**REPORT TITLE**

EMPLIB: A SEQUENTIAL FILE PROGRAM LIBRARIAN

**ABSTRACT**

EMPLIB, written for use on a CDC 6000 computer operating under Scope 3, is a librarian program whose function is to maintain an active library and a separate permanent archive of program UPDATE and object files on a sequential storage device such as a magnetic tape reel. The ENPLIB librarian can perform readout or alteration of the library or archive, and also certain file-positioning actions and program object file editing.
<table>
<thead>
<tr>
<th>KEY WORDS</th>
<th>LINK A</th>
<th>LINK B</th>
<th>LINK C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ROLE</td>
<td>WT</td>
<td>ROLE</td>
</tr>
<tr>
<td>Computer Programs</td>
<td>8</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Librarians</td>
<td>8</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Information Retrieval</td>
<td>8</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Utility Programs</td>
<td>8</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FORTRAN</td>
<td>8</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
HDL-TR-1591

EMPLIB: A SEQUENTIAL FILE PROGRAM LIBRARY

by

William T. Wyatt, Jr.

April 1972

This work was sponsored by the Defense Nuclear Agency under subtasks EA091 and EA094.

U.S. ARMY MATERIEL COMMAND,
HARRY DIAMOND LABORATORIES
WASHINGTON, D.C. 20438

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.
ABSTRACT

EMPLIB, written for use on a CDC 6000 computer operating under Scope 3, is a librarian program whose function is to maintain an active library and a separate permanent archive of program UPDATE and object files on a sequential storage device such as a magnetic tape reel. The EMPLIB librarian can perform readout or alteration of the library or archive, and also certain file-positioning actions and program object file editing.
# CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>3</td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>7</td>
</tr>
<tr>
<td>2. THE LIBRARIAN</td>
<td>7</td>
</tr>
<tr>
<td>3. LIBRARIAN DIRECTIVES</td>
<td>8</td>
</tr>
<tr>
<td>3.1 CREATE</td>
<td>8</td>
</tr>
<tr>
<td>3.2 CREATEARCH</td>
<td>8</td>
</tr>
<tr>
<td>3.3 End of librarian input</td>
<td>8</td>
</tr>
<tr>
<td>3.4 ADD and ADDB</td>
<td>9</td>
</tr>
<tr>
<td>3.5 TOC</td>
<td>9</td>
</tr>
<tr>
<td>3.6 FILLS</td>
<td>9</td>
</tr>
<tr>
<td>3.7 SKIP and SKIPB</td>
<td>9</td>
</tr>
<tr>
<td>3.8 NOREWIND</td>
<td>9</td>
</tr>
<tr>
<td>3.9 CHANGE and CHANGEB</td>
<td>9</td>
</tr>
<tr>
<td>3.10 RENAME and RENAMEB</td>
<td>10</td>
</tr>
<tr>
<td>3.11 DROP and DROPB</td>
<td>10</td>
</tr>
<tr>
<td>3.12 KEEP and KEEPB</td>
<td>10</td>
</tr>
<tr>
<td>3.13 HISTORY</td>
<td>10</td>
</tr>
<tr>
<td>3.14 RCH</td>
<td>10</td>
</tr>
<tr>
<td>3.15 COPY and COPYB</td>
<td>10</td>
</tr>
<tr>
<td>3.16 SELECT</td>
<td>10</td>
</tr>
<tr>
<td>3.17 REFUSE</td>
<td>11</td>
</tr>
<tr>
<td>3.18 ENDFILE</td>
<td>11</td>
</tr>
<tr>
<td>3.19 REWIND</td>
<td>11</td>
</tr>
<tr>
<td>3.20 FIND and FINDB</td>
<td>11</td>
</tr>
<tr>
<td>3.21 REPLACE and REPLACEB</td>
<td>11</td>
</tr>
<tr>
<td>4. LIBRARIAN ERROR MESSAGES</td>
<td>11</td>
</tr>
<tr>
<td>4.1 Directive format errors</td>
<td>12</td>
</tr>
<tr>
<td>4.2 Other errors</td>
<td>13</td>
</tr>
<tr>
<td>5. USER HINTS</td>
<td>13</td>
</tr>
<tr>
<td>5.1 List of directives</td>
<td>14</td>
</tr>
<tr>
<td>5.2 File actions</td>
<td>14</td>
</tr>
<tr>
<td>5.3 Examples of usage</td>
<td>15</td>
</tr>
<tr>
<td>Appendix A. Sample Output</td>
<td>18</td>
</tr>
<tr>
<td>Appendix B. Program Listing</td>
<td>23</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

