LIB Reference Manual

Microsoft® LIB

Library Manager

Reference Manual

Microsoft Corporation

If you have comments about the software or this manual, complete the Software Problem Report at the back of this manual and return it to Microsoft Corporation.

Document Number 8440-300-00

Contents

Chapter 1
Introduction

1.1 About This Manual

This manual describes the Microsoft® LIB Library Manager. This utility enables you to create and maintain your own libraries of useful functions. You can use these libraries to customize the runtime support available to your programs.

Notational Conventions

The following notational conventions are used throughout this manual:

italics

Italics mark the places in command line and option specifications and in the text where specific terms appear in an actual command. For example, in

```
/W number
```

number is italicized to indicate that this is a general form for the /W option. In an actual command, the user supplies a particular number for the placeholder *number*.

Italics are also used when referring to specific identifiers supplied for functions, variables, types, and labels.

Occasionally, italics are used to emphasize particular words in the text.

brackets [ ]

Brackets enclose optional fields in command line and option specifications. For example, in

```
/D identifier[=[string]]
```

the brackets around the phrase "=[*string*]" indicate that you are not

required to supply this phrase when you use the /D option. Furthermore, within this phrase, *string* is enclosed in brackets. Thus, when you give an equal sign (=), the *string* is optional. Notice, however, that you may not give a *string* without first giving the equal sign.

ellipses... — Ellipses following an item indicate that more items having the same form may appear. For example, in

LIB *library* [*/pagesize*] *operations*...

the ellipses indicate that one or more *operations* are allowed.

CAPITALS — Capital letters are used for the names of files, directories, environment variables, and manifest constants and macros. Commands typed at the MS™-DOS level are also capitalized.

SMALL CAPITALS — Small capital letters are used for the names of keys and key sequences, such as RETURN and CONTROL-C.

"Quotation marks" — Quotation marks set off terms defined in the text. For example, the term "module" appears in quotation marks the first time it is defined.

Quotation marks are also used to set off program fragments and to refer to command line prompts.

```
programming
examples
```

Programming examples are displayed without proportional spacing so that they look like the programs you create with a text editor.

1.2 Managing Libraries

The Microsoft LIB Library Manager is a utility designed to help you create, organize, and maintain runtime libraries. Runtime libraries are collections of compiled or assembled functions that provide a common set of useful routines. Any program can call a runtime routine, exactly as if the function were included in the program. The program is linked with the runtime library file and the call to the runtime routine is resolved by finding the routine in the library file.

Runtime libraries are created by combining separately compiled object files into one library file. Library files are usually identified by their ".LIB" extension, although other extensions are allowed.

In addition to accepting MS-DOS™ object files and library files, Microsoft LIB accepts 286 XENIX™ archives and INTEL®-style libraries. You can add the contents of a 286 XENIX archive or an INTEL-style library to an MS-DOS library by using the append operator (+).

Once an object file is incorporated into a library, it becomes an object "module." LIB makes a distinction between object files and object modules: an object "file" exists as an independent file while an object "module" is part of a larger library file. An object file can have a full pathname, including a drive designation, directory pathname, and filename extension (usually ".OBJ"). Object modules have just a name. For example, "B:\RUN\SORT.OBJ" is an object filename, while "SORT" is the corresponding object module name.

Using LIB, you can create a new library file, add object files to an existing library, delete library modules, replace library modules, and create object files from library modules. LIB also lets you combine the contents of two libraries into one library file.

The command syntax is straightforward, and LIB prompts you for responses. Once you have learned how LIB works and what its prompts mean, you can use one of the alternate methods of invoking LIB, described in Sections 2.5, "Using the Command Line," and 2.6, "Using a Response File." These alternatives let you give LIB commands without waiting for the LIB prompts.

