PART 10

THE DOS/BATCH LIBRARIAN

LIBR

PART 10

CHAPTER 1

INTRODUCTION TO LIBRARIAN

The DOS/BATCH Librarian (LIBR) is a system program that provides facilities for creating, modifying, deleting, and listing the contents of libraries. A library is a file consisting of one or more object modules which are accessible through a library directory. An object module is the binary output of the DOS/BATCH Assembler or FORTRAN Compiler.

LIBR is a valuable program for the DOS/BATCH user because:

1. It eliminates having separate directory entries in a User File Directory (UFD) for each object module.
2. It expedites the linking process in conjunction with the Linker's library search capabilities.
3. It allows for standardization and controlled updating of frequently used routines, e.g., FORTRAN cosine routine.

The user controls the operating of LIBR through command strings typed on the keyboard. Command strings may specify devices, library and object module names, and switches that indicate the LIBR operation desired. The user can direct LIBR to:

a. Create a library,
b. Update a library,
c. Insert one or more object modules in a library,
d. Replace one or more object modules in a library,
e. List the contents of a library,
f. Delete one or more object modules from a library, and
g. Delete an entire library.

A directory listing of the object modules of a library can be obtained by merely specifying the device on which the directory is to appear and the name of the library.

The flexibility of LIBR enables the user to specify certain combinations of operations in a single command string. For example, a library can be modified, renamed, and listed in one command string.

The switch options which direct LIBR operations are:

Switch	Operation
/D	Delete object module
/DL	Delete input library
/I	Insert object module
/LO	List object modules
/R	Replace object module

If the user types an illegal command string (e.g., illegal format, excessive switches, nonexistent file or object module, etc.), LIBR prints an appropriate error message on the teleprinter.

The following discussion assumes that the reader is familiar with the DOS/BATCH Monitor, EDIT Text Editor, MACRO Assembler, ODT-11R Debugging Program, and LINK Linker.

PART 10

CHAPTER 2

OPERATING PROCEDURES

2.1 CALLING LIBR

The Librarian is called into core by typing the RUN command in response to the Monitor's dollar sign. The Librarian is stored as LIBR; when called it prints its name, version number, and a # sign, and then waits for the user to issue a command string. For example:

```
$RUN LIBR
LIBR  Vxxx
#
```

2.2 COMMAND STRING

When the Librarian is in core and has printed the # sign, it is ready to accept a user command string. The format is:

```
output library,listing file<input library, input file(s)
```

LIBR performs two passes over all input files. For non-file-structured devices (e.g., paper tape reader), the system informs the user to reload the device for the second pass. For file-structured devices, both passes are performed automatically without requiring user intervention.

2.2.1 Creating a Library

Creating a library requires an output library specification and the input files to be placed in the library. The input files must be preceded by a comma; the input files are not considered to be an input library.

```
output library(,listing file)<,input file(s)
```

A library is created on the device specified in the output library specification and named as specified. The listing file specification is optional and, if it is present, the contents of the output library will be listed. The format of the listing is discussed under the heading "Listing a Library."

An input library need not appear, but the comma and one or more input files must appear (each of which contains one or more object modules). For example:

```
#DT1:FIL.LIB<,FIL.1,FIL.2
```

creates a library named FIL.LIB on DECtape 1. The library consists of all object modules in FIL.1 and FIL.2 in that order, and in the order in which the object modules appear in their respective input files.

A restriction placed on the user by the Librarian is that it accepts a command specification of only one line. It is therefore suggested when creating a Library from a large number of object modules, that the object modules be concatenated into a reasonable number of files.[1] Because the length of the Librarian's command string is restricted to one teleprinter line, creating libraries from many object modules must be accomplished with an intermediate step. The step is to first concatenate the object modules in their desired order using PIP. Having done this, the command string to the Librarian is reduced to one input file which contains the concatenated object modules. An example of this follows.

```
$RU PIP
#FILE.001<FILE.A,FILE.B,FILE.C
#FILE.002<FILE.D,FILE.E,FILE.F
#FILE.003<FILE.G,FILE.H,FILE.I
#↑C
.KI
$RU LIBR
#NAME.LIB<,FILE.ØØ1,FILE.ØØ2,FILE.ØØ3
$RU PIP
#FILE.001,FILE.002,FILE.003/DE
```

The Librarian does not supply default extensions (e.g., .OBJ); the user must remember to supply them.

