Advanced Operating System (AOS)

Library File Editor

User's Manual

093-000198-01

For the latest enhancements, cautions, documentation changes, and other information on this product, please see the Release Notice (085-series) supplied with the software.

Ordering No. 093-000198

Printed in the United States of America
Revision 01, April 1977

DataGeneral
SOFTWARE DOCUMENTATION

NOTICE

Advanced Operating System (AOS)
Library File Editor
User's Manual

Original Release - March 1976
First Revision - April 1977

Contents

Chapter 1 - Introduction to the Library File Editor

Terms and Conventions 1-1
LFE Functions 1-1

Chapter 2 - How to Operate the Library File Editor

Operating Procedures 2-1
Command Descriptions 2-1
(A) Analyze One or More Libraries 2-2
(D) Delete Objects from a Library 2-4
(I) Insert Objects into a Library 2-4
(M) Merge Libraries 2-5
(N) Create a Library 2-6
(R) Replace a Library Object 2-6
(T) List Library Object Titles 2-7
(X) Extract Modules From a Library 2-7
Error Messages 2-8

Appendix A - Library File Format

Library Start Block A-1
Library End Block A-2

Chapter 1
Introduction to the Library File Editor

Terms and Conventions

The Library File Editor (LFE) utility creates unshared library files, edits them, and analyzes them. (Shared libraries are built by the Shared Library Builder, discussed in the *AOS Shared Library Builder User's Manual.*) The unqualified term "library" in this manual refers to unshared libraries only.

An unshared *library file* is a series of *object modules (OBs).* A *library start block* precedes the series and a *library end block* terminates it. The LFE builds library start and end blocks when it constructs a library; they are described in Appendix A.

You produce an object module by assembling a source module (using MASM, the AOS Macroassembler) or compiling a program (using FORTRAN, etc.). You can bind one or more objects to form an executable program file, or you can merge them together using LFE to form an unshared library file. Collecting objects into libraries provides a convenient way to group objects for common reference by many programs. For example, mathematical routines are often grouped into a common library. Thus, a programmer can write code tailored for specific application problems, without duplicating or incorporating into each compilation the source code for commonly-used functions.

High-level languages, such as FORTRAN, use runtime libraries containing modules (SINE, COSINE, formatting, etc.) which your compiled object modules require for a complete program. When the Binder builds a program file, it uses libraries for a supply of objects with entries (.ENTs) to resolve external references (e.g., .EXTD or .EXTN) made in other objects.

Unless noted otherwise, all numbers in this publication are decimal except for core memory addresses which are octal values.

LFE Functions

LFE accepts input objects to create initial libraries, and can also:

- analyze a library
- list titles of OBs in a library
- merge libraries
- update libraries with new OBs
- extract OBs from a library.

Analyzing a library lists the core sizes for a library's ZREL, NREL shared code, NREL unshared code, etc. portions. LFE analysis also yields the size of each binary in each of its relocatability types, lists each binary's title, and lists all symbols in each OB which are arguments to to .ENT, .ENTO, .EXTD, .EXTN, and .PENT pseudo-ops. If any external references in one OB cannot be resolved by entries in succeeding OBs within the library, then it flags these unresolved references. Finally, it gives the total size of the entire library in all its categories of relocatability.

LFE also builds new libraries, and inserts or removes OBs from existing libraries. You must approach library building or modification with caution, however, since the order of OBs in a library can be critical; the Binder extracts OBs from libraries in sequential order when it builds a program file.

The .ENT pseudo-op makes library entries available to programs. When the Binder detects an .EXTN or .EXTD statement naming these entries, it binds library modules (containing these entries) into a program file. Thus if module A in a library makes an external reference to an entry in module B, module B must follow A in the library (see Figure 1-1).

