 2 has the same form as positional parameter 1.

Examples:

```
1        10        20        30        40        50
PCHS ALL
PCHL EPG0002,WMP0001
```

3.3.3. PUD FUNCTION

The PUD function 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 an entire library or to display and punch an individual module or group of modules. Any number of modules may be displayed and punched during a single run. When object modules are converted to cards using this function, the header block is punched; however, when source modules are converted to cards, the header block is ignored. All header blocks are displayed, however, regardless of the type of module.

The PUD statement has two formats. The first is used to display and punch an entire library with header records; the second, to display and punch specific modules with header records. In the PUD format, the lowercase x represents the library designator. The formats are:

OPERATION♭	OPERAND
PUDx	ALL
PUDx	name-1 [,name-2]

Positional Parameter 1

ALL – causes each module of the designated library to be printed and punched with its header record and associated records.

name-1 – specifies the module to be displayed and punched. The gang operation option (see 1.3) allows all modules having the same specified leading characters to be displayed and punched.

Positional Parameter 2

name-2 — if present, positional parameter 2 has the same form as positional parameter 1.

Examples:

```
1        10        20        30        40        50
PUDS ALL
PUDC SUPE0002,WILL0001
```

APPENDIX A. SAMPLE CONTROL STREAM FOR LIBS AND DAPS FUNCTIONS

```
1        10        20        30        40        50        60
// JOB   jobname
// DVC   3
// LFD   PRNTR
// DVC   RES       ASSIGNS SYSRES AS INPUT LIBRARY TAPE
// LFD   LIBIN
// DVC   5
// LFD   OBJFIL
// DVC   6
// LFD   LIBOUT
// DVC   7
// LFD   ALTLIB
// EXEC  LIBS
/$
   LIB   IPL       GENERATES IPL AND BOOTSTRAP RECORDS
   CPYL  TO,CONV0001  COPIES ALL LOAD LIBRARY UP TO BUT
                       NOT INCLUDING CONV0001
   DELL  TO,CONV0004  DELETES ALL LOAD MODULES UP TO BUT
                       NOT INCLUDING CONV0004
   ADDL  CONV0004(ALT) REPLACES CONV0004 WITH VERSION FROM
                        ALTLIB AND COPIES ENTIRE RESERVE
                        LIBRARY
   DELC  ALL       SKIPS ENTIRE COPY LIBRARY;NULL FILE IS
                   GENERATED ON LIBOUT
         ADDS LB$PROG1(CARD) ADDS SOURCE MODULE,LB$PROG1,
                             FROM CARDS. LB$PROG1 IS THE
                             VOLUME NAME USED IN CREATION OF
                             THE HEADER RECORD.

         VER 07,01           VER STATEMENT SPECIFIES USER'S
                             VERSION NUMBER. IF OMITTED,
                             BINARY ZEROS ARE STORED IN
                             VERSION NUMBER FIELD.
```

```
1        10        20        30        40        50        60
         START 0000
         USING R2,*
LP$PMO   BALR R2,0             } DATA CARDS
         LB$PROC1
         END  LB$PMO
         ENDCARD              SENTINEL INDICATING TERMINA-
                              TION OF ADD OPERATION.
   CORS    UTILITY        UPDATES UTILITY SOURCE MODULE.
   INS     3
   BCR     15,RE$         INSERTS SOURCE STATEMENT AFTER
                          LINE 3.
   ENDCARD                END SENTINEL.
   REP     29,29          REPLACES LINE 29 WITH FOLLOW-
                          ING COMMENT CARD.
   *RESTART SCAN
   ENDCARD                END SENTINEL.
   DELP3   ALL            DELETES ALL OF PROC GROUP 3.
   ADDP4   TAPE.ALL(ALT,7) ADDS TO PROC GROUP 4 ANY NAME
                            BEGINNING WITH TAPE IN PROC
                            GROUP 7 OF ALTLIB.
/*                        CONCLUDES LIBRARY UPDATE BY COPY-
                          ING REMAINING LIBRARY ITEMS.
// DVC     2
// LFD     PUNCH
// DVC     6              ASSIGNS DEVICE 6 (FROM PREVIOUS
                          STEP) AS LIBIN.
// LFD     LIBIN
// EXEC    DAPS
/$
   DISL    DIR            DISPLAYS ALL HEADER RECORDS IN
                          LOAD LIBRARY.
   DISP4   TAPE.ALL       DISPLAYS ANY NAME BEGINNING WITH
                          TAPE IN PROC GROUP 4.
   PCHP5   LIBLIST        PUNCHES LIBLIST IN PROC GROUP 5.
   PUDP6   ALL            DISPLAYS AND PUNCHES ALL OF PROC
                          GROUP 6.
/*
/&
```

APPENDIX B. ERROR CODES

This appendix describes the types of errors, the associated codes, and suggested error recovery procedures.

