SPERRY UNIVAC

PUBLICATIONS UPDATE

Operating System/3 (OS/3)

System Service Programs (SSP)

Programmer Reference

(Series 90)

For System 80 see UP-8842

UP-8209 Rev. 6-A

This Library Memo announces the release and availability of Updating Package A to "SPERRY Operating System/3 (OS/3) System Service Programs (SSP) Programmer Reference", UP-8209 Rev. 6.

This update documents the following new SSP features for the 8.1 release:

- A new parameter for the SMCLIST canned job control stream to sort the listing by the time alone that SMCs were applied, not time and date
- Enhancement to the condensed listing produced by SMCLIST to show which SMCs were backed out, replaced, or were not installed because of error during installation

  Change in the default for the FMT parameter on the SMCLIST job control stream from full to condensed.
- Clarification of F- and FC-type modules

All other changes are corrections or expanded descriptions applicable to features present in SSP prior to the 8.1 release.

Copies of Updating Package A are now available for requisitioning. Either the updating package only or the complete manual with the updating package may be requisitioned by your local Sperry representative. To receive only the updating package, order UP-8209 Rev. 6–A. To receive the complete manual, order UP-8209 Rev. 6.

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS
Mailing Lists BZ, CZ, and MZ	Mailing Lists A00, A01, 18, 18U, 19, 19U, 20, 20U, 21, 21U, 75, 75U, 76 and 76U (Package A to UP-8209 Rev. 6, 8 pages plus Memo)	Library Memo for UP-8209 Rev. 6–A
		RELEASE DATE: June, 1983

SPERRY UNIVAC
COMPUTER SYSTEMS

PUBLICATIONS REVISION

Operating System/3 (OS/3)

System Service Programs (SSP) (Series 90)

For System 80 see UP-8842

Programmer Reference

UP-8209 Rev. 6

This Library Memo announces the release and availability of "SPERRY UNIVAC® Operating System/3 (OS/3) System Service Programs (SSP) Programmer Reference", UP-8209 Rev. 6.

This revision for the 8.0 release, incorporates the following changes:

- SAT Librarian:
  - New PAGE statement
  - New // PARAM statements: ERROR, PRINT, PRTOBJ
  - COP statement allows listing of an entire file
  - COP statement is expanded to replace ADD statement.
  - New A option to process all groups in a file.
  - Load module patch addresses can be expressed relative to the start of a phase.
  - New ORG directive for object and load module corrections.
- New UNXFC parameter for disk and diskette preps.
- New FDATA and PARTL parameters for diskette prep.
- Executing SU$C16, SU$CSL, and DMPRST interactively.

Other changes applicable to system service program routines for 8.0.

Destruction Notice: If you are going to OS/3 release 8.0, use this revision and destroy all previous copies. If you are not going to OS/3 release 8.0, retain the copy you are now using and store this revision for future use.

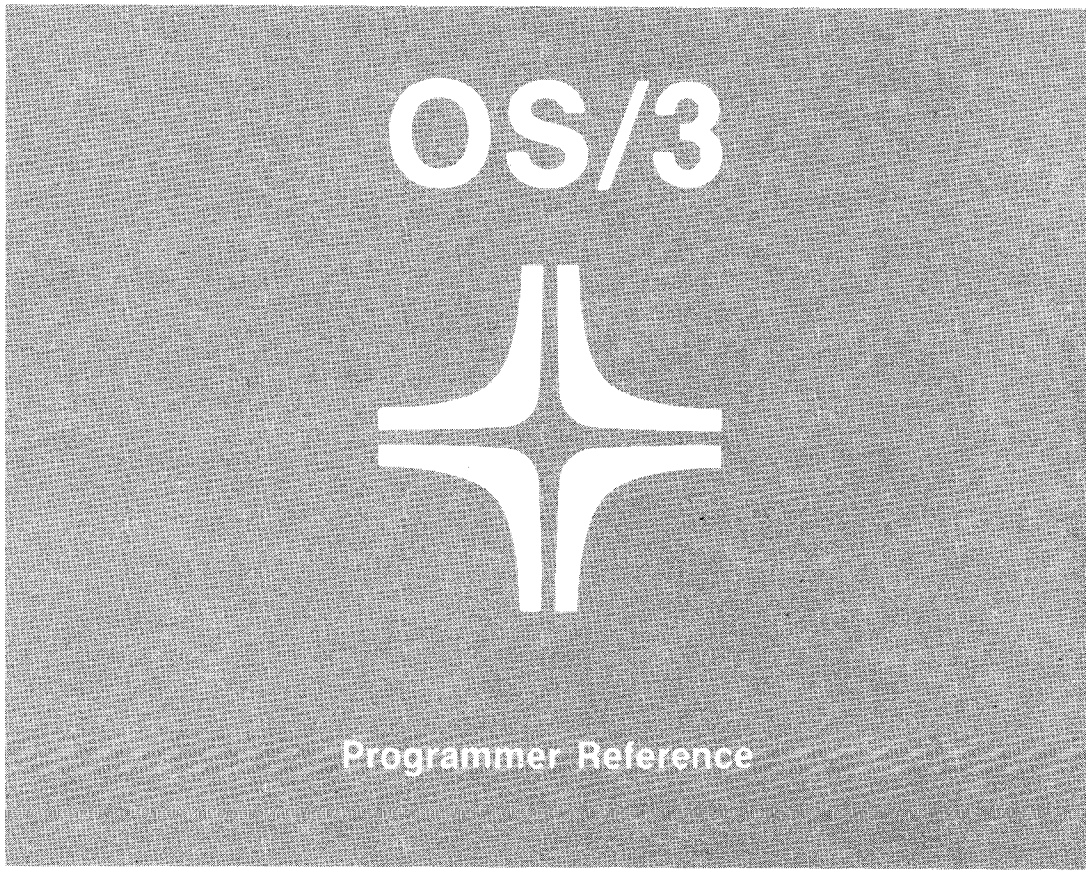
Copies of UP-8209 Rev. 5, UP-8209 Rev. 5–A and UP-8209 Rev. 5–B will be available for 6 months after the release of 8.0. Should you need additional copies of this edition, you should order them within 90 days of the release of 8.0. When ordering the previous edition of a manual, be sure to identify the exact revision and update

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS
Mailing Lists BZ, CZ and MZ	Mailing Lists A00, A01, 18, 18U, 19, 19U, 20, 20U, 21, 21U, 75, 75U, 76 and 76U (Cover and 158 pages)	Library Memo for UP-8209 Rev. 6
		RELEASE DATE: September, 1982

packages desired and indicate that they are needed to support an earlier release.

Additional copies may be ordered by your local Sperry Univac representative.

System Service Programs (SSP)



Environment: 90/25, 30, 30B, 40 Systems

SPERRY UNIVAC

This document contains the latest information available at the time of preparation. Therefore, it may contain descriptions of functions not implemented at manual distribution time. To ensure that you have the latest information regarding levels of implementation and functional availability, please consult the appropriate release documentation or contact your local Sperry Univac representative.

Sperry Univac reserves the right to modify or revise the content of this document. No contractual obligation by Sperry Univac regarding level, scope, or timing of functional implementation is either expressed or implied in this document. It is further understood that in consideration of the receipt or purchase of this document, the recipient or purchaser agrees not to reproduce or copy it by any means whatsoever, nor to permit such action by others, for any purpose without prior written permission from Sperry Univac.

Sperry Univac is a division of the Sperry Corporation.

FASTRAND, SPERRY UNIVAC, UNISCOPE, UNISERVO, and UNIVAC are registered trademarks of the Sperry Corporation. ESCORT, MAPPER, PAGEWRITER, PIXIE, and UNIS are additional trademarks of the Sperry Corporation.

This document was prepared by Systems Publications using the SPERRY UNIVAC UTS 400 Text Editor. It was printed and distributed by the Customer Information Distribution Center (CIDC), 555 Henderson Rd., King of Prussia, Pa., 19406.

PRINTED IN U.S.A.

PAGE STATUS SUMMARY

ISSUE: Update A – UP-8209 Rev. 6
RELEASE LEVEL: 8.1 Forward

Part/Section	Page Number	Update Level
Cover/Disclaimer		Orig.
PSS	1	A
Preface	1, 2	Orig.
Contents	1 thru 6	Orig.
Section 1	Title Page 1 thru 6	Orig. Orig.
Section 2	Title Page 1 thru 20 20a 21 thru 29	Orig. Orig. Orig. Orig.
Section 3	Title Page 1 thru 5	Orig. Orig.
Section 4	Title Page 1 thru 17	Orig. Orig.
Section 5	Title Page 1 thru 9	Orig. Orig.
Section 6	Title Page 1 thru 4	Orig. Orig.
Section 7	Title Page 1, 2 3	Orig. Orig. A
Section 8	Title Page 1 thru 5	Orig. Orig.
Section 9	Title Page 1 thru 3	Orig. Orig.
Section 10	Title Page 1 thru 3	Orig. Orig.
Section 11	Title Page 1 thru 7	Orig. Orig.
Section 12	Title Page 1, 2	Orig. Orig.
Section 13	Title Page 1, 2	Orig. Orig.
Section 14	Title Page 1, 2	Orig. Orig.
Section 15	Title Page 1, 2	Orig. A
Appendix A	Title Page 1, 2	Orig. Orig.
Appendix B	Title Page 1, 2	Orig. Orig.
Appendix C	Title Page 1	Orig. Orig.
Appendix D	Title Page 1 thru 3	Orig. Orig.
Appendix E	Title Page 1	Orig. Orig.
Appendix F	Title Page 1	Orig. Orig.
Appendix G	Title Page 1, 2	Orig. Orig.
Glossary	1, 2 3 4 thru 13	Orig. A Orig.
User Comment Sheet		

Part/Section	Page Number	Update Level

All the technical changes are denoted by an arrow (→) in the margin. A downward pointing arrow (↓) next to a line indicates that technical changes begin at this line and continue until an upward pointing arrow (↑) is found. A horizontal arrow (→) pointing to a line indicates a technical change in only that line. A horizontal arrow located between two consecutive lines indicates technical changes in both lines or deletions.

Preface

This programmer reference manual is one in a series designed to be used as a quick-reference document for programmers familiar with the SPERRY UNIVAC Operating System/3 (OS/3). This particular manual describes the system service programs.

The information presented here is limited to facts; no introductory information or examples of use are provided. This type of information is in the current versions of two other OS/3 manuals: an introduction to the system service programs, UP-8043, and a system service programs user guide, UP-8062.

The information in this manual is presented as follows:

- Section 1. General Information

  Gives a brief description of the system service programs and describes the statement conventions used in writing the parameters associated with OS/3 in general and the system service programs in particular.

- Section 2. SAT Librarian Control Statements

  Describes the function and parameters of each SAT librarian control statement, in alphabetic order by statement mnemonic.

- Section 3. MIRAM Librarian Control Statements

  Describes the function and parameters of each MIRAM librarian control statement, in alphabetic order by statement mnemonic.

- Section 4. Linkage Editor Control Statements

  Describes the functions and parameters of each linkage editor control statement, in alphabetic order by statement mnemonic.

- Sections 5 through 15. System Utility and DUMP Routines

  The system utilities include the initialize disk routine, the assign alternate track routine, the tape prep routine, the sectored and nonsectored disk copy routines, a stand-alone disk copy routine, the disk dump/restore routine, the hardware utilities, the system utility symbiont, the diskette utility, and the list software maintenance corrections.

- Appendix A. SAT Librarian Functions

  Lists the tasks performed by the SAT librarian.

- Appendix B. System Librarian Correction Cards

  Describes the method by which source and nonsource level modules can be corrected.

- Appendix C. MIRAM Librarian Functions

  Lists the tasks performed by the MIRAM librarian.

- Appendix D. Basic Linkage Editor Statement Processing

  Summarizes the use and placement of the linkage editor control statements.

- Appendix E. System Service Program Names

  Identifies the names of all the system service programs.

- Appendix F. Other System Utility Program Names

  Identifies the name and function of each system utility and the documents where they are described.

- Appendix G. Canned Job Control Streams

  Identifies the name and function of each canned job control stream used to perform a system service function and the documents where they are described.

- Glossary

  Defines the terms peculiar to the system service programs.

Contents

4. LINKAGE EDITOR CONTROL STATEMENTS

5. INITIALIZE DISK (DISK PREP) ROUTINE (DSKPRP)

6. INITIALIZE DISKETTE ROUTINE (DSKPRP)

7. ASSIGN ALTERNATE TRACK ROUTINE (DSKPRP)

8. TAPE PREP ROUTINE (TPREP)

9. SECTORED DISK COPY ROUTINE (SU$C16)

10. NONSECTORED DISK COPY ROUTINE (SU$CSL)

11. DISK DUMP/RESTORE ROUTINE (DMPRST)

12. DISKETTE UTILITY (CREATE)

13. SYSTEM UTILITY SYMBIONT (SL$$SU)

14. STAND-ALONE DISK COPY ROUTINES (SU$IDA OR SU$SEL)

15. LIST SOFTWARE MAINTENANCE CORRECTION ROUTINE (SMCLIST)

APPENDIXES

A. SAT LIBRARIAN FUNCTIONS

B. SAT LIBRARIAN CORRECTION CARDS

C. MIRAM LIBRARIAN FUNCTIONS

1. General Information

SAT LIBRARIAN

The SAT librarian is used to maintain the system library files that contain user and system program elements. These elements consist of program source code, macro/jproc source code, object code, and load code module elements used for program generation or execution. They are created and used by components of OS/3 during the normal course of system operation.

MIRAM LIBRARIAN

The MIRAM librarian is used to maintain screen format (F-type or FC-type) modules and saved run library (J-type) modules, help screen (HELP) modules, and menu (MENU) modules. Although the MIRAM librarian is primarily a disk utility, the MIRAM library files may also exist on magnetic tape, diskette, or punched cards and may be copied from one medium to another.

LINKAGE EDITOR

The linkage editor builds load modules by linking object modules and object module elements residing in the system object library file (OOBJ), the system job run library file (YRUN), or the user object library file. The linkage editor control statements govern load module construction by:

- defining the object modules and module elements to be included in the load module being built;
- specifying the order in which the object modules are to be linked (load module structure); and
- specifying the linkage editor options to be in effect during the building period.

The link-edit function is usually the last step to be performed in the program preparation process. The object modules cannot be loaded or executed until they have been processed by the linkage editor.

SYSTEM UTILITY ROUTINES

The system utility routines covered in this manual are as follows:

- Initialize disk (disk prep) and assign alternate track routine (DSKPRP)

  Used to test newly manufactured, sectored, and nonsectored disk packs for surface defects, generate initial track records recognized by the operating system, and test for defective tracks without destroying the data on that track. Also used to prep diskettes.

- Tape prep routine (TPREP)

  Used to prep magnetic tape volumes.

- Sectored disk copy routine (SU$C16)

  Used to copy any part or all of a sectored disk pack to another sectored disk pack in both batch and interactive environments.

- Nonsectored disk copy routine (SU$CSL)

  Used to copy any part or all of a nonsectored disk pack to another nonsectored disk pack in both batch and interactive environments.

- Disk dump/restore routine (DMPRST)

  Used for dumping disk to tape, restoring disk from tape, copying disk to disk, and copying tape to tape. DMPRST can be used in either batch or interactive environments.

- Hardware utilities (HU)

  Permits the interactive use of the dump/restore and disk copy routines.

- System utility symbiont (SL$$SU)

  Permits a multitude of system utility functions to be initiated through the system console. Card-to-card, tape-to-tape, and volume table of contents (VTOC) listings are typical of the functions that can be performed by this utility.

- Stand-alone disk copy routine (SU$IDA or SU$SEL)

  Copies any disk pack to another disk pack of a similar type. Executes as a dialog with the operator via console messages.

- Diskette utility (CREATE)

  This is a 2-part utility to create and maintain data files on diskette devices. File space is created through a canned job control stream (WRT) and file data is added, deleted, etc, through user commands entered in a dialog format at the system console.

- Listing of Software Maintenance Corrections (SMCLIST)

  Lists the software maintenance corrections contained in the SMCLOG file.

STATEMENT CONVENTIONS

The conventions used in writing general, SAT librarian, MIRAM librarian, linkage editor, and system utilities OS/3 statements are as follows:

General Conventions

- Positional parameters must be written in the order specified in the operand field and must be separated by commas. When a positional parameter is omitted, the comma must be retained to indicate the omission, except for omitted trailing parameters. Positional parameters must precede keyword parameters.

- A keyword parameter consists of a word or a code immediately followed by an equal sign, which, in turn, is followed by a specification. Keyword parameters can be written in any order in the operand field. Commas are required only to separate parameters.

- A positional or keyword parameter may contain a sublist of parameters, called subparameters, separated by commas and enclosed in parentheses. The parentheses must be coded as part of the list. Subparameters within the parentheses retain the comma if a parameter is omitted, except for trailing parameters.

- Capital letters, commas, equal signs, and parentheses must be coded exactly as shown. The exceptions are acronyms, which are part of generic terms representing information to be supplied by the programmer.

- Lowercase letters and words are generic terms representing information that must be supplied by the user. Such lowercase terms may contain hyphens and acronyms (for readability).

- Information contained within braces represents mandatory entries of which one must be chosen.

- Information contained within brackets represents optional entries that (depending upon program requirements) are included or omitted. Braces within brackets signify that one of the specified entries must be chosen if that parameter is to be included.

- An optional parameter containing a list of optional entries may have a default specification, which is supplied by the operating system when the parameter is not specified by the user. Although the default may be specified by the user with no adverse effect, it is considered inefficient to do so. For easy reference, when a default specification occurs in the format delineation, it is printed on a shaded background. If, by parameter omission, the operating system performs some complex processing other than parameter insertion, it is explained in an "if omitted" sentence in the parameter description.

- An ellipsis (series of three periods) indicates the omission of a variable number of entries.

- Commas are required when positional parameters are omitted, except after the last parameter.

SAT and MIRAM Librarian Conventions

Control statements may be written in free form.

- Each operation code is composed of an operation identifier, which describes the function. The operation code may be followed by a character string signifying options that alter normal processing of the function. The character string is separated from the operation identifier by a period. Twenty command functions are supported to accomplish the various processes for the SAT librarian, and five are supported for MIRAM librarian functions.

- The operand field of each statement is composed of a variable set of positional parameters, some of which are optional. Optional parameters are indicated by brackets; choice alternatives are indicated by braces. Operands must be separated from the operation field by at least one blank space.

- Commas are required when positional parameters are omitted, except after the last parameter specified.

  Example:

  ```
  positional-parameter-1,positional-parameter-2,,positional-parameter-4
  ```

- Prime librarian control statements may appear in any logical sequence within the librarian update control stream. Subfunction control statements must follow their associated prime control statements.

- Each file and module name may be composed of up to eight characters. Inserted comments describing specific modules may consist of up to 30 characters, including embedded blanks.

The coding format of all the librarian control statements is:

1 LABEL	10 △OPERATION△	16 OPERAND	73 SEQUENCE
unused	function [.options]	p1,p2,p3,p4,p5	seq-no

where:

function
Is the mnemonic of the librarian process to be performed.

.options
Is a string of one or more of the following letters, depending on the specified function.

A Specifies that all groups are to be processed. (G option must be used.)

C Specifies that the name parameter in the operand field is a module name prefix or a group name prefix rather than a complete name.