EMPLIB is a program written in CDC Fortran Extended and Compass for use on CDC 6000-series computers. It has been tested and run under Scopes 3.2 and 3.3, and requires about 54,200 words (octal) to load and execute. EMPLIB is a librarian program whose function is to maintain a library of frequently used program UPDATE files (called "source" files here) and program object files (called "binary" files here, i.e., the compiler or assembler object output). The term "file" is defined here as a string of data terminated by an EOF. The library is kept on a magnetic tape or other permanent sequential data storage device. EMPLIB also maintains an archive magnetic tape of program source or binary files to be saved indefinitely. The user may run the librarian program EMPLIB and cause it to perform certain library or archive functions by placing directive cards in the input card stream to be read by the librarian. These directive cards are processed sequentially, allowing library alteration, program file readout, user-assigned filenames for readin and readout functions, certain filename actions such as rewind, endfile, and skipfile, and archive additions or readout. The term "filename" is defined here as a logical file name (i.e., LGO, TAPE1, OLOPL, etc.).

2. THE LIBRARIAN

The librarian uses nine working filenames for various functions. All functions but one are assigned a one- or two-letter mnemonic and are associated by default with certain filenames which may be altered by the user during execution of the librarian. The file functions, mnemonics, default filenames, and purpose are listed below:

<table>
<thead>
<tr>
<th>Function</th>
<th>Mnemonic</th>
<th>Filename</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>card input</td>
<td>I</td>
<td>INPUT</td>
<td>Contains EMPLIB directives.</td>
</tr>
<tr>
<td>print output</td>
<td>O</td>
<td>OUTPUT</td>
<td>Contains printed output.</td>
</tr>
<tr>
<td>library</td>
<td>L</td>
<td>EMPLIB</td>
<td>Contains the program library.</td>
</tr>
<tr>
<td>archive</td>
<td>A</td>
<td>ARCHIV</td>
<td>Contains the program archive.</td>
</tr>
<tr>
<td>source input</td>
<td>SI</td>
<td>NEWPL</td>
<td>Source files read from NEWPL.</td>
</tr>
<tr>
<td>source output</td>
<td>SO</td>
<td>OLOPL</td>
<td>Source files written to OLOPL.</td>
</tr>
<tr>
<td>binary input</td>
<td>BI</td>
<td>LGO</td>
<td>Binary files read from LGO.</td>
</tr>
<tr>
<td>binary output</td>
<td>BO</td>
<td>XQT</td>
<td>Binary files written to XQT.</td>
</tr>
<tr>
<td>scratch</td>
<td>none</td>
<td>TAPE40</td>
<td>Scratch file for librarian.</td>
</tr>
</tbody>
</table>

All of the file functions, with the exception of the scratch function, may be assigned different filenames by use of the FILES directive described later. The filenames accessed by the librarian must all be odd-parity files as distinct from even-parity BCD files. The librarian can, of course, access an odd parity file onto which a BCD file has been copied. The terms file and binary file as used here both refer to files with odd parity. The difference between the two types of files is one of name only, and is conventionalized so that program UPDATE files are designated as source files and program object files are designated as binary files. The directives SELECT and REFUSE work properly only with binary files that are, in fact, program object files. Otherwise, any data file may be treated as a program source or binary file and manipulated by the librarian. The first file on the library filename is intended to be the librarian program object file, where it may be easily copied off and executed. (If the library tape is executed directly, the system loader will unload the tape, preventing later access to the library.) The second file contains a table of the library contents. Subsequent files are source and binary files previously placed in the library. Each file is identified in the table of contents by a name identifier, a version identifier, a mode identifier (to distinguish whether it is a source file or a binary file), and date of entry into the library. The name and version
identifiers must be from one to ten characters with no imbedded blanks or commas. The version identifier is optional and will be all blanks if not specified by the user. The archive filename contains two data files for each source or binary file kept on it. The first is a file containing the table of contents information about the source or binary file, and the second data file is the source or binary file itself. The end of the archive is denoted by a file containing just the one word "LAST."