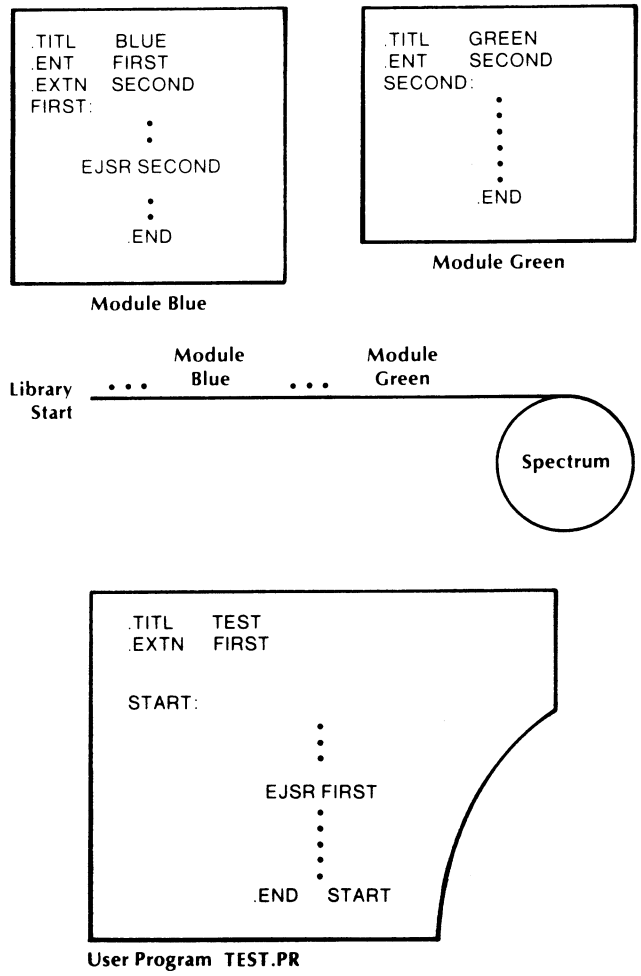


Figure 1-1. Order of OBs in a Library

If you gave the Binder a command line like "XEQ BIND TEST SPECTRUM.LB", it would build the user program file TEST.PR. If module BLUE precedes module GREEN in the library entitled SPECTRUM, then the Binder can resolve BLUE's external reference to SECOND in GREEN. TEST requests FIRST, FIRST is found in BLUE which, in turn, requests SECOND in module GREEN. BLUE is bound into the program file, and then GREEN is too. However, if the positions of BLUE and GREEN were reversed in SPECTRUM, then after the Binder bound BLUE from the library it would search in vain to the end of the library for GREEN. The Binder does not back up and rescan a library. Note that TEST did not explicitly request module GREEN, and it might well have been unaware that GREEN would ultimately become part of the program file.

End of Chapter

Chapter 2
How to Operate the Library File Editor

Operating Procedures

Each LFE command starts with the CLI command XEQ, then the LFE, followed by a single-letter function argument, followed by one or more other arguments. You can apply function switches to the function letter, and argument switches to arguments following the function letter:

```
XEQ LFE function-letter argument ...)
```

Each LFE function has one function letter.

Function Letter	Function
A	Analyze one or more libraries.
D	Delete one or more OBs from a library.
I	Insert one or more OBs into a library.
M	Merge two or more libraries to form a new library.
N	Create a new library.
R	Replace library OBs with new OBs.
T	List the titles of OBs in a library.
X	Extract OBs from a library, and place them into a new OB file.

If filenames you supply in an LFE command have no extensions, then LFE searches first for the names without extensions. If LFE does not find the files, then it searches for library names with the .LB extension and OB modules with the .OB extension.

Depending upon variables such as the amount of memory available to you, there is a maximum number of arguments that you can supply in each LFE command. If you exceed that maximum in any LFE command, LFE will output the error message "TOO MANY ARGUMENTS". All LFE error messages go to the generic file @OUTPUT. LFE error messages are listed and described at the end of this chapter.

Command Descriptions

This manual uses the following symbol conventions in describing LFE command sequences:

Symbol	Meaning	Name
REQUIRED or required	Required argument	boldface
...	Optional repetition	ellipsis
[]	Optional argument	italic brackets
)	Command terminator	new line
&)	CLI command line extender	ampersand and new line

In formats, REQUIRED (uppercase) means a literal entry such as XEQ LFE. required (lowercase) means you must supply an argument such as a filename; for example XEQ LFE X library.

The ellipsis indicates the optional repetition of a previous argument or group of arguments. A pair of italic brackets delimits the optional selection. Thus *"[title]"* indicates that you can supply an optional module .TITLE in the command sequence.