D Specifies that the entire module or module group being processed is to be listed on the librarian map. (This may also be used to obtain a table of contents for a specified library file.)

E Specifies that the card modules are terminated when the librarian detects the first EOD statement following the ELE statement in the control stream.

G Specifies that the name parameter in the operand field is a group name rather than a module name. This option initiates processing of only one group unless the C or A option is also specified. Whenever this option is used, the module type parameter must be omitted from the operand field.

M Specifies that the module identified in the operand field is to be processed only if a like module is in the output file.

N Specifies that the printing of header records on the librarian map is to be suppressed.

P Specifies that the entire module, or module group being processed, is to be reproduced on punched cards.

Q Specifies that the module identified in the operand field is to be processed only if no like module is in the output file.

U Specifies that processing is to be performed on all modules from the current position of the file up to and including the module identified in the name parameter. Whenever this option is used, the type of the module identified in the name parameter must also be specified in the operand field, unless the G option is also being used.

X Extend an unblocked, single phase, load module.

NOTES:

1. *If contradictory options are specified for a single SAT or MIRAM librarian function, a diagnostic message is printed on the librarian map and the last specified option is honored.*

2. *Unless the RES control statement is specified, the file pointer is positioned at the current position of the file rather than at the beginning of the file. Therefore, when using options C, G, or U, specify the RES control statement to access the file.*

p1

1. Is a logical file name.
2. Is a group name.

p2

1. Is a module type.
2. Is a logical file name.
3. Is a sequence control field.

p3

1. Is a module name.
2. Is a sequence control field.

p4

1. Is a logical file name.
2. Is a comment.
3. Is a sequence control field.

p5

Is a comment.

seq-no

Is a 1- to 8-character alphanumeric sequence control number of which at least one character must be numeric.

Linkage Editor Conventions

The coding format of all the linkage editor control statements is:

1 LABEL	10 △OPERATION△	16 OPERAND	72
1 to 8 characters	Must be delimited by blank columns	Cannot extend beyond column 71; continuation statements are not allowed	Not used

- The label field begins in column 1, is ended by a blank column, and may contain up to eight alphanumeric characters. This field is blank for all linkage editor control statements except the equate statement and comment cards, which contain an asterisk in column 1.

- The operation field, which must be preceded and followed by at least one blank column, contains the operation code of the function to be performed. If the label field is blank, this field may start in column 2.

- The operand field begins with the first nonblank character following the operation field, is ended by a blank column, and cannot extend beyond column 71. The field may contain multiple operands, depending on the function to be performed, and these operands must be separated by commas.

- Continuation statements are not allowed.

- Comment statements, which are identified by an asterisk in column 1, may appear anywhere in a linkage editor control stream. Comments are printed on the process map portion of the link-edit map. They do not begin any processing by the linkage editor.

2. SAT Librarian (LIBS) Control Statements

BLK

Function:

Used to convert your standard load module to a block load module. Block load modules increase the efficiency of program loading in that all or large parts of overlay phases may be loaded by a single I/O operation.

Format:

LABEL	△OPERATION△	OPERAND
unused	BLK	input-lfn,module-name[,output-lfn]

Parameters:

input-lfn
Logical file name of the disk file.

module-name
Is a 1- to 6-alphanumeric character string specifying the name of the load module being blocked.

output-lfn
Logical file name of the disk file being used as output.

If omitted, the input file contains the blocked module and the original load module is nullified.

BOG

Function:

Begins a module group by writing a beginning-of-group record in a specified disk or tape file.

Format:

LABEL	△OPERATION△	OPERAND
unused	BOG	group-name [{lfn / YRUN}]

Parameters:

group-name
One to eight alphanumeric characters specifying the name of the module group being started.

lfn
Logical file name of the disk or tape file on which the beginning of module group record is written.

If omitted, YRUN is used.

COM

Function:

Permits the comparison of a single source or proc module in two separate files on a record-by-record basis or the comparison of two complete files on a block-by-block basis. No other options are available with this statement.

Format:

LABEL	△OPERATION△	OPERAND
unused	COM	[{prim-lfn / SYSRUN}][,{S / M}][,{n-n / 73-80}] [,name],sec-lfn

Parameters:

prim-lfn
Logical file name of the disk or tape file used in the comparison.

If omitted, YRUN is used.

S, M
Specifies the type of modules being compared as either a program source module (S) or macro/jproc module (M).

If omitted, all modules in both files are compared.

n-n
Two decimal numbers separated by a hyphen; specifies the starting and ending columns of the sequence control field used if a source level module is compared.

If *name* is not specified, this parameter is ignored.

name
One to eight alphanumeric characters specifying the name of a source module being compared. The module named is first located in both files, then compared. Each module must be a source or proc module.

If omitted, the files designated are compared on a block-by-block basis from beginning to end, and the name parameter is ignored.

sec-lfn
Logical file name of the second disk or tape file used in the comparison.

COP

Function:

Used to copy an entire library file, module groups, or specific individual program modules. It effectively produces a new library file and allows for an exclusive reproduction of one module or group or the inclusive reproduction from current position up to and including the named module or group. The copy is selective if all code of a certain type is being copied. Modules or groups of modules may be listed, punched, or copied. File compression is performed as the new file is created, thus eliminating file fragmentation created by nullified modules or module groups.

NOTE:

When you use the copy facility to create a new tape file and want only the modules being copied to make up the new file, you must use a previously prepped tape free of any data. You must follow this procedure because the copy facility does not reinitialize an output file. All modules being copied are automatically written to the end of your output file.

Format:

LABEL	△OPERATION△	OPERAND
unused	COP[.options]	[{input-lfn / SYSRUN}] [,{S / M / O / L}] [name][,output-lfn]

Options:

A Process all groups specified by the name parameter. (The G option must also be used.)

C Name specified in the *name* parameter is either a module name prefix or group name prefix.

D List all the modules copied, or if a table of contents is being produced, list all the records in the file directory.

G Name specified in the name parameter is a module group name or a module group name prefix if the C option is also specified. If neither the C or A option is used, process only the first group found with the specified name. If the C option is used, process all groups with the specified prefix. If the A option is used, process all groups with the specified name. If the G option is used, the module type parameter should be omitted.

M Copy a specified module from the input file to the output file only if it causes an existing module in the output file to be deleted.

N Do not list any header records.

P Punch all modules processed. This option cannot be used when requesting a table of contents for a file.

Q Copy a specified module from the input file to the output file only if it does not cause an existing module in the output file to be deleted.

COP

U Process the modules from the current position of the input file, up to and including the specified module or module group. This option is ignored when producing a table of contents for a file.

Parameters:

`input-lfn`
Logical file name of the disk or tape input file.

If omitted, YRUN is used. If used with the D option, one trailing comma, and no other parameters, all modules in the file are listed on the librarian map but no copy operation is performed.

`S, M, O, L`
Type of module being copied as program source (S), macro/jproc (M), object (O), or load (L).

If omitted, all modules with the specified name are copied.

`name`
One to eight alphanumeric characters specifying the module name, name prefix (C option), or name of the group (G option) copied.

If omitted, all modules from current position to the end of file of the specified file type are copied. If no type or name is specified, all modules from current position to end of file are copied.

`output-lfn`
Logical file name of the disk or tape output file used in the copy operation. The output file specification is not necessary to position a file, to list a disk file directory, or to list and punch specified modules in a file.

If omitted, only a subfunction (list, punch, position) of the COP statement is performed. However, if the input-lfn, module-type, and module-name are specified and the output-lfn is omitted, the file pointer is positioned to the next module after the specified module in the file.

COR

Function:

Used to correct either program source (S), macro/jproc (M), object (O), or load (L) modules. Correction cards follow this statement, specifying how the module is to be corrected. The librarian EOD statement indicates the end of the correction cards. The corrected module may be output to the same or another disk file.

Correction cards are based upon the type of module (source or nonsource). For a description of the correction cards used with the COR statement, see Appendix C.

Deletion or reorganization of source module records is accomplished by using REC, SEQ, and SKI as subfunctions of the statement.

Format:

LABEL	△OPERATION△	OPERAND
unused	COR[.options]	[{input-lfn / YRUN}],{S / M / O / L},name[,output-lfn]

Options:

E Terminate at the first EOD card.

N Do not list header records, subfunction control statements, or records added or deleted.

P Punch module.

X Extend the load module if any of the supplied patch addresses are beyond the end of the module. This option can only be used for unblocked single phased load modules.

Parameters:

`input-lfn`
Logical file name of the disk or tape file containing the module being corrected.

If omitted, YRUN is used.

`S, M, O, L`
Type of module being corrected as program source (S), macro/jproc (M), object (O), or load (L).

`name`
One to eight alphanumeric characters specifying the name of the module being corrected. If tapes are being used, two unique files are required.

`output-lfn`
Logical file name of the disk or tape file into which the corrected module is placed.

If omitted, a delete/add operation is performed within the input file.

DEL

Function:

Used to mark modules or module groups for deletion. Once marked for deletion, the information in the module or module group cannot be read or changed. The actual physical deletion is done by the COP and PAC control statements.

Format:

LABEL	△OPERATION△	OPERAND
unused	DEL[.options]	[{lfn / SYSRUN}][,{S / M / O / L}][,name]

Options:

A Delete all groups with the specified group name. (The G option must also be used.)

C Delete all modules whose name begins with prefix appearing in *name* parameter.

D List deleted modules.

G Name specified in the name parameter is a module group name or a module group name prefix if the C option is also specified. If neither the C or A option is used, delete only the modules in the first group found with the specified name. If the C option is used, delete all module groups with the specified group name prefix. If the A option is used, delete all module groups with the specified group name. If the G option is used, the module type parameter should be omitted. When a module group is deleted, the BOG and EOG records associated with the group are also deleted.

N Do not list header records of deleted modules.

P Punch deleted modules.

U Deletion starts at current file position and continues up to and including the named module. If a module name is specified, then a type must also be included. If module name is omitted, delete rest of file.

Parameters:

`lfn`
Logical file name of the disk file.

If omitted, YRUN is used.

`S, M, O, L`
Type of module to be deleted - program source (S), macro/jproc (M), object (O), or load (L).

DEL

If omitted, delete all modules explicitly named regardless of type.

name
One to eight alphanumeric characters specifying a name for the module or module prefix being deleted.

If omitted, all modules of specified type are deleted. If both type and name are omitted, all remaining modules are deleted.

ELE

Function:

Adds a program source (S), object (O), macro/jproc (M), or load (L) module (contained on cards) to a disk or tape file. If a card file element is being added to a disk file that already contains a module of the same name and type, the old module is replaced by the new module. A module header record is inserted in the specified output file. All cards immediately following the ELE card down to the EOD card are assumed to be the module being added. Librarian control streams are valid source modules, but each EOD card that is a part of that control stream must be associated with its own correct module (COR) or ELE control statement. The librarian does not terminate the add module operation until an unattached EOD card is detected, unless the E option is specified.

Format:

LABEL	△OPERATION△	OPERAND
unused	ELE[.options]	[{lfn / YRUN}],{S / M / O / L},name[,comments]

Options:

D List the module being added.

E Terminate at the first EOD.

N Do not list the header record.

P Punch the module being added.

Parameters:

`lfn`
Logical file name of the disk or tape file to which this card module is added.

If omitted, YRUN is used.

`S, M, O, L`
Type of module being added as program source (S), macro/jproc (M), object (O), or load (L).

`name`
One to eight alphanumeric characters that specify a name for the module being added.

For object and load modules, the name on the ELE card must be the same as the name of the module.

`comments`
Up to 30 characters of comments for insertion into the module header record.

If omitted, no comment is included in the header record.

EOD

Function:

Ends the correction card data that follows either an ELE, COR, or REPRO control statement. Each EOD card must be associated with one and only one ELE or COR card.

Format:

LABEL	△OPERATION△	OPERAND
unused	EOD	unused

EOG

Function:

Ends a module group by writing an end-of-group record in a specified file.

Format:

LABEL	△OPERATION△	OPERAND
unused	EOG	group-name [,{lfn / YRUN}]

Parameters:

group-name
Name of the module group being ended.

lfn
Logical file name of the disk or tape file on which the end-of-group record is written.

If omitted, YRUN is used.

ESC

Function:

Causes the librarian to read all subsequent librarian control statements from the user-specified tape, from disk or diskette SAM file, or from a disk source module. Processing ends when an end of file or end of module is detected. There are two formats for the ESC control statement:

- Format 1 to read from user-created SAM files.
- Format 2 to read from source module.

Format 1:

LABEL	△OPERATION△	OPERAND
unused	ESC	filename,{TP / DK / DT}[,{FU / FB / VU / VB}][,{record-length / 80}] [,{block-length / 80}]

Parameters:

`filename`
Name of the user-created SAM file containing the librarian control stream to be processed. The first character must be alphabetic.

`TP, DK, DT`
Specifies the type of file to be processed.

`TP`
Tape files.

`DK`
Disk files.

`DT`
Diskette file.

`FU, FB, VU, VB`
Specifies the record type for the file being read.

`FU`
Fixed, unblocked.

`FB`
Fixed, blocked.

ESC

VU
Variable, unblocked.

VB
Variable, blocked.

If omitted, fixed, unblocked records are assumed.

`record-length`
Specifies the length of fixed records in bytes. Maximum permissible entry is decimal 128. For tape, the minimum entry is decimal 18. If omitted, a record length of 80 bytes is assumed. No entry required for variable length records. This field is ignored for fixed, unblocked records.

`block-length`
Indicates the length in bytes of the file blocks, including all block header and record header fields. If block length exceeds 1024 decimal bytes, additional storage is required. For tapes, the minimum entry is decimal 18. If your job requires additional storage as a result of ESC processing, it must have been previously allocated on the // JOB job control command. For information on determining the amount of storage required, see the system service programs user guide, UP-8062 (current version).

This entry indicates the maximum block size for variable length records. If omitted, a block size of 80 bytes is assumed.

Format 2:

LABEL	△OPERATION△	OPERAND
unused	ESC	`filename,LD,modulename`

Parameters:

`filename`
Specifies the name of the file containing the librarian control stream.

`LD`
Indicates the control stream is in a librarian-created source module.

`modulename`
Specifies the name of the librarian source module containing the control stream to be processed.

FIL

Function:

Declares to the librarian all the tape and disk files that are referenced subsequently in the control stream. It also assigns to each file a type code (D for disk and T for tape) and a logical file number (0–15), which together form a logical file name that is used for all subsequent file references within the librarian control stream. File declarations may be strung out on one FIL card or may be made individually on separate FIL cards. An appropriate job control file declaration card is required in the job control stream for each file described by the FIL statement unless a system or job run library file is being used.

Format:

LABEL	△OPERATION△	OPERAND
unused	FIL	{Tn / Dn}=filename-1[,...,{Tn / Dn}=filename-n]

Parameters:

`Tn=filename`
Equates a tape file (file name) with a logical file name of T0 through T15.

`Dn=filename`
Equates a disk file (file name) with a logical file name of D0 through D15.

NOTE:

The file name specification may not exceed eight alphanumeric characters and must begin with an alphabetic character. Equating the names of files to be processed with logical file names allows a single LIBS control stream to be used to maintain any number of different files. (Functions performed by the control stream use the logical file specifications.) When files change, only the FIL statements need be modified; thus each command to the librarian need not specify the actual file name used.

LST

Function:

Displays the contents of a file in alphabetical sequence.

Format:

LABEL	△OPERATION△	OPERAND
unused	LST	[{input-lfn / SYSRUN}][,{S / M / O / L}]

Parameters:

input-lfn
Logical file name of disk file containing the modules to be listed.

If omitted, YRUN is used.

S, M, O, L
Type of module being printed is either program source (S), macro/jproc (M), object (O), or load (L).

If omitted, entire file is listed.

PAC

Function:

Used to condense a library file by discarding any modules that were marked for deletion and squeezing the remaining modules together. This function may be used with DEL and ADD control statements to build a reordered, updated, and packed library file. The printing of header records is done for both the modules being packed and the modules not being packed under the headings MODULES MOVED and MODULES NOT MOVED, respectively.

PAC should not be specified to pack a file containing an ICAM symbiont while that symbiont is active.

Format:

LABEL	△OPERATION△	OPERAND
unused	PAC[.options]	[{lfn / YRUN}]

Option:

N Do not list header records.

Parameter:

lfn
Logical file name of the disk file being compressed.

If omitted, YRUN is compressed.

PAGE

Function:

Causes the librarian to start a new page on the librarian map. It may also specify a header line to be printed at the top of that new page and succeeding pages for the duration of the job step or until another PAGE statement is encountered.

Format:

LABEL	△OPERATION△	OPERAND
unused	PAGE	['header line']

Parameter:

'header line'
Specifies the contents of the header line to be printed on the top of the new page and succeeding pages. The header line must be enclosed in single quotes and can be 64 characters in length.

PARAM ERROR

Function:

This // PARAM statement either causes the librarian to terminate the job step (normal termination, STOP) or to cancel the entire job (CANCEL) after encountering an error.

Format 1:

```
// PARAM ERROR=STOP
```

Parameter:

ERROR=STOP
Terminates the job step when an error is encountered.

Format 2:

```
// PARAM ERROR=CANCEL
```

Parameter:

ERROR=CANCEL
Terminates the entire job when an error is encountered.

PARAM PRINT

Function:

This // PARAM statement is used to suppress the printing of the librarian map. It must be the first parameter in the job step (it must precede the /$ card in the control stream). The printer device assignment set need not be specified.

Format:

```
// PARAM PRINT=OFF
```

Parameter:

PRINT=OFF
Suppresses printing of the librarian map.

NOTE:

This // PARAM statement must precede the /$ card in the control stream.

PARAM PRTOBJ

Function:

This // PARAM statement causes source or macro/jproc modules to be printed in hexadecimal format. It remains in effect for the entire librarian job step.

Format:

```
// PARAM PRTOBJ=ON
```

Parameter:

PRTOBJ=ON
Causes source or macro/jproc modules to be printed in hexadecimal format.

NOTE:

If used, this // PARAM statement must be placed between the // EXEC LIBS statement and the /$ statement in the job control stream.

PARAM TAPEFILES

Function:

This // PARAM statement is used to allow multiple files to be written to the same tape volume. All files must be written to this volume in a single session when this statement is used. These files cannot be extended later.

As the files are being created, only one file on the tape can be open at a time. Therefore, the same logical file name (Tn) must be used for every tape file, and a FIL statement must be used to redefine this logical file name each time another file is to be processed.

If this statement is not used, only one file can be written to each tape. This statement is not required to read a file on a multifile tape.

Format:

```
// PARAM TAPEFILES=MULTI
```

Parameter:

`TAPEFILES=MULTI`
Allows multiple files to be written to the same tape volume.

NOTE:

If used, this // PARAM statement must be placed between the // EXEC LIBS statement and the /$ statement in the job control stream.

PARAM UPDATE

Function:

This // PARAM statement is used to specify the date and time to be in effect during the execution of a librarian job. The date and time are inserted in the header records of modules being corrected by the librarian. If a // PARAM UPDATE statement is not included in the librarian control stream, the date and time contained in the system information block (SIB) are used. The date and time remain in effect until the librarian job is terminated.

Format:

```
// PARAM UPDATE=yymmdd/hhmm
```

Parameter:

UPDATE=yymmdd/hhmm
Specifies a date and time to be used for modules being corrected during the execution of the job.