3. LIBRARIAN DIRECTIVES

The various functions the librarian can perform will be illustrated through their use in the following examples. The completed output is listed in Appendix A. It is assumed in the first example that the librarian object file has been copied to the filename EMPLIB (so as to allow creation of the library) and that a magnetic tape has been assigned to the filename ARCHIV. Execution of the librarian causes the filename EMPLIB to be rewound if the first directive is not a FILES directive; thus the library filename must be changed immediately if it is not to be EMPLIB.

Directives are free-field, but must have a dollar sign in column one. Directives and identifiers must be separated by blanks, unless commas are required. The librarian will copy each directive card to the print file as encountered and then add a description of any action taken. On the print file, directives can be recognized by the single dollar sign, whereas statements originated by the librarian begin with "EMPLIB $$.

3.1 CREATE

CREATE causes a library to be created on the filename attached to the library function. Physically, the first file is skipped on the library filename and a table of contents file is written which records the first file as "EMPLIB" and the second file as "TOC." The library must be created (establishing a table of contents) before any library additions can be performed. In fact, a table of contents is required by all but the following directives: CREATE, CREATEARCH, FILES, SKIP, SKIPB, HISTORY, ENDFILE, REWIND, FIND, AND FINDB. The directives SELECT and REFUSE may or may not require a table of contents. $CREATE

3.2 CREATEARCH

CREATEARCH causes an archive to be established on the filename attached to the archive function. Physically, the hollerith word "LAST" is written on the archive. The archive must be established before any archive additions can be performed. The archive is rewound before and after the creation. $CREATEARCH

3.3 End of librarian input

The sequence of directives is terminated by a 7-8-9 card. If the last operation on the source output or binary output filename was a write-end-of-record, the filename is EOF'd and backspaced before execution is ended. All the following directives may be given in the same or any subsequent execution of the librarian once the library has been created. The file name/versions specified must be in the library when the directive is processed by the librarian, except for new name/versions in ADD (ADDB) and RENAME (RENAMEB), and
except for the archive directive FIND (FINDB). File name/versions appearing with the FIND (FINDB) directive must already be on the archive when the directive is processed.

3.4 ADD and ADDB

ADD (ADDB) causes the source (binary) file on the source (binary) input filename to be added to the library, and assigns it to the name and version specified. The source (binary) input filename is rewound before reading is begun, unless suppressed by a NOREWIND directive (discussed later). $ADD PROG VERS

3.5 TOC

TOC causes a table of contents of the library to be printed. $TOC

3.6 FILES

FILES causes the file functions whose mnemonics are specified on the directive card to be reassigned different filenames. A reassignment consists of the mnemonic, one or more blanks, and the new filename, in that order. Multiple reassignments must be separated by commas. Old filenames whose last operation was a write-end-of-record are EOF’d and back-spaced before being detached from a file function when the reassignment is made. This directive may be issued even if the library has not been created. $FILES SI OLD, BI AGO, SO NEW

3.7 SKIP and SKIPB

SKIP (SKIPB) causes the number of files specified to be skipped in a forward direction on the source (binary) input filename. Up to 999 files may be skipped with one directive. If the number of files to be skipped is not specified, one file is skipped. $SKIPB 2

3.8 NOREWIND

NOREWIND suppresses the automatic rewind of the source (binary) input filename for the next (and only the next) ADD (ADDB) or CHANGE (CHANGEB) directive encountered. $NOREWIND

3.9 CHANGE and CHANGEB

CHANGE (CHANGEB) causes the source (binary) file name/version specified to be replaced on the library by the next file encountered on the source (binary) input filename. The filename is automatically rewound before reading unless suppressed, as in this example, by a NOREWIND directive. The present data is placed in the library table of contents for the file changed. The file changed must already be in the library. $CHANGE PROG VERS
3.10 **RENAME** and **RENAMEB**