You must provide a command terminator after each command; the manual shows the terminator (new line) as ") ". Press the new-line or carriage return key on your console keyboard to type a new-line character. You can extend a single command beyond the length of a single line by typing an ampersand followed immediately by the new line character. The CLI will then display an ampersand and prompt on the next line, where you can continue typing. Insert a space as a delimiter, either before your ampersand, or at the start of the new line.

(A) Analyze One or More Libraries

Formats:

XEQ LFE *[/F][/L=listfile]* A library *[title...]*)
XEQ LFE *[/F][/L=listfile]* A/M library library...)

where: listfile is the name of an output file to receive the analysis. If you specify no output file, then the analysis appears in the generic file @OUTPUT. If you use the /F switch, the analysis of each OB will be listed on a separate page; i.e., a form feed precedes the analysis of each OB.

library is an input library to be analyzed (/M indicates that more than one library is to be analyzed).

title is the title of a module to be analyzed. If you specify one or more titles, then only the modules with these titles will be analyzed.

Purpose:

This command analyzes one or more libraries, or one or more OBs within a single library. The output analysis lists symbols, naming the symbol type, and provides certain flags indicating potential error conditions. Also provided is the amount of each type of data in the OB, the total library size (in each type of relocatability), and the total size of unlabeled common, if any.

You can analyze any group of OBs after merging them into a library (with the LFE N command).

Analysis describes the type of each of the symbols displayed on output using the following flags:

AC accumulating symbol
EN .ENT
EO .ENTO
NC named common symbol
PN .PENT (OB destined for a shared library, merged into a dummy library for analysis only)
T .TITL
XT .ENTN or .EXTD

Analysis also notes library errors or cautionary conditions with the following flags:

* multiply-defined symbol (warning). The first module in a multiply-defined series will flag with a star (*) in its EN row the titles of all other modules defining this same symbol. Each other module defining this symbol will precede the EN row with a star, and will list the title of only the first module in the multiply-defined series. All external references (XT) list the title of only the first module in the series. If you can guarantee that only one of the modules containing the symbol with the same name will ever be bound, then you can ignore this warning.

P phase warning. A symbol is .ENTered in a module that occurs before another module externally referencing this symbol. If you can guarantee that the first module will always be bound (and thus the external reference will always be satisfied), then you can ignore the phase warning.

U An .ENTry was not found for an .EXTN or .EXTD symbol. In analyzing part of a library, you can ignore this warning if you know that the .ENTry occurs elsewhere in this library or in another library that would always be bound.

If you name an existing list file in the command line, then listing output produced by this command will be appended to whatever is in that file.

Example:

XEQ LFE/L=@LPT A LIBRARY1)

The analysis report is:

```
      T          IOPKT
      EN     IOPKT
      EN     SPOOL    MACR     SYMB     XREF
      EN     DIR      SYMB     CLANG
      XT     DOUBLE   MACR
      XT     WAIT     XREF

      PAGE ZERO RELOCATABLE DATA=0
      NON-SHARED CODE          = 98
      NON-SHARED DATA          = 0
      SHARED DATA              = 0
      SHARED CODE              = 0

      T          MACR
      EN     DOUBLE   IOPKT    *SYMB    CLANG
      EN     MACR
P     XT     SPOOL    IOPKT
U     XT     ASMVA

      PAGE ZERO RELOCATABLE DATA= 2
      NON-SHARED CODE          = 56
      NON-SHARED DATA          = 0
      SHARED DATA              = 0
      SHARED CODE              = 0

      T          SYMB
 *    EN     DOUBLE   MACR
      EN     SYMB
 P    XT     DIR      IOPKT
 P    XT     SPOOL    IOPKT

      PAGE ZERO RELOCATABLE DATA= 2
      NON-SHARED CODE          = 126
      NON-SHARED DATA          = 0
      SHARED DATA              = 0
      SHARED CODE              = 0
```

```
     T          XREF
     EN       WAIT     IOPKT    CLANG
     EN       EP1
     EN       IOPK1
P    XT       SPOOL    IOPKT
     XT       BLIP     CLANG
U    XT       BLOP

     PAGE ZERO RELOCATABLE DATA = 12
     NON-SHARED CODE            = 69
     NON-SHARED DATA            = 0
     SHARED DATA                = 0
     SHARED CODE                = 0
     ABSOLUTE DATA              =14

     T          CLANG
     EN       BLIP     XREF
     EN       BLAP
P    XT       DOUBLE   MACR
P    XT       DIR      IOPKT

     PAGE ZERO RELOCATABLE DATA = 0
     NON-SHARED CODE            =90
     NON-SHARED DATA            = 0
     SHARED DATA                = 0
     SHARED CODE                = 0
     ABSOLUTE DATA               =0

     TOTAL PAGE ZERO RELOCATABLE DATA=14
     TOTAL NON-SHARED CODE            =439
     TOTAL NON-SHARED DATA            =0
     TOTAL SHARED DATA                =0
     TOTAL SHARED CODE                =0
     TOTAL ABSOLUTE DATA              =14
```