2.2.2 Updating a Library

To update a library, both output and input library specifications are required, along with input files that may consist of one or more object modules.

Libraries can be updated in one of three ways:

1. Delete one or more object modules.
2. Insert one or more object modules
3. Replace one or more object modules.

Listing the updated library is optional.

---

[1]Note that an input file of concatenated object modules differs from a library in that it does not have a directory of the object modules it contains.

2.2.2.1 To Delete One or More Object Modules

```
output library(,listing file)<input library/D:M1:...Mn
```

The output library is created as a result of deleting the object modules named M1..., Mn from the input library. The listing file is optional.

The name associated with an object module is the symbol assigned to the module by the MACRO Assembler's .TITLE directive.

The object modules to be deleted must appear in the same order as they appear in the library; their order can be determined from the listing.

For example:

```
#DT1:LIBR.1<DT2:LIBR.0/D:M1:M2
```

creates a library named LIBR.1 on DECtape 1 as a result of deleting the object modules M1 and M2 from LIBR.0 on DECtape 2.

Insert and/or Replace operations cannot accompany a Delete request.

2.2.2.2 To Insert One or More Object Modules

```
output library(,listing file)<input library,input file(s)/I(:v)
```

The output library is created as a result of combining the object modules of the input file with the input library. If v is specified, the object module(s) in the input file are inserted starting at position v, otherwise, they are inserted at the end. v is treated as a decimal integer, and is always relative to the input library.

If more than one input file is specified for insertion, the positions at which the files are to be inserted must appear in ascending order. For example:

```
#DT1:LIBR.1<DT2:LIBR.0,FIL.1/I:2,FIL.2/I
```

creates an output library on DECtape 1 as a result of inserting the object modules of FIL.1 into LIBR.0, beginning at position 2, and then inserting the object modules of FIL.2 into LIBR.0 at the end.

Insert and Replace operations can appear in the same command as long as the order restriction is observed.

2.2.2.3 To Replace One or More Object Modules

```
output library(,listing file)<input library, input file(s)/R
```

The output library is created as a result of replacing the object module(s) in the input library by those in the input file(s).

The object modules to be replaced must have the same name as those replacing them, and they must be in the same order. For example:

```
#DT1:LIBR.1<DT2:LIBR.0,FIL.1/R,FIL.2/R
```

creates the output library LIBR.1 on DECtape 1 as a result of replacing the object modules in the input library LIBR.0 with those in FIL.1 and FIL.2.

2.2.3 Listing a Library

To list the contents of a library requires only a listing file specification and an input library specification. The listing file specification must be preceded by a comma, so that it is not interpreted as an output library.

```
#,listing file(/LO)<input library
```

The directory of the input library is listed. The presence of the /LO switch directs the Librarian to produce an object module listing. This is a means of double-checking the accuracy of the library; the directory listing must correspond exactly to the object module listing.

The output library is listed if one was created; otherwise, the input library is listed. The format of the listing is:

```
Library Name and Extension
Decimal Order Number     Object Module Name (first module)
          .                           .
          .                           .
          .                           .
Decimal Order Number     Object Module Name (last module)
```

For example, if LIB.1 contains object modules M1, M2, and M3 in that order, the command:

```
    #,LP:FIL.LST<DT1:LIB.1
```

produces on the line printer:

```
    LIBR  Vxxx      (xxx is the LIBR version number)
        FIL          .LST      date      time
        SEQ.         NAME      VERSION
        00001        M1        021
        00002        M2        031
        00003        M3        041
```

If the /LO switch appears, for example:

```
    #,LP:FIL.LST/LO<DT1:LIB.1
```

the listing above is followed by a form feed and a similar table, except that the name of the second table is always OBJMOD.LST.

The library name that is printed at the head of the listing is the name specified in the listing file specification. For example:

```
    #LIB.ABC,LP:NAME<,FIL.1,FIL.2
```

The listing is titled NAME, not the newly created library LIB.ABC. When the listing file name is not specified, then the listing is titled with the name of the newly created file (LIB.ABC).