1.3 Overview of LIB Operation

You can perform a number of library management functions with Microsoft LIB. LIB can

- Create a library file
- Delete modules
- Extract a module and place it in a separate object file
- Extract a module and delete it
- Append an object file as a module of a library, or append the contents of a library
- Replace a module in the library file with a new module
- Produce a listing of all public symbols in the library modules

For each library session, LIB first reads and interprets the user's commands. It determines whether a new library is being created or an existing library is being examined or modified.

Deletion and extraction commands (if any) are the first commands processed. LIB does not actually delete modules from the existing file. Instead, it marks the selected modules for deletion, creates a new library file, and copies only the modules *not* marked for deletion into the new library file.

Next, LIB processes any addition commands. Like deletions, additions are not performed on the original library file. Instead, the additional modules are appended to the new library file. (If there were no deletion or extraction commands, a new library file is created in the addition stage by copying the original library file.)

As LIB carries out these commands, it reads the object modules in the library, checks them for validity, and gathers the information necessary to build a library index and a listing file. The library index is used by the linker to search the library.

The listing file contains a list of all public symbols in the index and the names of the modules in which they are defined. LIB produces the listing file only if you ask for it during the library session.

LIB never makes changes to the original library; it copies the library and makes changes to the copy. Thus, when you terminate LIB for any reason, you do not lose your original file. It also means that when you run LIB, enough space must be available on your disk for both the original library file and the copy.

When you modify a library file, LIB gives you the option of specifying a different name for the file containing the modifications. If you use this option, the modified library is stored under the name you give, and the original, unmodified version is preserved under its own name. If you choose not to give a new name, LIB gives the modified file the original library name, but keeps a backup copy of the original library file. This copy has the extension ".BAK" instead of ".LIB".

Example

```
LIB LANG HEAP+HEAP;
```

This command first deletes the library module HEAP from the library file LANG.LIB, then adds the file HEAP.OBJ as the last module in the library. The same command could be given as

```
LIB LANG+HEAP HEAP;
```

The effect is the same because delete operations are always carried out before append operations, regardless of the order of the operations in the command line. This order of execution prevents confusion in LIB when a new version of a module replaces an old version in the library file.

After the library is modified, the modified file is written back to LANG.LIB. A copy of the original LANG.LIB is stored in LANG.BAK.

Chapter 2
Running LIB

LIB requires two types of input: a command to start LIB and responses to command prompts. Start LIB at the MS-DOS command level by typing

```
LIB
```

LIB prompts you for the input it needs by displaying the following four messages, one at a time. LIB waits for you to respond to each prompt, then prints the next prompt.

```
Library name:
Operations:
List file:
Output library:
```

The responses you can make to each prompt are described in Sections 2.1 through 2.4.

Once you understand the LIB prompts and operations, you may want to use one of the two alternate methods of running LIB. The command line method lets you type all commands, options, and filenames on the line used to start LIB. With the response file method, you create a file that contains all the necessar commands, then tell LIB where to find that file.

Both of the alternate methods require that you understand how LIB works and what your responses to its prompts mean. For this reason it is recommended that you allow LIB to prompt you for responses until you are comfortable with its commands and operations.

2.1 Library Name Prompt

At the "Library name" prompt, give the name of the library file you want. Usually library files are named with the ".LIB" extension. You can omit the ".LIB" extension when you give the library filename since LIB assumes that the filename extension is ".LIB".

If your library file does not have the ".LIB" extension, be sure to include the extension when you give the library filename. Otherwise, LIB cannot find the file.

Pathnames are allowed with the library filename, so you can give LIB the pathname of a library file in another directory or on another disk.

Because LIB manages only one library file at a time, only one filename is allowed in response to this prompt. There is no default response, so LIB produces an error message if you do not give a filename.

If you give the name of a library file that does not exist, LIB displays the prompt

```
Library file does not exist.  Create?
```

Type "y" (yes) to create the library file. If you answer "n" (no), LIB returns control to the MS-DOS level.

If you type a library filename and follow it immediately with a semicolon (;), LIB performs only a consistency check on the given library. A consistency check tells you whether all the modules in the library are in usable form. LIB prints a message only if it finds an invalid object module; no message appears if all modules are intact.