The first module analyzed in this library, LIBRARY1, is entitled IOPKT.

This module defines entries IOPKT, SPOOL and DIR. Although IOPKT is not externally referenced by any other module in the library, SPOOL is referenced by modules MACR, SYMB and XREF in the library. Likewise, DIR is referenced by SYMB and CLANG. Symbols DOUBLE and WAIT are externally referenced by this module; they are defined as entries in modules MACR and XREF respectively. There are 98 storage words in this module, and they are all non-shared code.

LFE gives two warnings in the next module, MACR. Entry SPOOL is externally referenced, yet this symbol occurs in a previous module (IOPKT). Unless every program using module MACR also binds IOPKT, these two modules must be merged or their positions in the library reversed; otherwise, Binder resolution errors will occur. Likewise, entry ASMVA is externally referenced yet there is no corresponding entry in the library. This will cause a Binder error unless, in every case, a module follows this library that resolves ASMVA. These comments apply also to the remaining P and U errors flagged in this example.

Finally, the module entitled SYMB has an entry defined, DOUBLE, which was also defined in a previous module (MACR). The Binder will flag this as an error unless in every case modules DOUBLE and MACR will not both be bound. The storage totals for all kinds of relocatability are listed at the end of the analysis.

```
XEQ LFE A FORT.LB SINE COSINE)
```

LFE analyzes objects SINE and COSINE of FORT.LB. The analysis is listed on the generic file @OUTPUT.

```
XEQ LFE/L=@LPT/F A/M TRIG1.LB TRIG2.LB)
```

LFE analyzes libraries TRIG1.LB and TRIG2.LB as a complete library. It outputs the listing on the line printer, with a form feed output before the analysis of each module.

(D) Delete Objects from a Library

Format:

XEQ LFE D input/I output/O title *[...]*)

where: input is the name of the unshared library file containing OBs to be deleted.

output is the name to be assigned to the new, reduced library.

title is the title of each module to be deleted from the input library.

Purpose:

This command deletes one or more OBs from an input library, and creates an output library lacking these OBs. The original input library is left intact.

Example:

) XEQ LFE D FORT.LB/I FORT1.LB/O TEMP)

This command deletes the module entitled "TEMP" from library FORT.LB to produce a new library named FORT1.LB.

(I) Insert Objects into a Library

Format:

XEQ LFE I input/I output/O *[object [/F][/C]...]* &)
[title/A object [/F][/C]...]... &)
[title/B object [/F][/C]...]...)

where: input is the name of the library into which OBs are to be inserted.

output is the name to be assigned to the new, expanded library.

object is the name of a file containing an OB to be inserted into the library.

Switches:

/F force-binds this module (sets the force-bind flag); when you specify this library in a BIND command, this OB will be bound whether or not it contains the .FORC pseudo-op.

/C instructs the Binder to ignore a .FORC pseudo-op in this module when you specify this library in a BIND command (clears the force-bind flag); prevents this module from being bound unless it resolves an .EXTN reference.

title is the title of the library module immediately before (/B) or after (/A) which LFE inserts the OBs. LFE inserts OBs named in the command line before a title argument at the beginning of the library.

Purpose:

This command creates a new library consisting of an old library and inserted modules. Modules which you name before a title argument are inserted at the beginning of the new library. Modules following a title argument are inserted either immediately before (/B) or after (/A) the module with that title in the old library. Contents of the input library always remain unchanged.