REC

Function:

Repositions the record pointer of the original program source or macro/jproc module back to the first record in the module. Used with the COR control statement to correct source or proc modules. Also used with the SKI control statement to rearrange major segments of a program source or macro/jproc module. Repositioning of the record pointer occurs immediately if the sequence field of the REC statement is blank (omitted).

Format:

LABEL	△OPERATION△	OPERAND	SEQUENCE
	REC	unused	[last-sequence-no.]

Parameter:

last-sequence-no
One to eight alphanumeric characters identifying the sequence number of the last record to be copied into the new data set before the record pointer is repositioned back to the first record in the module. This field begins in column 73 unless a SEQ control statement dictates otherwise.

If omitted, the repositioning operation takes place without any records being copied into the new data set.

REN

Function:

Used to rename a specific module, module group, or record; to change the comment field in a header record; or to mark object and load modules as sharable or unsharable. The records that can be renamed include control section, common section, and ESD records within object modules and alias phase names in load modules.

Format:

LABEL	△OPERATION△	OPERAND
unused	REN[.options]	[{lfn / SYSRUN}] [,{S / M / O / L}] ,old-name[,{record-type-and-name / RON / ROFF}] [,new-name] [,comments]

Options:

G Names specified are group names. The first module group encountered with the name identified as *old-name* is to be renamed.

N Do not list header records.

Parameters:

`lfn`

Logical file name of the disk file containing the modules being renamed or identified as reentrant or nonreentrant.

If omitted, YRUN is assumed to contain the modules.

`S, M, O, L`

Type of modules being operated on as program source (S), macro/jproc (M), object (O), or load (L).

If omitted, all modules of the specified old name are affected.

REN

old-name[,{record-type-and-name / RON / ROFF}]

Identifies the module group, or record to be processed, or the object to be marked as reentrant (RON) or nonreentrant (ROFF). The record type codes that may be specified are:

C Indicates COM.

E Indicates ENTRY.

N Indicates procedure name.

P Indicates alias phase name.

S Indicates CSECT.

V Indicates V-CON.

X Indicates EXTRN.

Record names can be one to eight characters long. The record type and name specification cannot contain any embedded blanks.

When program source, macro/jproc, object, or load modules are being renamed or their header record comment field is being changed, the first 1- to 8-character name is sufficient; if a record within an object module is being renamed, record type and old record name also must be provided. If an alias phase name is being changed, record type and old alias phase name must be specified.

new-name

Specifies the new name to be substituted for the old name. If renaming a multiphase load module, only the first six characters can be changed; the last two remain the same. If you are changing the shareability status of a module or only the comment field of a header record, the new name is not necessary.

comments

A string of up to 30 characters of comments that is to be inserted into the header record of the identified module.

If omitted, current comments remain unchanged.

REPRO

Function:

Used to add or delete linkage editor control statements within object modules. If addition is specified, the new linkage editor control statements are inserted in the object module immediately after any control statements that may already be present in the named object module. If deletion is specified, the linkage editor control statements are deleted from either the end of the object module header or transfer record and may be replaced with new control statements.

Two EOD control statements are always required when the REPRO function is specified. The linkage editor control statements prior to the first EOD are inserted following the object module header record. The control statements after the first EOD are inserted following the object module transfer record. The second EOD delimits the control statement insertions.

Format:

LABEL	△OPERATION△	OPERAND
unused	REPRO[.options]	[{lfn / YRUN}],module-name[,#deletions] [,#deletions]

Options:

D List entire module.

N Do not list header records.

P Punch module.

Parameters:

`lfn`
Disk file where object module is located.

If omitted, YRUN is used.

`module-name`
Object module being modified.

`#deletions`
Decimal value indicating the number of control statements to be deleted following the object module header record.

`#deletions`
Decimal value indicating the number of control statements to be deleted following the object module transfer record.

RES

Function:

For disk files, resets the current position pointer to the beginning of the file; for tape files, rewinds the tape to load point. When an output tape file is rewound, a tape mark is written before rewinding occurs. If a module name and type are specified, the current position pointer in disk or tape files points to the first record of the named module. If a module of the name and type specified is not found, the current position pointer remains as it was before the RES statement was processed (search ends at point of origin) and an appropriate diagnostic is printed on the map.

Format:

LABEL	△OPERATION△	OPERAND
unused	RES[.options]	[{lfn / YRUN}][,{S / M / O / L}][,name]

Options:

G *Name* parameter is the name of a group. The file position pointer points to the first record of the first group encountered with the name specified.

Parameters:

lfn
Logical file name of the disk or tape file being reset.

If omitted, resets YRUN.

S, M, O, L
Type of module being reset as program source (S), macro/jproc (M), object (O), or load (L).

If omitted, it is assumed that the reset operation is directed to a file rather than a module or module group.

name
Name of the module or module group to which the current position pointer points.

If omitted and a module type is not specified, it is assumed that the reset operation is directed to a file. Otherwise, an error message is listed to indicate its omission.

SEQ

Function:

Permits the sequence checking, sequencing, or resequencing of program source or macro/jproc modules. It does not apply to object or load modules. This statement can be used by itself or with the COR or ELE control statements. Used with ELE, it sequence checks (sequence numbers already assigned), sequences (generates new sequence numbers), or resequences (disregards old sequence numbers and generates new ones) on source or proc modules being filed. Used with COR, it enables the correction of a source or macro/jproc module by using the sequence numbers for control.

When the SEQ control statement is used with a tape library, the SEQ control statement must be used as a subfunction control statement to the COR or ELE control statement. When the SEQ control statement is used as a subfunction control statement, any options specified are ignored.

Format:

LABEL	△OPERATION△	OPERAND
unused	SEQ[.options]	[{lfn / **SYSRUN**}],{S / M}[,module-name] [,{column-position / **73**}][,{content / SAME / **00000000**}] [,{increment / **1**}]

Options:

D List sequenced module.

N Do not list header records.

P Punch sequenced module.

Parameters:

lfn

Logical file name of the disk file in which the program source or macro/jproc module being sequenced or resequenced resides.

If omitted, YRUN is assumed to contain the module. (If a primary function, a delete/add operation is performed.)

S,M

Specifies the type of module being sequenced as either a program source module (S) or a macro/jproc source module (M).

SEQ

module-name
Name of the program source or macro/jproc module being sequenced or resequenced. This parameter is required when the SEQ control statement is used as a primary function. If the module-name is specified on a SEQ command that immediately follows an ELE or COR statement, the module is sequenced as it is added or resequenced as it is corrected, respectively.

If omitted:

- and the SEQ control statement immediately follows an ELE control statement, the SEQ control statement checks the sequence of a program source or macro/jproc module being filed; or
- if the SEQ control statement immediately follows a COR control statement, the SEQ control statement identifies a sequence field that is in the program source or macro/jproc module being corrected and that is used to insert corrections.

column-position
First column position in the program source or macro/jproc record where the sequence field begins and where the sequence data is incorporated.

If omitted, column 73 is assumed to be the first column of the sequence field.

content
A 1- to 8-character value specifying the initial value placed into the sequence field of the first record in the module. The length of this value determines the length of the sequence field. The rightmost portion of the sequence field must be removed. The mixing of sequence numbers with alphabetic and numeric characters is permitted, provided that the alphabetic and numeric string remain intact and the alphabetic characters are left-justified.

SAME
Contents of the sequence field of the first record of the module being resequenced are to remain as they were. This specification assumes that this field occupies eight character positions. If it does not, this parameter should not be specified. Instead, the initial sequence field content should be respecified.

If omitted, the initial sequence contents are assumed to be 00000000.

increment
A decimal number, not to exceed 255, that specifies the sequence increment used in the sequencing process.

If omitted, the increment is assumed to be 1.

SKI

Function:

Used only with the COR control statement to correct program source or macro/jproc modules. The SKI statement allows one or more original records in a program source or macro/jproc module to be skipped by the COR function. The beginning and ending of the skip are specified by record sequence number.

Format:

LABEL	△OPERATION△	OPERAND	SEQUENCE
	SKI[.options]	last-sequence-no.	[starting-sequence-no.]

Options:

D List the records skipped.

Parameters:

last-sequence-no
One to eight alphanumeric characters identifying the sequence number of the last record to be bypassed.

starting-sequence-no
One to eight alphanumeric characters identifying the sequence of the first record to be bypassed. This field begins in column 73 unless a SEQ control statement dictates otherwise.

If omitted, the skip begun immediately, starting with the record immediately following the last record operated on by the COR function.

3. MIRAM Librarian (MLIB) Control Statements

CHG

Function:

Used to change the name of an existing screen format (F-type or FC-type) module, saved run library (J-type) module, help screen module (HELP), or menu module (MENU) or used to insert comments on the header record. Only one change operation per command can be performed.

Format:

LABEL	△OPERATION△	OPERAND
unused	CHG	input-lfn,module-type,old-name,{N / C},{new-name / comments}

Parameters:

`input-lfn`
Specifies the file number (as defined by the FIL statement) of the file to be used as input. Also, the updated header record will be written to this file.

`module-type`
Specifies the type of module being processed as a screen format module (F or FC), a saved run library module (J), help screen module (HELP), or menu module (MENU). No default is allowed.

`old-name`
Specifies the name of the module being processed. No default is allowed.

`N, C`
Specifies the type of change being performed as either a name change (N) or a comment insert (C). No default is allowed.

`new-name,comments`
Specifies the information required by the type of change parameter. If N was specified, a valid module name must be specified. If a module with the same name and type already exists in the file, an error will result and the change will not occur. If C was specified, a comment (up to 30 characters) enclosed by single quotes must be specified.

COP

Function:

Used to copy the nondeleted contents of an entire library file to another library file, creating a compressed output file, or to copy from one file to another individual modules based on module names or types or on module name prefixes.

Format:

LABEL	△OPERATION△	OPERAND
unused	COP[.options]	input-lfn,[module-type],[name],output-lfn

Options:

C Specifies that the name specified in the *name* parameter is a module name prefix.

Parameters:

`input-lfn`
Specifies the file number (as defined by the FIL statement) of the file to be used as input.

`module-type`
Specifies the type of modules being copied as screen format modules (F or FC), saved run library modules (J), help screen modules (HELP), or menu modules (MENU). If no type is specified, all modules with the specified name are copied.

`name`
Specifies the name or prefix (up to eight characters) of the module. If no name is specified, all modules of the specified type are copied.

`output-lfn`
Specifies the file number (as defined by the FIL statement) of the file to be used as output. No default is allowed.

NOTE:

If the name and type specification are both omitted, the entire file is copied.

DEL

Function:

Used to delete the active contents of an entire MIRAM library file or to delete individual modules based on prefixes, module names, and/or types.

Format:

LABEL	△OPERATION△	OPERAND
unused	DEL[.options]	input-lfn,[module-type],[name],output-lfn

Options:

C Indicates that the name specified in the *name* parameter is a module name prefix.

Parameters:

`input-lfn`
Specifies the logical file name (as defined by the FIL statement) of the input file.

`module-type`
Specifies the type of modules being deleted as screen format modules (F or FC), saved run library modules (J), help screen modules (HELP), or menu modules (MENU). If omitted, all modules with the specified name are deleted.

`name`
Specifies the name or prefix of the module being deleted.

NOTE:

If no type is specified, the module name must be specified or the entire file is deleted.

FIL

Function:

Declares to the librarian all MIRAM files that will be referenced subsequently in the control stream. At the same time, each file is assigned a logical file number (0–29), which forms a logical file name that is to be used (rather than the filename) for all subsequent file references within the control stream. File declarations may be strung out on one FIL statement or be made individually on separate FIL statements. Up to 30 files can be declared in your control stream.

Format:

LABEL	△OPERATION△	OPERAND
unused	FIL	Fn=filename-1[,...filename-n]

Parameter:

Fn=filename

Specifies that the MIRAM file (filename) is equated with the logical file name (F0–F29), which forms a logical file name that is to be used for all subsequent file references within the control stream.

NOTE:

The file name specification may not exceed eight alphanumeric characters and must begin with an alphabetic character.

PRT

Function:

Used to print modules or file directories.

Format:

LABEL	△OPERATION△	OPERAND
unused	PRT[.options]	input-lfn,[module-type],[name]

Options:

C Indicates that the specified name is a prefix.

D Indicates that a file directory is printed. If specified, the *name* parameter must be omitted.

Parameters:

`input-lfn`
Specifies the file number (as defined by the FIL statement) of the input file. No default is allowed.

`module-type`
Specifies the type of modules being printed as screen format modules (F or FC), saved run library modules (J), help screen modules (HELP), or menu modules (MENU). If the D option is used and F is specified, screen format directory entries are printed. If the D option is used and J is specified, saved run library directory entries are printed. Whenever the D option is specified and the type is omitted, the entire directory is listed. If both the D option and type parameter are omitted, all modules of the specified name are printed.

`name`
Specifies the name or the prefix of the module to be printed. When you are using the D option, this parameter must be omitted. If the C option is used, the name is used as a prefix, and all modules of the specified type beginning with this prefix are printed. If this parameter is omitted, all modules of the specified type are printed.

NOTE:

If no options are specified and name and type are omitted, all modules in the specified file are printed.

4. Linkage Editor Control Statements

ENTER

Function:

Provides the address of the entry point of the load module phase being built. This is the address to which control is normally transferred if the phase is loaded by a supervisor FETCH macroinstruction. The ENTER statement is usually the last control statement for a phase. It may be followed by a MOD, EQU, RES, or comment statement in addition to an OVERLAY statement for the next phase, a REGION statement for new region, a LOADM or LINKOP statement for a new load module, or an end-of-data (/*) statement, which ends execution of the linkage editor.

If the ENTER control statement is immediately followed by an INCLUDE control statement, a diagnostic warning indicating the sequence error is listed on the link-edit map, but the INCLUDE control statement is processed and included in the current phase.

If no ENTER statement is provided for a phase, the transfer address is obtained from the first object module transfer record in the phase that has a valid transfer address. If no object module supplies a valid transfer address, the entry point address is the relocated address assigned to the first CSECT specifically included in the phase.

Format:

LABEL	△OPERATION△	OPERAND
	ENTER	[expression]

Parameter:

expression

Transfer address for the phase. This expression, which usually represents a relative phase address, may have one of the following forms:

- A decimal number from one to eight digits long
- A hexadecimal number from one to six digits long, in the form X'nnnnnn'
- A previously defined label (the name of a control section or an entry point in an object module that was previously included or defined by a previous EQU statement). For the link edit of a nonreentrant module, this symbol must not be a shared definition.
- A previously defined label plus or minus a decimal or hexadecimal number as previously described

If omitted, the transfer address is that address assigned to the base (node point) of the phase.

EQU

Function:

Provides the linkage editor with the value of a label that would not otherwise be defined. The definition of a symbol by an EQU control statement is subject to the same rules for automatic deletion as entry points.

Format:

LABEL	△OPERATION△	OPERAND
symbol	EQU	expression

Label:

symbol
One to eight alphanumeric characters specifying the label to be defined.

Parameter:

expression
Value to be assigned to the label. The expression may have one of the following forms:

- A decimal number from 1 to 10 digits long
- A hexadecimal number from one to eight digits long, in the form X'nnnnnnnn'
- A previously defined label (the name of a control section or an entry point in an object module that was previously included or defined by a previous equate statement)
- A previously defined label plus or minus a decimal or hexadecimal number, as previously described. For the link edit of a nonreentrant module, this label must not be a shared definition.
- An asterisk (*) to indicate a reference to the current value of the location counter

INCLUDE

Function:

Requests that a specific object module or selected control sections of a specific object module be included in the current phase of the load module being built. This control statement may follow any linkage editor control statement except the ENTER control statement, or it may be embedded in an object module. Nesting of embedded INCLUDE control statements may continue indefinitely. Nested INCLUDE control statements are identical in format and capability to those not nested and may thus specify full or partial inclusion, as well as alternate files. The same applies during the automatic inclusion process, although the initial module located and used to satisfy reference always is included in full.

If this control statement is omitted from an otherwise valid linkage editor control stream, all the object modules currently residing in the system job run library (YRUN) are included in the load module phase being built.

Format:

LABEL	△OPERATION△	OPERAND
	INCLUDE	[module-name][,(s1,...s9)][,filename]

Parameters:

module-name
One to eight alphanumeric characters identifying the object module to be included.

If omitted, it is assumed that the INCLUDE control statement is nested and that the object module being referenced immediately follows the previously referenced object module in the library being accessed. Otherwise, an error message is output on the link-edit map.

s1 through s9
Is a list of one to nine control section labels that identify the CSECTs to be included in the load module phase being built. The control sections referenced in these subparameters must be contained in the object module referenced by the INCLUDE control statement; otherwise, an error diagnostic is output on the link-edit map. The order in which the CSECTs appear in the object module is the order in which they will appear in the load module, regardless of the order in which they are listed in these subparameters.

When a subparameter list is specified, no unnamed CSECT in the module being scanned is ever included in the load module, but all requests for common storage are honored, as are all embedded control statements.

If omitted, the entire object module referenced is included in the load module except for those CSECTs that may be automatically deleted by the linkage editor.

INCLUDE

`filename`

One to eight alphanumeric characters identifying the symbolic name of the file in which the referenced object module is stored. This name must be specified exactly as it is specified in the job control label file directive (LFD) that identified the file; otherwise, an error message is output on the link-edit map.

If omitted, the object module is assumed to be in YRUN or, if not there, in the last library specified in an INCLUDE control statement or, if not there and a file name was specified in a previous // PARAM statement, in the file specified in the // PARAM statement; otherwise, the object module is assumed to be in the system object library file (YOBJ).

LINKOP

Function:

Specifies the options to be used in creating the load module. This control statement may appear only within the linkage editor control stream, within the data definition, and must not be specified in the job control stream proper. The first LINKOP control statement processed for a job establishes the initial linkage editor options for that job. These options take effect as soon as they are detected and remain in effect until changed by a succeeding LINKOP control statement or until the job is ended. This applies to system-supplied (default) parameter specifications, as well as programmer-supplied specifications. (Once a default specification is overridden by the programmer, the programmer-supplied specification remains in effect for the remainder of the job or until it is explicitly respecified by another LINKOP control statement.)

The linkage editor starts the building of a new load module each time it processes a LINKOP control statement. Conflicting specifications are resolved by assuming that the last statement specified is correct and the linkage editor functions accordingly.

If linkage editor options are not specified in the control stream for a given job, the linkage editor functions under the direction of the default parameter specifications as follows:

1. Performs automatic inclusion of required object modules, as necessary, and assumes that YOBJ is the only file that might contain the required modules.

2. Processes V-CON references, as required.

3. Stores the output load modules in YRUN.

4. Uses only the control statements contained in the primary control stream to produce load modules.

5. Generates phase header records that contain blank comment fields and does not require the system loader to clear main storage before the phase is loaded.

6. Processes all the control statements contained in the control stream, even if processing errors that would render a load module useless are detected.

7. Provides for the promotion of common storage areas.

8. Generates a link-edit map for each module generated that lists all the information normally desired in the link-edit map.

9. Produces nonreentrant load modules.

10. Recognizes reentrant object modules and creates the shared records needed to link their load module counterparts with the load module being created.

11. Creates internal symbol dictionary (ISD) records in the load module. At execution time, these records enable JOBDUMP to print a formatted dump of the load module area.

