TOPS-10 MAKLIB User's Guide

AA-M472A-TB

January 1982

The MAKLIB guide describes the four functions that the MAKLIB utility performs on library files.

The Guide supersedes the *DECsystem-10 MAKLIB User's Guide*, order number DEC-10-UMUGA-A-D

OPERATING SYSTEM:	TOPS-10 V7.01
SOFTWARE:	MAKLIB V2B

January 1982

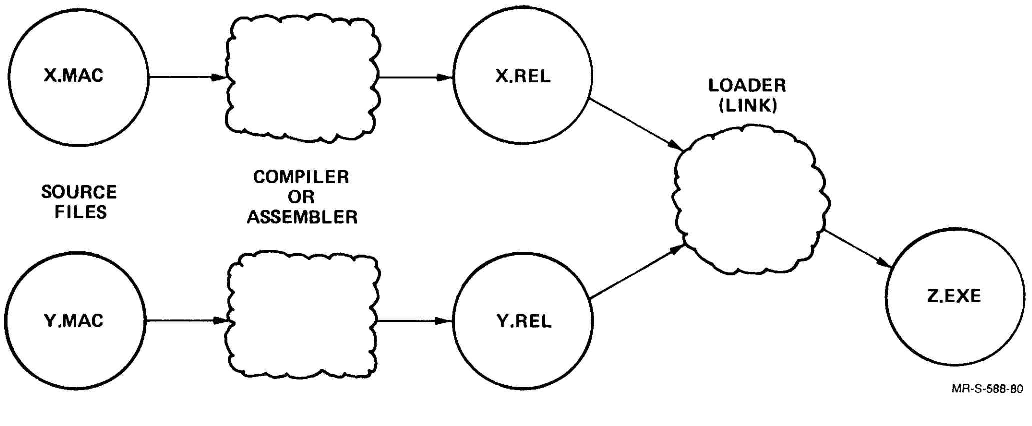
Contents

1.0 Introduction

The MAKLIB program organizes and manipulates files of relocatable object (REL) modules. These REL modules are the output from a source language translator, such as the FORTRAN compiler or the MACRO assembler.

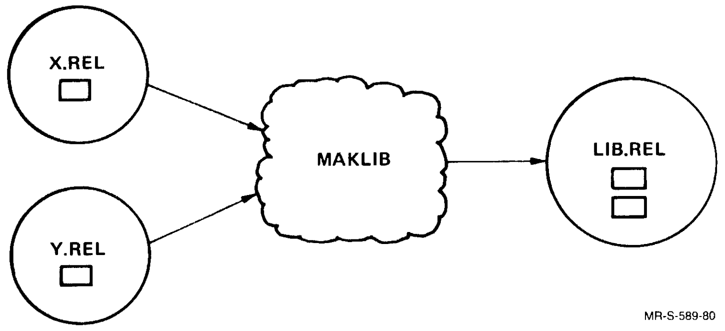
At load time, the modules are linked together to build an executable program. As shown in Figure 1, the MACRO assembler processes two source files, X.MAC and Y.MAC, and produces two corresponding .REL files, X.REL and Y.REL. The LINK program then loads the resulting object modules from these .REL files and produces an executable program, Z.EXE.

Figure 1: Generation of an .EXE File



Multiple modules can be concatenated into a single file called a library. A file containing a single module can also be called a library. (See Figure 2.)

Figure 2: Generation of a Library



MAKLIB performs four functions on library files. Each function has switches that cause the MAKLIB program to do a specific task. The four functions are:

Obtaining Information about Libraries

These switches cause MAKLIB to give information about the status and contents of the library.

Manipulating Libraries

These switches cause MAKLIB to create new libraries by combining modules. Other switches cause MAKLIB to add, delete, extract, or replace modules.

Modifying Libraries

These switches cause MAKLIB to create new libraries from existing libraries, either by adding an index or removing local symbols. By modifying libraries you can reduce the amount of processing time required by the LINK program.

Editing Libraries

These switches cause MAKLIB to edit (or patch) modules within the library. You selectively change the object code in a module by supplying MAKLIB with the required MACRO assembly language code changes.

For more information on the contents of .REL files and REL Block types, refer to the *TOPS–10 LINK Reference Manual.* For more information on MACRO assembly language, refer to the *TOPS–10 MACRO Assembler Reference Manual.*

2.0 Running MAKLIB

To run MAKLIB, type MAKLIB after the TOPS–10 prompt . and press the RETURN key. The program prompts you with an asterisk:

```
.R MAKLIB(RET)
*
```

After this prompt, enter a command string. MAKLIB takes commands in the following format:

Destination file spec=Source file spec1/Switches,Source file spec2/Switches... Source file specn/Switches

where:

Destination file spec is the output file that MAKLIB produces. It can be either a text file or a library, depending upon the function you perform.

If you do not specify a Destination file name, MAKLIB uses the name of the file in Source file spec1. If you omit the Destination file type, the default depends on the function you perform.

and:

Source file spec1 is the master library. This file specification is always required in a MAKLIB command string.

You must specify a file name in Source file spec1. The default file extension is always .REL.

and:

Source file spec2 ...Source file specn are the transaction files. These are additional input files required to perform some MAKLIB functions. A function usually requires only one transaction file.

You include switches in the command string to instruct MAKLIB to perform a specific function. You specify switches in one of three formats.

and:

/Switch

/Switch:argument

/Switch:(arg1,arg2...argn)

You can perform only one action with a switch in a single command string, but MAKLIB allows up to 100 switch arguments for each command string.

You can use MAKLIB switches in abbreviated form as long as they remain unique. However, arguments to switches are usually module names and hence cannot be abbreviated. Parentheses are optional when you specify only one argument, but are required to enclose two or more switch arguments.

Table 1 in Section 3 is an alphabetical list of the MAKLIB switches; they are discussed in detail in Sections 2.1 through 2.4.

You can use an indirect file in a MAKLIB command string to reference another file. The indirect file can contain a complete or partial MAKLIB command string (file names and switches). For more information on using indirect files, refer to the *TOPS–10 Operating System Commands Manual.*

If you always want to use specific switches when you run MAKLIB, you can put these switches in a SWITCH.INI file. For more information on creating a SWITCH.INI file, refer to the *TOPS–10 Operating System Commands Manual.*

MAKLIB allows you to type a string of commands on one or more lines. If the command string takes more than one line, type a hyphen (–) at the end of the first line, and press RETURN. Then, when MAKLIB prompts with a pound sign (#), continue to type the command string on the next line. You can also use multi–line commands in an indirect file.

To exit from MAKLIB and return to system command level, type either /EXIT or CTRL/Z after the asterisk prompt.

2.1 Running MAKLIB to Obtain Information About Libraries

MAKLIB contains four switches that allow you to obtain information on the status and contents of the master library (first input file). The four switches are: /LIST, /POINTS, /TRACE, and /LOAD.

Command String Requirements

Files:	A master library and an output file are required in the command string for each of the four switches. None of the command strings require a transaction file.
Default file extension:	Output file extension – .LST Master library extension – .REL
Arguments:	(Modules affected by the switch) – None

The output file (.LST) is a text file that can be written to any output device that supports text files, such as TTY: or LPT:. The discussion of /POINTS in this section illustrates the use of this option.

/LIST – /LIST Switch

This switch lists the names of the modules that are contained in the master library. In addition to the names, MAKLIB also lists the two data values from the END block (REL Block type 5) of the module. If the module is a two–segment program, the first value is the high–segment break, and the second value is the low–segment break. If the module is a one–segment program, the first value is the program break, and the second value is the absolute break. If the second value is zero, it is not printed.

For example, if you want to create a file, REAP.LST, showing the names of all modules in the library IAGO.REL, give the following command string:

```
.R MAKLIB(RET)
*REAP.LST=IAGO.REL/LIST(RET)
*
```

When MAKLIB finishes the task you request, it prompts you with another asterisk. You can now enter another command string.

You get the following when you type the new file:

```
.R TYPE (FILE) REAP.LST(RET)
Listing of Modules
Produced by MAKLIB Version 2B(104) on 12-Dec-81 at 14:02:41
**************************
DSK:IAGO.REL[4,244] created on 8-Dec-81 at 14:32:00
IAGO   401725   023746
.
```

The output file REAP.LST shows that the file IAGO.REL contains one module named IAGO. This module is a two–segment program. The first value listed, 401725, is the high–segment break. The second value listed, 023746, is the low–segment break.

/POINTS – POINTS Switch

This switch lists all entry points in the specified library. Entry points are usually subroutine starting addresses. They are used by the LINK program to determine whether a global request can be satisfied by loading a module from a library.

For example, if you want to know all the entry points in the library NICE.REL and have the output written to the device TTY: (your terminal), do the following:

```
.R MAKLIB(RET)
*TTY:=NICE.REL/POINTS(RET)
Listing of Modules and Entry Points
Produced by MAKLIB Version 2B(104) on 12-Dec-81 at 16:33:45
**************************
DSK:NICE.REL[4,244] created on 12-Dec-81 at 16:11:00
NICE    BEGIN    LOOP    NICE
*
```

In this example, the output shows that the library NICE.REL contains one module, NICE, which has three entry points: BEGIN, LOOP, and NICE. After the output, MAKLIB returns with the asterisk prompt and waits for another command string.

/TRACE – TRACE Switch

This switch lists all the edits you have made to a library. This information is contained in the TRACE blocks in the specified library. MAKLIB creates these TRACE blocks (REL Block type 1060) when you use /FIX to edit a module in the library. The TRACE blocks contain information about the edits you insert and the changes you make to the library. For more information on TRACE blocks, refer to Section 5. /TRACE allows you to determine the exact binary patching status of the library.