C 0	Description: The object module being added from cards is missing a header record or the module name is incorrectly specified. Suggested Recovery Procedure: Insert the missing header record or correct the module name on the control statement; resubmit the job.
C 3	Description: Incorrect sequence of module name. Names must be in ascending order. Suggested Recovery Procedure: Reposition control card into correct sequence.
C 5	Description: The /* card was encountered before the ENDCARD statement when adding a module from cards or correcting source code. The LIBIN tape, if used, is copied to completion. Suggested Recovery Procedure: Insert the missing ENDCARD statement; resubmit the job.
E 0	Description: The specified module was not found on the OBJFIL tape. Suggested Recovery Procedure: Either the module does not exist on the OBJFIL tape or it is not the correct type of module (a load module). Correct and resubmit job.
E 1	Description: Punched card hole count error detected. Suggested Recovery Procedure: The incorrectly punched card is sent to the error stacker. Eight attempts are made to punch a card correctly before the job step is terminated. Continued incorrect punching indicates a punch hardware failure.
E 2	Description: LIBS statement is missing. A printer is required and the following files are assumed to exist: an IPL systems library tape, a non-IPL library input tape (LIBIN), an ALTLIB, and an OBJFIL. Suggested Recovery Procedure: LIBS program processing continues. If any of the devices required are not specified, the run is terminated. If a systems dump occurs, insert the appropriate LIB parameter statement; resubmit the job.
E 3	Description: Two or more logical library files are assigned to the same device. The job step is aborted. Suggested Recovery Procedure: LIBIN, LIBOUT, and the ALTLIB files must be assigned to separate devices. Correct the assignments and resubmit the job.

E 4	Description: The length of a module being converted to a transient routine exceeds 1536 bytes. The transient routine processing is terminated. Suggested Recovery Procedure: Reduce the module size. Reassemble the module and resubmit the job.
E 5	Description: The module being processed to produce a transient routine is neither an object nor a load module. Processing continues. If the data records conform to the formats in object or load modules, the transient processing produces a useful transient routine. If the data records do not conform, the routine is not constructed. Suggested Recovery Procedure: Correct the module and resubmit the job.
E 6	Description: Records being processed for a transient routine are not text records. Possibly an incorrect module is being used. Suggested Recovery Procedure: The records are ignored, but processing continues. Correct the module and resubmit the job.
E 7	Description: The maximum number of bytes allowable in the transient routine being created has been exceeded. Transient processing is terminated. Suggested Recovery Procedure: Reduce the module size. Reassemble the module and resubmit the job.
E A	Description: Sequence numbers in a source statement are not in ascending sequential order. (Check is made when the statement PARAM SEQCHKER is specified.) Processing continues. Suggested Recovery Procedure: Correct the error and resubmit the job.
E B	Description: A data error was encountered when converting string data from decimal to binary. The library control expression in error is ignored; processing continues when the next expression is encountered. The error may be a double punch in the decimal field or a field contains alphanumeric characters. Suggested Recovery Procedure: Correct the expression and resubmit the job.
E E	Description: Module could not be found in ALTLIB file. Either a module name was found that is greater than the specified module name, or a tapemark was encountered. Suggested Recovery Procedure: Correct the error and resubmit the job.
S 0	Description: An invalid delimiter was found in an expression on a control statement. The expression is ignored but the succeeding expressions in the control statement are evaluated. Suggested Recovery Procedure: Correct delimiter error in control statement and resubmit the job.
S 1	Description: The module name exceeds eight characters. The expression is ignored but the succeeding expressions in the control statement are evaluated. Suggested Recovery Procedure: Modifying the module name; resubmit the job.

S 2	Description: Two or more expressions specified in a control statement are mutually exclusive. The expressions are ignored but the succeeding expressions in the control statement are evaluated.
	Suggested Recovery Procedure: Correct expression in the control statement; resubmit the job.
S 3	Description: Incorrect sequence of module name. Names must be in sequence in ascending order (except for $name entries). The name is ignored but the succeeding expressions in the control statement are evaluated.
	Suggested Recovery Procedure: Reposition the control statement and correct sequence; resubmit the job.
S 4	Description: One of these conditions has occurred: an invalid function was specified (the function is valid in type construction but does not meet the requirements of the function's options); a special character is misused if the delimiter following the function is other than a space; invalid file designator is specified; Proc file designator exceeds a value of 250; invalid option for corresponding function; line number is not specified with INS or REP statement; incorrect delimiter before or after line number. The specifications in the statement are ignored.
	Suggested Recovery Procedure: Correct the expression in control statement; resubmit the job.
S 5	Description: Invalid delimiter or an element in control statement.
	Suggested Recovery Procedure: Correct the expression in control statement and resubmit the job.
S 6	Description: Module name is all zeros or all blanks. When an ADD operation was being performed using the OBJFIL tape, the name element was omitted.
	Suggested Recovery Procedure: Correct error; resubmit job.
S 7	Description: Prefix for module group was not found during a COPY-GANG operation (a greater prefix value was found). Either an incorrect tape was used or an incorrect prefix was specified.
	Suggested Recovery Procedure: Correct prefix; resubmit job.
S 8	Description: Prefix for module group not found. Tape reached end-of-file configuration. Either an incorrect tape was used or an incorrect prefix specified.
	Suggested Recovery Procedure: Correct prefix; resubmit job.
S 9	Description: Specified module not found. Tape reached end-of-file configuration. Either the wrong tape was used or the wrong prefix specified.
	Suggested Recovery Procedure: Correct module name; resubmit job.
S A	Description: Prefix for module group not found when a CPY name or CPY TO option was specified (a prefix value greater was found). Either an incorrect tape was used or an incorrect prefix specified.
	Suggested Recovery Procedure: Correct prefix; resubmit job.