RENAME (RENAMEB) causes the first source (binary) file name/version given on the card to be renamed in the table of contents file with the second source (binary) file name/version given on the card. The first and second file name/version must be separated by a comma. $RENAME PROG VERS, PROGA NEWNAME

3.11 **DROP** and **DROPB**

DROP (DROPB) causes the source (binary) file name/version to be removed from the library and its entry in the table of contents file to be deleted. The first file on the library (the EMLIB binary file) will never be dropped, since this will cause the library to be scrambled. $DROP NEWPROG

3.12 **KEEP** and **KEEPB**

KEEP (KEEPB) causes the source (binary) file name/version specified to be added to the archive. The specified file name/version must already be in the library. Once added to the archive, a file cannot be removed from the archive by the librarian. $KEEPB PROG VERS

3.13 **HISTORY**

HISTORY causes the contents of the archive to be scanned and a list of the file name/versions encountered to be printed. This directive may be processed by the librarian even if the library has not been created; only the archive need exist. $HISTORY

3.14 **RUN**

RUN causes the first binary file on the library with the specified name to be copied to the binary output filename irrespective of the program version. Thus, only the program name need be specified. The terminating EOF is not copied, so further material may be copied to the binary filename to complete the desired load module. $RUN PROG

3.15 **COPY** and **COPYB**

COPY (COPYB) causes the source (binary) file name/version specified on the library to be copied to the source (binary) output filename. The terminating EOF is not copied, just as for the RUN directive. $COPY PROGA NEWNAME

3.16 **SELECT**

SELECT causes the specified binary file name/version on the library to be scanned for the named object programs or subprograms, which are copied as encountered to the binary output filename with no terminating EOF. On the directive card the binary file name/version must be the first identifiers after the SELECT word, followed by a comma, and then followed by the program or subprogram names separated by commas. If the file name/version is
omitted so that the next non-blank character after the directive is a comma, the next file on
the binary input filename will be scanned for the named programs and subprograms; this ac-
tion does not require a table of contents. If the last non-blank character on the card is a
comma, continuation cards will be read until the final non-blank card character is not a comma.
Continuation cards must not contain a dollar sign in column one, and must contain information
in columns one through 79 only. Up to 100 program or subprogram names may be specified in
the directive. A list of all object routines encountered and their selection or refusal is
printed. The largest object routine that can be processed by SELECT or REFUSE must be
less than 6000 words long. A statement of the maximum size processed is printed at end of
execution. $SELECT PROG VERS, ISO, SPLITR, SPLITC.

$SELECT, ISO, SPLITR, SPLITC.

3.17 REFUSE

REFUSE causes the same action as SELECT, except that specified object program and
subprogram names are not copied to the binary output filename and all others encountered
are copied. Empty records are not copied. Up to 100 names may be specified for re-

REFUSE causes the file function whose mnemonic is specified to have an EOF written
on the filename assigned to the file function. Only one file function may be specified on the
directive card and only the file functions BO and SO may be endfiled with this directive.
$ENDFILE FO

3.19 REWIND

REWIND causes the file function whose mnemonic is specified to have its assigned file-
name rewound. If information had been written to the filename, end-of-information termina-
tors are written to the filename before it is rewound. The file functions I,O, and L cannot be
rewound with this directive. $REWIND SO

3.20 FIND and FINDB

FIND (FINDB) causes the archive to be searched for the source (binary) file name/
version specified, which is then copied to the source (binary) output filename. No EOF is
written, just as for the COPY (COPYB) directive. The directives FIND and FINDB may be
processed by the librarian even if the library has not been created; only the archive is re-
quired to exist. $FINDB PROG VERS

3.21 REPLACE and REPLACEB

REPLACE (REPLACEB) causes the source (binary) file name/version specified to be
replaced on the library on the next file encountered on the source (binary) input filename,
and given a new name/version label. This combines the functions of CHANGE (CHANGEB)
and REPLACE (REPLACEB). The directive format is the same as for the RENAME
(RENAMEB) directive. The source (binary) input filename is rewound before reading is
begun, unless suppressed by a NOREREWIND directive.