LINKOP

Format:

LABEL	△OPERATION△	OPERAND
	LINKOP	[ALIB=filename] [,CLIB=modulename/filename] [,CMT={'character-string' / '△'}] [,OUT={filename / (N) / SYSRUN}] [,RLIB={filename / SYSOBJ}] [,{CNL / NOCNL}] [,{ZRO / NOZRO}] [,{NOAUTO / AUTO}] [,{NOV / V}] [,{NOPROM / PROM}] [,{NOLIST / LIST}] [,{NOCNTCD / CNTCD}] [,{NOERR / ERR}] [,{NODICT / DICT}] [,{NODEF / DEF}] [,{NOPHS / PHS}] [,{DEL / NODEL}] [,{RCNTCD / NORCNTCD}] [,{REF / NOREF}] [,{SHARE / NOSHARE}] [,{RNT / NORNT}] [,{ISD / NOISD}]

NOTES:

1. *System-supplied parameters are in effect only until changed by the user. Once changed, they must be reset to their default state by the user before the default option is effected again.*

2. *Keywords may be specified in any order but must be separated by commas, with no spaces between the keywords unless they identify a delimited character string.*

3. *Private file names declared are always logical and must be accompanied by the appropriate job control DVC/LFD statements.*

LINKOP

Parameters:

`ALIB=filename`
File to be searched during automatic inclusion processing.

If omitted, only the RLIB-specified file is searched during the automatic inclusion process.

`CLIB=modulename/filename`
Name of a source module, and the file in which it resides, that contains the linkage editor control statements to be processed for this link-edit job.

If omitted, the current control stream source (primary input or source module input) continues to be accessed for control statements.

`CMT='character-string'`
A character string of up to 30 characters to be inserted in the comment field of each phase header record produced for the generated load modules. The character string must be enclosed in apostrophes and may contain blanks, commas, and other special symbols. Phase header comment fields are listed in the allocation map portion of the link-edit map.

`CMT='Δ'`
Phase header comment fields are to be blank.

If omitted, the phase header comment fields are left blank unless a previous CMT keyword specified otherwise.

`OUT=filename`
Library file name in which the output load module is to be stored. If a load module with the same name already exists in the output file specified by this keyword, it is replaced by the new load module.

`OUT=(N)`
Output load modules produced by the linkage editor are not to be stored in any library file; however, the link-edit map is still produced if not inhibited by specification of the NOLIST keyword parameter.

`OUT=$Y$RUN`
Output load modules produced by the linkage editor are stored in YRUN.

If omitted, the load modules produced are output to YRUN unless a previous OUT keyword specified otherwise.

NOTE:

The file designated to store the output of the linkage editor is logically extended to accommodate the load modules produced by the linkage editor.

LINKOP

RLIB=filename
Identifies the file to be searched by the linkage editor, under the following conditions.

- During the automatic inclusion process, when no automatic include library file (ALIB) has been specified (no previous ALIB keyword specification present) or when the specified ALIB does not contain the required module.
- During the specific inclusion process, when no file name is specified on an INCLUDE control statement and:
  - the module is not found in YRUN; or
  - the module is not found in the file last specified on a prior INCLUDE control statement or no prior INCLUDE control statements exist.

RLIB=**YOBJ**
File to search under the conditions just described.

If omitted, YOBJ is searched under the foregoing conditions, unless a previous RLIB keyword specified otherwise.

CNL
Cancel link-edit job step at the end of the link-edit operation for the current load module if any diagnostic processing has been triggered during the generation of the load module; otherwise, processing continues, regardless of the number or type of error diagnostic triggered, until the normal end-of-job function for the link-edit job step is detected.

NOCNL
Process link-edit job step to completion, regardless of the number or type of diagnostic errors detected.

If omitted, NOCNL is assumed unless the CNL keyword was previously specified.

ZRO
Clear main storage to all 0's prior to loading the root phase of the load module.

NOZRO
Do not clear main storage to 0's before the root phase of the load module is loaded.

If omitted, NOZRO is assumed unless the ZRO keyword was previously specified.

AUTO
Automatic inclusion processing is performed for the load modules being built.

NOAUTO
Automatic inclusion processing is not performed for the load modules being built.

If omitted, AUTO is assumed unless the NOAUTO keyword was previously specified.

LINKOP

NOTE:

The NOAUTO specification does not affect the automatic inclusion of the overlay control routine if V-CON processing is not inhibited (NOV keyword not specified) and valid V-CON references exist in the object code being included in the load modules being produced.

V

Specifies that V-CON references are permitted to appear in the object module elements to be included in the load modules and that automatic loading of required phases is started.

NOV

Specifies that V-CON references are treated as A-CON references and that automatic loading of phases is inhibited because it is not required. This specification is significant only when valid V-CON references are present in the object module elements being included.

If omitted, V is assumed unless the NOV keyword was previously specified.

PROM

Place common storage areas in their most efficient phases within the load module being built.

NOPROM

Place all common storage areas in the root phase of the load module being built.

If omitted, PROM is assumed unless the NOPROM keyword was previously specified.

LIST

Produce standard link-edit map for current link-edit job.

NOLIST

Do not produce standard link-edit map for current link-edit job.

If omitted, LIST is assumed unless NOLIST was previously specified.

CNTCD

Produce process map portion of the link-edit map.

NOCNTCD

Suppress process map.

If omitted, CNTCD is assumed unless NOCNTCD was previously specified.

ERR

Link-edit map includes diagnostic messages and unresolved references.

NOERR

Suppress this information.

If omitted, ERR is assumed unless NOERR was previously specified.

LINKOP

DICT
/ Produce definitions dictionary and phase structure diagram portions of the link-edit map.

NODICT
Suppress definitions dictionary and phase structure diagram portions of the link-edit map.

If omitted, DICT is assumed unless NODICT was previously specified.

DEF
List CSECT, COM, and ENTRY items in the allocation map portion of the link-edit map. These ESDs are associated with their respective object module origin and ESID, length, linked base address, and high limit after linking.

NODEF
Suppress ESDs.

If omitted, DEF is assumed unless NODEF was previously specified.

PHS
List phase data (phase name, alias name, origin, length, transfer address, etc) in allocation map portion of the link-edit map.

NOPHS
Suppress phase data.

If omitted, PHS is assumed unless NOPHS was previously specified.

DEL
List all definitions, including those automatically deleted due to redundant inclusions on identical paths or items not included because of partial include specifications, in the allocation map so long as definitions are being listed (NODEF is not specified).

NODEL
Automatically deleted definitions are not listed.

If omitted, NODEL is assumed unless DEL was previously specified.

RCNTCD
List control statements that are to be included in the allocation map portion of the link-edit map so that each may be located easily to see its effect on the load module. Only action-type control statements are included in the allocation map; // PARAM and LINKOP statements are never listed.

NORCNTCD
No listing of control statements in the allocation map.

If omitted, NORCNTCD is assumed unless RCNTCD was previously specified.

LINKOP

REF

List in the allocation map portion of the link-edit map all references (EXTRNs) and transfer records processed, the object module assigned address and ESID, and the resolved value, if appropriate.

NOREF

Suppress this information.

If omitted, NOREF is assumed unless REF was previously specified.

SHARE

Specifies that object modules marked reentrant are to be recognized during the inclusion process (specific and automatic). The text from such modules is not to be included. Instead, shared records are to be created for references made to reentrant code. The load module produced is nonreentrant but may contain references to reentrant code. This parameter is ignored if a reentrant load module is being generated (the RNT parameter is specified).

NOSHARE

Specifies that all object modules are to be treated as nonreentrant and no shared records are to be generated. A nonreentrant load module is produced that contains no references to shared code.

If omitted, SHARE is assumed unless NOSHARE was previously specified.

RNT

Specifies that a reentrant load module is to be produced and SENTRY records are to be generated. Normally, only one object module would be included in the link-edit, in which case, the load module name must be the same as object module name. Parameter NOAUTO should be specified when using this option.

NORNT

Specifies that a nonreentrant load module is to be produced though references may be made to reentrant code.

If omitted, NORNT is assumed unless RNT was previously specified.

ISD

Specifies that type 3 and type 4 ISD records from the included object module are relocated and passed to the load module. It also specifies that type 1 and type 2 ISD records are generated based on the undeleted CSECTs and COMMONs detected during the link-edit. If execution of the load module terminates, these records are used by JOBDUMP (if specified) to print a formatted dump, segmenting it by CSECTs and printing user program symbols.

NOISD

Specifies that ISD record generation is to be suppressed.

If omitted, ISD is assumed unless NOISD was previously specified.

LOADM

Function:

Specifies the program name for the module being built. The LOADM control statement may follow only a // PARAM, LINKOP, or comment statement or a complete set of control statements when used in a control stream that is generating more than one load module.

If this statement is omitted from an otherwise valid linkage editor control stream and no LOADM control statement is embedded in an included object module, the load module produced by the linkage editor is assigned the name LNKLOD by default.

Format:

LABEL	△OPERATION△	OPERAND
	LOADM	[{name / LNKLOD}]

Parameter:

name

One to six (eight, if a reentrant load module is being created) alphanumeric character specifying the name to be assigned to the load module. The first character must be alphabetic. If the specified name is less than six or eight characters, as applicable, it is padded on the right with EBCDIC 0's. If the specified name is more than six characters (eight, if it is the link-edit of a reentrant module), it is truncated to the maximum allowable limit.

If omitted, the default name LNKLOD is assumed to the load module.

MOD

Function:

Tells the linkage editor to modify its location counter to agree with a specific power of 2 and a specific remainder.

Format:

LABEL	△OPERATION△	OPERAND
	MOD	power [,{remainder ■}]

Parameters:

power
A decimal number specifying the power of 2 relative to which the location counter is to be adjusted. The only acceptable powers of 2 that may be specified are 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, and 32768.

remainder
A decimal number that is a multiple of 4 and specifies the desired remainder of the new value of the location counter relative to the specified power of 2. If the decimal number specified is not a multiple of 4, it is rounded to the next higher multiple of 4, then truncated to a value less than the specified power of 2.

If omitted, the remainder is assumed to be 0.

OVERLAY

Function:

Identifies the beginning of an overlay phase (a phase other than the root phase) and defines the relative position of the phase within the load module structure. The object modules included thereafter establish a single, separate phase until the next OVERLAY, REGION, LOADM, LINKOP, or end-of-data (/*) control statement is detected.

This statement cannot be used in the link edit of a reentrant module.

Format:

LABEL	△OPERATION△	OPERAND
	OVERLAY	symbol[,alias-phasename]

Parameters:

symbol
: Name of a logical or relative node point that defines the starting address (but not necessarily the entry point) of the phase. It may consist of one to eight alphanumeric characters. If the symbol is relative (a CSECT, ENTRY, or EQU name), it must be on the path of the phase. Further, the relative symbol must not be a shared definition.

alias-phasename
: A 1- to 6-character, user-supplied phase name that can be used in place of linkage-editor-supplied phase name to address the phase being created by the OVERLAY control statement. If an alias phase name longer than six characters is supplied, it is truncated to six characters.

PARAM

Function:

Specifies the initial linkage editor options that are to be in effect during the building of a load module. Performs the same function as the LINKOP control statement but can only be specified in the job control stream proper, not within the linkage editor data set. It can only be used to define the initial options and may not be used to change or add to those options within a linkage editor job step.

Format:

```
// PARAM      Same keyword parameters as those specified for the
              LINKOP control statement
```

REGION

Function:

Begins building the first phase in a new region, starting at the end of the longest path currently built for the load module. Once a region has been started, no prior region structure may be continued. All parts of a given region should be fully specified and built before beginning a new region. The only phase common to all regions in a load module is the root phase.

OVERLAY control statements may be interspersed among REGION control statements to structure the phases within each region. The first phase of any region, including the initial phase, may be overlaid by referencing the region node (or LOADM node) or a symbol with that address, using an OVERLAY control statement. Inasmuch as the REGION control statement effectively replaces an OVERLAY, INCLUDE and other control statements used for the building of a phase follow immediately. Up to 10 regions may be declared for a single load module.

Format:

LABEL	△OPERATION△	OPERAND
	REGION	symbol[,alias-phasename]

Parameters:

Refer to parameter description for the OVERLAY control statement. Note that inasmuch as the implied origin of a new region always is at the end of the longest path of the current region, a symbol in a REGION control statement may specify only a logical node name and not a relative definition name, as in the case of an OVERLAY control statement.

RES

Function:

Tells the linkage editor to reserve additional space in main storage following the end of the longest path in the highest region of the load module. The additional storage requested by this statement adds to the total length of the module and is recorded in each phase header record, but is not included in the size requirements for any particular phase. This statement, therefore, may be placed anywhere within the control stream for a load module.

Processing this control statement, the linkage editor automatically assigns two entry points to the reserve storage area. These entry points may be addressed by the user to access the reserve storage area by declaring them as EXTRAS in your program. The two entry points assigned are:

1. KE$ALP – effective address of the end of the longest path in the load module, or the starting address of the reserve storage area.

2. KE$RES – effective address of the end of the longest path plus the sum of all the reserve storage area size specifications (RES control statements) or the address of the last four bytes (one word) of the reserve storage area.

These two entry points cannot be used in the operand field of the EQU control statement.

Format:

LABEL	△OPERATION△	OPERAND
	RES	value

Parameter:

value
Specifies, in one of the following forms, the number of bytes of storage to be reserved.

- A decimal number from 1 to 10 digits long
- A hexadecimal number from 1 to 8 digits long, in the form X'nnnnnnnn'

5. Initialize Disk (Disk Prep) Routine (DSKPRP)

INSERT

Function:

Identifies the address of the track suspected to be defective for which an alternate track is assigned. Only one track address may be specified on an INSERT control statement; however, multiple INSERT control statements are permitted. These control statements must follow the VOL1 card in the control stream and must be used whenever the keyword INSRT=X or INSRT=Y has been specified. If the keyword TRCON=N was specified, an INSERT control statement with the word NONE starting in column 10 must be specified for IDA disk systems whenever no defective tracks are known.

Format:

1	10	72
INSERT	{ccccchh {NONE	

Parameters:

ccccchh
Six hexadecimal numbers indicating the cylinder (cccc) and head (hh) address of a possible defective track.

NONE
If the keyword TRCON=N was specified for an IDA disk, this parameter must be specified. No defective tracks are known to exist on the disk pack.

VOL1

Function:

Creates the standard volume labels (VOL1) on the disk pack or initializes the data set labels on a diskette.

Format 1:

```
1    11  42        51                                          72
VOL1 [r] [aaaaaaaaaa]
```

Parameters:

r
This entry is reserved for future use.

aaaaaaaaaa
Alphanumeric name and address of the owner. If RPVOL=Y is specified, specifies new owner name and address.

NOTE:

VOL1 is the only entry required for diskettes.

KEYWORD PARAMETERS FOR DISK

Function:

There are 17 keyword parameters that supply the needed information for disk prepping. When prepping a disk, the SERNR keyword parameter always must be specified.

Format:

```
[ALTRK={N|Y}][,ILOPT={C|N|1|2|3}][,INSRT={N|X|Y}]
[,IPLDK={N|Y}][,PARTL={N|S|V}]
[,PREPT={1|2|3|4|C|F}][,PTBEG={ccchh|000000}] [,PTEND=ccchh]
[,RETRY={nn|0A}][,RPVOL={N|Y}],SERNR=volume serial number (six characters)
[,TRCON={C|D|N|S}][,TRKCT={C|D|L|X|Z}][,UNXFC={N|Y}][,VERFY={N|Y}]
[,VTOCB={ccchh
         000001 for all non-IPL volumes
         006400 for 8411/8414 IPL volumes
         00CA00 for 8415/8416/8418/8424/8425/8430/8433 IPL volumes}]
[,VTOCE={ccchh
         000009 for 8411 non-IPL volume
         000013 for 8414/8424/8425 non-IPL volume
         000002 for 8415 fixed, non-IPL volume
         000001 for 8415 removable, non-IPL volume
         000006 for 8416/8418 non-IPL volumes
         000012 for 8430 non-IPL volume
         006409 for 8411 IPL volume
         006413 for 8414 IPL volume
         00CA02 for 8415 fixed, IPL volume
         00CA01 for 8415 removable, IPL volume
         00CA06 for 8416/8418 IPL volumes
         00CA13 for 8424/8425 IPL volumes
         00CA12 for 8430/8433 IPL volumes}]
```

ALTRK
ILOPT
INSRT

ALTRK Keyword Parameter:

Specifies whether the alternate track area of the disk pack is to be tested.

ALTRK=N
Track area is not tested.

ALTRK=Y
Track area is tested.

ILOPT Keyword Parameter:

An alphanumeric character (C, N, 1, or 2) denoting the input type of initial load control storage (ILCS) written on the disk pack (IPL volume).

ILOPT=C
ILCS is cards. These cards must be within the data delimiters (/$ and /*) and follow immediately after the first /*.

ILOPT=N
No ILCS is written.

ILOPT=1
ILCS is disk input and 1K capacity. The ILCS module in the system source library file (YSRC) must be named IL1K0000.

ILOPT=2
ILCS is disk input and 2K capacity. The ILCS module in YSRC must be named IL2K0000.

ILOPT=3
ILCS is disk input and 2K fast capacity. The ILCS module in YSRC must be named IL2FK000.

INSRT Keyword Parameter:

Indicates if specific tracks (as identified by INSERT control statements) are to be automatically recorded as defective.

INSRT=N
No INSERT control statements appear in the control stream.

INSRT=X
Only INSERT control statements are used to designate defective tracks.

INSRT=Y
INSERT control statements are used together with the normal surface analysis function to flag defective tracks for the disk pack.

IPLDK
PARTL
PREPT

IPLDK Keyword Parameter:

Indicates if disk pack being prepped is to be used for an IPL volume.

IPLDK=N
Disk pack is not an IPL volume.

IPLDK=Y
Disk pack is an IPL volume.

PARTL Keyword Parameter:

Indicates only a partial prep is being done or indicates changing your volume serial number while also removing all entries in the VTOC of your disk pack.

PARTL=N
Indicates no partial prep or change in the volume serial number.

PARTL=S
Indicates a partial prep used with the PTBEG and PTEND keyword parameters. The prep is done under this option without regard to VTOC, labels, or files.

PARTL=V
Indicates a change in the volume serial number and VTOC (with or without COS).

PREPT Keyword Parameter:

Specifies the type of disk prep to be performed.

PREPT=1
Least accurate prep.

PREPT=2
Less accurate than 3 or C.

PREPT=3
Extreme accuracy prep. Same as C.

PREPT=C
Extreme accuracy prep. Automatically selected in place of options 1, 2, or 3 when an IDA disk is being prepped.

PREPT=F
Fast prep. No surface analysis is performed.

PTBEG
PTEND
RETRY
RPVOL
SERNR

PTBEG Keyword Parameter:

PTBEG=ccchh
Six hexadecimal numbers representing the primary track address at which disk prepping is to start.

If omitted, hexadecimal 0's are assumed.

PTEND Keyword Parameter:

PTEND=ccchh
Six hexadecimal numbers representing the primary track address at which disk prepping is to end. It must be equal to or greater than the starting address specified.

If omitted, all primary tracks are prepped.

RETRY Keyword Parameter:

Indicates the number of times an I/O command is to be reissued during surface analysis testing when errors are encountered. May also be used with PREPT=F.

RETRY=nn
Two hexadecimal numbers specifying the number of retries to be made during surface analysis testing.