For example, if you want to see all the edits in the library OLDLIB.REL, and have the output written to the device TTY:, do the following:

```
.R MAKLIB(RET)
*TTY:=OLDLIB.REL/TRACE(RET)
Listing of TRACE blocks
Produced by MAKLIB Version 2B(104) on 10-Dec-81 at 9:33:59
***************************
DSK:OLDLIB.REL[4,601] created on 1-Dec-81 at 9:31:00
Module: SORT Edit: 341
Status is Active
Last affected by DRB
Created by HAS On 2-Dec-81 Installed by DRB On 10-Dec-81 Program
changes:
Inserts 4 instruction(s) at location 001046
```

This example shows that the module SORT in the library OLDLIB.REL has one edit inserted. The status of the edit is active. You can change the status of a particular edit with the .REMOVE or .REINSERT pseudo–ops. The example also shows that the edit was last affected by DRB. This information comes from /WHO when you install or change an edit. The edit was created by HAS on 2–Dec–81. This information comes from the .NAME and .DATE pseudo–ops, respectively. The edit was installed by DRB on 2–Dec–81. This information comes from /WHO and the date that the system supplies when you install an edit. For more information on these pseudo–ops and /WHO, refer to Section 2.4.

/LOAD – LOAD Switch

This switch shows additional loading instructions that are embedded within the library in either REQUEST (REL Block type 17), REQUIRE (REL Block type 16), or ASCII text blocks.

The library QUICK.REL was produced from a MACRO program containing the following statements:

```
TITLE   QUICK
.TEXT   "/VERSION:2(111)"
.REQUEST
SYS:FORLIB
.REQUIRE
SYS:MACREL
```

To see the additional loading instructions embedded within QUICK.REL in the blocks, do the following:

```
.R MAKLIB(RET)
*TTY:=QUICK.REL/LOAD(RET)
Listing of Internal loading instructions
Produced by MAKLIB Version 2B(104) on 10-Dec-81 at 9:06:23
*************************
DSK:QUICK.REL[4,601] created on 6-Dec-81 at 13:28:00
Module: QUICK
Text string: /VERSION:2(111)
Requests SYS:FORLIB
Requires SYS:MACREL
```

2.2 Running MAKLIB To Manipulate Libraries

For handling and creating libraries, MAKLIB includes six switches that allow you to work with individual modules within libraries. The six switches are: /MASTER, /APPEND, /DELETE, /EXTRACT, /INSERT, and /REPLACE.

Command String Requirements

Files:	A master library (first input file) and an output file are required in the command string for each of the six switches. A transaction file is required with some switches.
Default file extension:	.REL for all files
Arguments:	All switches accept arguments. /APPEND and /INSERT are two switches that do not always require arguments. For more information, refer to the discussions of these two switches in this section.

/MASTER – MASTER Switch

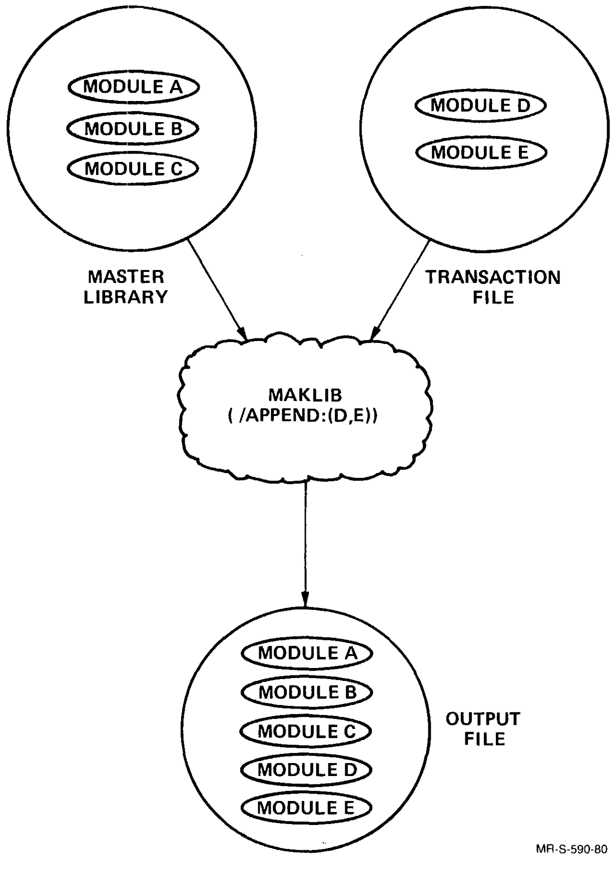
This switch identifies modules within the master library that correspond to those in the transaction file being used to effect the update. /MASTER takes at least one argument, and requires that another switch be given in the same command string. It is the only switch within this function that causes no real manipulation of a library. It is mentioned here because some of the switches used to manipulate libraries require /MASTER in their respective command strings.

You include /MASTER in the command strings for only two switches, /INSERT and /REPLACE. These switches are discussed later in this section.

/APPEND – APPEND Switch

This switch adds new modules to the end of an existing library. The output file is the master library plus the appended modules. MAKLIB reads these appended modules from the transaction file. You specify them as arguments to the switch. You must specify modules as arguments in the same physical order as they occur in the transaction file. Figure 3 illustrates the function of /APPEND. Note that, in the figure, modules D and E from the transaction file are appended to the master library.

Figure 3: Function of /APPEND



NOTE

When you do not specify an argument to /APPEND, the entire transaction file is appended to the master library.

For example, you want to append the module IAGO in the library IAGO.REL to the library GRAF.REL. You name the output file EXEN.REL. The command string is:

```
.R MAKLIB(RET)
*EXEN.REL=GRAF.REL,IAGO.REL/APPEND:IAGO(RET)
*
```

MAKLIB returns with the asterisk prompt for you to enter another command string. To check the new file, use /LIST and have the output written to the device TTY:.

```
*TTY:=EXEN.REL/LIST(RET)
Listing of Modules
Produced by MAKLIB Version 2B(104) on 14-Dec-81 at 10:45:13
**************************
DSK:EXEN.REL[4,244] created on 14-Dec-81 at 10:43:00
CRIG     004582
PASZ     003217
TENP     036651
IAGO     402420    023746
*
```

The example shows that the new file, EXEN.REL, contains four modules: the three modules from the library GRAF.REL (CRIG, PASZ, TENP), and the appended module IAGO.

/APPEND also allows you to initially create a library. For example, you wish to combine the following four .REL files into a single library:

GLOBAL.REL
DTABLE.REL
INOUT.REL
IO.REL

Type the following MAKLIB command string to produce the file LIB.REL, which contains all modules from the four specified files:

```
.R MAKLIB(RET)
*LIB=GLOBAL,DTABLE/APPEND,INOUT/APPEND,IO/APPEND(RET)
*
```

In this example, MAKLIB copies the file GLOBAL to a file named LIB, and then appends all modules from the files DTABLE, INOUT, and IO to LIB. This also illustrates the use of more than one transaction file.

/DELETE – DELETE Switch

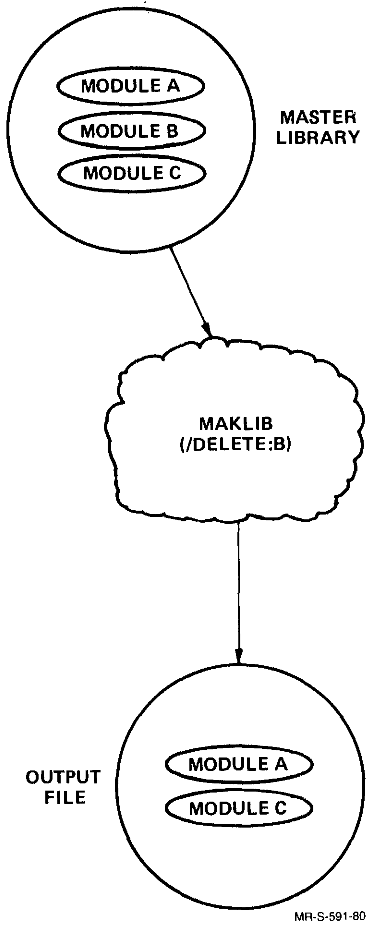
This switch removes one or more modules from an existing library. The output file is the master library minus the deleted module(s). All modules, except those specified as arguments to /DELETE, are read from the master library and copied to the output file. No transaction file is required.

NOTE

> You must specify modules as arguments in the same physical order as they occur in the master library.

Figure 4 illustrates the function of /DELETE. Note that, in the figure, module B is deleted from the master library.

Figure 4: Function of /DELETE



Type the following command string to delete the module TENP from the library EXEN.REL. The output file name is DIPEP.REL.

```
.R MAKLIB(RET)
*DIPEP.REL=EXEN.REL/DELETE:(TENP)(RET)
*
```

The program returns with the asterisk prompt for you to enter another command string. To check the new file, use /LIST and have the output written to the device TTY:.

```
*TTY:=DIPEP.REL/LIST(RET)
Listing of Modules
Produced by MAKLIB Version 2B(104) on 14-Dec-81 at 14:10:41
**************************
DSK:DIPEP.REL[4,244] created on 14-Dec-81 at 14:10:00
CRIG     004582
PASZ     003217
IAGO     402420    023746
*
```

/EXTRACT – EXTRACT Switch

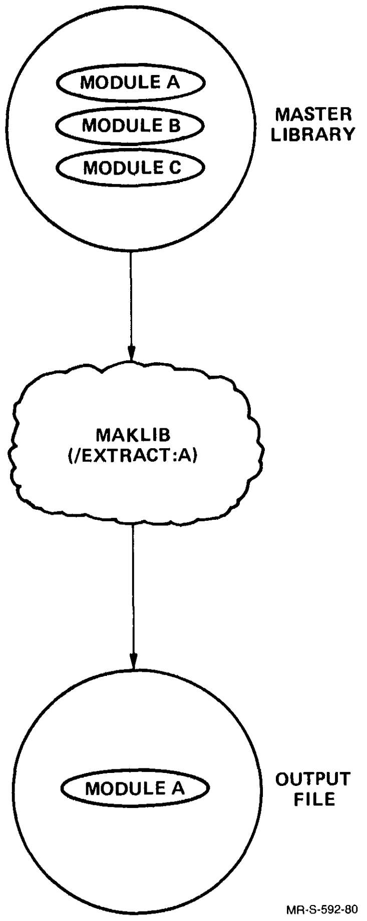
This switch produces an output file that is a subset of modules in the master library. You specify the modules as arguments to the switch. No transaction file is required.