S B

Description: One of the following conditions has occurred: the file specified in this control statement is not the file in which LIBIN is positioned; the ALTLIB file was not specified using the group notation [ADDx NAME(ALT,GROUP)]; a tape block in the DAPS operation does not have a file ID consistent with the file in which it is located; when a module was being corrected, a source type file was not designated.

Suggested Recovery Procedure: Check the control statements for proper sequence and for correct specifications. Correct error; resubmit job.

S C

Description: During a correct operation, the first statement after the COR statement is not a VER, INS, or REP card. The COR statement is ignored. The LIBIN file is copied onto the LIBOUT file through the specified module. Processing resumes when an ADD, DEL, CPY, or COR statement is encountered.

Suggested Recovery Procedure: Correct the erroneous statement; resubmit the job.

S D

Description: A line number specified on an INS or REP statement does not exist in the module specified. The previous COR statement is ignored. The LIBIN file is copied onto the LIBOUT file through the specified module. Processing resumes when an ADD, DEL, CPY, or COR statement is encountered.

Suggested Recovery Procedure: Correct the erroneous line number or module; resubmit the job.

S E

Description: After a read operation, a card image was not returned by the system because of a control stream discrepancy. Error caused by a missing /$ or /* statement or misplacement of job control statements.

Suggested Recovery Procedure: Correct error; resubmit the job.

S F

Description: A block which contains an inconsistent file identification code has been read. DAPS processing continues.

Suggested Recovery Procedure: Re-create the module in which the error occurred and place it in the correct file; resubmit the job.

S G

Description: The Proc definition does not begin with a PROC directive. DAPS processing continues.

Suggested Recovery Procedure: Correct the Proc module before submitting the job to the Assembler.

S H

Description: The Proc END statement is missing from a DIS DIR function. DAPS processing continues.

Suggested Recovery Procedure: Correct the error before submitting the job to the Assembler.

S I

Description: More than one END statement is detected for a DIS DIR function. DAPS processing continues.

Suggested Recovery Procedure: Remove unnecessary END statement. Reprocess Proc.

T 0 Description: The input block number does not match the internal block number counter. For DAPS, the LIBIN tape volume is rewound and the following console message is produced:

DP30 I/O TROUBLE – JOB ABORTED

For LIBS, two tape marks and the ENDLIB record block are written on LIBOUT, all applicable tapes are rewound, and the following console message is produced:

LB40 I/O TROUBLE – JOB ABORTED

Suggested Recovery Procedure: Resubmit the job. For DAPS, properly processed modules need not be specified in the rerun. For LIBS, the aborted run output may be used as the LIBIN, provided that there is no need for modules from the previous LIBIN. The module in which the error occurred cannot be used.

T 1 Description: Block count error on LIBS tape.

Suggested Recovery Procedure: Reprocess the module in error by submitting another job.

T 2 Description: Block count error on ALTLIB tape. The module is truncated and processing is terminated.

Suggested Recovery Procedure: Reprocess the module in error.

T 3 Description: Incorrect header record format in the first block of LIBIN file.

Suggested Recovery Procedure: Ensure that the correct volume is mounted according to the LIBIN file assignment.

T 4 Description: Incorrect header record format in first block of OBJFIL.

Suggested Recovery Procedure: Ensure that the correct volume is mounted according to the OBJFIL file assignment.

T 5 Description: Incorrect header record format in first block of ALTLIB.

Suggested Recovery Procedure: Ensure that the correct volume is mounted according to the ALTLIB file assignment.

T 6 Description: The end of the LIBOUT tape has been reached. The tape is rewound and the following console message is produced:

LB40 I/O TROUBLE – JOB ABORTED

Two tape marks and the ENDLIB record block are written on LIBOUT and the tape is rewound.