If omitted, hexadecimal 0A is assumed.

RPVOL Keyword Parameter:

Indicates if the disk pack volume serial number is to be changed in the VOL1 label. The file information and file serial number of the files currently in the VTOC are not changed.

RPVOL=N
Volume serial number is not to be changed.

RPVOL=Y
Volume serial number or user address is to be changed.

SERNR Keyword Parameter:

SERNR=volume-serial-number
Six alphanumeric characters representing the disk pack number. This number may already have been assigned to the disk volume or may specify a new assignment. No embedded blanks are permitted.

TRCON
TRKCT

TRCON Keyword Parameter:

Indicates the method of generating or transferring the track condition table to the root segment of the routine.

TRCON=C
Punched card method. Used when the track condition table on the disk pack has been destroyed.

TRCON=D
Disk method. Used when the disk pack has been in use and the current information is to be retained. Existing track condition table must be updated and used to create new track condition table.

TRCON=N

or

TRCON=S
Surface analysis method. Used when prepping the disk pack for the first time or if the track condition table cannot be recovered by using either TRCON=C or TRCON=D. Either N or S can be coded with the same results.

TRKCT Keyword Parameter:

Specifies if the track condition table is to be printed, punched in cards, or written on an alternate track.

TRCT=C
Output to cards.

TRKCT=D
Output to alternate track.

TRKCT=L
Output to printer and no other prepping functions take place. Disk pack must have been prepped using TRCON=D.

TRKCT=P
Output to printer. This option is illegal when using IDA disks.

TRKCT=X
Output to cards printer.

TRKCT=Z
Output to alternate track and printer.

UNXFC
VERFY
VTOCB

UNXFC Keyword Parameter:

Checks the expiration date for all files on the volume. DSKPRP compares the system date to the expiration date. Whenever the expiration date is not expired, a message is displayed asking the user to either cancel the prep or to ignore the date and continue the prep.

UNXFC=N
Expiration date is not checked.

UNXFC=Y
Expiration date is checked.

VERFY Keyword Parameter:

Assures that the area to be prepped as specified by keywords PTBEG and PTEND does not contain area already in use. This option can only be used on a pack previously prepped and used by OS/3. The following information must be supplied when using this option: PTBEG, PTEND, SERNR, and TRCON cannot be N, and TRKCT cannot be P. The following is automatic: ALTRK=N. The following are ignored: VTOCB, VTOCE, ILOPT, and IPLDK. The keywords INSRT and PREPT may be used as desired. The VOL1 card must also be included in the deck.

VERFY=N
Area is not tested.

VERFY=Y
Area is to be tested.

VTOCB Keyword Parameter:

Six hexadecimal numbers representing the primary track address at which the VTOC starts. It must be less than the ending VTOC address.

VTOCB=(ccchh
- **000001** for all non-IPL volumes
- **006400** for 8411/8414 IPL volumes
- **00CA00** for 8415/8416/8418/8424/8425/8430/8433 IPL volumes

VTOCE

VTOCE Keyword Parameter:

Six hexadecimal numbers representing the primary track address at which the VTOC ends.

VTOCE=(ccchh

- 000009 for 8411 non-IPL volume
- 000013 for 8414 non-IPL volume
- 000002 for 8415 fixed, non-IPL volume
- 000001 for 8415 removable, non-IPL volume
- 000006 for 8416/8418 non-IPL volumes
- 000012 for 8430 non-IPL volume
- 006409 for 8411 IPL volume
- 006413 for 8414 IPL volume
- 00CA02 for 8415 fixed, IPL volume
- 00CA01 for 8415 removable, IPL volume
- 00CA06 for 8416/8418 IPL volumes
- 00CA13 for 8424/8425 IPL volumes
- 00CA12 for 8430/8433 IPL volumes

6. Initialize Diskette Routine (DSKPRP)

KEYWORD PARAMETERS FOR DISKETTE

Function:

There are five keyword parameters that supply the needed information for diskette prepping. When prepping a diskette, the SERNR keyword parameter must always be specified as well as the VOL1 statement. Diskette prep is a function of the DSKPRP routine.

Format:

```
[,FDATA={N}][,PARTL={N}]
        {Y}         {V}
[,RPVOL={N}][,SERNR=volume-serial-number]
        {Y}
[,UNXFC={N}]
        {Y}
```

FDATA
PARTL
RPVOL

FDATA Keyword Parameter:

Specifies whether to automatically allocate the entire DSL diskette as one file (named DATA) or have the files allocated via // EXT job control statement or ALLOCATE command.

FDATA=N
Specifies that the diskette is to be allocated by the user.

FDATA=
Specifies that the entire diskette is automatically allocated as one file.

PARTL Keyword Parameter:

Indicates changing your volume serial number while also removing all entries in the VTOC of your data set labels on your diskette.

PARTL=
Indicates no change in the volume serial number.

PARTL=V
Indicates a change in the volume serial number and VTOC (with or without COS).

RPVOL Keyword Parameter:

Indicates if the diskette volume serial number is to be changed in the VOL1 label. The file information and file serial number of the files currently in the VTOC are not changed.

RPVOL=
Volume serial number is not to be changed.

RPVOL=Y
Volume serial number or user address is to be changed.

SERNR
UNXFC

SERNR Keyword Parameter:

SERNR=volume-serial-number
Six alphanumeric characters representing the diskette number. This number may already have been assigned to the diskette volume or may specify a new assignment. No embedded blanks are permitted.

UNXFC Keyword Parameter:

Specifies whether file expiration date checking is to be performed.

UNXFC=N
Specifies that file expiration date checking is not to be performed.

UNXFC=Y
Specifies that file expiration date checking is to be performed.

DISK PREP JOB CONTROL REQUIREMENTS

The following depicts the job control requirements for DSKPRP showing the location of optional cards in the job stream.

```
// JOB jobname
// DVC number // LFD PRNTR
// DVC number // VOL vsn // LFD DISKIN
// EXEC DSKPRP
/$
    (keyword option cards inserted here)
VOL1
    (track condition cards inserted here, if needed)
    (INSERT cards inserted here, if needed)
/*
/$
    (IL(COS) cards inserted here, if needed)
/*
/&
// FIN
```

If the output is to go to cards, the following statement must be added:

```
// DVC number // LFD PUNCH
```

7. Assign Alternate Track Routine (DSKPRP)

KEYWORD PARAMETERS

Function:

These are five keyword parameters that supply the information needed for assigning an alternate track. The ASGTK and SERNR keywords must be specified in each control stream.

Format:

```
ASGTK=cccchh[,ASGPR={A}][,ASUPD={N}][,ASURF={N}],SERNR=volume-serial-number
                    {E}        {Y}        {S}
```

ASGTK
ASGPR
ASUPD
ASURF
SERNR

ASGTK Keyword Parameter:

Indicates the assign alternate track routine is to be loaded and the suspected defective track is to be checked. Note that a VOL1 card must not be present when specifying this keyword.

ASGTK=ccchh
Six hexadecimal numbers representing the suspected defective track. Must be specified for assigning an alternate track.

ASGPR Keyword Parameter:

Indicates whether all the records being read or just those records detected in error are to be listed on the printer.

ASGPR=A
All records read are listed.

ASGPR=E
Only records detected in error are listed.

ASUPD Keyword Parameter:

Indicates whether update records are present in the control stream.

ASUPD=N
No update records are present in the control stream.

ASUPD=Y
Update records are present in the control stream.

ASURF Keyword Parameter:

Indicates whether a surface analysis is to be performed on the track identified by ASGTK keyword.

ASURF=N
No surface analysis is performed. Alternate track is assigned automatically.

ASURF=S
Surface analysis is performed. Alternate track is assigned only if track identified by ASGTK keyword is found to be defective.

SERNR Keyword Parameter:

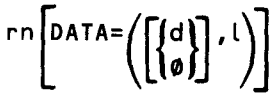
SERNR=volume-serial-number
Six alphanumeric characters representing the disk pack volume serial number.

UPDATE RECORDS

Function:

Indicates the record on the alternate track to be patched that was detected in error by a previous assign alternate track run. The keyword parameter ASUPD=Y must be specified.

Format:



Parameters:

rn

Indicates the record number (2-byte hexadecimal) in error.

DATA=([d],l)

Indicates the displacement value (0–3 positions, relative to 0) and the length value (1–4 positions) of the data field.

NOTES:

1. *The actual data record to be written on the track must be in hexadecimal format containing no embedded blanks.*

2. *The rn parameter may be the first card in the data set and the only entry on the card.*

8. Tape Prep Routine (TPREP)

DVC

Function:

Indicates the appropriate logical unit number of the tape drive being used for the tape prep.

Format:

```
// DVC nn
```

Parameter:

nn
Logical unit number from 90–127.

EXEC

Function:

Calls the tape prep utility from YLOD.

Format:

```
// EXEC TPREP
```

Parameter:

TPREP
Program name of the tape prep utility.

LBL

Function:

Assigns a file identifier to the tape being prepped. This statement is optional.

Format:

```
// LBL file-identifier
```

Parameter:

`file-identifier`
One to 17 characters; the first character is alphabetic.

LFD

Function:

Indicates a unique file name required by TPREP.

Format:

```
// LFD TAPExy
```

Parameter:

```
TAPExy
```

where:

x
Is any alphanumeric character A through Z, or 0 through 9.

y
Is either the character A indicating ASCII mode or blank indicating EBCDIC mode.

VOL

Function:

Indicates the volume serial number of the tape being prepped.

Format:

```
// VOL[N,]volume-serial-number(PREP)
```

Parameters:

N
Indicates that the tapes are to be prepped for use without block numbers.

If omitted, the tapes will be prepped according to SYSGEN-supplied parameters; that is, if tape block numbers are being supported in the system, the tape prep routine will do likewise.

volume-serial-number
Six alphanumeric characters, other than SCRTCH, representing the tape serial number.

(PREP)
This character string must start immediately following the last character in the volume-serial-number.

9. Sectored Disk Copy Routine (SU$C16)

SU$C16 – BATCH ENVIRONMENT (PARAM)

Function:

There are seven keyword parameters associated with copying a sectored disk pack. All of the keyword parameters have default values; however, the // PARAM statement must be supplied even if all the defaults are assumed.

Format:

```
// PARAM [COPY={n}] [,BGAD={ccc16    }]
               {1}          {ccch16   }
                            {ccchrr16 }
                            {000001   }

        [,EDAD={ccc16                          }] [,OVEF={NO }]
               {ccch16                         }         {YES}
               {ccchrr16                       }
               {193628 for 8416 and 8418 low   }
               {327628 for 8418 high           }
               {193128 for 8415R               }
               {327228 for 8415F               }

        [,PRNT={NO }] [,UNXF={NO }] [,VEFY={NO }]
               {YES}         {YES}         {YES}
```

COPY Keyword Parameter:

Indicates the number of copies of the input disk pack to be made, where n may be from 1 to 7.

If omitted, one copy is made.

BGAD Keyword Parameter:

Indicates the starting address, in hexadecimal, of the disk copy. The starting address can be specified in cylinder (ccc), cylinder and head (ccch), or cylinder, head, and record (ccchrr) formats.

If omitted, cylinder 000, head 0, and record 01 is assumed.

EDAD Keyword Parameter:

Indicates the ending address, in hexadecimal, of the disk copy. The ending address can be specified in cylinder (ccc), cylinder and head (ccch), or cylinder, head, and record (ccchrr) formats.

If omitted, the following addresses are assumed:

- Cylinder 193, head 6, record 28 for 8416 and 8418 low
- Cylinder 327, head 6, record 28 for 8418 high
- Cylinder 193, head 1, record 28 for 8415 removable
- Cylinder 327, head 2, record 28 for 8415 fixed

OVEF Keyword Parameter:

OVEF=NO
One or more copies are made, with or without verification.

OVEF=YES
A verify-only operation is performed.

PRNT Keyword Parameter:

PRNT=NO
Only fatal errors are displayed on the system console.

PRNT=YES
All errors detected are printed, listing both the input and output records. Normally used in conjunction with VEFY=YES.

UNXF Keyword Parameter:

UNXF=NO
No file expiration date checking is performed.

UNXF=YES
File expiration date checking is performed.

VEFY Keyword Parameter:

VEFY=NO
No verification is performed on the output disk pack.

VEFY=YES
Verification is performed on all disk packs being copied.

SU$C16 – INTERACTIVE ENVIRONMENT (PARAM)

Function:

Create and verify up to seven copies of a single sectored disk (regardless of the disk's contents) interactively from either a UNISCOPE 100, UNISCOPE 200, or UTS 400 terminal.

Operation:

Enter the LOGON command. After successfully logging on to the system, enter HU in system mode to initiate the hardware utility. The following MENU screen appears:

```
                    HARDWARE UTILITIES          HUØØA

1. DUMP FILES FROM A DISK
2. RESTORE FILES TO A DISK
3. COPY FILES FROM DISK TO DISK
4. COPY AND/OR VERIFY IDA DISK
5. COPY AND/OR VERIFY SELECTOR DISK
6. NONE OF THESE

                    ENTER SELECTION 4
```

Select 4 from the MENU screen and press the transmit key. (When using the UNISCOPE 100 and 200 terminals, use the TRANSMIT key. When using the UTS 400 terminal, use the XMIT key.) After pressing the XMIT key, the following information screen is displayed.

```
                                                          HUØØAIØ4
A CONVERSATIONAL JOB (HU$C16) TO COPY THE CONTENTS OF ONE IDA
DISK TO ONE OR MORE DISKS OR TO VERIFY THE CONTENTS OF ONE OR
MORE IDA DISKS HAS BEEN INITIATED IN YOUR BEHALF. YOU MUST BE
IN SYSTEM MODE FOR THE JOB TO BE SCHEDULED. IF YOU ENTERED
HARDWARE UTILITIES THROUGH THE HU COMMAND YOU WILL BE IN SYSTEM
MODE AFTER TRANSMITTING. IF YOU ENTERED THROUGH THE MENU
COMMAND YOU ARE RESPONSIBLE FOR GOING INTO SYSTEM MODE.
                ***** TRANSMIT TO CONTINUE *****
```

Press the XMIT key to continue. An appropriate set of screens will be displayed asking for the required device and file information. After the information is given, the COPY operation is performed.

10. Nonsectored Disk Copy Routine (SU$CSL)

SU$CSL – BATCH ENVIRONMENT (PARAM)

Function:

There are seven keyword parameters associated with copying a nonsectored disk pack. All the keyword parameters have default values; however, the // PARAM statement must be specified even if all the defaults are assumed.

Format:

```
// PARAM [COPY={n}] [,BGAD={ccchh,16}]
               {1}          {00000   }
         [,EDAD={ccchh,16              }] [,OVEF={NO }] [,PRNT={NO }] [,UNXF={NO }]
         [      {0C709 for 8411        }]        {YES}          {YES}          {YES}
         [      {0C713 for 8414        }]
         [      {18F13 for 8424/25     }]
         [      {19312 for 8430        }]
         [      {32712 for 8433        }]
         [,VEFY={NO }]
                {YES}
```

COPY Keyword Parameter:

Indicates the number of copies of the input disk pack to be made, where n may be from 1 to 7.

If omitted, one copy is made.

BGAD Keyword Parameter:

Indicates the starting address, in hexadecimal, at which the copy operation is to end. The starting address must be specified in cylinder and head (ccchh) format.

If omitted, cylinder 000, head 00 is assumed.

EDAD Keyword Parameter:

Indicates the ending address, in hexadecimal, in which the copy operation is to end. The ending address must be specified in cylinder and head (ccchh) format.

If omitted, the following addresses are assumed:

- Cylinder C7, head 09 for 8411
- Cylinder C7, head 13 for 8414
- Cylinder 18F, head 13 for 8424/25
- Cylinder 193, head 12 for 8430
- Cylinder 327, head 12 for 8433

OVEF Keyword Parameter:

OVEF=NO
One or more copies are made, with or without verification.

OVEF=YES
A verify-only operation is performed.

PRNT Keyword Parameter:

PRNT=NO
Only fatal errors are displayed on the system console.

PRNT=YES
All errors detected are printed, listing both the input and output records. Normally used in conjunction with VEFY=YES.

UNXF Keyword Parameter:

UNXF=NO
No file expiration date checking is performed.

UNXF=YES
File expiration date checking is performed.

VEFY Keyword Parameter:

VEFY=NO
No verification is performed on the output disk pack.

VEFY=YES
Verification is performed on all disk packs being copied.

SU$CSL – INTERACTIVE ENVIRONMENT (PARAM)

Function:

Create and verify up to seven copies of a single nonsectored disk (regardless of the disk's contents) interactively from either a UNISCOPE 100, UNISCOPE 200, or UTS 400 terminal.

Operation:

Enter the LOGON command. After successfully logging on to the system, enter HU in system mode to initiate the hardware utility. The following MENU screen appears:

```
                    HARDWARE UTILITIES          HUØØA
1. DUMP FILES FROM A DISK
2. RESTORE FILES TO A DISK
3. COPY FILES FROM DISK TO DISK
4. COPY AND/OR VERIFY IDA DISK
5. COPY AND/OR VERIFY SELECTOR DISK
6. NONE OF THESE

                    ENTER SELECTION 5
```

Select 5 from the MENU screen and press the transmit key. (When using the UNISCOPE 100 and 200 terminals, use the TRANSMIT key. When using the UTS 400 terminal, use the XMIT key.) After pressing the XMIT key, the following informational screen is displayed.

```
                                                          HUØØAIØ5
A CONVERSATIONAL JOB (HU$CSL) TO COPY THE CONTENTS OF ONE
SELECTOR DISK TO ONE OR MORE DISKS OR TO VERIFY THE CONTENTS OF
ONE OR MORE SELECTOR DISKS HAS BEEN INITIATED IN YOUR BEHALF.
YOU MUST BE IN SYSTEM MODE FOR THE JOB TO BE SCHEDULED. IF YOU
ENTERED HARDWARE UTILITIES THROUGH THE HU COMMAND YOU WILL BE IN
SYSTEM MODE AFTER TRANSMITTING. IF YOU ENTERED THROUGH THE MENU
COMMAND YOU ARE RESPONSIBLE FOR GOING INTO SYSTEM MODE.
                ***** TRANSMIT TO CONTINUE *****
```

Press the XMIT key to continue. An appropriate set of screens will be displayed asking for the required device and file information. After the information is given, the copy operation is performed.

11. Disk Dump/Restore Routine (DMPRST)

DUMP – BATCH ENVIRONMENT

Function:

Writes from a disk pack to a magnetic tape. Multiple disks and tapes are permitted. Either data or library files are accepted. Volumes or files may be specified; however, file names containing embedded blanks must be delimited with quotation marks. File prefixes are also accepted. Dump routine checks for the correct job control statements and any fatal error is displayed on system console or printer (if specified). The // PARAM statements indicate the operation. Adding the MUX parameter to the // PARAM OUT statement allows an output tape, which is being produced on a selector channel tape unit, to be blocked in a format that can be read by either a multiplexer or selector channel tape unit. The // PARAM END statement can only be used in the volume environment and the ccc must be specified in decimal.

If dumping to multivolume tapes, the output tape file names may be specified on the same // VOL card or they may be entered on separate // VOL cards. If the former method is chosen, only one // LFD card is required with the name TAPEOT specified. In this case, only one tape drive is required and the output tapes are mounted as needed. If separate // VOL cards are used, a // LFD card must be coded for each output tape with the numbers 01 to 99 appended to the additional output tape LFD name (TAPEOT01). All tapes must be mounted if this method is chosen.