You can also set the library page size following this prompt. See Section 3.9, "Setting the Library Page Size," for details.

2.2 Operations Prompt

Following the "Operations" prompt, you can type one of the command symbols for manipulating modules (+, −, −+, *, −*) followed immediately (no space) by the module name or the object filename. You can specify more than one operation following this prompt, in any order. The default for the "Operations" prompt is no changes.

When you have a large number of modules or files to manipulate (more than can be typed on one line), type an ampersand (&) as the last symbol on the line, immediately before the carriage return. The ampersand must follow a filename; you cannot give an operator as the last character on a line to be continued. The ampersand causes LIB to repeat the "Operations" prompt, allowing you to specify more operations and names.

The following list describes the command symbols and their meanings and uses:

Command Symbol	Meaning and Use
+	The plus sign appends an object file as the last module in the library file. Give the name of the object file immediately after the plus sign. You can use pathnames for the object file. LIB automatically supplies the ".OBJ" extension, so you can omit the extension from the object filename. You can also use the plus sign to combine two libraries. When you give a library name after the plus sign, a copy of the contents of the given library is added to the library file being modified. You must include the ".LIB" extension when you give a library filename. Otherwise, LIB uses the default ".OBJ" extension when it looks for the file.
−	The minus sign deletes a module from the library file. Give the name of the module to be deleted immediately after the minus sign. A module name has no pathname and no extension.
−+	Type a minus sign followed by a plus sign to replace a module in the library. Give the name of the module to be replaced after the replacement symbol. Module names have no pathnames and no extensions. To replace a module, LIB first deletes the given module, then appends the object file

having the same name as the module. The object file is assumed to have an ".OBJ" extension and to reside in the current working directory.

* Type an asterisk followed by a module name to copy a module from the library file into an object file of the same name. The module remains in the library file. When LIB copies the module to an object file, it adds the ".OBJ" extension and the drive designation and pathname of the current working directory to the module name to form a complete object filename. You cannot override the ".OBJ" extension, drive designation, or pathname given to the object file, but you can later rename the file or copy it to whatever location you like.

\-* Use the minus sign followed by an asterisk to move an object module from the library file to an object file. This operation is equivalent to copying the module to an object file, as described above, then deleting the module from the library.

2.3 List File Prompt

After the "List file" prompt, you can give a filename for a cross-reference listing file. You can specify a full pathname for the listing file to cause it to be created outside your current working directory. You can give the listing file any name and any extension. LIB does not supply a default extension if you omit the extension.

A cross-reference listing file contains two lists. The first is an alphabetical listing of all external (public) symbols in the library. Each symbol name is followed by the name of the module in which it is referenced.

The second list is a list of the modules in the library. Under each module name is an alphabetical listing of the public

symbols referenced in that module. The default when you omit the response to this prompt is the special filename NUL., which tells LIB *not* to create a listing file.

2.4 Output Library Prompt

After the "Output library" prompt, you can give the name of a new library file to create with the specified modifications. The default is the current library filename; the original, unmodified library is saved in a library file with the same name but with a ".BAK" extension replacing the ".LIB" extension. This prompt appears only if you specify modifications to the library following the "Operations" prompt.

2.5 Using the Command Line

The command line method of starting LIB has the form

LIB *library* [*/pagesize*] [;] *operations*... [, *listing*, *newlibrary*]

The entries following LIB are responses to the LIB command prompts.

The *library* entry, with the optional */pagesize* specification, corresponds to the "Library name" prompt. If you want LIB to perform a consistency check on the library, follow the *library* entry with a semicolon (;).

The *operations* entries are any of the operations allowed following the "Operations" prompt. The *listing* entry, if you include it, tells LIB to create a listing file with the given name. The *newlibrary* entry, if it appears, is the name of the revised library.

If you want to create a cross-reference listing, the name of the listing file must be separated from the last *operations* entry by a comma. If you give a filename in the new library field, the library name must be separated from the listing filename or the last *operations* entry by a comma.