NOTE

> You must specify the modules as arguments in the same physical order as they occur in the master library.

Figure 5 illustrates the function of /EXTRACT. Note that, in the figure, module A is extracted from the master library.

Figure 5: Function of /EXTRACT



The following example illustrates /EXTRACT. The library FOO.REL contains four modules. To get a listing of the module names, use /LIST and have the output written to the device TTY:.

```
.R MAKLIB(RET)
*TTY:=FOO.REL/LIST(RET)
Listing of Modules
Produced by MAKLIB Version 2B(104) on 18-Dec-81 at 14:37:45
**************************
DSK:FOO.REL[4,244] created on 18-Dec-81 at 13:54:00
SQUARE    000023
BOX       000014
MAIN      000433
DRAW      000505
*
```

You want to extract two of these modules, SQUARE and BOX, from the library FOO.REL, and put them in a new library, 2FOO.REL. After using /EXTRACT, check on the new file with /LIST, and have the output written to the device TTY:. Type the MAKLIB command string after the asterisk prompt:

```
*2FOO.REL=FOO.REL/EXTRACT:(SQUARE,BOX)(RET)
*TTY:=2FOO.REL/LIST(RET)
Listing of Modules
Produced by MAKLIB Version 2B(104) on 18-Dec-81 at 14:57:17
**************************
DSK:2FOO.REL[4,244] created on 18-Dec-81 at 14:56:00
SQUARE    000023
BOX       000014
*
```

The example shows that the new library, 2FOO.REL, contains the two modules that you extracted from the master library, FOO.REL.

/INSERT – INSERT Switch

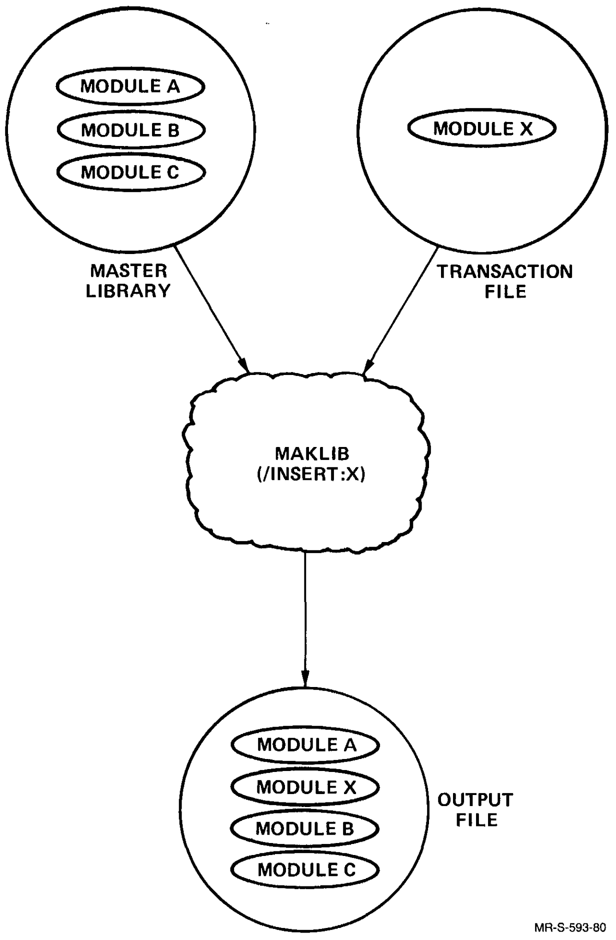
This switch inserts new modules into a master library. /MASTER is required in the command string. The output file is formed as follows: MAKLIB copies the master library to the output file up to but not including the module named as the first argument to /MASTER. Next, MAKLIB copies the module named as the first argument to /INSERT from the transaction file to the output file. The process repeats until the argument list specified to /MASTER and /INSERT is exhausted. At this point, MAKLIB copies the remaining modules in the master library to the output file. There must be one argument to /MASTER for every argument to /INSERT.

NOTE

You must specify the module names in the argument lists for /MASTER and /INSERT in the same physical order as they occur in the master library and the transaction file, respectively. When you do not specify an argument to /INSERT, the entire transaction file is inserted before the module you specify to /MASTER. You must always specify an argument to /MASTER.

Figure 6 illustrates this function of /INSERT. Note that, in the figure, module X in the transaction file is inserted before module B in the master library.

Figure 6: Function of /INSERT with One Module



For example, the library FOO.REL contains four modules: SQUARE, BOX, MAIN, and DRAW. The library NICE.REL contains one module, NICE. You want to insert the module NICE before the module BOX in FOO.REL. The name of the output file containing the five modules is CLAR.REL. The command string to insert this module is as follows:

```
.R MAKLIB(RET)
*CLAR.REL=FOO.REL/MASTER:BOX,NICE.REL/INSERT:NICE(RET)
*
```

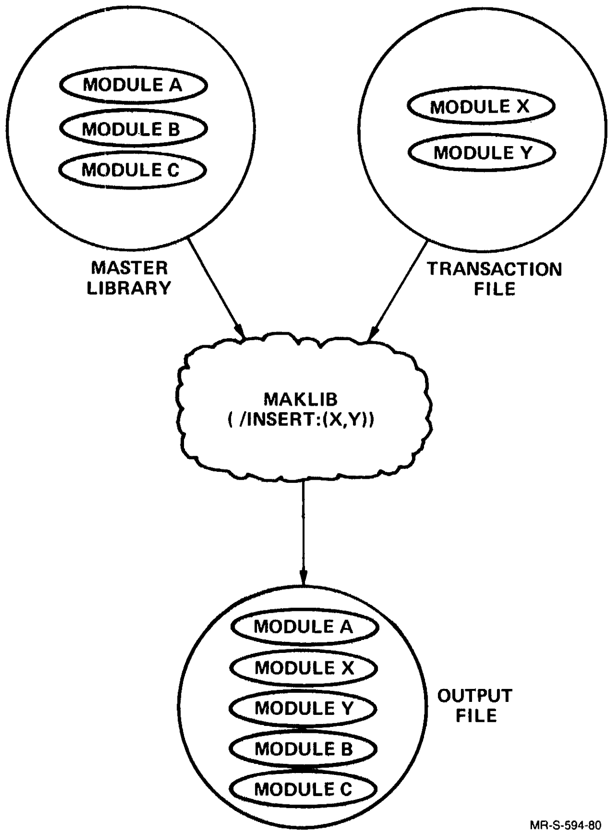
MAKLIB returns with the asterisk prompt. You can now check on the new library, CLAR.REL, with /LIST. The output from /LIST is written to the device TTY:

```
*TTY:=CLAR.REL/LIST(RET)
Listing of Modules
Produced by MAKLIB Version 2B(104) on 27-Dec-81 at 15:40:28
*************************
DSK:CLAR.REL[4,244]created on 27-Dec-81 at 15:40:00
SQUARE    000023
NICE      005557
BOX       000014
MAIN      000433
DRAW      000505
*
```

You may also insert more than one module in front of a module in a master library. However, the master library module name must appear repeatedly in the argument list to /MASTER. This produces a one–to–one correspondence between the module in the master library and the modules you wish to insert. In this case, you must list the argument names for both /MASTER and /INSERT in the same physical order that they appear as modules in the master library and transaction file, respectively.

Figure 7 illustrates this function of /INSERT. Note that, in the figure, modules X and Y in the transaction file are inserted before module B in the master library.

Figure 7: Function of /INSERT with Multiple Modules



For example, the library SFJA.REL contains two modules: ILJA, and HLBET. The library FOO.REL contains four modules: SQUARE, BOX, MAIN, DRAW. You want to insert both modules in SFJA.REL in the library FOO.REL, before the module DRAW. The name of the new library is SFOO.REL . The command string is:

```
.R MAKLIB(RET)
*SFOO.REL=FOO.REL/MASTER:(DRAW,DRAW),SFJA.REL/INSERT:(ILJA,HLBET)(RET)
*
```

After the asterisk prompt, check on the contents of the new library, SFOO.REL, with /LIST and have the output written to the device TTY:.

```
*TTY:=SFOO.REL/LIST(RET)
Listing of Modules
Produced by MAKLIB Version 2B(104) on 28-Dec-81 at 16:30:22
**************************
DSK:SFOO.REL[4,244] created on 28-Dec-81 at 16:30:00
SQUARE    000023
NICE      005557
BOX       000014
MAIN      000433
ILJA      001047
HLBET     000433
DRAW      000505
*
```

/REPLACE – REPLACE Switch

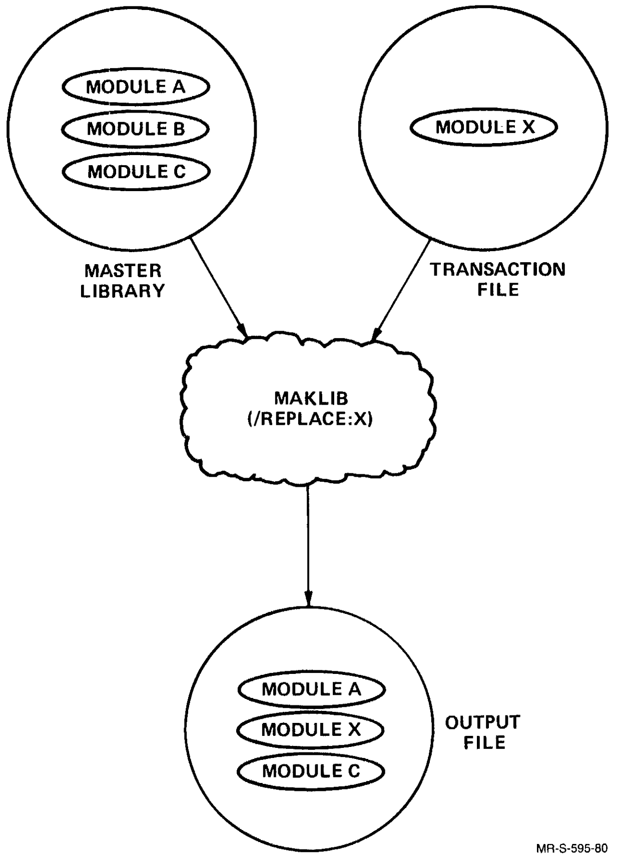
This switch replaces modules in the master library with those specified in the transaction file. /MASTER is required in the command string so that the program can identify the modules in the master library that are to be replaced by those named as arguments to /REPLACE. There must be a one–to–one correspondence between the number of arguments to /MASTER and /REPLACE. The output file is the entire master library, with modules replaced by those read from the transaction file (and named as arguments to the switch).