Examples:

```
) XEQ LFE I LIBRARY1/I LIBRARY2/O FILE1 )
```

The OB in FILE1 becomes the first module in LIBRARY2; OBs in LIBRARY1 follow as the remainder of LIBRARY2. Error messages go to the generic file @OUTPUT.

```
) XEQ LFE I LIBRARY1/I LIBRARY2/O &)
&) .FIVE/A FILE1 FILE2)
```

OBs in FILE1 and FILE2 are inserted immediately after the OB entitled ".FIVE" in LIBRARY1 to form LIBRARY2.

(M) Merge Libraries

Format:

```
XEQ LFE M output/O input1 input...)
```

where: output is the name to be assigned to the new library.

input1 is the name of the first input library.

input represents the names of subsequent libraries that are merged with input1.

Purpose:

This command merges two or more libraries to form a new output library. Contents of input libraries remain intact.

Example:

```
) XEQ LFE M LIBRARY/O LIBRARY1 LIBRARY2 &)
&) LIBRARY3)
```

Libraries LIBRARY1, LIBRARY2, and LIBRARY3 are merged to form a new library named LIBRARY.

(N) Create a Library

Format:

XEQ LFE N output/O object *[/F][/C]...*)

where: output is the name of the new library.

object is the name of each file containing an OB that will go into the new library.

Switches:

/F force-binds this module (sets the force-bind flag); when you specify this library in a BIND command, this OB will be bound whether or not it contains the .FORC pseudo-op.

/C instructs the Binder to ignore a .FORC pseudo-op in this module when you specify this library in a BIND command (clears the force-bind flag); prevents this module from being bound unless it resolves an .EXTN reference.

Purpose:

This command takes one or more OBs and forms a library. You can use this command to form a dummy library consisting of a collection of OB files to be analyzed. These OB files can contain the .PENT pseudo-op even though they are eventually destined to form a shared library file.

Examples:

```
) XEQ LFE N NEWLIB.LB/O FILE.OB FILE1.OB &)
&) FILE2.OB)
```

This command creates a new library, NEWLIB.LB, containing the OBs in FILE, FILE1 and FILE2. This command would work exactly the same if you omitted the extensions ".LB" and ".OB" in the command line.

```
) XEQ LFE N LIB.LB/O OB1/C OB2 OB3/F)
```

This example builds new library LIB.LB, containing OB1, OB2 and OB3. OB1 has the force-bind flag cleared, OB2's flag is not modified, and OB3 has the flag set.

(R) Replace a Library Object

Format:

XEQ LFE R input/I output/O title1 object1 *[/F] [/C]* &)
[titleN objectN[/F][/C]]...)

where: input is the name of the input library.

output is the name of the output library.

title is the title of a library module to be replaced by the object whose filename immediately follows it in the command line.

object is the name of the file containing the object replacing the one whose title is the preceding argument.

Switches:

/F force-binds this module (sets the force-bind flag); when you specify this library in a BIND command, this OB will be bound whether or not it contains the .FORC pseudo-op.

/C instructs the Binder to ignore a .FORC pseudo-op in this module when you specify this library in a BIND command (clears the force-bind flag); prevents this module from being bound until it resolves an .EXTN reference.

Purpose:

This command replaces objects within a library. You specify library objects and replacement objects in pairs. You identify by .TITL the library objects to be replaced, and you identify replacement objects by filename. Contents of the input library remain intact.

Examples:

```
) XEQ LFE R TRIG.LB/I .SINE FILE1 &)
&) TRIG2.LB/O)
```

This command replaces the object entitled ".SINE" in library TRIG.LB with the object in FILE1, and forms a new library, TRIG2.LB.

```
) XEQ LFE R OLDLIB/I NEWLIB/O OLD NEW/F)
```

This command creates library NEWLIB, from the contents of OLDLIB, with object OLD replaced by NEW. The Binder will force-bind object NEW whenever library NEWLIB appears in a BIND command line.

(T) List Library Object Titles

Format:

XEQ LFE *[/F] [/L=listfile]* T input...)

where: listfile is the name of an output file to receive the titles of the objects. If you specify no listing file, then the list appears in the generic file @OUTPUT. If you use the /F switch, a form feed will be output between the title lists of each separate library.

input is the name of a library file whose objects' titles are to be listed.