4. LIBRARIAN ERROR MESSAGES

When the librarian detects an error involving the directive card being processed, a mes-
sage describing the nature of the error: i.e., printed and the rest of the librarian card input file
is copied to the print output file, after which execution is terminated by a call to the nonexist-
ent subroutine ABORT which causes a mode one (address-out-of-range) error termination.
Terminators are assured to be on any source or binary output filename if the filename has been written on, just as for a normal termination.

If another kind of error is detected, an informative message is printed and execution is terminated immediately with a CALL ABORT. Terminators are not assured for filenames assigned to output functions at the time the error was detected.

4.1 Directive format errors

The following errors use the ABORT termination after checking file terminators:

1. Missing or misplaced dollar sign on directive card. The dollar sign must be in column one.

2. Improper directive. A directive cannot be found on the card.

3. Unrecognized directive. The specified directive is not familiar to the librarian.


5. Missing program filename. A program file name cannot be found on the card when one is required.

6. Program file name too long. The specified program file name is longer than 10 characters.

7. Program file version too long. The specified program file version is longer than 10 characters.

8. Program name/version not in table of contents. The specified name/version is not on the library and the directive cannot be executed.

9. Adding file already in table of contents. The specified name/version/mode's already in the library; a unique name/version/mode must be specified.

10. Missing comma. A needed comma is missing between the old name/version and the new name/version on a RENAME, RENAMES, REPLACE, or REPLACEB directive.

11. Word is too long. A word is longer than 10 characters on a FILES directive card. In fact, SCOPE can handle filenames only up to seven characters long, so care should be taken not to use 8, 9, or 10 character filenames.

12. Unrecognized file type. The file function type specified is not recognized.

13. More than 100 record names. Too many program and subprogram names are listed in a SELECT or REFUSE directive.


15. Illegal directive for the file type. The directive is not allowed for the file function type specified.
16. Illegal number. Unrecognizable number on a SKIP or SKIPB card; 999 is the maximum allowed.

17. Program file name/version not on archive. The name/version specified by a FIND or FINDB directive is not in the archive.

4.2 Other errors

The following errors cause an immediate CALL ABORT termination:

1. KEEP read parity error. A read parity error occurred while reading the library for a KEEP or KEEPB directive.

2. KEEP write parity error. A write parity error occurred while writing to the archive for a KEEP or KEEPB directive. The former contents of the archive are intact, but an end-of-archive record no longer exists.

3. FIND read error. A read parity error occurred while reading the archive for a FIND or FINDB directive.

4. HISTORY read error. A read parity error occurred while reading the archive for a HISTORY directive.

5. GETTOC parity error. A read parity error occurred while the librarian was trying to read the table of contents file.

6. Empty file. The filename specified as the location of a program file was empty.

7. CPYFIL read parity error. A read parity error occurred while the librarian was skipping a file.

8. I/O error in CPYBUF. An I/O error occurred while the librarian was copying a file.

9. End-of-information encountered. An EOI was encountered while trying to copy a file; i.e., the filename was short-terminated.

10. TOC write error in PWDFIL. A write parity error occurred while the librarian was writing the table-of-contents file to the library for a library alteration directive.

11. Read error in CPYREC. A read parity error occurred while the librarian was reading or binary input filename during processing of a SELECT or REFUSE directive.

12. Write error in CPYREC. A write parity error occurred while the librarian was copying a program or subprogram record to the binary output filename during processing of a SELECT or REFUSE directive.

5. USER HINTS

The following information will be useful to the EMPLIB user.
5.1 List of directives

PROG and PROGA are program file names, and VERS and VERSA are program file versions in the following examples. Items enclosed in parenthesis are optional. An asterisk denotes the file is rewound before reading unless suppressed by a NOREWIND directive.