NOTE

You must specify the names in both argument lists (/MASTER and /REPLACE) in the same physical order as they appear as modules in the master library and transaction file, respectively.

Figure 8 illustrates the function of /REPLACE. Note that module X in the transaction file replaces module B in the master library.

Figure 8: Function of /REPLACE



For example, the library FOO.REL contains four modules: SQUARE, BOX, MAIN, and DRAW. The library NICE.REL contains one module, NICE. You want to replace the module MAIN in FOO.REL with the module NICE in NICE.REL. The output file name is WELW.REL. The command string is the following:

```
.R MAKLIB(RET)
*WELW.REL=FOO.REL/MASTER:MAIN,NICE.REL/REPLACE:NICE(RET)
*
```

After MAKLIB completes the action you requested, it prompts you with an asterisk. Now check on the contents of the new library, WELW.REL, with /LIST and have the output written to the device TTY:

```
*TTY:=WELW.REL/LIST(RET)
Listing of Modules
Produced by MAKLIB Version 2B(104) on 6-Dec-81 at 14:49:22
*************************
DSK:WELW.REL[4,244] created on 6-Dec-81 at 14:48:00
SQUARE    000023
BOX       000014
NICE      005557
DRAW      000505
*
```

NICE replaced MAIN in the master library FOO.REL. The new library, WELW.REL, contains the four modules: SQUARE, BOX, NICE, and DRAW.

2.3 Running MAKLIB to Modify Libraries

The two switches within this function facilitate the processing of requests by the LINK program when it loads modules from libraries. The two switches are: /INDEX and /NOLOCALS.

Command String Requirements

Files:	A master library (first input file) and an output file are required in the command string for both switches. Transaction files are not allowed. Both switches appear with the master library in the command string.
Default file extension:	.REL for both the master library and the output file for both switch command strings.
Arguments:	None

/INDEX – INDEX Switch

This switch produces an output file, which is identical to the master library, except with INDEX blocks (REL Block type 14) inserted in the file. Normally, programs make external requests to library subroutines they need, and LINK must search completely through the library to decide which modules to load to satisfy the requests. INDEX blocks list the entry point names and corresponding modules, allowing LINK to determine quickly which modules to load. LINK searches more efficiently, and loading time is shorter because the amount of I/O is reduced.

/NOLOCALS – NOLOCALS Switch

This switch produces an output file which is the master library with all local symbols deleted from the file SYMBOL blocks (RELBlock type 2). Local symbols are useful for debugging purposes, and also when modules are edited with MAKLIB. (Refer to FIX files, Section 2.4.) In a production–mode library, local symbols are usually deleted, because they serve no purpose. This reduces the amount of mass storage space the library occupies. In addition, loading time is faster because the amount of I/O is reduced. Global symbols are not deleted because they are used in the linking of modules.

In the following example you create a new library, L2, from L1. The new library has an index but no local symbols. You can use /INDEX and /NOLOCALS together in the command string.

```
.R MAKLIB(RET)
*L2=L1/NOLOCALS/INDEX(RET)
*
```

2.4 Running MAKLIB to Edit Libraries

MAKLIB provides a mechanism for you to patch (or edit) the code of a relocatable object module. This patching facility allows you to make program changes directly to a library. Although MAKLIB provides the facility for editing a program without having to change the source code, good programming practice requires that the same edits also be made at the source level.

To edit a library in this way, you must first create a text file called a .FIX file. This .FIX file contains one or more edits that you want to insert in the library. Each edit has a unique identifier and consists of a sequence of control pseudo–ops and MACRO assembly language code. The control pseudo–ops tell MAKLIB where and how to make the changes. Any new code is supplied as a sequence of MACRO assembly language statements. Each edit begins with an .EDIT pseudo–op and ends with an .ENDE pseudo–op. Figure 9 shows the order that pseudo–ops must appear within an edit.

When MAKLIB processes the .FIX file, it creates a new library with any new edits inserted. Although each edit is now a permanent part of the library, you can use MAKLIB to deactivate an edit. This operation effectively removes any code changes inserted by the edit. Therefore, edits in the library can be either active or inactive.

MAKLIB maintains edit history information on the library by generating TRACE blocks (REL Block type 1060) for each edit you insert. TRACE blocks are part of the module. Therefore, when you use /REPLACE, /EXTRACT, or /DELETE, the TRACE blocks move with the module. Section 2.1 describes how you can determine the edit status of the library using /TRACE.

NOTE

MAKLIB does not handle PSECTS.

Pseudo–ops for .FIX files

.EDIT xxxxxx This pseudo–op is an identifying name for the edit you insert in the specified module. The edit name (xxxxxx) can be up to six (6) SIXBIT characters and is stored in the TRACE block for any module affected by the edit. .EDIT is the first pseudo–op for each edit.

.DATE dd–mmm–yy This pseudo–op gives the date that the edit was made. The day (dd) and year (yy) entries are numeric. The month entry (mmm) is alphabetic. .DATE is an optional pseudo–op. If you supply this information, it is stored in the edit TRACE block.

.NAME xxx This pseudo–op gives the three initials (xxx) of the person who wrote the edit. .NAME is an optional pseudo–op. If you supply these initials, they are stored in the TRACE block for the edit.

.MODULE xxxxxx This pseudo–op gives the name (xxxxxx) of the module that you wish to edit. It is the module name as it appears in the library, up to six Radix–50 characters. Once you give a name, that module is loaded. Editing continues with this module unless a new .MODULE pseudo–op is given. You do not have to edit modules in the same order that they reside in the master library (first input file). However, each module may be named only once within an edit.

.ASSOCIATED +edit1,–edit2,+edit3,+edit4..... This pseudo–op gives information on which other related edits must be present in the specified module. The edit names here are the same as those designated under the .EDIT pseudo–op. The "+" indicates that the edit is required. The "–" indicates an edit that conflicts with the current edit; you cannot have both edits active simultaneously. You receive notification if the specified module does not contain the correct combination of associated edits. The default is "+" if no sign precedes the edit name. Information you supply with the .ASSOCIATED pseudo–op must precede any information supplied in the .FIX file by the .INSERT, .REMOVE, or .REINSERT pseudo–ops.

.VERSION nnnxx(nnnnnn)–g This pseudo–op allows you to specify the version number of the edit. The version is in standard TOPS–10 version format. The nnn designates the major version number, which consists of three octal digits maximum. The xx designates the minor version, which consists of two letters maximum. The (nnnnnn) designates the edit number, and can be up to six octal digits. The g designates the code for the group that last edited the module, and it is one octal digit maximum. All fields are optional.

.ALTER location,<new value>,<original value> This pseudo–op changes the contents of a specific location. All code is written in angle brackets, < >. The original value at the specified location is replaced by a new value. The first argument, location, is where you wish to place the new value. Once you enter the new value, it is evaluated and put into the specified location. You can specify a third argument, <original value>, to check whether the actual original value differs from the expected original value. If it does, MAKLIB gives you an error message and the location is not altered.

.INSERT location, keyword:n,<original value> This pseudo–op allows you to add code to a module. You precede the new code with an .INSERT pseudo–op and terminate the sequence with an .ENDI pseudo–op. MAKLIB assembles the code and adds it to the module in the output file.

The .INSERT pseudo–op takes three arguments. The first gives the location at which you want the new code executed. This argument can be a numeric or symbolic expression. This location is assumed to be relocatable, and may be in either the low or the high segment. The second argument is a keyword that specifies how the code is to be located with respect to this location. This keyword is one of the following:

BEFORE	Insert the new instructions so that they are executed before the instruction at the specified location.
AFTER	Insert the new instructions so that they are executed after the instruction at the specified location.
REPLACE	Delete one instruction from the existing code for each one included in the edit, beginning with the instruction at the location of the edit. Then, insert the new instructions so that they are executed in place of the deleted code.
REPLACE:n	Delete n instructions from the existing code, starting with the instruction at the edit address. Then, insert the new instructions so that they are executed in place of the deleted code. This applies no matter how many instructions you insert. The argument may be an expression, and is evaluated in the current radix.

You can specify a third argument, <original value>, to check whether an actual original value differs from the expected original value. If it does, MAKLIB gives you an error message and the editing does not take place. This argument gives the line of code at the location specified by the first argument. The code is written in angle brackets, and is evaluated. It must exactly match the code at the specified location. If the code at the location is a literal, you must give only the first word.

MAKLIB always inserts the new code at the end of the current segment. It replaces the referenced instruction with an unconditional jump to the new code. The format of code insertion appears in Section 5, Technical Notes.