You can use a semicolon after any entry but the first to tell LIB to use the default responses for the remaining entries. The semicolon should be the last character on the command line.

Examples

```
1.  LIB LANG +HEAP;

2.  LIB C;

3.  LIB LANG,LCROSS.PUB
```

The first command instructs LIB to replace the HEAP module in the library LANG.LIB. LIB first deletes the HEAP module in the library, then appends the object file HEAP.OBJ as a new module in the library. The semicolon command symbol at the end of the command line tells LIB to use the default responses for the remaining prompts. This means that no listing file is created and that the changes are written back to the original library file instead of creating a new library file.

The second command causes LIB to perform a consistency check of the library file C.LIB. No other action is performed. If LIB displays any consistency errors it finds and returns to the operating system level. The third command tells LIB to perform a consistency check of the library file LANG.LIB, then output a cross-reference listing file named LCROSS.PUB.

2.6 Using a Response File

The command to start LIB with a response file has the form

```
LIB @response-file
```

The *response-file* field is the name of a response file. The *response-file* name may be qualified with a drive and directory specification to name a response file from a directory other than the current working directory.

Before you use this method you must set up a response file containing answers to the LIB prompts. This method lets you conduct the library session without typing responses at the keyboard.

A response file has one text line for each prompt. Responses must appear in the same order as the command prompts appear. Use command symbols in the response file the same way you would for responses typed on the keyboard.

When you run LIB with a response file, the prompts are displayed along with the responses from the response file. If the response file does not contain answers for all the prompts, LIB uses the default responses.

Example

```
SLIBC
+CURSOR+HEAP HEAP*FOIBLES
CROSSLST
```

This response file causes LIB to delete the module HEAP from the SLIBC.LIB library file; extract the module FOIBLES and place it in an object file named FOIBLES.OBJ; and then append the object files CURSOR.OBJ and HEAP.OBJ as the last two modules in the library. Finally, LIB creates a cross-reference file named CROSSLST.

2.7 Extending Lines

If you have many operations to perform during a library session, use the ampersand (&) command symbol to extend the operations line. Give the ampersand symbol after an object module or object filename; do not put the ampersand between an operations symbol and a name.

If you use the ampersand with the prompt method of invoking LIB, the ampersand will cause the "Operations" prompt to be repeated, allowing you to type in more operations. With the response file method, you can use the ampersand at the end of a line and then continue typing operations on the next line.

2.8 Terminating the Library Session

At any time, you can use CONTROL-C to terminate a library session. If you type an incorrect response, such as a wrong or incorrectly spelled filename or module name, you must enter CONTROL-C to exit LIB. You can then restart the program.

2.9 Selecting Default Responses to Prompts

After any entry but the first, use a single semicolon (;) followed immediately by a carriage return to select default responses to the remaining prompts. You can use the semicolon command symbol with the command line and response file methods of invoking LIB, but it is not really necessary, since LIB supplies the default responses wherever you omit responses.

The default response for the "Operations" prompt is no operation. The library file is unchanged.

The default response for the "List file" prompt is the special filename NUL., which tells LIB not to create a listing file.

The default response for the "Output library" file is the current library name. This prompt appears only if you specify at least one operation following the "Operations" prompt.

Chapter 3
Library Tasks

This section summarizes the library management tasks you can perform with LIB.

3.1 Creating a Library File

To create a new library file, simply give the name of the library file you want to create following the "Library name" prompt. LIB supplies the ".LIB" extension.

The name of the new library must not be the name of an existing file, or else LIB will assume you want to modify the existing file. When you give the name of a library file that does not currently exist, LIB displays the following prompt.

```
Library file does not exist.  Create?
```

Type "yes" to create the file, "no" to terminate the library session.

You can specify a page size for the library when you create it. The default page size is 16 bytes. See Section 3.9, "Setting the Library Page Size," for a discussion of this option.