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Directive</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>L</td>
<td>$CREATE</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>$CREATEARCH</td>
</tr>
<tr>
<td>SI*</td>
<td>L</td>
<td>$ADD PROG ,VERS)</td>
</tr>
<tr>
<td>BI*</td>
<td>L</td>
<td>$ADDB PROG (VERS)</td>
</tr>
<tr>
<td>L</td>
<td></td>
<td>$TOC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$FILES BI ABC, A PQR7826</td>
</tr>
<tr>
<td>SI</td>
<td></td>
<td>$SKIP (5)</td>
</tr>
<tr>
<td>BI</td>
<td></td>
<td>$SKIPB (999)</td>
</tr>
<tr>
<td>SI*</td>
<td>L</td>
<td>$CHANGE PROG (VERS)</td>
</tr>
<tr>
<td>BI*</td>
<td>L</td>
<td>$CHANGEB PROG (VERS)</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>$RENAME PROG (VERS), PROGA (VERSA)</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>$RENAMEB PROG (VERS), PROGA (VERSA)</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>$DROP PROG (VERS)</td>
</tr>
<tr>
<td>L</td>
<td>A</td>
<td>$DROPB PROG (VERS)</td>
</tr>
<tr>
<td>L</td>
<td>A</td>
<td>$KEEP PROG (VERS)</td>
</tr>
<tr>
<td>A</td>
<td></td>
<td>$KEEPB PROG (VERS)</td>
</tr>
<tr>
<td>L</td>
<td>BO</td>
<td>$HISTORY</td>
</tr>
<tr>
<td>L</td>
<td>SO</td>
<td>$HISTORY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$RUN PROG</td>
</tr>
<tr>
<td>L</td>
<td>BO</td>
<td>$RUNB PROG (VERS)</td>
</tr>
<tr>
<td>L</td>
<td>BO</td>
<td>$COPY PROG (VERS)</td>
</tr>
<tr>
<td>L</td>
<td>SO</td>
<td>$COPYB PROG (VERS)</td>
</tr>
<tr>
<td>L</td>
<td>BO</td>
<td>$SELECT PROG (VERS), SUBA, SUBB, SUBC</td>
</tr>
<tr>
<td>BI</td>
<td>BO</td>
<td>$SELECT, SUBA, SUBB, SUBC</td>
</tr>
<tr>
<td>L</td>
<td>BO</td>
<td>$REFUSE PROG (VERS), SUBA, SUBB, SUBC</td>
</tr>
<tr>
<td>BI</td>
<td>BO</td>
<td>$REFUSE, SUBA, SUBB, SUBC</td>
</tr>
<tr>
<td>L</td>
<td>BO or SO</td>
<td>$ENDFILE BO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$REWIND SI</td>
</tr>
<tr>
<td>A</td>
<td>SO</td>
<td>$FIND PROG (VERS)</td>
</tr>
<tr>
<td>A</td>
<td>BO</td>
<td>$FINDB PROG (VERS)</td>
</tr>
<tr>
<td>SI*</td>
<td>L</td>
<td>$REPLACE PROG (VERS), PROGA (VERSA)</td>
</tr>
<tr>
<td>SI*</td>
<td>L</td>
<td>$REPLACEB PROG (VERS), PROGA (VERSA)</td>
</tr>
</tbody>
</table>

All directives except CREATE which use the library (L) as input or output require a created library. All directives except CREATEARCH which use the archive (A) as input or output require a created archive.

5.2 File Actions

The librarian checks the first directive encountered and, if it is not FILES directive, rewinds the library (which has the filename EMPLIB) and looks to see if a table of contents exists. If it is a FILES directive, rewinding the library file is deferred to just prior to processing the next directive.

All directives which use the library as output cause the entire library to be copied to the scratch filename TAPE40 and recopied in its modified form back to the library filename. If the library is of substantial length and if more than one or two directives of this kind are to
be processed, much PP time will be saved if the library tape is copied to a disk filename before librarian execution and then recopied from the disk filename back to the library tape after librarian execution. The library filename must be the disk filename, of course. This also helps protect the library tape from write parity errors.

A good practice is periodically to copy the entire library and the entire archive to a backup library tape and a backup archive tape, to avoid loss of program files if the first-line copies are impaired by permanent parity errors.