Because MAKLIB does not physically insert the code at the location you specify, you need to consider a restriction when you use this facility.

MAKLIB constructs a patch that has the effect of a skipping instruction, and assumes that it follows an instruction that can potentially skip, at most, one instruction. If the intent of the patch does not fit these assumptions, it may not work correctly. Further, for REPLACE functions, MAKLIB assumes that program control can pass only to the first instruction of the deleted code.

For example consider the following code segment,

```
        JRST    FOO
BAR:    JFCL
        JFCL
FOO:    JFCL
        JFCL
        JFCL
```

and a MAKLIB patch to it:

```
.INSERT BAR,REPLACE:4<JFCL>
        JFCL
.ENDI
```

Note that the JRST FOO instruction at BAR–1 still causes the old code to be executed in spite of the patch.

.REMOVE edit1,edit2,edit3.... This pseudo–op deactivates the specified edits from the selected module. The original instructions displaced by the jumps to the edit area for each .INSERT are returned to that location. No changes are made to the symbol table. The arguments are the edits you wish to remove.

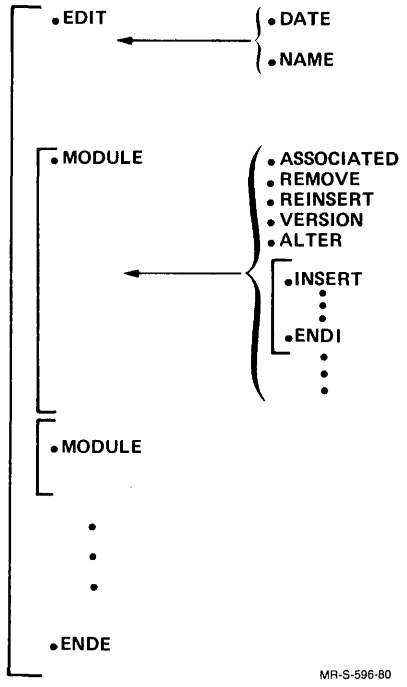
.REINSERT edit1,edit2,edit3... This pseudo–op activates any edits you previously removed with the .REMOVE pseudo–op.

.ENDI This pseudo–op marks the ending point of the code following the last .INSERT pseudo–op.

.ENDE This pseudo–op marks the ending point of the complete edit. It also instructs MAKLIB to check for undefined labels or other invalid entries within the edit.

Figure 9 illustrates the order that pseudo–ops appear in a .FIX file:

Figure 9: Order of Pseudo–ops in a .FIX File



The MAKLIB program has a one–pass assembler. Because of this, forward references to labels and expressions are restricted to simple addition and subtraction on the halfword boundary. References to undefined labels or symbols are valid where references to external symbols would be valid in MACRO (with no Polish fix–ups). Literals are treated as forward references, because the actual location of the literal is not known until the .INSERT pseudo–op ends. Defining a label inside of a literal is not valid. Finally, the value you place in the right–hand side of an assignment must not be forward or external.

It is not required that assignments be inside .INSERT in the .FIX file. It is required, however, that the .EDIT and .MODULE pseudo–ops precede any assignments, because these define new symbols in the symbol table. MAKLIB does not allow redefinitions of existing symbols, because it is impossible to backtrack references to a symbol in the relocatable binary file. So, any label or symbol you create with /FIX must be new to the program.

To simplify editing and to keep the appearance of binary edits as close as possible to the source level, the following pseudo–ops are implemented in the MAKLIB .FIX file assembler and operate as they do in MACRO:

ASCII	ASCIZ	BLOCK
BYTE	COMMENT	DEC
EXP	IOWD	OCT
POINT	PURGE	RADIX
RADIX50	REMARK	SIXBIT
SQUOZE	SUBTTL	TITLE
XWD		

NOTE

> The pseudo–ops BYTE, DEC, OCT, and EXP are limited to a maximum generation of one word of data.

All MACRO operators and qualifiers are available except ^F. MAKLIB also supports the following MACRO pseudo–ops for writing conditional code:

IFN	IFG	IFDEF
IFE	IFLE	IFNDEF
IFL	IFGE	

You may follow symbols with ## (double pound sign) to indicate that they are EXTERNAL quantities. However, if the symbol is already defined as EXTERNAL (in the symbol table), you do not have to use ##. It is not necessary to follow undefined symbol names with # (single pound sign), since it is assumed that any undefined symbol is a forward reference. If a symbol name is already assigned and followed by the #, you receive an error message (see Section 4, MAKLIB Messages). You may define labels as internal (available to other programs) if they are followed by :: (double colon). Entry points may not be defined. The full facilities available in MACRO for combinations of DDT suppression and internal declaration are available for both labels and assignments.

Command String Requirements

Files:	A master library (first input file) and an output file are required in the command string. The .FIX file is the required transaction file.
Default file extension:	.REL for both the output file and the master library. The default for the transaction file is .FIX.
Arguments:	None.

The editing command string accepts two switches. They are /FIX and /WHO.

/FIX – FIX Switch

This switch makes changes to the actual code and symbol table of a module. It appears with the transaction (.FIX) file specification.

/WHO:xxx – WHO Switch

This switch is optional and you use it only with /FIX. You can enter it in either the master library or the transaction (.FIX) file specification. The argument to /WHO can be up to three characters (xxx). These are usually the initials of the person using MAKLIB at the time the edit is installed. If you include this information, it appears in the TRACE block of all new edits (in the last affected field and in the last installed field). If any of these edits change the status of an existing edit (such as .REMOVE or .REINSERT), this information is entered in the last affected field of the TRACE block of the affected edit. If you use /WHO without /FIX, MAKLIB ignores it in the command string.

In order to edit a library with a .FIX file, you can use the following command string. In this example, you use the .FIX file FIX1.FIX to edit the library OLDLIB.REL, and create an updated library NEWLIB.REL.

```
.R MAKLIB(RET)
*NEWLIB=OLDLIB,FIX1/FIX/WHO:SFA(RET)
*
```

The following are sample .FIX files. This first example illustrates the use of the .INSERT and .VERSION pseudo–ops. One module in the library is modified.

```
.EDIT 341
.NAME HAS
.MODULE GLOB..
.VERSION 4C(341)
.INSERT START,AFTER,<RESET>
MOVE      P,[IOWD PDLEN,PDLIST]
.ENDI
.INSERT   LOOP+3,BEFORE,+<JRST LOOP>
       PUSHJ P,NEXTCR
.ENDI
.ENDE
```

The following edit illustrates the use of .ALTER pseudo–op to change the value of a table entry. Location RAD50 + 46 is changed from a "." to a "$". In addition, this edit uses the .ASSOCIATED pseudo–op to specify that this edit also requires edit 343 to the module TABLES.

```
.EDIT 344
.NAME HAS
.MODULE TABLES
.ASSOCIATED 343
.ALTER RAD50+46,<"$">,<"."> .ENDE
```

This edit uses the .REMOVE pseudo–op to deactivate edit 345 in the module FSORT. As a result of this operation, any code that was changed by edit 345 will be restored to its previous state.

```
.EDIT 346
.NAME HAS
.MODULE FSORT
.REMOVE 345
.ENDE
```

3.0 MAKLIB Switch Options

Table 1 is a reference table of all MAKLIB program switches. These switches are listed alphabetically. The function that each switch performs appears beside it in the table.

Table 1: MAKLIB Switches

Switch	Function
/APPEND	Adds new modules to the end of an existing library.
/DELETE	Removes one or more modules from an existing library.
/EXIT	Terminates MAKLIB and returns you to system command level.
/EXTRACT	Produces an output file that is a subset of modules in the master library.
/FIX	Makes changes to the actual code and symbol table of a module.
/INDEX	Produces an output file identical to the master library except with INDEX blocks inserted in the file.
/INSERT	Inserts new modules into the master library.
/LIST	Lists the names of the modules that are contained in the master library.
/LOAD	Shows additional loading instructions that are embedded within the library in either REQUEST, REQUIRE, or ASCII text blocks.
/MASTER	Identifies files within the master library that correspond to those in the transaction file being used to effect the update.
/NOLOCALS	Produces an output file which is the master library with all local symbols deleted from the file symbol blocks.
/POINTS	Lists all entry points in the specified library.
/REPLACE	Replaces modules in the master library with those specified in the transaction file.
/TRACE	Lists all the edits made to a library.
/WHO	Specifies the initials of the person using MAKLIB when an edit is installed.

4.0 MAKLIB Messages

The MAKLIB program issues two types of messages: fatal errors and warning messages. Fatal errors are preceded by a question mark (?) and cause the current command to be aborted. Warning messages are preceded by a percent sign (%) and indicate that the command will be completed, but the operation may not have been performed as you intended.

All messages are typed on your terminal. They begin with a six–character code that identifies the error. This is followed by a short description of the problem.

MAKLIB uses the command scanner routines SCAN and WILD. The following list contains the most common messages that these routines produce. These messages begin with SCN or WLD.

Some of the messages contain information that is dependent on the exact command string, switch, or file you wish to process. The key to these message variables follows:

[edit]	The name of a specific edit.
[file]	A file specification.
[label]	The name of the label which caused the error.
[location]	The location where the error was detected. This is expressed as either a symbolic address or as a line number in the .FIX file.
[module]	The name of a specific module.
[pseudo–op]	A specific pseudo–op.
[statement]	A specific statement related to or causing the error.
[status]	A specific numeric file status code.
[switch]	A specific MAKLIB command switch.
[symbol]	A symbol. (Refer to the *TOPS–10 MACRO Assembler Reference Manual* for an exact definition.)
[type]	A REL Block type.
[value]	A specific value.

All MAKLIB messages are listed here alphabetically by the six–character code. Suggested user responses are provided for most fatal errors and those warning messages that require correction.