2.2.4 Naming Libraries

The output library can have the same name as the input library. In this case, however, the input library has an implied /DL; that is, the input library is deleted. For example:

```
    #LIB.1<LIB.1/D:OM1
```

is the same as:

```
    #LIB.TMP<LIB.1/D:OM1/DL
```

and then rename LIB.TMP to LIB.1

NOTE

The user must never name a Library LIBR.TMP. This name is reserved for use by the Librarian.

2.2.5 Legal File Specification Combinations

In a command string, various combinations of file specifications are possible; legal combinations and their operation are shown below.

Operation		Output Library	Listing	Input Library	Input File(s)	Note
Insert or Replace Object Modules; List Output Library	(1)	P	P	P	P	SE if/D on input library
Delete Object Modules; List Output Library	(2)	P	P	P	NP	SE if/D not on input library
Create Library; List Output Library	(3)	P	P	NP	P	SE if switch on input file
List Input Library	(4)	NP	P	P	NP	SE if /D on input library
Same as (1) except no listing	(5)	P	NP	P	P	Same as (1)
Same as (2) except no listing	(6)	P	NP	P	NP	Same as (2)
Same as (3) except no listing	(7)	P	NP	NP	P	Same as (3)

Legend: P = present
NP = not present
SE = syntax error

PART 10

CHAPTER 3

EXAMPLES

Example 1

Assume FIL.1 contains object modules OM1, OM2, and OM3 in that order, FIL.2 contains OM4 and OM5 in that order, FIL.3 contains OM5 and OM3 in that order, and FIL.4 contains OM6. Then:

```
#LIB.1,LP:LIB.1<,FIL.1,FIL.2
```

Creates a library named LIB.1 containing object modules OM1, OM2, OM3, OM4, and OM5 in that order. The listing appears on the line printer as:

```
LIBR  Vxxx
    LIB         .1        date         time
    SEQ.        NAME      VERSION
    00001       OM1       025
    00002       OM2       032
    00003       OM3       041
    00004       OM4       054
    00005       OM5       027
```

Files FIL.1 and FIL.2 remain unaltered. The listing is produced after all other actions have been performed. Consequently,

```
#LIB.1,LIB.1<,FIL.1,FIL.2
```

produces an error message (file already exists) when an attempt is made to write the listing to the disk.

Example 2

Using the assumption above:

```
#LIB.2<LIB.1/D:OM1:OM4
```

creates a library named LIB.2 containing object modules OM2, OM3, and OM5 in that order. No listing is produced and LIB.1 is not deleted.

Example 3

```
    #LIB.3<LIB.2/D:OM3:OM2
```

produces an error message because the modules to be deleted are not in the order in which they appear in the library.

Example 4

The command string:

```
    #,LP:LIB2.LS/LO<LIB.2
```

produces a listing on the line printer which appears as:

```
    LIBR   Vxxx
         LIB2        .LS         date        time
         SEQ.        NAME        VERSION
         00001       OM2         032
         00002       OM3         041
         00003       OM5         027
```

Example 5

The command string:

```
    #LIB.3<LIB.2/DL,FIL.4/I:2
```

creates a library named LIB.3 containing OM2, OM6, OM3 and OM5 in that order. No listing is produced and LIB.2 is deleted.

Example 6

The command string:

```
    #LIB.4<LIB.3,FIL.4/R
```

creates a library named LIB.4, which is really LIB.3 with OM6 replaced (i.e., removed from LIB.3 before creating LIB.4).

Example 7

The command string:

```
#LIB.5<LIB.4,FIL.3/R
```

produces an error message because the object modules in FIL.3 are not in the same order as in LIB.4.

Example 8

The command string:

```
#LIB.5<LIB.3/DL,FIL.4/I
```

creates a library named LIB.5 containing OM2, OM6, OM3, OM5 and OM6 in that order. No listing is produced and LIB.3 is deleted. Note that a library can contain multiple copies of the same object module, e.g., two OM6 modules, as above.

Example 9

The command string:

```
#LIB.6<LIB.5/D:OM6
```

creates a library named LIB.6 containing OM2, OM3, OM5 and OM6 in that order. No listing is produced and LIB.5 is not deleted. When a library contains multiple copies of the same object module, the copies are deleted one at a time in their order of occurrence.

Example 10

To delete all occurrences of OM6, the command string should be either:

```
#LIB.6<LIB.5/D:OM6:OM6
```

or

```
#LIB.6<LIB.5/D:OM6/D:OM6
```