If the library is of short length, it may be practical to have the library reside on a permanent disk file instead of on a magnetic tape. The archive will generally be too large for this, however.

5.3 Examples of usage

Although it would not be possible to illustrate all the possible uses of the EMPLIB librarian, a few examples will be useful to convey the flexibility and simplicity of the program. The examples are for a Scope 3.3 system. All TOC directives are optional, but are recommended.

1. Update, compilation of changes, and execution.

   JOB, CM54000, TP1
   REQUEST, EMPLIB, (540/NORING)
   COPYBF (EMPLIB, LIB, 1)
   LIB.
   RETURN (EMPLIB)
   UPDATE (P)
   FTN (I=COMPLETE)
   REWIND (XQT)
   COPYBF (XQT, LGO, 1)
   LGO.
   7-8-9
   $TOC
   $COPY NEPHI CORRQ
   $REFUSE NEPHI CORRQ, PHOTON, GROUND
   7-8-9
   (Update input with changes for subroutines PHOTON and GROUND.)
   7-8-9
   (Input data.)
   6-7-8-9

   This could also be accomplished by the following cards between the FTN card and UPDATE input cards:

   LGO.
   7-8-9
   $TOC
   $FILES BO LGO
   $COPY NEPHI CORRQ
   $REFUSE NEPHI CORRQ, PHOTON, GROUND
   7-8-9

15
2. Update, compilation of changes, and alteration of library.

```plaintext
JOB, CM54000, TP1.
REQUEST, EMPLIB (540/RING)
COPYBF (EMPLIB, LIB, 1)
LIB.
UPDATE (P, N, W) (W makes new UPDATE library sequential.)
FTN (I = COMPILB)
LIB.
UNLOAD (EMPLIB)
$TOC
$COPY NEPHI CORRQ
$FILES BO LGO
$REFUSE NEPHI CORRQ, PHOTON, GROUND
7-8-9
(Update input with changes for subroutines PHOTON and GROUND.)
7-8-9
$DROP NEPHI CORRQ
$DROPB NEPHI CORRQ
$ADD NEPHI CORRR
$ADDDB NEPHI CORRRR
$TOC
6-7-8-9
```

More efficient use of the greater speed of disk files would be made by using the following control cards in the previous example:

```plaintext
JOB CM54000, TP1.
REQUEST, ZAP. (540/RING)
COPYBF (ZAP, EMPLIB, 100) (less than 100 files on ZAP)
EMPLIB.
UPDATE (P,N,W)
FTN (I = COMPILB)
EMPLIB.
REWIND (EMPLIB, ZAP)
COPYBF (EMPLIB, ZAP, LGO)
UNLOAD (ZAP)
7-8-9
```

3. Execution of one program.

```plaintext
JOB, CM54000, TP1.
REQUEST, EMPLIB (540/NORING.)
COPYBF (EMPLIB, LIB, 1)
LIB.
RETURN (EMPLIB)
RFL100000.
REDUCE.
XQT.
7-8-9
$RUN NEPHI
7-8-9
```
4. Execution of several programs.

JOB, CM54000, TP1.
REQUEST, EMPLIB (540/NORING.)
COPYBF (EMPLIB, LIB, 1)
LIB.
RETURN (EMPLIB)
XQT.
NEXT.
LAST.
7-8-9
$TOC
$COPYB PROG FIRST
$FILES BO NEXT
$RUN PROGSEC
$FILES BO LAST
$RUN PROGFIN
7-8-9
(Data for PROG/FIRST.)
7-8-9
(Data for PROGSEC.)
7-8-9
(Data for PROGFIN.)
6-7-8-9
Appendix A. SAMPLE OUTPUT
ENLIB $S THF DATE IS 10/29/71 AND THE WORK FILES ARE
BINARY OUTPUT = NOT  GINO OUTPUT = OCT
SOURCE INPUT = LGO  SOURCE INPUT = NEWPL
EMPL LIBRARY = EMPLIB  ARCHIVE KEEP = ARCHIV
EMPL OUTPUT = OUTPUT  EMPLIB INPUT = INPUT

$ADD PROG VERS
ENLIB $S ADDED 3TH FILE (SOURCE PROG  VERS  10/29/71 ) TO ENLIB FROM NEWPL FILE.
$ADD PROG VERS
ENLIB $S ADDED 4TH FILE (BINARY PROG  VERS  10/29/71 ) TO ENLIB FROM LGO FILE.