Format 1. Volume Environment:

Disk to Tape:

```
1         10    16
// JOB DUMP
// DVC 50
// VOL OS3RES
// LBL RESPAK
// LFD DISCIN
// DVC 90
// VOL BACKUP
// LBL BRESPAK
// LFD TAPEOT
// DVC 20
// LFD PRNTR
// EXEC DMPRST
// PARAM IN=DISC
// PARAM OUT=TAPE
// PARAM END=ccc or LAST
/&
// FIN
```

Format 2. File Environment:

Disk to Tape:

```
1         10    16
// JOB DUMP
// DVC 50
// VOL OS3RES
// LBL RESPAK
// LFD DISCIN
// DVC 90
// VOL BACKUP
// LBL BRESPAK
// LFD TAPEOT
// DVC 20
// LFD PRNTR
// EXEC DMPRST
// PARAM IN=DISC
// PARAM OUT=TAPE,MUX
// PARAM TYPE=FILE,NOWAIT
/$
   FILE PARTS
   FILE.P BANK
/*
/&
// FIN
```

Disk to Tape:

```
// JOB DUMP
// DVC 20
// LFD PRNTR
// DVC 50
// VOL OS3RES
// LFD DISCIN
// DVC 90
// VOL BACK01,BACK02
// LFD TAPEOT
// EXEC DMPRST
// PARAM IN=DISC
// PARAM OUT=TAPE
// PARAM TYPE=FILE,ALL
/$
// FIN
```

RESTORE – BATCH ENVIRONMENT

Function:

Writes from a magnetic tape to a disk pack. Multiple tapes and disks are permitted. Either data or library files are accepted. Volumes or files may be specified; however, file names containing embedded blanks must be delimited with quotation marks. Files must be restored in the order they appear on the tape. Restore routine checks for correct job control statement and any fatal error is displayed on system console or printer (if specified). The // PARAM statements indicate the operation.

Several options are available for file placement if performing a disk copy or restore operation in a file environment. These options are implemented through the specification of the following parameters on the FILE card. The formats of the FILE statement are:

Format 1:

```
FILE old-name[,{PRE}][,new-name]
              [{ABS}]
              [{LOG}]
              [{REL}]
```

Format 2:

```
FILE.P prefix-name[,{PRE}]
                   [{ABS}]
                   [{LOG}]
                   [{REL}]
```

The old-name parameter names the file being processed. The prefix-name specifies the prefix name of all the files to be restored. The new-name parameter changes the name of the file being restored to disk. The second parameter is the allocation parameter. This parameter controls the processing of restored files.

The PRE parameter indicates that the file space for the file has been previously allocated. The ABS parameter indicates that the file is to be allocated on the output disk in the same absolute extents as it occupied on the input disk. The LOG parameter indicates that the file will be allocated with all unused space deleted. The REL parameter is used to specify that the file is to be relocated. If enough file space to hold the file is not available, the unused space is deleted and the file is relocated. Unused space is that space allocated for a file but not yet assigned to a partition of the file.

If restoring from multivolume tapes, the input tape names may be specified on separate // VOL cards or on the same // VOL card. If entered on separate // VOL cards, a separate // LFD must be entered for each tape volume with the first volume named TAPEIN and the numbers 01 through 99 appended to each additional volume (TAPEIN01).

Format 1. Volume Environment:

```
1         10     16
// JOB RESTORE
// DVC 20
// LFD PRNTR
// DVC 90  // VOL PAY002  // LFD TAPEIN
// DVC 50  // VOL DISK01  // LFD DISCOT
// EXEC DMPRST
// PARAM IN=TAPE
// PARAM OUT=DISC
/&
// FIN
```

Format 2. File Environment:

```
// JOB RESTORE
// DVC 20
// LFD PRNTR
// DVC 90  // VOL PAY002  // LFD TAPEIN
// DVC 50  // VOL DISK01  // LFD DISCOT
// EXEC DMPRST
// PARAM IN=TAPE
// PARAM OUT=DISC
// PARAM TYPE=FILE
// PARAM NOEXPCK
/$
   FILE filename,ABS
/*
/&
// FIN
```

DISK COPY – BATCH ENVIRONMENT

Function:

Copies a disk pack to another disk pack of the same device type. Either data or library files are accepted. Volumes or files may be specified; file names containing embedded blanks must be delimited with quote marks. Copy routine checks for the current job control statements and any fatal error is displayed on the system console or printer (if specified). The // PARAM statements indicate the operation. The // PARAM END statement can only be used in the volume environment and the ccc must be specified in decimal.

Format 1. Volume Environment:

```
1        10    16
// JOB COPY
// DVC 20  // LFD PRNTR
// DVC 50  // VOL PACK01  // LFD DISCIN
// DVC 51  // VOL PACK02  // LFD DISCOT
// EXEC DMPRST
// PARAM IN=DISC
// PARAM OUT=DISC
// PARAM END=LAST
/&
// FIN
```

Format 2. File Environment:

```
// JOB COPY
// DVC 20  // LFD PRNTR
// DVC 50  // VOL PACK01  // LFD DISCIN
// DVC 51  // VOL PACK02  // LFD DISCOT
// EXEC DMPRST
// PARAM IN=DISC
// PARAM OUT=DISC
// PARAM TYPE=FILE
/$
   FILE filename,LOG
/*
/&
// FIN
```

TAPE COPY – BATCH ENVIRONMENT

Function:

Copies a tape that was created by a previous running of DMPRST to another tape. The tape being copied could have been created on either a multiplexer or selector channel tape unit. Input tapes created on a selector channel unit, however, cannot be read by a multiplexer channel tape unit unless they were created in multiplexer blocked format. Input tapes created on a multiplexer channel tape unit or in multiplexer format on a selector channel unit can be read by any tape unit. The output tape may be on either a multiplexer or selector channel tape unit. Output tapes produced on a selector channel unit can be blocked in multiplexer format by adding the MUX parameter to the // PARAM OUT statement.

This function is only applicable for tapes produced in the volume environment. DMPRST will not copy tapes produced in the file environment.

Format:

```
1         10    16
// JOB TAPECOPY
// DVC 2Ø // LFD PRNTR
// DVC 9Ø // VOL RELØ3Ø // LFD TAPEIN
// DVC 91 // VOL SP1234 // LFD TAPEOT
// EXEC DMPRST
// PARAM IN=TAPE
// PARAM OUT=TAPE
/&
// FIN
```

DUMP, RESTORE, AND COPY – INTERACTIVE ENVIRONMENT

Function:

Creates backup program and data libraries stored on disk to disk or tape interactively from either a UNISCOPE 100, UNISCOPE 200, or UTS 400 terminal.

Operation:

Enter the LOGON command. After successfully logging on to the system, enter HU in system mode to initiate the hardware utility. The following MENU screen appears:

```
                    HARDWARE UTILITIES          HUØØA

1. DUMP FILES FROM A DISK
2. RESTORE FILES TO A DISK
3. COPY FILES FROM DISK TO DISK
4. COPY AND/OR VERIFY IDA DISK
5. COPY AND/OR VERIFY SELECTOR DISK
6. NONE OF THESE

                    ENTER SELECTION  __
```

Depending on the number selected (1 or 2), an appropriate set of screens is displayed for that function. Enter the requested information on the screen and the operation is performed.

12. Diskette Utility (CREATE)

CANNED CONTROL STREAM

The following canned control stream is entered at the system console. It is used to initialize the utility, to locate or allocate file space, and to specify the functions to be performed.

```
RV WRT,[,D={n  }],V=vsn[,F={filename}][,B={n  }][,O={Y}][,S={Y}]
       {130}          {IMAGE   }      {52 }     {N}     {N}
       [,E={Y}][,R={Y}][,A={Y}]
          {N}     {N}     {N}
```

where:

RV WRT
Initializes the utility.

D={n }
  {130}
Specifies the logical unit number assigned to the desired diskette. The default value is 130.

V=vsn
Specifies the volume serial number of the desired diskette. This is the only required parameter.

F={filename}
  {IMAGE   }
Assigns a file name to the diskette file being created. The default file name is IMAGE.

B={n }
  {52}
Specifies the number of sectors to be assigned to your file. The default value is 52.

O={Y}
  {N}
Opens the file.

S={Y}
  {N}
Scratches all or part of the file.

E={Y}
  {N}
Indicates that the file is to have records added to it.

R={Y}
  {N}
Re-creates an existing file.

A={Y}
  {N}
Causes the file to be numerically sorted.

DISKETTE UTILITY COMMANDS

L Lists the contents of the file.

C Indicates the line is to be changed. Entered with sequence number of line.

A Adds a record to the file.

R Replaces contents of a record with new data.

D Causes system to display the diskette utility commands.

E Terminates the diskette utility.

I Causes utility to ignore previously issued change (C) command.

F Closes file.

DISKETTE INDEX SCAN

The format for the RUN command to perform an index scan is:

```
RV ISCAN,[,D={n  }][,V=vsn]
           {130}
```

where:

RV ISCAN
Initializes the utility.

D={n }
** {130}**
Specifies the logical unit number assigned to the desired diskette device. The default value is 130.

V=vsn
Specifies the volume serial number of the desired diskette. The default value is the volume serial number of the diskette mounted on the device specified by the D parameter.

DISKETTE INDEX SCAN RESPONSES

The permissible responses to the index scan sector query (?SS=07-26) are:

07 through 26
Specifies the sector to be displayed.

C
Indicates that the information in the displayed sector is to be changed. This entry is followed by the new information to be placed in the sector.

I
Indicates that no changes are to be made to the displayed sector.

END
Terminates the index scan.

13. System Utility Symbiont (SL$$SU)

13. System Utility Symbiont (SL$$SU)

SL$$SU CAPABILITIES

There are two versions of the system utility symbiont: SU and TU. TU should be used for tape while SU should be used for all other functions.

Table 13-1 lists the SL$$SU functions available.

Table 13-1. SL$$SU Functions (Part 1 of 2)

Function Code	Function Performed
CC	Reproducing cards punched in Hollerith code
CC96	Reproducing 96-column cards
CCB	Reproducing cards punched in binary and Hollerith code
CCS	Reproducing and resequencing source programs
CS96	Reproducing and resequencing source programs contained on 96-column cards
CT	Writing card to tape in unblocked format
CT96	Writing 96-column cards to tape in unblocked format
CTR	Writing card to tape in blocked format
CP	Listing cards
CP96	Listing 96-column cards in character format
CH	Listing cards containing compressed mode
CH96	Listing 96-column cards in vertical hexadecimal format
JCP	Punching cards from the system console
TT	Copying a tape to another tape
TH	Printing a tape in character and hexadecimal format
THR	Printing a tape in character, hexadecimal deblocked format
TP	Printing a tape containing only standard characters
TPR	Printing a tape in character and deblocked format
TRS	Locating a specific record on tape
TRL	Changing existing records on tape
TC	Punching cards from tape
INT	Prepping a tape
FSF	Forward space to a specific file
BSF	Backward space to a specific file
FSR	Forward space to a specific record

Table 13-1. SL$$SU Functions (Part 2 of 2)

Function Code	Function Performed
BSR	Backward space to a specific record
WTM	Writing tape marks
REW	Rewind a tape
RUN	Rewind a tape with interlock
ERG	Erasing a portion of a tape
DD	Printing a disk or diskette in unblocked format or a VTOC from a disk not created by OS/3
DDR	Printing a disk in reblocked format
VTP	Printing the volume table of contents of a disk or the data set labels on a diskette
SVT	Printing a short format VTOC file from disk
AVX	Displaying available disk extents on console screen
DID	Changing the volume serial number of a disk or diskette

14. Stand-Alone Disk Copy Routines (SU$IDA or SU$SEL)

SCOPE

Copies any disk pack to another disk pack of a similar type. Executes as a dialog with the operator via console messages. The permissible types of input and output devices are:

Input	Output		Program Name
8415R	8415R		SU$IDA
8415F	8415F		
8415R	8415F		
8415F	8415R	①	
8416	8416		
8418L	8418H	②	
8418	8418		
8418H	8416	②	
8416	8418H	③	
8411	8411		SU$SEL
8414	8414		
8424/25	8424/25		
8430	8430		
8433	8433		
8430	8433	②	
8433	8430	②	

NOTES:

① *It requires three removable disk packs to store the data on one fixed disk pack.*

② *Only one-half of the disk is copied to the new disk. The VTOC will show that 808 cylinders are available.*

③ *The VTOC for the 8418 disk will show 404 cylinders are available.*

EXECUTING SU$IDA OR SU$SEL

The following procedure permits the system operator to execute the stand-alone disk copy routine:

1. Perform the following on the operator/maintenance panel:
   a. Set the DATA ENTRY ROTARY switches to the address of SYSRES.
   b. Set INITIAL LOAD CONTROL ROTARY switches to program load.
   c. Press the SYSTEM RESET switch.
   d. Press the RUN switch.

2. Perform the following on the system console:

   a. When the following message is displayed

      ```
      IPL OF THE STANDARD SUPERVISOR UNLESS THE NEW NAME IS KEYED IN_______
      ```

      key in the proper program name, depending on the device type, as follows:

      ```
      SU$IDA,L or SU$SEL,L
      ```

   b. Press the TRANSMIT button.

   c. Answer the system console messages regarding:

      Type of operation (copy or verify)

      Input device type and address

      Output device type and address

      Operational type (full, partial, copy, allocated, or restore-fixed)

      Beginning and ending extent addresses (ccc,hh)

      If an invalid reply is entered, an error message is displayed and the original message can be submitted.

See OS/3 system service programs user guide, UP-8062 (current version) for detailed information.

15. List Software Maintenance Correction Routine (SMCLIST)

SMCLIST FUNCTION

You can use the SMCLIST canned job control stream to print a listing of the software maintenance corrections contained in the SMCLOG file. This listing is printed in either a full or condensed format depending on which option you choose. Also, by specifying certain parameters, you can produce listings sorted by SMC number, component number, program-product-type number, date, and time.

EXECUTING SMCLIST

The format of the SMCLIST canned job control stream is:

```
RV SMCLIST,[,FMT={F}] [,SEQ1=(COMP   )] [,SEQ2=(COMP   )]
                 {C}          (DATE   )         (DATE   )
                              (TIME   )         (TIME   )
                              (PP-TYPE)         (PP-TYPE)
                              (SMC#   )         (SMC#   )
```

where:

FMT={F | C}

Specifies the format of the listing being produced.

FMT=F

Specifies that a full listing is printed.

FMT=C

Specifies that a condensed listing is printed.

A full listing is a listing sorted primarily by component number and then by SMC number. It gives more information about each SMC than a condensed listing, such as the regenerations an SMC requires or the method used to install it.

Since you don't always need as much information as the full listing shows, we also provide a condensed listing. A condensed listing contains only SMC numbers in ascending order and an indication of whether any SMCs were backed out, replaced, or were not installed because of an error during installation. See Figures 16–1 and 16–2 for an example of each listing.

The default for the FMT parameter is C for condensed. Unless you specify FMT=F on your SMCLIST run command, you will always receive a condensed listing of the SMCs in the SMCLOG file.

SEQ1={COMP | DATE | TIME | PP-TYPE | SMC#}

Is the major sort key used to specify the sorting method used.

SEQ1=COMP

Specifies component number.

SEQ1=DATE
Specifies the date the SMCs were applied.

SEQ1=TIME
Specifies the time the SMCs were applied.

SEQ1=PP-TYPE
Specifies program-product-type number.

SEQ1=SMC#
Specifies SMC number.

If you omit this keyword, a listing sorted by component number is printed.

SEQ2={**COMP** | **DATE** | **TIME** | **PP-TYPE** | **SMC#**}
Is the minor sort key that specifies the sorting method used.

SEQ2=COMP
Specifies component number.

SEQ2=DATE
Specifies the date the SMCs were applied.

SEQ2=TIME
Specifies the time the SMCs were applied.

SEQ2=PP-TYPE
Specifies program-product-type number.

SEQ2=SMC#
Specifies by SMC number.

If you omit the SEQ2 keyword, a listing sorted by SMC number is printed.

NOTE:

The SEQ1 specification may not be the same as the SEQ2 specification.

Appendix A. SAT Librarian Functions

SCOPE

The functions performed by the SAT librarian are supplied by means of a set of integrated subroutines, file tables, and overlay segments associated with the support of individual functions.

The following outlines the tasks performed by the SAT librarian:

- Maintains all program elements within the system complex:
  - User programs
  - System programs
- Manages all program elements comprising the system complex: source code, proc code, object code, and load code.
- Performs, effectively (on all or specific portions of SAT library files), tasks such as:
  - Copying one library file to another library file by duplication or selectively as to module or module group
  - Building a new library file composed of merged modules or groups from other libraries and media
  - Adding to an existing library file
  - Deleting from an existing library file
  - Compressing an existing library file
  - Punching modules or groups of modules into cards
  - Creating the elements from card modules or groups of card modules
  - Converting standard load modules to block load modules
  - Correcting loadable and object code elements based on phase/section definitions
  - Renaming a specific group, module, phase, control section, or entry name
  - Identifying object and load modules as reentrant or nonreentrant
  - Listing a specific module or module group in the appropriate format, depending on code type
  - Providing a library map of supplied control directives and status information concerning the content of the files being handled
  - Validating library file and program structure

- Supplying meaningful, noncryptic diagnostics on a library map
- Providing gang operations involving certain specific file maintenance options
- Aiding in the system generation process
- Listing the contents of a library file in alphabetical order

- Accomplishes specific tasks in accordance with control statements directed to the SAT librarian through the control stream.

Appendix B. SAT Librarian Correction Cards

COR CORRECTION CARDS FOR PROGRAM SOURCE AND MACRO/JPROC SOURCE MODULES

The corrections made to program source or macro/jproc source modules consist of insertions and replacements. Deletions are accomplished as a function of insertion or replacement or as a function of three control statements that are used as subfunctions of the COR control statement. The three control statements are the recycle (REC), the sequence (SEQ), and the skip (SKI). These same subfunctions are also used for correcting and recording source or proc modules.

When making source or proc module insertions or replacements, the card containing the correction is the actual source record. The type of correction is indicated by the sequence number specified in the sequence number field of the correction card. Replacements are performed on a one-to-one basis by using a correction card with the same sequence number as the record being replaced. Insertions are performed by using a correction card with a sequence number that falls between the sequence numbers of the records between which the correction is inserted. If an insertion consists of more than one record, only the first correction card must contain the sequence number. Any number of unsequenced correction cards may then follow. Records are inserted into the source or proc module in the same sequence in which they are arranged in the correction deck; cards that are out of sequence in a correction deck are inserted out of sequence in the source or proc module, resulting in the printing of an error message on the librarian map.

If the corrections to a source module include the /$-/* job control statements, they must be paired and be without sequence numbers.

From the preceding discussion, it can be determined that source or proc modules must contain record sequence identifiers in order to be corrected by the librarian. It is not a requirement, however, that source or proc modules carry sequence numbers to be in a given library. Sequence numbers may be added at any time, during module creation from cards via the add element (ELE) control statement or thereafter via the sequence (SEQ) control statement when it is used as a primary subfunction to the COR statement.