Once you have given the name of the new library file, you can insert object modules into the library by using the add operation (+) following the "Operations" prompt. You can also add the contents of another library if you wish. These options are discussed in Sections 3.3, "Adding Library Modules," and 3.6, "Combining Libraries."

3.2 Modifying a Library File

You can modify an existing library file by giving the name of the library file following the "Library name" prompt. All operations you specify following the "Operations" prompt are performed on that library.

However, LIB lets you keep both the unmodified library file and the newly modified version, if you like. You can do this by giving the name of a new library file following the "Output library" prompt. The modified library file is stored under the new library filename, while the original library file is preserved unchanged.

If you don't give a filename following the "Output library" prompt, the modified version of the library file replaces the original library file. Even in this case, LIB saves the original, unmodified library file. The unmodified library file has the extension ".BAK" instead of ".LIB". Thus, at the end of the session you have two library files: the modified version with the ".LIB" extension and the original, unmodified version with the ".BAK" extension.

3.3 Adding Library Modules

Use the plus sign (+) following the "Operations" prompt to add an object module to a library. Give the name of the object file to be added, without the ".OBJ" extension, immediately after the plus sign.

LIB strips the drive designation and the extension from the object file specification, leaving only the basename. This becomes the name of the object module in the library. For example, if the object file B:\CURSOR.OBJ is added to a library file, the name of the corresponding object module is "CURSOR".

Object modules are always added to the end of a library file.

3.4 Deleting Library Modules

Use the minus sign (−) following the "Operations" prompt to delete an object module from a library. Give the name of the module to be deleted immediately after the minus sign. A module name has no pathname and no extension; it is simply a name, like "CURSOR".

Replacing Library Modules

Use a minus sign followed by a plus sign (−+) to replace a module in the library. Give the name of the module to be replaced after the replacement symbol (−+). Remember that module names have no pathnames and no extensions.

To replace a module, LIB first deletes the given module, then appends the object file having the same name as the module. The object file is assumed to have an ".OBJ" extension and to reside in the current working directory.

3.5 Extracting Library Modules

Use an asterisk (*) followed by a module name to copy a module from the library file into an object file of the same name. The module remains in the library file. When LIB copies the module to an object file, it adds the ".OBJ" extension and the drive designation and pathname of the current working directory to the module name to form a complete object filename. You cannot override the ".OBJ" extension, drive designation, or pathname given to the object file, but you can later rename the file or copy it to whatever location you like.

Use the minus sign followed by an asterisk (−*) to move an object module from the library file to an object file. This operation is equivalent to copying the module to an object file, as described above, then deleting the module from the library.

3.6 Combining Libraries

You can add the contents of one library to another by using the plus sign (+) with a library filename instead of an object filename. Following the "Operations" prompt, give the plus sign (+) followed by the name of the library whose contents you wish to add to the library being modified. When you use this option you must include the ".LIB" extension of the library filename. Otherwise, LIB assumes that the file is an object file and looks for the file with an ".OBJ" extension.

In addition to allowing MS-DOS libraries as input, LIB also accepts 286 XENIX archives and Intel-format libraries. Thus, you can use LIB to convert libraries from either of these formats to the Microsoft format.

LIB adds the modules of the library to the end of the library being modified. Notice that the added library still exists as an independent library. LIB copies the modules without deleting them.

Once you have added the contents of a library or libraries, you can save the new, combined library under a new name by giving a new name following the "Output library" prompt. If you omit the "Output library" response, LIB saves the combined library under the name of the original library being modified.

3.7 Creating a Cross-Reference Listing

Create a cross-reference listing by giving a name for the listing file following the "List file" prompt. If you omit the response to this prompt, LIB uses the special filename NUL., which means that no listing file is created.

You can give the listing file any name and any extension. You can specify a full pathname, including drive designation, for the listing file to cause it to be created outside your current working directory. LIB does not supply a default extension if you omit the extension.

A cross-reference listing file contains two lists. The first is an alphabetical listing of all public symbols in the library. Each symbol name is followed by the name of the module in which it is referenced.