Purpose:

This command lists the titles of object modules in one or more library files.

LFE prints an asterisk before the title of any module in which the force-bind flag is set.

Example:

```
) XEQ LFE/L=TITLELIST/F T MATH1.LB &)
&) MATH2.LB)
```

This command outputs the titles of objects in two math libraries, MATH1 and MATH2, to a disk file named TITLELIST. A form feed (blank page) separates the lists of MATH1 and MATH2 titles.

(X) Extract Modules From a Library

Format:

XEQ LFE X input title...)

where: input is the name of the library from which copies of modules will be extracted.

title is the title of each module whose copy is to be extracted from the library.

Purpose:

This command extracts copies of modules from a library, and constructs OB files from these copies. Each OB file has the extension ".OB" and bears the name of the module's title. The original input library is left intact.

If an OB file already exists with the name which is given to the extracted module, then no extraction occurs and you get an error message.

Note that since you don't flag the input argument with a /I switch, input must appear as the first argument in the command line.

Example:

```
) XEQ LFE X MATH.LB SINE)
```

This command extracts the module whose title is "SINE", and builds an OB file named "SINE.OB". MATH.LB remains unchanged.

Error Messages

All error conditions output in an error message go to the generic file @OUTPUT. Some error conditions abort the LFE command; other errors permit some processing to continue.

The following error messages indicate that the LFE command has been aborted:

ILLEGAL FUNCTION KEY: key.

The single-letter function key following immediately "LFE" in each command must be A, D, I, M, N, R, T, or X. You supplied some other key.

INPUT FILE NOT A LIBRARY FILE.

LFE required an input library file, yet the input you supplied was not a library.

INPUT FILE NOT SPECIFIED.

An LFE command required an input filename with a /I switch, yet you provided none.

OUTPUT FILE NOT SPECIFIED.

An LFE command required an output filename with a /O switch, yet you provided none.

NOT ENOUGH ARGUMENTS.

Either you gave too few arguments (such as only two arguments in a Merge), or you did not pair the title and object arguments in a Replace command.

SWITCH MAY NOT APPEAR TWICE: switch.

This message appears when the named switch appears more than once in a single LFE commnand.

TOO MANY ARGUMENTS.

If you need to name more arguments (as in creating a library with more OBs), issue several commands and combine their results (for example, create several small libraries and then merge them into one large library).

TOO MANY SYMBOLS IN THE LIBRARY.
ANALYSIS IS NOT POSSIBLE

This condition arises when the combined length of all unique symbols in modules being analyzed exceeds 131,072 characters. It also arises if memory is sufficient for LFE to build itself a temporary table. You can correct this condition by making the library smaller (perhaps by extracting some modules), or by increasing the amount of memory available to the process in which LFE is running.

The following error messages indicate that the LFE has been able to execute part of the command:

BLOCK NUMBERS NOT IN SEQUENCE,
FILE:pathname

This error message indicates that the named file was either not an OB file (which it had to be), or it had gross structural errors (perhaps it was stored on a bad disk). Whatever the case, LFE does not process file named in the command sequence.

FILE ALREADY EXISTS: pathname

On an eXtract, a file to be created already exists. LFE ignores the argument.

NOT A LIBRARY FILE: pathname

This message is given only when you attempt to execute an Analyze or Title command, and one or more (but not all) of the input files was supposed to be library, yet was not. LFE ignores each offending file.

NOT AN .OB FILE: pathname

An OB module with title name was required, yet you supplied none. LFE ignores the argument.

OBJECT MODULE NOT FOUND: pathname

An input file did not contain a requested module with the title name. LFE ignores the object module (and any associated arguments) in executing the command. The following LFE commands can detect this error condition: Analyze, Delete, Insert, Replace, and eXtract.

End of Chapter

Appendix A
Library File Format

Most users will never be aware of internal library file structure. You need to understand this appendix only if:

1) you plan to build libraries without using LFE, or
2) you intend to read the headers (as the Binder does).

Otherwise, you may disregard this appendix.