?MKLAAC	**.ASSOCIATED seen after .INSERT, .REMOVE or .REINSERT in edit [edit]**
Description:	In the indicated .EDIT, the .ASSOCIATED pseudo–op is out of order.
Suggested User Response:	Move the .ASSOCIATED pseudo–op so that it appears before any .INSERTs, .REMOVEs, or .REINSERTs.
?MKLAAL	**.ALTER address is not in current module in edit [edit]**
Description:	In your edit, you used a .ALTER pseudo–op to change the contents of an address. However, you gave an address that does not exist in the specified module.
Suggested User Response:	Change the .ALTER pseudo–op so that it specifies a legal address.
%MKLAFI	**Arguments to /FIX switch are ignored**
Description:	In the command string, you gave arguments on /FIX. There are no defined arguments for /FIX, so MAKLIB ignores any that you specify.
%MKLAMI	**Assignment to [symbol] with no module selected was ignored:**
	[statement]
Description:	In a .FIX file you assigned a value to a symbol, but you did not specify a module.
Suggested User Response:	Change the .FIX file so that the assignment statement occurs after the .MODULE pseudo–op. This specifies which symbol belongs with a particular module.
?MKKLANA	**Asterisk not allowed as output file spec**
Description:	In the command string you gave an output file name that included a wildcard character.
Suggested User Response:	Since wildcard characters are illegal for the output file name, reissue the command with an explicit output file name.
?MKLASG	**FORWARD/EXTERNAL assignment to [symbol] at [location] (Edit [edit])**
	[statement]
Description:	In the specified edit, you assigned a forward or external reference.
Suggested User Response:	MAKLIB does not support forward or external references. Change the assignment statement.

?MKLBAM	**BEFORE, AFTER or REPLACE missing from .INSERT in edit [edit]**
Description:	You typed an incomplete .INSERT pseudo–op in a .FIX file.
Suggested User Response:	Include the required second argument for .INSERT indicating how the code should be inserted.
?MKLBDA	**Bad .DATE argument for edit [edit]**
Description:	In a .FIX file you specified a date that was not in a recognizable format for the .DATE pseudo–op.
Suggested User Response:	Specify the date in the form dd–mon–yy, such as 7–JUL–77.
?MKLBNI	**Binary patching tool not included in MAKLIB**
Description:	You attempted to use a .FIX file with a version of MAKLIB that does not support .FIX files.
Suggested User Response:	Rebuild MAKLIB from the MACRO source files and set feature switch FTBPT non–zero.
?MKLCDM	**Existing code does not match original code**
Description:	The original value you specified with either the .ALTER or .INSERT pseudo–op does not match the actual code at the location you were attempting to change.
Suggested User Response:	First, determine if this is an error in the original value field of the pseudo–op. If this is actually the value you expected at that location, this error could indicate that you are trying to edit a different version of the library file.
%MKLCII	**Code generated outside of range of .INSERT was ignored:** **[statement]**
Description:	You entered a .FIX file with a sequence of code that was not included between .INSERT and .ENDI pseudo–ops.
Suggested User Response:	Change the .FIX file so that the instructions are within the range of an .INSERT. They can then be inserted in the module.

%MKLCNF	**Insertion of edit [edit1] by edit [edit2] conflicts with edit [edit3]**
Description:	Edit2 contains a .REINSERT pseudo–op for edit1. This conflicts with the .ASSOCIATED list in edit3 which is currently an active edit for this module. This is only a warning message, and the actual .REINSERT does take place. You can verify the current status of any edits with /TRACE.
%MKLCNF	**Removal of edit [edit1] by edit [edit2] conflicts with edit [edit3]**
Description:	Edit2 contains a .REMOVE pseudo–op for edit1. This conflicts with the .ASSOCIATED list of edit3 which is currently an active edit for this module. This is only a warning message, and the actual .REMOVE does take place. To verify the status of any edits, use /TRACE.
?MKLCSR	**Command switch is required**
Description:	You typed an incomplete command string. Supply additional information with file switches.
Suggested User Response:	Reissue the command string including the necessary switches on either the master library or transaction files.
?MKLEEI	**.ENDE seen before .ENDI in edit**
Description:	Your .FIX file is missing an .ENDI pseudo–op, or the .ENDI pseudo–op is out of order.
Suggested User Response:	Make sure that each .INSERT pseudo–op is matched with an .ENDI pseudo–op to identify the code to be inserted. The .ENDE pseudo–op indicates the end of an edit. Therefore, it is always the last statement of an edit.
?MKLEFF	**End of file found before END block in module**
Description:	The input master file is bad. Although part of the file is readable, the END block (REL Block type 5) is missing for the particular module. This indicates that the file is damaged.
Suggested User Response:	Re–create the file or restore it from a backup copy.

%MKLEMA	**Entire MASTER file will be appended**
Description:	For the current command string, the entire MASTER file is being appended to the output file even though you specified an individual module.
?MKLEPM	**.EDIT pseudo–op is missing from FIX file**
Description:	The .FIX file is incorrect.
Suggested User Response:	Change the .FIX file so that each edit begins with the .EDIT pseudo–op and ends with an .ENDE pseudo–op.
?MKLERI	**Edit [edit] tried to .REMOVE or .REINSERT itself**
Description:	The .FIX file contains an edit that has a .REMOVE or .REINSERT pseudo–op referencing itself. These pseudo–ops must reference edits that have previously been applied to the module.
Suggested User Response:	Change the .FIX file so that the .REMOVE or .REINSERT pseudo–op does not reference the current edit.
?MKLETC	**MACRO code expression too complex at [location] (Edit [edit])**
Description:	The expression at the indicated location is too complex for MAKLIB to evaluate.
Suggested User Response:	Try to break the expression into several simpler expressions.
?MKLETL	**ENTRY block is too large to read in for module [module]**
Description:	MAKLIB does not have enough space allocated to process all the entry points for the specified module.
Suggested User Response:	Try to reduce the number of entry points in the module.
?MKLFF4	**Cannot apply FIX to F40 produced REL file**
Description:	You attempted to apply a .FIX file to a .REL file produced by the F40 FORTRAN compiler. This operation is not supported by MAKLIB. F40 generates .REL files that MAKLIB cannot edit from .FIX files.
?MKLFNI	**Qualifier ^F not implemented**
Description:	MAKLIB does not support ^F for entering a fixed–point decimal number.
Suggested User Response:	Enter the value in an alternate form.

?MKLFSI	**File status error on input [status] for file [file]**
Description:	An error occurred while MAKLIB was reading the specified file. The status value that appears is described in *TOPS–10 Monitor Calls.*
Suggested User Response:	This could indicate a more global problem with the system. See the error codes section of the ENTER call to determine the specific problem with the named file.
?MKLFSO	**File status error on output [status] for file [file]**
Description:	An error occurred while MAKLIB was writing the specified file. The status value that appears is described in *TOPS–10 Monitor Calls.*
Suggested User Response:	This could indicate a more global problem with the system. See the error codes section of the ENTER call to determine the specific problem with the named file.
?MKLIAA	**Illegal address in .ALTER in edit [edit]**
Description:	In the specified edit, you used a .ALTER pseudo–op to change the contents of an address. However, you gave an illegal value for an address.
Suggested User Response:	Change the .ALTER pseudo–op so that it specifies a legal address.
?MKLIAI	**Illegal address in .INSERT in edit [edit]**
Description:	You specified an illegal address in the location field for an .INSERT pseudo–op.
Suggested User Response:	See if the address you specified is the address of a literal, external, or undefined address. These are not allowed.
?MKLIAL	**.INSERT address is not in current module in edit [edit]**
Description:	You specified an address in the location field for an .INSERT pseudo–op that is not in the specified module.
Suggested User Response:	First determine if the address you specified is the one you intended. Then check the .FIX file to verify that you placed the .INSERT sequences after the correct .MODULE pseudo–ops.
?MKLIBT	**Illegal block type ([type]) was seen in file [file]**
Description:	MAKLIB encountered an illegal REL Block while reading the indicated file.
Suggested User Response:	The .REL file being read may be damaged. Re–create the .REL file and try again.

?MKLIED	**Internal error detected at [location] in MAKLIB**
Description:	This indicates that MAKLIB cannot perform the operation you were attempting.
Suggested User Response:	Contact your Software Specialist or send a Software Performance Report (SPR) to DIGITAL.
?MKLIIA	**.INSERT pseudo–op illegal inside range of .INSERT [edit]**
Description:	In the specified edit, you nested .INSERT pseudo–ops.
Suggested User Response:	These pseudo–ops cannot be nested. Use the .ENDI pseudo–op to end the first .INSERT sequence before you begin another.
?MKLIII	**Illegal pseudo–op in range of .INSERT: [value] at [location] (edit[edit])**
Description:	In your .FIX file you used a MACRO pseudo–op which is illegal in an .INSERT sequence.
Suggested User Response:	Verify that this is the correct MACRO pseudo–op for the operation you wish to perform.
?MKLILS	**Illegal use of long string or BLOCK in .ALTER at [location] (edit[edit])**
Description:	You tried to use a multi–word value with the .ALTER pseudo–op.
Suggested User Response:	The .ALTER pseudo–op is for changing single word values only. This applies to the original value as well as the new value. Use a separate .ALTER pseudo–op for each word to be altered.
?MKLIPM	**.ENDI seen without .INSERT in edit [edit]**
Description:	The .FIX file is incorrect.
Suggested User Response:	Change the .FIX file so that each set of instructions you wish to insert begins with an .INSERT pseudo–op and ends with an .ENDI pseudo–op.
?MKLIRF	**Illegal relocation in FORWARD reference to [symbol] in edit [edit]**
Description:	MAKLIB could not process the reference to the specified relocatable symbol.