The second list is an alphabetical list of the modules in the library. Under each module name is an alphabetical listing of the public symbols referenced in that module.

3.8 Performing Consistency Checks

When you give just a library name followed by a semicolon following the "Library name" prompt, LIB performs a consistency check, displaying messages about any errors it finds. No changes are made to the library. This option is not usually necessary, since LIB automatically checks object files for consistency before adding them to the library.

To produce a cross-reference listing along with a consistency check, use the command line method of invoking LIB. Give the library name followed by a semicolon, then give the name of the listing file. LIB performs the consistency check and then creates the cross-reference listing.

3.9 Setting the Library Page Size

The page size of a library affects the alignment of modules stored in the library. Modules in the library are aligned so that they always start at a position that is a multiple of *n* bytes from the beginning of the file. The value of *n* is the page size. The default page size is 16 for a new library or the current page size for an existing library.

Because of the indexing technique used by LIB, a library with a large page size can hold more modules than a library with a smaller page size. However, for each module in the library, an average of $n/2$ bytes of storage space is wasted (where *n* is the page size). In most cases, a small page size is advantageous, and you are advised to use the smallest page size possible.

To set the library page size, add a page size option after the library filename following the "Library name" prompt:

library `/pagesize:`*n*

The value of *n* is the new page size. It must be a power of 2 and must fall between 16 and 32,768.

3.10 LIB Error Messages

The following are Microsoft LIB error messages:

***symbol* is a multiply defined PUBLIC. Proceed?**

Cause: Two modules define the same public symbol. You are asked to confirm the removal of the definition of the old symbol.

Cure: Remove the PUBLIC declaration from one of the object modules and recompile or reassemble. If you respond No, the library will be left in an indeterminate state.

Allocate error on VM.TMP

Out of disk space

Cannot create extract file

No room in directory for extract file

Cannot create list file

No room in directory for library file

Cannot nest response file

@filespec in response (or indirect) file

Microsoft LIB cannot open VM.TMP

There is no room for VM.TMP in disk directory

Cannot write library file

Out of disk space

Close error on extract file

Out of disk space

Error: An internal error has occurred

Contact Microsoft Corporation

Fatal Error: Cannot open input file

You mistyped an object filename

Fatal Error: Module is not in the library

You tried to delete a module that is not in the library

Input file read error

Bad object module or faulty disk

Invalid object module/library

Bad object module and/or library

Library Disk is full

No more room on disk

Listing file write error

Out of disk space

No library file specified

No response to Library File: prompt

Read error on VM.TMP

Disk not ready for read

Symbol table capacity exceeded

Too many public symbols (about 30K chars in symbols)

Too many object modules

More than 500 object modules

Too many public symbols

1024 public symbols maximum

Write error on library/extract file

Out of disk space

Write error on VM.TMP

Out of disk space

Index

MICROSOFT®
10700 Northup Way, Bellevue, WA 98004

Software Problem Report

Name _____

Street _____

City _____ State _____ Zip _____

Phone _____ Date _____

Instructions

Use this form to report software bugs, documentation errors, or suggested enhancements. Mail the form to Microsoft.

Category

_____ Software Problem

_____ Software Enhancement

_____ Documentation Problem (Document #_____)

_____ Other

Software Description

Microsoft Product _____

Rev. _____ Registration # _____

Operating System _____

Rev. _____ Supplier _____

Other Software Used _____

Rev. _____ Supplier _____

Hardware Description

Manufacturer _____ CPU _____ Memory _____ KB

Disk Size _____" Density: Sides:

Single _____ Single _____

Double _____ Double _____

Peripherals _____

Problem Description

Describe the problem. (Also describe how to reproduce it, and your diagnosis and suggested correction.) Attach a listing if available.

Microsoft Use Only

Tech Support _____ Date Received _____

Routing Code _____ Date Resolved _____

Report Number _____

Action Taken:

MICROSOFT®