Library files are made from three types of blocks. The first block is always a library start block. This is followed by object modules, and the last object module is followed by a library end block.

Library Start Block

The library start block consists of a fixed 4-word header, followed by a series of module descriptors. There are as many module descriptors as there are OB modules in the library:

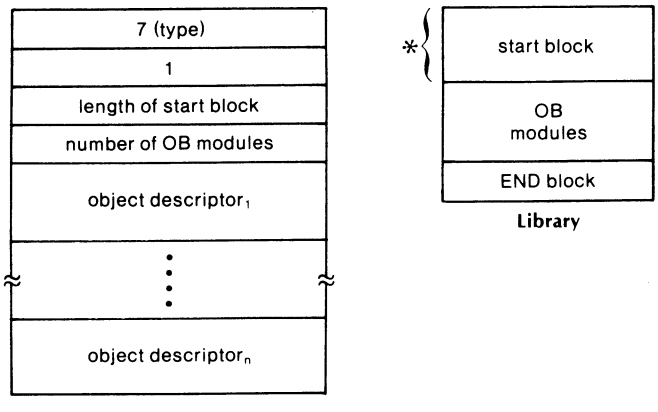


*Figure A-1. Library Start Block Overview**

The first word in the block contains the block type number, 7. The second word contains 1, since the library start block is always the first block in a library. (Note that sequence numbers in object module blocks within the library are unchanged from what they were when the objects were stand-alone OB files.) The third word of the start block contains the word length of the entire start block. The fourth word lists the number of OB modules contained within the entire library.

Each object descriptor is a variable-length block describing a single object module. Each descriptor consists of a fixed 6-word header, followed by a variable number of variable-length sections describing entries in the object module.

Object descriptor structure is shown in Figure A-2:

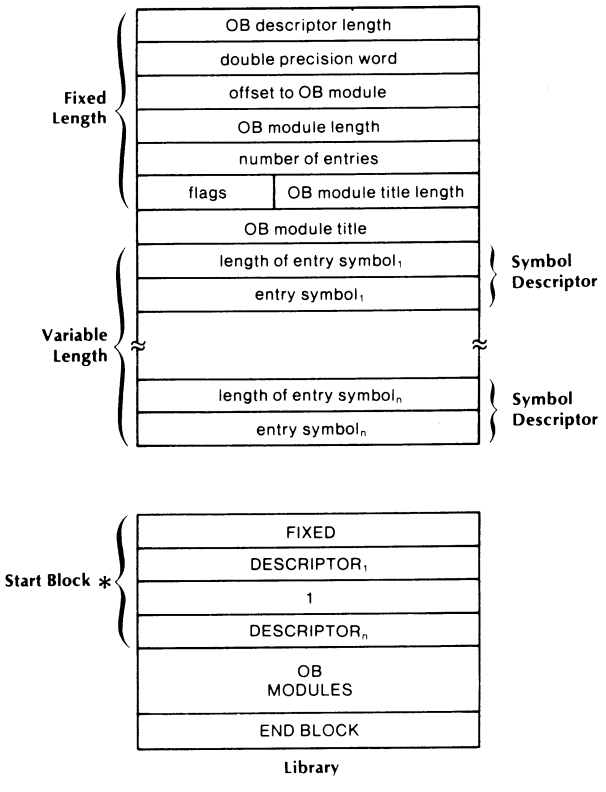


*Figure A-2. Object Descriptor Structure**

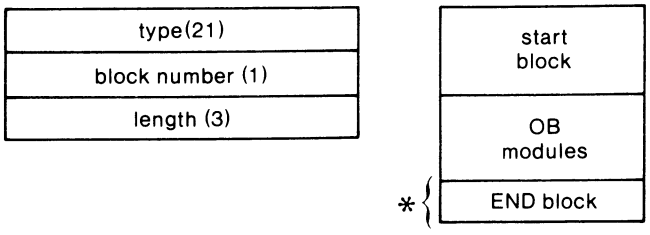
The second and third words of the object descriptor contain a double-precision word offset to the object module identified by this descriptor. This offset starts at the first OB module; thus the offset is 0, 0 for the first OB descriptor. If the first OB is *n* words long, and *n* is less than a 16-bit integer, then the second descriptor offset value is 0, *n*. Following this offset are the length of the OB module and the number of entries within this module. Entries include arguments of .ENT, .ENTO and .PENT pseudo-ops (bear in mind that .PENT pseudo-ops will be found only if you build a dummy library for analysis; see Chapter 2, "A" command).