?MKLIRM	**/INSERT requires at least one /MASTER specification**
Description:	The command string is incomplete.
Suggested User Response:	When you specify /INSERT on a Transaction file, you must include /MASTER with the master file. You must supply a specification on /MASTER for every module you wish to insert.
?MKLISM	**/INSERT,/REPLACE and /FIX are illegal switches on MASTER**
Description:	The command string is incorrect.
Suggested User Response:	Reissue the command string putting /INSERT, /REPLACE, or /FIX on the transaction file. When you use /INSERT or /REPLACE, you must include /MASTER on the master file.
?MKLIST	**Interim symbol table overflowed; code too complex in edit |edit|**
Description:	Your specified edit contains code that has too many symbols for MAKLIB to process.
Suggested User Response:	Try to break the single edit into several edits, each having fewer symbols.
?MKLITS	**Insufficient TRACE block storage in edit |edit|**
Description:	The specified edit exhausted all allocated storage for TRACE block information.
Suggested User Response:	Try to break the edit into several edits, thus reducing the amount of storage MAKLIB needs for TRACE blocks at any one time. For each edit, MAKLIB creates a TRACE block for each module that is changed.
?MKLIUN	**Illegal to have null address in .INSERT in edit |edit|**
Description:	In the specified edit, the location field of an .INSERT pseudo–op is null.
Suggested User Response:	The location field cannot be null. It specifies the location where you want to insert the patch code.
%MKLLII	**Label outside of .INSERT was ignored: |label|**
Description:	In a .FIX file you specified a label outside the range of an .INSERT.
Suggested User Response:	Change the .FIX file so that the label occurs within the range of an .INSERT–.ENDI pair. This specifies where you insert the new label in the program.

?MKLLTL	**MACRO code line is too long at [location] (Edit [edit])**
Description:	In the specified edit, a line of code in the .FIX file is too long for MAKLIB to process.
Suggested User Response:	Try to reduce the length of the line by breaking it into several shorter lines.
?MKLMCA	**Pseudo–operator argument error at [location] (Edit [edit])**
Description:	You gave an illegal pseudo–op argument. The values you can use depend on the particular pseudo–op.
Suggested User Response:	Supply a legal value for that pseudo–op.
?MKLMCB	**MASTER device must be capable of binary IO**
Description:	The input master file in your command string is not on a device that is capable of binary input/output. MAKLIB performs binary input/output on the master file.
Suggested User Response:	Keep the files you plan to use as MAKLIB master files on devices capable of binary input/output.
?MKLMCE	**Command error**
Description:	MAKLIB was not able to recognize a valid command from the command string that you typed. This error could occur if you improperly formatted the command string or if you used a non–unique abbreviation to specify a switch.
Suggested User Response:	Retype the command string in the correct format: Destination file spec=Source file spec1/Switches,Source file spec2/Switches,...Source file specn/Switches
?MKLMCF	**Illegal forward or external reference at [location] (Edit [edit])**
Description:	At the specified location you made a reference to an undefined symbol in this module. It could be a forward reference to a new symbol or it could be an external reference. MAKLIB cannot process forward references to new symbols. This error could also occur if you attempted to reference an existing symbol and supplied the wrong symbol name.
Suggested User Response:	Change your forward or external references to legal ones.

?MKLMCM **Attempt to redefine value of symbol [symbol] at |location| (Edit |edit|)**

Description: You tried to redefine the value of an existing symbol. MAKLIB does not support this.

Suggested User Response: You cannot use MAKLIB to change the value of an existing symbol. If this is not what you intended to do, you may be using a symbol that was already defined. Try another symbol.

?MKLMCN **MACRO code numeric error at |location| (Edit |edit|)**

Description: You supplied a numeric argument at the specified location that is illegal in that context.

Suggested User Response: Supply a legal numeric value.

?MKLMCQ **MACRO code is questionable at |location| (Edit |edit|)**

Description: MAKLIB cannot process the code at the specified location.

Suggested User Response: Check the code for legal MACRO syntax.

?MKLMCR **MACRO code relocation error at |location| (Edit |edit|)**

Description: The value at the specified location contains a half–word that is neither absolute nor simple relocatable. Expressions that would require Polish, such as A + A where A is relocatable, cannot be handled in MAKLIB.

Suggested User Response: Give absolute or simple relocatable values only.

?MKLMCU **Undefined symbol: |symbol| at |location| (Edit |edit|)**

Description: The symbol shown is not defined.

Suggested User Response: Define the symbol. If you are trying to use a symbol that you think is already defined, check for the correct symbol name.

?MKLMCW **BYTE,EXP,DEC,or OCT more than one word at |location| (Edit |edit|)**

Description: You used an expression that generates a multi–word value. In this context, MAKLIB supports only a single–word value.

Suggested User Response: Try to break the expression into simpler expressions, each generating a single–word value.

?MKLMEP **Missing .ENDE for edit |edit|**

Description: The specified edit in your .FIX file is incorrect.

Suggested User Response: Change the .FIX file so that each edit begins with the .EDIT pseudo–op and ends with an .ENDE pseudo–op.

?MKLMFR **Master file rejected by condition**

Description: The master file did not meet the condition you specified (as size, creation date); hence processing cannot continue.

?MKLMHE **Module already has an edit |edit|**

Description: You attempted to apply an edit to a module that already has an edit with a similar identifier.

Suggested User Response: Each edit to a module must have a unique identifier. You may already have applied this edit to the module.

?MKLMKM **[Pseudo–op| pseudo–op in edit |edit| without preceding .MODULE**

Description: In the specified edit, you gave a pseudo–op out of order. It applies to a specific module that you must specify with a preceding .MODULE pseudo–op.

Suggested User Response: Make certain that a .MODULE pseudo–op precedes those pseudo–ops that refer to a specific module.

?MKLMNF **Module |module| was not found in file |file|**

Description: MAKLIB cannot find the specified module in this file. If you are trying to apply a .FIX to a master library, you may have specified a non–existent module name on a /MODULE pseudo–op. You can also receive this error when you perform an operation that does not involve .FIX files. You may have specified several module names and supplied them out of order. The module in question may be in the file. But since MAKLIB processes files in a sequential manner, it will not find all modules.

Suggested User Response: Use /LIST to see if the specified module is really in the library. If not, you cannot perform this function.

%MKLMNI	**/MASTER module names are ignored when patching**
Description:	For editing, the correct way to specify module names is with the .MODULE pseudo-op in the .FIX file. /MASTER is not required for this type of command. MAKLIB ignores it.
?MKLMTF	**/MASTER switch cannot be used on transaction file**
Description:	The command string is incorrect.
Suggested User Response:	Retype the command string with the switch in the correct place. Include only /MASTER on the master file.
?MKLNEA	**Not enough arguments specified**
Description:	The command string is incomplete. This error usually means that you omitted a required file spec.
Suggested User Response:	Retype the command string in the correct format: Destination file spec=Source file spec1/Switches,Source file spec2/Switches,...Source file specn/Switches
?MKLNEC	**Not enough memory is available**
Description:	MAKLIB cannot obtain enough memory to process this command.
Suggested User Response:	Break the .REL files into smaller numbers of modules, or break large modules into smaller modules.
?MKLNEI	**Null argument to .EDIT is illegal**
Description:	You did not specify an edit identifier on the .EDIT pseudo–op.
Suggested User Response:	Each .EDIT pseudo–op requires an argument (up to 6 characters) that identifies the edit.
%MKLNIO	**Output file |file| will not be indexed**
Description:	The output file will not include an index. To add an index to the file, issue a separate MAKLIB command with /INDEX.
?MKLNMS	**Null specification to .MODULE |edit|**
Description:	In your edit, you included a .MODULE pseudo–op without the module name.
Suggested User Response:	Correct the .FIX file by supplying a module name on each .MODULE pseudo–op.

?MKLNPC	**No program code was found for module in edit [edit]**
Description:	You have specified a non–existent module in the master file.
?MKLNPS	**No program names were specified for file [file]**
Description:	You tried to manipulate some of the modules in the specified file, but you did not supply any module names.
Suggested User Response:	In order to /EXTRACT or /DELETE modules from a file, supply the module name on the file switch.
?MKLNRP	**Not a recognized position switch: [value]**
Description:	You gave an illegal position indicator on a .INSERT pseudo–op.
Suggested User Response:	Use one of the three legal position indicators: BEFORE, AFTER, or REPLACE.
?MKLNTM	**Not enough transaction modules were specified**
Description:	You did not supply enough replacement module names to perform the /REPLACE operation.
Suggested User Response:	When you give the MAKLIB command string, make certain that there is a corresponding replacement module for each module you intend to replace in the master library.
?MKLODD	**Output device must be DISK or DECTAPE**
Description:	In your MAKLIB command, you specified an illegal output library device.
Suggested User Response:	Retype the command string and use disk for the output device of the library file.
?MKLPEF	**Premature end–of–file during edit [edit] in file [file]**
Description:	While processing the indicated edit, MAKLIB encountered an unexpected end–of–file in the .FIX file. This error usually occurs when you omit the .ENDE pseudo–op.
Suggested User Response:	Check the .FIX file for errors, and look especially for .ENDI and .ENDE pseudo–ops.