COR CORRECTION CARDS FOR NONSOURCE MODULES

When object or load modules are being corrected, the correction cards build a text record containing the data and instructions required as patch corrections necessary to the specific object or load module. Text records are inserted in the corrected module just ahead of the transfer record. Then, whenever the object or load module is loaded in main storage, its correction records are inserted in their appropriate places in the module, overlaying any records that may have been nullified because of their replacement. When patched modules are listed, patches are flagged. When changes are being made to object and load modules, CSECT and phase sizes may not be altered. Patches must be correctly sequenced for phased load modules.

Subfunction patch corrections for object and load modules must immediately follow the COR control statement. A correction card for an object or load module must have the following format:

$$\left\{\begin{matrix}-\\P\end{matrix}\right\}\ \text{address}\left[,\left\{\begin{matrix}\text{esid-no}\\\text{phase-no}\\\text{ORG}\end{matrix}\right\}\right][,\text{text}[,\text{rld}]]$$

The specified address is the relative address to be assigned to the generated text record.

A hyphen in column 1 indicates that the address is relative to the object or load module base address. A letter P in column 1 indicates that the address is relative to the load module phase being patched.

The address value must be hexadecimal and may be positive or negative. A negative address is specified by a hyphen in column 2 followed immediately by the address. A positive address value must begin in column 2.

Any error detected in a correction card is flagged. The last correction card for the module must be followed by an EOD librarian control statement.

Both text and relocation data (RLD) records may be supplied for the patch. RLD masks must be represented in hexadecimal 3-byte multiples. Each patch supplied causes the generation of an appropriate text record. Contiguous patch addresses on succeeding patches do not cause the generated text to be merged.

Padding of a zero to the nearest half byte is automatic for the address, esid/phase-no, and text specifications.

The specified hexadecimal address is relative to the base address of the object or load module if a hyphen is used or relative to the address of the phase area if the letter P is used. This relative address is assigned to the generated text record. If an object module is being patched, the ESID is specified in the range $01–255_{10}$. If a load module is involved, the phase number is specified in the range 00–99. Text is supplied as a contiguous string of hexadecimal digits to be assigned starting at the indicated address. Optional RLD data (3-byte, six hexadecimal digit multiples) may be specified for the object or load text record being created. The minimum amount of text patchable is one byte. If no text is specified, the patch correction is flagged and RLD data, if present, is disallowed. As indicated, commas must separate each parameter. If esid/phase-no is omitted, 1 is assumed; however, the comma still must be coded.

The use of ORG causes the location counter to be set or reset to the address specified by positional parameter 1. The current value of the location counter will automatically be added to all addresses specified in subsequent correction cards. The location counter will remain set at that value until another ORG directive is used or the EOD librarian control statement encountered. If used, the text and rld parameters must be omitted.

Appendix C. MIRAM Librarian Functions

SCOPE

The functions performed by the MIRAM librarian are supplied by means of a set of integrated subroutines, file tables, and overlay segments associated with the support of individual functions.

The following outlines the tasks performed by the MIRAM librarian:

- Maintains all screen format (F-type or FC-type) modules, saved run library (J-type) modules, help screen (HELP) modules, and menu (MENU) modules.
- Performs effectively (on all or specific portions of MIRAM library files), tasks such as:
  - Copying modules from one file to another.
  - Deleting modules from a file.
  - Printing entire modules.
  - Printing of file directories.
  - Defining user files.
  - Changing the name of an existing module.
  - Inserting comments on the header record.
- Accomplishes specific tasks in accordance with control statements directed to the MIRAM librarian through the control stream.

Appendix D. Basic Linkage Editor Statement Processing

SCOPE

The linkage editor cycles through two control modes for each load module it generates. The exact processing done in each mode is determined by the parameter or control statements processed in each mode. For this discussion, the parameter statement and control statements that affect the operation of the linkage editor are divided into two groups. The first group contains all the statements that direct the basic operation of the linkage editor and includes:

- all job control statements directed to the linkage editor – start-of-data (/$), parameter specification (// PARAM), and end-of-data (/*);
- all LINKOP statements; and
- the LOADM statement.

These statements are processed in the first control mode.

The second group of statements consists of those which basically affect the structure and content of the load module, rather than the operation of the linkage editor. All remaining linkage editor control statements make up this group and are processed in the second control mode.

All the group 1 statements input to the linkage editor for the purpose of building any given load module may come from the primary control stream, plus any number of user source libraries. (See // PARAM and LINKOP CLIB keyword description.) All group 2 statements, however, must come from a single source. The first group 2 statement detected starts mode 2 processing, and mode 2 processing continues until INCLUDE processing is ended for the load module. A given load module may thus pick up options and its load module name from multiple sources, but its structure must be defined in a single input source (primary control stream input or user source library).

User source libraries for linkage editor statements are specified by the CLIB keyword of the // PARAM statement or the LINKOP control statement. The processing performed when building a load module depends, in some respects, on the source that contains the CLIB specification. In most cases, the statement containing the CLIB specification is the last statement processed for the current load module from the source that contains the CLIB specification. The only exception to this is if the source specified by the CLIB specification contains only group 1 statements. If the linkage editor statements are being input from the primary control stream when the CLIB specification is processed, the current location in the control stream is saved and is the point of return when the statements in the specified source are exhausted. In contrast, if a CLIB specification is processed while in a source library, the source library is disconnected and can never be returned. Thus, it can be seen that multiple CLIB specifications can be meaningfullly specified only in the primary control stream because only the first CLIB specification in a source module will ever be processed.

A single link-edit job step that produces multiple load modules proceeds as follows:

1. Enters mode 1 and processes group 1 control statements for first load module to be produced
2. Enters mode 2 and processes all group 2 control statements to produce first load module
3. Reenters mode 1 and processes group 1 control statements for second load module
4. Reenters mode 2 and processes group 2 control statements to produce second load module
5. Repeats steps 3 and 4 until all load modules are produced

The following is a list of the basic control statements used for directing the operation of the linkage editor:

- Specify linkage editor options (LINKOP or // PARAM)
- Begin load module (LOADM)
- Include object code (INCLUDE)
- Begin overlay phase (OVERLAY)
- Begin new region (REGION)
- Define phase entry point (ENTER)
- Define label (EQU)
- Modify location counter (MOD)
- Reserve storage (RES)

Table D-1 lists the linkage editor control statements, their use, and placement in the job stream.

Table D-1. Linkage Editor Statements (Part 1 of 2)

Mnemonic Form	Use	Placement Guidelines
ENTER	Specifies the start-of-execution (program entry) point for the phase currently being built in a load module. This is the point to which control is optionally transferred once the phase has been loaded in main storage. The transfer point is optionally assigned by the linkage editor if no such statement is supplied for a phase. This statement is normally the last specified for each phase defined in a load module.	Normally, the last statement specified for each phase defined in a load module. May not be embedded in an automatically included object module.
EQU	Equates an otherwise undefined label with a defined label in a load module. The normal method of defining and satisfying cross-references by the linkage editor is via the proper INCLUDE directives and external symbol dictionary (ESD) records in object code included in the load module. This statement, however, allows the programmer to equate two symbols, a symbol and a value, or a value only that could not otherwise be resolved in the link-edit run. If one symbol is being equated to another, the equating symbol must have been previously defined.	May follow a LINKOP, LOADM, OVERLAY, or REGION control statement.
INCLUDE	Tells the linkage editor to include in the load module being built a named object module or object module elements. Also specifies the name of the module and module elements, if applicable, required to be in the load module segment currently under construction and may also identify the file in which the specified module is located.	Must be followed by at least one object module header record; then may be followed by any number of embedded control statements, which are followed by a transfer record and, again, any number of embedded control statements. May follow a LINKOP, LOADM, OVERLAY, or REGION control statement. Must not immediately follow an ENTER control statement.
LINKOP	Specifies the linkage editor processing options that are to be in effect during building of a load module. These options include: – a method of determining which libraries are to be scanned by the INCLUDE control statement, if no filename is explicitly designated by the user on the INCLUDE control statement; – disallowing the automatic inclusion of object modules in the load module by the linkage editor; – disallowing the automatic overlay mechanism of the linkage editor from being included in the load module;	May precede an INCLUDE, EQU, MOD, or RES statement. May not be embedded in an automatically included object module. May not be specified outside of the linkage editor control stream proper (data set).

Table D—1. Linkage Editor Statements (Part 2 of 2)

Mnemonic Form	Use	Placement Guidelines
LINKOP (cont)	– disallowing the promotion of common storage sections by the linkage editor; – selecting the output file in which the load module created by the linkage editor is to be stored; – ending a link-edit job if a processing error is detected; – selecting the components of the link-edit map to be output for a load module; – inserting comments in the phase header records produced by the linkage editor; – disallowing the recognition of reentrant object modules during the include process; and – initiating the building of reentrant load modules. Causes linkage editor to begin building a new load module when it follows an INCLUDE, EQU, MOD, or RES statement. This feature is intended for use only when building multiple load modules in a single job step.	
LOADM	Requests the linkage editor to begin building an executable load module. Also initiates the creation of the start of the root phase segment and specifies a name for the load module.	Normally the first control statement in each link-edit job. May precede an INCLUDE, EQU, MOD, or RES control statement. May not be embedded in an automatically included object module.
MOD	Modifies the current program counter that the linkage editor maintains when building a load module. Permits the programmer to accomplish boundary adjustments at link-edit time outside the realm of the makeup of modules and code being included.	May follow a LINKOP, LOADM, OVERLAY, or REGION control statement.
OVERLAY	Directs linkage editor to begin building a new load module phase separate from the initial phase and any other previously defined phase. Identifies the object module elements to be included in the new load module phase.	May precede an INCLUDE, EQU, MOD, or RES control statement. May not be embedded in an automatically included object module.
REGION	Causes a new phase to be created in a new region of the load module. Has all the attributes of the OVERLAY control statement and, in addition, initiates the building of a new load module region.	May precede an INCLUDE, EQU, MOD, or RES control statement. May not be embedded in an automatically included object module.
RES	Directs linkage editor to reserve a blank (additional) storage area at the end of the longest path in the load module being built. This area may be used by the load module program as a temporary storage or scratch area.	May follow a LINKOP, LOADM, OVERLAY, or REGION control statement.
//PARAM	Performs the same function as the LINKOP control statement but cannot be specified as part of the linkage editor data set.	May be specified in the job control stream but not in the linkage editor control stream proper (data set).

NOTES:

1. Embedded control statements may be placed before or after the CSECTs in an object module but may not be placed within a CSECT. They must, however, be inserted in the object modules prior to their being link-edited. The system librarian is used to perform this function.

2. Embedded statements are processed as are other control statements, except when a statement that affects the structure of a load module (LINKOP, LOADM, OVERLAY, REGION, ENTER) is detected in an automatically included module. Because automatically included modules always are included in the root phase of a load module, these statements are not permitted to be embedded in automatically included modules and are flagged as errors in the link-edit map.

3. All the control statements embedded in an object module are processed by the linkage editor, even if the object module is being accessed for a partial inclusion of its object code.

4. None of the statements are required to appear in any linkage editor control stream. In fact, a linkage editor control stream need not exist at all for the linkage editor to build a load module; in this case, by default, the linkage editor, when executed, builds a single-phase load module named LNKLOD by using all the object module elements currently residing in the system job run library file (YRUN).

Appendix E. System Service Program Names

SCOPE

The following defines the name for all routines that make up the system service programs:

Routine	Name
SAT librarian	LIBS
MIRAM librarian	MLIB
Linkage editor	LNKEDT
Disk/diskette prep	DSKPRP
Assign alternate track	DSKPRP
Tape prep	TPREP
System utility copy (sectored disk)	SU$C16
System utility copy (nonsectored disk)	SU$CSL
Disk/dump/restore	DMPRST
Hardware utilities	HU
Diskette utility	CREATE (WRT is the name of the canned job control stream used to execute the diskette utility. See Appendix G.)
System utility symbiont	SU or TU (operator keyin) SL$$SU (load module name) SL$LOG (DUMPLOG and DUMPLOGT are the names of the canned job control streams and JPROCs used to execute SL$LOG. See Appendix G.)
List software maintenance corrections	SMCLIST

Appendix F. Other System Utility Program Names

SCOPE

The following describes the names of other system utility programs and shows where they are discussed:

Routine	Name	Document
System log accumulation utility	SL$LOG	Spooling and job accounting concepts and facilities, UP-8869
Joblog routine program	JOBLOG	Spooling and job accounting concepts and facilities, UP-8869
System dump routine	SYSDUMP	Dump analysis UG/PR, UP-8837
Job dump routine	JOBDUMP	Dump analysis UG/PR, UP-8837
Catalog manipulation routine	JC$CAT	File cataloging concepts and facilities, UP-8860

Appendix G. Canned Job Control Streams

Table G-1. Canned Job Control Streams (Part 1 of 2)

Job Name	Function	Described in Document*
ADD1KCOS	Adds 1K COS to YSRC on SYSRES	System service programs UG, UP-8062
ADD2KCOS	Adds 2K COS to YSRC on SYSRES	System service programs UG, UP-8062
ADD3KCOS	Adds 2K of fast COS to YSRC on SYSRES	System service programs UG, UP-8062
CGV	Changes, from the console, a volume serial number on a previously prepped disk pack.	System service programs UG, UP-8062
CHGVSN	Changes a volume serial number on a previously prepped disk pack.	System service programs UG, UP-8062
COPYREL	Copies SYSRES files to a new SYSRES volume	System installation UG/PR, UP-8074
COPY$10	Creates a backup copy of an 8410 disk	Emulation/conversion (9200/9300) UG/PR, UP-8063
COPY$11	Creates a backup copy of an IBM 2311 disk onto a Series 90 supported disk	Emulation/conversion (360/20) UG/PR, UP-8064
DCOP	Copies SYSRES from one disk to another	System installation UG/PR, UP-8074
DRDP	Prints directory partition of librarian disk files	System service programs UG, UP-8062
DUMPLOG	Dumps job or console log records to disk	Spooling and job accounting concepts and facilities, UP-8869
DUMPLOGT	Dumps job or console log records to tape	Spooling and job accounting concepts and facilities, UP-8869
DUMP20	Dumps the image of an IBM 360/20 disk pack	Emulation/conversion (360/20) UG/PR, UP-8064
ECDC	Feeds in cards with names for an emulation carriage tape loops display	Emulation/conversion (9200/9300 and 360/20) UG/PR, UP-8063 and UP-8064
ECDK	Keys in names for an emulation carriage tape loops display program	Emulation/conversion (9200/9300 and 360/20) UG/PR, UP-8063 and UP-8064
JBLOG	Produces a job accounting report with SYSLOG residing on disk	Spooling and job accounting concepts and facilities, UP-8869
JBLOGT	Produces a job accounting report with SYSLOG residing on tape	Spooling and job accounting concepts and facilities, UP-8869
LISTRES	Prints a directory for SYSRES modules	System service programs UG, UP-8062
LNKUPL	Executes UTS 400 upline linker routine	UTS 400-OS/3 interface UG/PR, UP-8611
MODLST	Lists the contents of the five system libraries	System service programs UG, UP-8062
PACKRES	Compresses all modules on SYSRES and prints directory	System service programs UG, UP-8062

*UG = user guide
PR = programmer reference

Table G-1. Canned Job Control Streams (Part 2 of 2)

Job Name	Function	Described in Document*
PIMAGE	Creates a copy of an 8410 disk image onto a Series 90 supported disk	Emulation/conversion (9200/9300) UG/PR, UP-8063
PNCH9300	Punches card deck to use 9300 as peripheral device	Emulation/conversion (9200/9300) UG/PR, UP-8063
PRNT9300	Prints source module needed to use 9300 as a peripheral device	Emulation/conversion (9200/9300) UG/PR, UP-8063
PRP1KCOS	Repositions 1K COS from YSRC for COS-IPL	System service programs UG, UP-8062
PRP2KCOS	Repositions 2K COS from YSRC for COS-IPL	System service programs UG, UP-8062
PRP3KCOS	Repositions 2K of fast COS from YSRC for COS-IPL	System service programs UG, UP-8062
SCLIST	Lists the shared code (YSCLOD) modules	System installation UG/PR, UP-8074
SETREL	Preps and allocates RELEASE/SYSRES files	System service programs UG, UP-8062
SYSDUMP	Prints a complete system dump from SYSRES or another system disk	Dump analysis UG/PR, UP-8837
SYSDUMPO	Prints a complete system dump or a portion of a system dump from SYSRES or another system disk	Dump analysis UG/PR, UP-8837
UPLCNV	Executes UTS 400 upline conversion routine	UTS 400-OS/3 interface UG/PR, UP-8611
UPLDELT	Deletes the specified UTS 400 upline dump file	UTS 400-OS/3 interface UG/PR, UP-8611
UPLDMPN	Prints the specified UTS 400 upline dump file	UTS 400-OS/3 interface UG/PR, UP-8611
UPLDUMP	Prints and deletes the specified UTS 400 upline dump file	UTS 400-OS/3 interface UG/PR, UP-8611
VTOC20	Prints a volume table of contents listing of an IBM 360/20 disk pack image	Emulation/conversion (360/20) UG/PR, UP-8064
WRT	Creates files on a diskette device	System service programs UG, UP-8062

*UG = user guide
PR = programmer reference

Glossary

A

automatic deletion

A function performed by the linkage editor that automatically deletes control sections and entry points previously defined either in the phase being built or in a phase that is on the path of the current phase.

automatic inclusion

A function performed by the linkage editor that automatically includes object modules in a load module for which no definitions exist and these modules are needed for referencing.

B

block load modules

Standard load modules that have been corrected to block format. This format increases the efficiency of program loading because all or large parts of the overlay phases may be loaded by a single I/O operation, resulting in fewer disk accesses. The program loader can read one track at a time until the entire phase is loaded or read only one track that contains one or more entire phases.

C

code set

A predefined series of records that make up either a source, proc, object, load module, or a module group.

common section

A unit of coding (constants only) that is, in itself, an entity. A common section (COM) is contained in an object module.

common storage areas

Load module areas that are common to more than one phase of a load module. The size of each storage area is equal to the largest size requested by all object module elements referring to a particular COM section.

control section

A unit of coding (instructions and constants if a CSECT) that is, in itself, an entity. A CSECT is the smallest unit of coding the linkage editor can process. It (CSECT) is contained in an object module.

current file position

The position of the file directory at which the librarian begins searching to find a module in a library file and continues until the module is found or the end of the file is reached. If the end of file is reached, the search begins anew at the beginning of the file directory and continues until the module directory record is found or the original current position of the file directory is reached again. (This is not true for gang operations.) The current position being arrived at again signifies no-find for that module on the file being searched.

The current position of a file can be affected:

- by the reset (RES) function; and
- by any librarian function except the EOD function.

The RES control statement can place the current position pointer at the first logical record on the file specified or at the first record in a named module in the specified file.

All librarian functions except EOD affect the current position. When the function is completed, the current position pointer for the processed file is the address of the record immediately following the last record processed.

A COP function may be initiated with no output file specified. This effectively places the current position pointer at the record after the last record of the module/group specified in the COP function, without actually copying the module or group.

D

definition

An item that can serve as an object of resolution. Such items are CSECTs, ENTRYs, COMs, and linkage editor EQUs.