Suggested Recovery Procedure: The remaining modules must be added to another LIBOUT tape in another job step. If the LIBOUT tape is faulty, rerun using a new LIBOUT tape.

T 7 Description: During transient processing, the next module header is reached before the transfer record of the module being processed. This indicates that some object code has been truncated. Processed information is placed in the transient format block.

Suggested Recovery Procedure: Reassemble module; resubmit job step.

T 8 Description: The file control block for the specified file was not found. The job is aborted and the name of the file and a specific error code are printed. The error codes are:

CODE	DESCRIPTION
01	The file control block is missing. Ensure that the LFD statement specified the correct filename.
02	The device type specified is not tape and is not acceptable.
03	The device type specified is disc and is not acceptable.
04	The specified output tape had no write-enable ring and a U code was returned by the operator to the OPR message informing him of the problem, causing control to be transferred to the user's error address. If the operator had inserted the ring and typed in an R (retry) response, the test for a ring would have been retried.
05	The specified output OBJFIL file is missing the ENDOBJ block.
06	The first block on the specified tape file does not conform to the standard VOL1 or BOOT labeled block and cannot be processed.
07	The volume serial number in the VOL1 block 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 block, which is the first block after the VOL1 or VOL blocks on any standard tape, was not found for the specified tape file.
09	The file ID in the file control block does not match the file ID in the HDR1 block (either the wrong file ID on the LBL statement was submitted or the wrong tape was mounted).
10	The creation date in the file control block and the HDR1 block do not match (either the creation date on the LBL statement is incorrect or the wrong tape was mounted).
11	The specified output tape file's expiration date did not match the expiration date (Julian date format) in the job preamble and a U (unrecoverable) indicator was returned by the operator in reply to the OPR message informing him of the problem (either the expiration date in the LBL statement is incorrect or the wrong tape was mounted or the Julian date part of the SET DATE command was not keyed in by the operator at the start of the run).

Suggested Recovery Procedure: Correct the job control statement or LIB librarian control statement and resubmit the job.

T 9 Description: Tape file definition error caused by one of these conditions: wrong device code; missing write-enable ring on output file; for LIBIN, ALTLIB, LIBOUT, and OBJFIL tape volumes the first block does not contain VOL1 or BOOT. The name of the file and a specific error code are printed. See the description of error code T8 for definitions of these error codes.

Suggested Recovery Procedure: Correct error and resubmit job.

TC	Description: The ALTLIB tape is positioned to the next file but the preceding file is referenced. Possible causes: a missing ALTLIB tape header record or a reference to a file which has been passed through in a search for nonexistent module. If the ALTLIB file is incorrectly positioned, the current librarian control statement is ignored and processing continues. If the header record in the desired file is missing, the ALTLIB tape is positioned to the next available header record, the current librarian control statement is ignored, and processing continues. Suggested Recovery Procedure: If the ALTLIB tape is incorrectly positioned, correct the error and resubmit the job. If the header record is missing, the first module of that file must be rebuilt.
TD	Description: The maximum block size for a file has been exceeded. The block number and file name are printed and processing continues. This error is usually caused by a faulty LIBIN tape. Suggested Recovery Procedure: Resubmit the LIBS program that created the LIBIN tape.
TE	Description: The maximum block size for a file has been exceeded and the PARAM statement BLKSZERR has been inserted before the /$ statement. This error is usually caused by a faulty LIBIN tape. Suggested Recovery Procedure: Resubmit the LIBS program that created the LIBIN tape.
TF	Description: Modules in the load file are missing a prefix loader block immediately after the header. Processing continues. Suggested Recovery Procedure: Either of these methods are used: reassemble modules using PARAM OUT=(P); employ the system alter routine which inserts a prefix loader block after each module header.
TG	Description: The output tape being used to create a user library volume is labeled BOOTSYSRES. The job step is terminated. Suggested Recovery Procedure: Resubmit the job step using an output tape labeled VOL1, HDR1, or specify IPL in the LIB statement.
TH	Description: Information on tape HDR1 record is inconsistent with information specified in the LBL statement. The name of the file and a specific error code are printed. See the description of error code T8 for definitions of these error codes. Suggested Recovery Procedure: Correct error; resubmit job.
TI	Description: Block count error or read parity error during the open phase of a tape operation. The name of the file and a specific error code are printed. See the description of error code T8 for definitions of these error codes. Suggested Recovery Procedure: Resubmit the job. DTF CCB may be set to accept read parity errors and interrogate the error status byte.
TJ	Description: Checksum character inconsistency. Processing continues. Suggested Recovery Procedure: Blocks in question may be displayed to determine whether the data was written or read incorrectly. If incorrectly written, resubmit the job after restoring erroneous modules. If incorrectly read, resubmit the job, preferably on a different tape device.