%MKLPEP	**Precluded edit [edit] is present in module**
Description:	The module you are editing contains an active edit that your current edit precludes with the .ASSOCIATED pseudo–op. MAKLIB still applies your edit.
%MKLPES	**Purging EXTERNAL symbol [symbol] may give bad REL file**
Description:	One of the symbols that this edit PURGEd from the symbol table was an EXTERNAL symbol. With this symbol removed, it may not be possible to LINK the file correctly.
?MKLRBF	**REQUEST or REQUIRE block is badly formatted**
Description:	In the master library being processed, MAKLIB encountered a REQUEST block (REL Block type 17) or REQUIRE block (REL Block type 16) that was not in the expected format.
Suggested User Response:	The library file may be damaged. Try to rebuild it.
%MKLREM	**Required edit [edit] is missing from module**
Description:	The module you are editing is missing an active edit that your current edit requires with the .ASSOCIATED pseudo–op. MAKLIB still applies your current edit.
%MKLRER	**Required edit [edit] is inactive in module**
Description:	The module you are editing contains an inactive edit that your current edit requires with the .ASSOCIATED pseudo–op. MAKLIB still applies your current edit.
%MKLRIA	**Edit [edit] tried to .REINSERT already active edit**
Description:	Your current edit contains a .REINSERT pseudo–op that attempts to activate an edit that is already active.
%MKLRIE	**Edit [edit] tried to .REMOVE already inactive edit**
Description:	Your current edit contains a .REMOVE pseudo–op that attempts to deactivate an edit that was previously deactivated.
%MKLRIN	**Edit [edit] tried to .REINSERT non–existent edit**
Description:	Your current edit contains a .REINSERT pseudo–op that attempts to activate a non–existent edit.

%MKLRNE	**Edit [edit] tried to .REMOVE non–existent edit**
Description:	Your current edit contains a .REMOVE pseudo–op that attempts to deactivate a non–existent edit.
?MKLRTL	**.INSERT's REPLACE argument of [value] too large for module [module]**
Description:	On an .INSERT pseudo–op you specified a number of instructions to be replaced. This number exceeds the number of instructions in the module beyond the starting replacement address.
Suggested User Response:	Correct the argument to the REPLACE keyword on the .INSERT pseudo–op.
?MKLSCE	**Storage for patch code was exhausted in edit [edit]**
Description:	MAKLIB allocates a fixed amount of space for processing the new code inserted from a .FIX file. Your .FIX file contains more code than MAKLIB can process in the allocated space.
Suggested User Response:	Try to break your .FIX file into several .FIX files with fewer lines of new code.
?MKLSIO	**Switches are illegal on output**
Description:	The command string is incorrect. Switches are not recognized on the output file.
Suggested User Response:	Retype the command string including any necessary switches with the appropriate input files.
%MKLSNF	**Symbols not found for module**
Description:	There are no symbols in the master file for the module that you wish to edit.
?MKLSSE	**Storage for patch symbols was exhausted during edit [edit]**
Description:	MAKLIB allocates a fixed amount of storage for processing the new symbols from .FIX files. Your .FIX file contains more symbols than MAKLIB can process in the allocated space.
Suggested User Response:	Try to break your .FIX file into several .FIX files with fewer symbols.

%MKLTBF	**TRACE block is badly formatted in module**
Description:	One of the TRACE blocks (REL Block type 1060) for this module is not in the expected format. The .REL file may be damaged. Since LINK ignores trace blocks when reading a file, you may still be able to load from this .REL file.
%MKLTFI	**Transaction file ignored**
Description:	You included a transaction file in the command string to create an indexed library.
Suggested User Response:	You must create the indexed library as a single operation. If there are several operations you must perform on the library, the command to index the library should be the last command that you issue.
?MKLTFR	**All transaction files rejected by condition.**
Description:	Processing cannot continue because the transaction files did not meet the condition you specified (as size, creation date, etc.).
?MKLTMN	**Too many module names ..... stopped at |module|**
Description:	The command string is too long.
Suggested User Response:	Retype the command string as several shorter commands. It is unlikely that this message will appear, since MAKLIB now allows up to 100 switch arguments for each command string.
?MKLTMS	**Too many switches**
Description:	You included too many switches on a file specification, or you specified some of the switches in an illegal combination.
Suggested User Response:	Retype the command string in a correct format.
?MKLUDF	**Module |module| in edit |edit| contains undefined symbol(s)**
Description:	Your current edit contains undefined symbols in the indicated module.
Suggested User Response:	Make certain that each new symbol introduced in your .FIX file has a value associated with it.

?MKLWIO	**Wild cards illegal for output file specification**
Description:	You gave an output file name in the command string that included wildcard characters (either * or ?).
Suggested User Response:	Since wildcard characters are illegal for the output file name, retype the command string with an explicit output file name.
?SCNSVR	**Switch value required on [switch]**
Description:	The specified switch requires a value.
Suggested User Response:	Retype the command string and supply a value for the switch. The format for providing a value is /switch:value or /switch:(value1,value2).
?WLDLKE	**Protection failure file [file]**
Description:	The specified file is protected. You do not have the privileges to access it.
Suggested User Response:	If this is the output file, you need privileges to create a file in the directory you specified. If it is an input file, you need privileges to read that file from that particular directory.
?WLDLKE	**Non-existent file [file]**
Description:	The specified file does not exist.
Suggested User Response:	This error usually occurs when the name of an input file is incorrect. Retype the command string with the correct input file name.

5.0 Technical Notes

The following is supplementary information related to editing libraries with MAKLIB. Section 5.1 describes TRACE blocks (REL Block type 1060). Section 5.2 contains the format for code insertion in .FIX files.

5.1 Format of TRACE Block Data (REL Block Type 1060)

MAKLIB uses the TRACE block to include, in the .REL file, information for verifying and changing the patch status of a program. The format of the TRACE block follows.

The first part of the TRACE block is the static area. This area appears in each module affected by the particular edit. The static areas give information common to all modules affected by an edit and the variable gives the changing data on the particular edit as it goes from module to module.

TB$HED	REL Block Type	Length of Block
TB$EDT	SIXBIT Edit Name (Up to 6 chars.)	
TB$STA	–1 if active	Who Last Affected
TB$MAK	Who Created	Date (15 BIT)
TB$INS	Who Installed	Date (15 BIT)
TB$FUT	Reserved for Future Use	
TB$LEN	# of Assoc. Edits	# of PCO Groups

The static area, which repeats in each module, is followed by a variable area. The variable area consists of two parts. The first gives data on the associated edit status for this module, and the second gives the actual program change orders (PCO's). The length of each of these areas appears in the static area of the TRACE block.

For each associated edit, the following group appears:

TB$AEN	SIXBIT Edit Name of Assoc. Edit		
TB$AES	X	Reserved for Future Use	0B0 If can't be present 1B0 If must be present

After the associated edit groups appear (if there are any), the PCO groups for that module appear. There are currently three types of program change groups: INSERT, REMOVE, and REINSERT. They can appear in any order and the total number is variable.

INSERT PCO:

TB$PCO	PCO Type Code (1)	Length of Group
TB$DAT	Instrs. INSERTed	Addr. of INSERT
TB$PAT	New Addr. of Original Code	Addr. of Patch Code

REMOVE PCO:

TB$PCO	PCO Type Code (2)	Length of Group
TB$REN	SIXBIT Edit Name	

RE–INSERT PCO:

TB$PCO	PCO Type Code (3)	Length of Group
TB$RIN	SIXBIT Edit Name	

ALTER PCO:

TB$PCO	PCO Type Code (4)	Length of Group
TB$DAT	Unused	Addr. of Alter
TB$PAT	New Addr. of Original Code	Unused

5.2 Format of Code Insertion

The four formats of code insertion in a .FIX file are shown here. Notice that, in all cases, the patch ends with exactly two JUMPA instructions. Thus, the last instruction of the patch can at most skip a single instruction and still return control to the original code. The diagrams show how MAKLIB generates the instructions to insert your patch.

1. To insert any instruction or series of instructions (code) BEFORE a location, use this format:

   .INSERT location, BEFORE, <original instruction>

```
LOCATION:    JUMPA    %PATCH
%PATCH:      First Patch Instruction
             Second Patch Instruction
                   .
                   .
                   .
             Last Patch Instruction
             Original Patch Instruction
             JUMPA 1, LOCATION+1
             JUMPA 2, LOCATION +2
             Any "Literals"
```

   The actual label created at the location of the patched–in code is of the form:

   "%"<edit–name><edit–part>

   where the edit–part is from "A" to "Z", incremented for each .INSERT in the edit.

2. To insert any instruction or series of instructions (code) AFTER a location, use this format:

   .INSERT location, AFTER, <original instruction>

```
LOCATION:      JUMPA    %PATCH
%PATCH:        Original Instruction
               First Patch Instruction
               Second Patch Instruction
                         .
                         .
                         .
               Last Patch Instruction
               JUMPA 1, LOCATION+1
               JUMPA 2, LOCATION +2
               Any "Literals"
```

3. To REPLACE a single instruction, use the format:

   .INSERT location,REPLACE, <original instruction>

```
LOCATION:      JUMPA    %PATCH
               Original Instruction
%PATCH:        First Patch Instruction
               Second Patch Instruction
                         .
                         .
                         .
               nTH (Last) Patch Instruction
               JUMPA 1, LOCATION+n
               JUMPA 2, LOCATION +n+1
               Any "Literals"
```

   If you do not insert instructions (n=0), then the return is to LOCATION+1 and LOCATION+2.

4. To REPLACE more than one instruction at a location, use the format:

   .INSERT location,REPLACE:m, <original instruction>

```
LOCATION:     JUMPA    %PATCH
              Original Instruction
%PATCH:       First Patch Instruction
              Second Patch Instruction
                    .
                    .
                    .
              Last Instruction of Patch
              JUMPA 1, LOCATION+m
              JUMPA 2, LOCATION +m+1
              Any "Literals"
```

   If you do not specify m, or it is zero, the effect is the same as the REPLACE keyword without an argument. In other words, one word is skipped over on return for every one inserted.

Index

TOPS–10
MAKLIB
User's Guide
AA–M472A–TB

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) ______

Name ______ Date ______

Organization ______ Telephone ______

Street ______

City ______ State ______ Zip Code or Country ______

---------Do Not Tear - Fold Here and Tape---

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE PUBLICATIONS
200 FOREST STREET MR1–2/L12
MARLBOROUGH, MASSACHUSETTS 01752

--------- Do Not Tear - Fold Here and Tape---

Cut Along Dotted Line