E

entry

An object module symbol dictionary record that supplies a definition for one or more possible external references in other object modules.

exclusive phases

Phases that the linkage editor has determined to be not in the same path (possibly in contention for identical storage).

exclusive reference

A reference between exclusive phases.

external symbol dictionary record (ESD)

A linkage editor input record containing the information used to define cross-reference (EXTRN, CSECT ENTRY, V-CON, COM) when object modules are linked.

external symbol identification (ESID)

An index number within a module representing special information with regard to the module's makeup. This information is used by the linkage editor in relocating object modules and object module sections and in resolving references between object modules. The system loader also can use this information to relocate load modules at execution time. Each of the following items is considered to be related to an ESID:

- Each symbol associated with an external reference or definition
- Each text record base address
- Each V-type address constant
- Related relocation masks for text
- Each control or common section name

EXTRN

An object module symbol dictionary record that requests an external definition for one or more references existing in the same object module.

F

F-type modules

Screen format modules created by the screen format generator. They contain data type codes and input and output instructions used by the application program as it processes the data entered and displayed on the screens. (See also FC-type modules.)

FC-type modules

Screen format modules generated by the screen format generator. They contain the text displayed at the workstation when the screens are used in an application program. (See also F-type modules.)

file compression

The squeezing together, by the librarian, of fragmented files (interspersed voided elements), thus providing space at the end of the file for new elements. The compression is automatic if merging or copying involving the file in question occurs. If not, an existing file may be compressed by using certain specific librarian functions. File compression can occur piecemeal within a given librarian job stream. Any associated directories also are compressed in the update job.

file deletion

The removal of individual modules, groups of modules, or entire code sets from library files by using the facilities of the librarian. Deletions can occur during the updating of existing files or the creating of new ones. Deletions applied to existing files can cause file fragmentation (as in the case of module replacement), which can, in turn, be remedied by later file compression.

file extension

The updating (or effective extension) of a current library file without creating a new output file. This may involve replacement of a given element within the file by a new copy of the same element. Replaced elements are flagged as nullified and may be removed via a subsequent file compression operation. Directory entries for replaced elements in extended files are altered accordingly.

file merging

The combining of one or more library files, module groups, or individual modules into a new output library (or libraries) by the librarian. Multiple file merging is permitted, and the number of files involved is a function of the user requirements. The librarian can merge up to six files concurrently (including output files).

Reference to a seventh file (or more) causes the first file (and any succeeding files) to be reopened whenever a new, interspersed file referenced is detected. Thus, merging of multiple files beyond a sixth might necessarily be on an exclusive rather than inclusive basis.

G

gang operations

Certain functions of the librarian in which types of code may be copied, deleted, added, punched, compared, or displayed as a group. These options are initiated via the following command statements and the omission of the name parameter (or name and type parameters) in the operand field.

COP	COM
DEL	REN
ADD	

When the gang mode of the librarian is initialized in one of the foregoing operations, the referenced file is scanned from the current position for the code set type designated (unless all modules are being scanned). When the module of the type indicated is detected, the requested operation commences. Unless this preselected requirement is met at least once, the operation is aborted.

group

A set of modules that exist within a specified library file and that may be handled as an entity by the librarian. Such module sets may be composed of mixed or nonmixed program elements. There may be many groups of the same name in a given library file.

group management

The management of mixed module types in a library file; that is, object code, load code, and source-level code can be intermingled within a given library file. The librarian can process the elements in a file individually or by groups. Via the appropriate gang operation, the librarian can service groups of modules of the same or different type through specific functions. Gang operations allow servicing of all modules of a specified type within a given library or all modules regardless of type within certain designated file limits. For library management, groups may be handled as specified entities by using the library group reference method.

H

help screen modules

Modules containing text displayed at the workstation when an interactive services HELP command is issued. These are contained in the system file YHELP.

I

inclusive phases
Phases that the linkage editor has determined can be in main storage simultaneously.

inclusive reference
A reference between inclusive phases.

internal symbol dictionary record (ISD)
A record describing an internal symbol of the program being link-edited.

J

jproc module
A set of one or more job control statements organized in a specific manner (as dictated by the job control processor) and used as input to the job control run processor.

J-type module
Saved run library modules generated by the run processor.

L

library file
A specific set of programs residing within the physical limits described by an appropriate file label present in a volume table of contents. Library files are herein distinguished from data files in that they are composed only of program elements. A library may be composed of multiple library files.

library file directory
An index within each library file, that facilitates locating elements within that file. Each library file contains partitions, and the directory (first) partition is an index into the two prime data (second and third) partitions.

load module
A single, multiphase, or multiregion program produced by the linkage editor and ready for system execution.

load module management
Management, by the librarian, of load modules generated by the linkage editor. The facilities provided for load module management are much the same as those provided for object module management, except that specific load module phases may be patched. Applied patches are inserted at the end of the designated phase. Load modules also may be listed, punched, filed, and renamed. Load module listings are hexadecimal printouts of load module records. Load elements may be serviced via all standard librarian functions. Phases within a load module also can have an alias phase name, which was given to it at link-edit time, in addition to the phase name assigned to the load segment. This alias phase name also can be renamed by the librarian.

logical file name
As viewed by the librarian, it is a type code (T=tape, D=disk) and a number (0–15). This symbolic name identifies all files referenced within the librarian control stream. This name normally used to identify a file is equated to a logical file name at the beginning of each librarian run.

M

macro module

A set of one or more source code statements (BAL instructions) used as input to the macro facility of the assembler. Macro modules can be written in two separate formats: macro and proc. The name given the macro in the PROTOTYPE statement in the macro format can be the same name as the one given on a NAME statement in proc format in the same file; however, the name on the PROTOTYPE statement cannot have the same name as the one given in the PROC statement in the proc format.

mapping facilities

Facilities that produce a map of the functions the librarian performs each time it is executed. The map is output on the system printer for the user. The map normally includes:

- A listing of all the librarian control statements processed
- A printout of all the header records processed
- Any appropriate diagnostic messages

Additionally, the map can include:

- Source module listings
- Object and load module listings
- Module correction results (insertions versus deletions)

The map normally reflects the state or content of the output library files if one or more were produced; otherwise, it reflects the state or content of the input file serviced by the respective librarian function. In comparison functions, discrepancies are listed one above the other on a record-by-record or block-by-block basis.

module

A program element existing in either source, proc, object, or loadable program format. These elements serve as input and output of various system functions. Different module types may be mixed within a given library file or segregated in separate library files at the discretion of the user.

module gang mode

The mode in which gang operations are to be performed on modules of a specified type. In this mode, the module name is omitted and the type positional parameter is set as follows:

S For source modules

P For proc module

O For object module

L For load module

By setting the type as shown and by omitting the name, the user instructs the librarian to perform the designated operation on all modules of the type specified from the current position of the library file up through end of file.

module group

A group of one or more source, proc, object, or load modules that are prefixed with a beginning-of-group demarcator record and an end-of-file sentinel record or end-of-group demarcator. Like a module, a module group may be treated as a single entity by the librarian.

multiphase load modules

Modules constructed by a programmer to minimize the main storage requirements of a program. They consist of more than one program segment, with each segment being a phase that may be loaded into main storage and executed individually as required by the logic of the program. Each phase of a multiphase load module is composed of individually assembled and/or compiled sets of code that may be thought of as a program subroutine. Further, each phase in a multiphase load module can be made to overlay one or more previously executed phases in main storage.

The main storage location at which a phase is loaded is called a node point. All the phases in a multiphase load module, excluding the root phase, are loaded in main storage at a node point. Node points and phases are defined through the linkage editor OVERLAY and REGION control statements.

The INCLUDE statements following an OVERLAY or REGION control statement identify the object module elements that are to comprise the phase. Ignoring the root phase, the number of phases in a multiphase load module coincides with the number of OVERLAY and REGION control statements present in the control stream that caused its generation. The root phase of all load modules is initiated with the initiation of the load module, normally in response to a LOADM control statement.

multiregion load modules

Modules that are basically the same as multiphase load modules, except that a multiregion load module is so constructed that the origin of the first phase of each region is at the end of the longest path defined in the previous region, rather than at the end of the phase previously defined. This feature prevents the phases in one region from overlaying any portion of a phase in any other region.

In general, a load module constructed as a multiregion load module normally requires more main storage space for execution than the same program configured as a multiphase load module. Multiregion structures are most useful, however, when a need exists for a phase to reside in an area where it will not be overlaid by other phases that may not be directly associated with it. Also, it is sometimes possible to realize an actual saving in main storage space with a multiregion construction when one or more control sections are required for two or more distinctly separate phases but are not required by any other phases. In this case, these CSECTs could be placed in a separate region, rather than being embedded in a phase common to the phases requiring them. Placing them in a common phase could unnecessarily affect the origins of succeeding phases even though the majority of these phases do not require these CSECTs. The opposite is true when these CSECTs are placed in a separate region. Region origins always are assigned at the end of the longest path of the preceding region, and unnecessary placement of CSECTs in separate regions may have an adverse effect on the overall length of the load module.

Regions are declared by the programmer with the REGION control statement in much the same way a phase is declared with an OVERLAY control statement. Both statements initiate construction of a new phase at some symbolic starting address, or node point, specified in the control statement. The only difference between the two statements is the way they cause the linkage editor to assign an origin to the phase being created.

N

naming conventions

The conventions used to name modules. Modules within library files (regardless of type) contain an 8-character EBCDIC identifier that is used as the name of the module. (Modules of the same name and type are not allowed in one file.) If the name assigned has less than eight characters, it is left-justified and space-filled. The exceptions are nonalias load module phase names, which always contain only six significant digits (EBCDIC, left-justified and space-filled); the least significant two characters specify the phase number (00-99 EBCDIC). Naming of specific modules can be performed at:

- assemble/compile time for object modules;
- link-edit time for load modules;
- library services time for source-level and macro definition modules; or
- file time for jprocs and job control streams.

The librarian also can be used to rename specific modules or module groups. It can:

- rename a source-level or macro definition module;
- rename an object module or a specific CSECT;
- rename common sections and ESD records in object modules;
- rename all phases of a load module (retaining phase numbers); and
- rename the alias phase name of a load module phase.

node point

The starting, or origin, address of each loadable phase (which also is the terminal address of a previous phase), or the address of a definition in the same path of a phase.

nonreentrant code

Code that is self-modifying. Consequently, only one copy of this code may be executing at any one time. If more than one execution is occurring simultaneously, the code will not produce the desired results because one execution path will be using information pertinent to another execution path.

nullified module

A module logically marked for future physical deletion by the librarian. A nullified module cannot be accessed by either system or user programs.

O

object module

A set of one or more control sections of code produced by a language processor.

object module management

The procedure by which language processor output modules can be maintained by the librarian, in that object code can be patched, listed, punched, filed, and renamed. Specific CSECTs or ESDs also may be renamed. Patch corrections are inserted at the end of the object module. Listings of object modules are hexadecimal printouts of object records. All standard librarian functions regarding module manipulation apply to object elements.

Whenever nonsource elements are serviced, they are checked for proper content and record sequence. Discrepancies trigger diagnostic processing.

Object modules can be designated as being reentrant by the librarian, so that the linkage editor can set up linkages to a shared load module rather than including the object module in the load modules needing it.

P

partition

A logical portion of a library file that contains either an index for the file or the program data for the file.

path

A path determined by one phase leading into the next phase and so on. This allows label references in a load module to be traced from one phase backward through each preceding phase between it and the root phase. No more than 14 phases are allowed in any one path.

phase

A portion of a load module that can be loaded as an overlay by a single execution of a supervisor LOAD or FETCH macroinstruction; also called a load segment. A load module can consist of up to 100 phases.

phase dependencies

Whenever a phase is in main storage or is being loaded in main storage, all the phases in its path from start of module or from start of region also should be in main storage. Phases may be loaded in any numerical sequence whatsoever, excluding the root phase, and reloaded any number of times, as required by the logic of the program. The assigned location of the phases has no bearing on the order in which the phases are executed. Any part of a phase that is modified during its execution will remain so only until the phase is overlaid.

phase names

Names that identify the various phases of a load module when they are loaded in main storage for program execution. (Programmers who wish to do their own program loading, rather than have the automatic overlay region control mechanism of the linkage editor embedded in their load module, must reference a phase name in a FETCH or LOAD macroinstruction whenever a phase is to be loaded for execution.) Each phase is automatically assigned by name by the linkage editor. This name is based on the name assigned to the load module by either the programmer or the linkage editor. An alias phase name also may be assigned to each phase through the OVERLAY or REGION control statement that causes its generation. The assignment of alias phase names allows the programmer to reference the phases of a load module in his subroutines or phases without knowing the order in which the phases will be defined in the linkage editor control stream. The linkage editor automatic overlay control mechanism always refers to the linkage-editor-generated phase names.

preamble SEXTRN processor

A piece of code that resides in the user prologue that completes the transfer of control from a program to a shared code module outside the program. The SCON value in register 15 is converted to the machine address of the shared code ENTRY. The shared code routine is given control exactly as if it were included with the calling program. The content of the 4-byte SCON in main storage is not altered during this process; only register 15 is altered.

program length

The length, of an overlay program, that must be considered in building a user program. The length of the longest path is the minimum storage requirement of an overlay program. Also, when a program is built with the automatic overlay mechanism, the storage requirements of the necessary control routine, entry table, phase table, and possible region table also must be considered.

program library

All program modules existing within a given system complex or user environment.

program source module

A set of one or more source code statements used as input to language processors or as data for a user program.

R

reentrant code

Code that does not write to its own area. Reentrant code can be used by more than one program in a multiprogramming environment, simultaneously. When more than one program is using reentrant code, only one copy of the code needs to be present in main storage. (See *shared code.*)

region

A contiguous area of main storage within which assigned load module phases can be loaded independently of phases in other regions. Up to 10 regions may be in a single load module.

resource

A reentrant load module that is needed by a nonreentrant load module for its proper execution. The operating system schedules the nonreentrant load module when the resources (reentrant load modules) are available.

resource number

A number, unique to each load module, that is assigned to each resource record by the linkage editor.

resource record

A record produced by the linkage editor that names a shared module required for the successful execution of the load module being created. There may be several resource records per load module.

RLD mask

An object or load module relocation mask used to specify text modification.

root phase

The phase that is the first order of storage allocated to a load module.

root segment

The segment that controls the phases and the order in which they run as specified by phase 0. The root segment contains common constants and subroutines used by more than one phase and storage space for information being passed from one phase to another.

When the routine is executed, control is passed to the root segment. The root segment passes control first to phase 0 and then to the other phases. When a phase is completed, control always is returned to the root segment.

S

SCON (shared constant)

An ACON or VCON that got resolved to a shared module. It has a 4-byte value – the high byte contains the SINDEX number of the SEXTRN, and the three low bytes contain a negative address (address of preamble SEXTRN processor).

SENTRY (shared ENTRY)

All definitions (CSECTs, EQUs, and ENTRYs) encountered in the link of a shared module.

SENTRY number

A number, unique to each load module, that is assigned to each SENTRY of a shared module.

SENTRY record

A record produced for each SENTRY found during the link of a shared object module. It will contain the SENTRY name, its link origin, and SENTRY number.

SEXTRN record

A record produced by the linkage editor that describes a SEXTRN. This record contains the SEXTRN name, its SINDEX number, and the resource number of the module it successfully got resolved to.

SEXTRN (shared EXTRN)

An EXTRN that successfully got resolved to a shared module.

shared code

A piece of code that does not write to its own area: that is, reentrant code. Normally, this code expects its user to pass it a pointer to a work area through a specified register. In OS/3, shared code may be delivered with the operating system (e.g., shared data management) or written by the user himself. Shared code is assumed to be in object module form and its linked version as a load module.

shared definition

A definition that is contained in a reentrant object module.

shared record

A resource, SEXTRN, or SENTRY record.

SINDEX number

A number, unique to each load module, that is assigned to a SEXTRN.

single-phase load modules

Modules that consist of a single program segment that is loaded into main storage each time the program is executed. The storage structure of a single-phase load module can be represented by a single horizontal line whose length is relative to the amount of serial storage locations required to store the load module in main storage. All load modules generated by the linkage editor start out as single-phase load modules and are created only as multiphase or multiregion load modules if so directed through OVERLAY and REGION control statements in the input control stream. Thus, single-phase load modules are modules that consist solely of a root phase; multiphase and multiregion load modules consist of a root phase plus one or more phases.

source module

A set of one or more source code statements used for input to language processors or the job control run procedure.

source/proc module management

The facilities provided by the librarian, for the maintenance of source or proc code modules. The librarian can maintain copy, proc, and macro definition modules as source-level code modules for processing. Source-level code modules can be listed, filed, punched, corrected, and renamed, as well as handled with the standard librarian-provided functions. Specific source records can be added and deleted from a source element. Updated source-level modules may be mapped as corrections are applied. Source records are printed individually in 80-byte EBCDIC format.

system library file

A permanent system file containing system programs and existing in support of the operating system. The OS/3 system library is composed of five permanent system library files and a temporary system job run library file, as follows:

1. System load library file (YLOD)
2. System object library file (YOBJ)
3. System source library file (YSRC)
4. System macro library file (YMAC)
5. System JCS library file (YJCS)
6. System job run library file (YRUN)

The preceding breakdown is defined by the operating system. OS/3 supports the system job run library file only for the duration of each job. This file acts as the default file in cases involving different system programs. Library files may be composed of both user and system programs. Such a mix is transparent to the librarian.

system program library

All system program modules for a given system. This library exists in support of the operating system.

T

text
Records containing the actual instructions and data associated with a particular object module control section or load module phase.

total gang mode
If the function to be performed does not concern itself with a specific module or code set, the type and name positional parameters must not be specified. In this case, an entire library (or remainder of one prepositioned) may be manipulated via the facility desired. The gang operations always process the library file from its current position as defined by the respective file table (DTF) contained within the program.

transfer record
A unique record signaling the end of an object module or load module phase.

tree
The graphic representation that shows how load module phases can use main storage at different times.

type
A specific indication of the classification of the program module or record being referenced within a given library file.

U

update records
As defined by the assign alternate track routine, records correcting errors that are encountered when reading from a defective to an alternate track.

user library file
A private user file containing user worker programs existing in any or all of the specified formats. This is a permanent file existing in support of the user. A user library may be composed of multiple user library files.

user program library
All user program modules for a given system. This library is private and comprises all user worker programs.

V

V-CON
A special reference to an external symbol that may exist in an exclusive phase when the program has been link-edited and that may require automatic loading of an exclusive path. Such a constant may not be used to reference data; it is used only for branching.

SPERRY UNIVAC

USER COMMENT SHEET

Your comments concerning this document will be welcomed by Sperry Univac for use in improving subsequent editions.

Please note: This form is not intended to be used as an order blank.

_____

(Document Title)

_____ _____ _____

(Document No.) *(Revision No.)* *(Update No.)*

Comments:

Cut along line.

From:

_____

(Name of User)

_____

(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SPERRY UNIVAC

ATTN.: SYSTEMS PUBLICATIONS

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424

CUT

FOLD

SPERRY UNIVAC

USER COMMENT SHEET

Your comments concerning this document will be welcomed by Sperry Univac for use in improving subsequent editions.

Please note: This form is not intended to be used as an order blank.

_____
(Document Title)

_____ _____ _____
(Document No.) *(Revision No.)* *(Update No.)*

Comments:

Cut along line.

From:

_____
(Name of User)

_____
(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SPERRY UNIVAC

ATTN.: SYSTEMS PUBLICATIONS

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424

CUT

FOLD