Following the entry count, the OB flags make up the first byte of the next word. The OB flags include the *force-bind flag*. You can set this flag, which force-binds a module, either by including the .FORC pseudo-op in the module, or by setting the LFE /F switch, described earlier. You can clear the flag, and restore normal Binder loading of the module, with the LFE /C switch. The *AOS Macroassembler* manual describes .FORC. The second byte of this word is the OB title length.

The next word contains the title name string. Following this are as many symbol descriptors as there are entry symbols to be described. Each descriptor consists of a one-word symbol length followed by the entry symbol string itself. Following the last descriptor for the last OB module is the start of the first OB module in the library.

Library End Block

Following the end of the last OB module is a three-word block, the library end block. This fixed-length block consists of three words as shown in Figure A-3.



SD-00304

*Figure A-3. Library End Block Structure**

The first word, block type, contains "21_8 ". The second word, sequence number, contains "1"; note that the original sequence numbers in each OB module do not differ from the values they had in each stand-alone OB file. Since the block is three words long, the block length (given in the third word) is three.

End of Appendix

Index

Within this index, the letter "F" after a page number means "and the following page."

A (Analyze) command 2-2F
analyze symbols (LFE) 2-2F
  *
  AC
  EN
  EO
  NC
  P
  PN
  T
  U
  XT
analyzing an OB 2-6

Binder 1-1F, 2-4, 2-6
block (library)
  end A-1F
  number sequence 2-8
  start A-1

CLI
  command line extension (&) 2-1
  LFE command line (XEQ LFE) 2-1

D (delete) command 2-4
dummy library 2-6, A-2

.ENT pseudo-op 1-1F, A-2
.ENTO pseudo-op 1-1, A-2
errors
  Binder 1-2, 2-3
  LFE message 2-8
external definition see .EXTN, .EXTD
.EXTD pseudo-op 1-1
.EXTN pseudo-op 1-1F, 2-4, 2-6
eXtract command 2-7

filename extensions 2-1
.FORC pseudo-op 2-4, 2-6, A-2
force-bind flag 2-4, 2-6
  definition of A-2
form feed
  inserting (/F switch) 2-2, 2-7

I (Insert) command 2-4F

LFE
  command descriptions 2-1 to 2-8
  definition of 1-1
  functions 1-1
  function letter 2-1
  operating procedures 2-1
library
  creating 2-6
  end block 1-1, A-1F
  modifying 2-2 to 2-7
  object modules 1-1
  shared 1-1, 2-6
  start block 1-1, A-1
  unshared 1-1

M (Merge) command 2-5
MASM assembler 1-1
multiply-defined symbol 2-2

N (Create a library) command 2-6
new-line character 2-1
NREL 1-1

.PENT pseudo-op 1-1, 2-6, A-2
phase warning 2-2F

R (Replace) command 2-6

shared code 1-1, 2-2
switches, definition of 2-1
symbols
  maximum for LFE 2-8
  notation conventions 2-1
  undefined 2-2F

T (List titles) command 2-7

undefined symbol 2-2F
unshared code 1-1, 2-2

X (Extract) command 2-7

ZREL 1-1

DataGeneral

SOFTWARE DOCUMENTATION
REMARKS FORM

Document Title	Document No.	Tape No.

SPECIFIC COMMENTS: List specific comments. Reference page numbers when applicable. Label each comment as an addition, deletion, change or error if applicable.

GENERAL COMMENTS: Also, suggestions for improvement of the Publication.

FROM:

Name	Title	Date

Company Name

Address (No. & Street)	City	State	Zip Code

FOLD DOWN FIRST FOLD DOWN

FIRST
CLASS
PERMIT
No. 26
Southboro
Mass. 01772

BUSINESS REPLY MAIL

No Postage Necessary If Mailed In The United States

Postage will be paid by:

Data General Corporation

Southboro, Massachusetts 01772

ATTENTION: Software Documentation

FOLD UP SECOND FOLD UP

STAPLE