$TOC
ENLIB $S TABLE OF CONTENTS OF ENLIB
1  EMPLIB  10/29/71  BINARY
   TOC  10/29/71  4 FILES ON LIBRARY
3  PROG VERS  10/29/71  SOURCE
4  PROG VERS  10/29/71  BINARY

$FILES SI OLO, 81 AGO, SO NEW
ENLIB $S ADDED SOURCE INPUT FILE OLD INSTEAD OF NEWPL *
ENLIB $S ADDED BINARY INPUT FILE AGO INSTEAD OF LGO *
ENLIB $S ADDED SOURCE OUTPUT FILE NEW *

$SKIP 2
ENLIB $S SKIPEO 2 FILES ON AGO.

$SNOREWIND

$CHANGER PROG VERS
ENLIB $S CHANGED 4TH FILE (HAS BINARY PROG  VERS  10/29/71 ) IS NOW BINARY PROG  VERS  10/29/71 ) ON ENLIB USING CONTENTS OF AGO FILE.

$TOC
ENLIB $S TABLE OF CONTENTS OF ENLIB
1  EMPLIB  10/29/71  BINARY
   TOC  10/29/71  4 FILES ON LIBRARY
3  PROG VERS  10/29/71  SOURCE
4  PROG VERS  10/29/71  BINARY

$RENAME PROG VERS, PROGA NEWNAME
ENLIB $S RENAMED 3TH FILE (HAS SOURCE PROG  VERS  10/29/71 ) IS NOW SOURCE PROGA NEWNAME 10/29/71 ).

$ ADD NEWPROG
ENLIB $S ADDED 5TH FILE (SOURCE NEWPROG  10/29/71 ) TO ENLIB FROM OLD FILE.

$TOC
ENLIB $S TABLE OF CONTENTS OF ENLIB
1  EMPLIB  10/29/71  BINARY
   TOC  10/29/71  5 FILES ON LIBRARY
3  PROGA NEWNAME  10/29/71  SOURCE
4  PROG VERS  10/29/71  BINARY
9  NEWPROG  10/29/71  SOURCE

$DROP NEWPROG
ENLIB $S DROPPED 5TH FILE (SOURCE NEWPROG  10/29/71 ) FROM ENLIB.

$ TOC
ENLIB $S TABLE OF CONTENTS OF ENLIB
1  EMPLIB  10/29/71  BINARY
   TOC  10/29/71  4 FILES ON LIBRARY
3  PROGA NEWNAME  10/29/71  SOURCE
IAA PROG VERS 10/29/71 BINARY

S KEEP PROG VERS
EMPLIB $SS KEPT 4TH FILE (BINARY PROG VERS 10/29/71) FROM EMPLIB ON ARCHIV FILE.
S KEEP PROG NAME
EMPLIB $SS KEPT 5TH FILE (SOURCE PROG NAME 10/29/71) FROM EMPLIB ON ARCHIV FILE.

S HISTORY
EMPLIB $SS HISTORY OF ARCHIV
KEEP NO. 1 PROG VERS 10/29/71 BINARY
KEEP NO. 2 PROGA NAME 10/29/71 SOURCE

S RUN PROG
EMPLIB $SS COPIED 4TH FILE (BINARY PROG VERS 10/29/71) FROM EMPLIB TO XOT FILE.

S COPY PROG NAME
EMPLIB $SS COPIED 5TH FILE (SOURCE PROGA NAME 10/29/71) FROM EMPLIB TO NEW FILE.

S SELECT PROG VERS,157, SPLITR, SPLITC
EMPLIB $SS COPYING THE FOLLOWING BINARY RECORDS ONTO XOT FROM THE 4TH FILE (BINARY PROG VERS 10/29/71) ON EMPLIB.

SELECTED REFUSED
157 PLOT
158 LLL
SPLITR QUBIT
SPLITC FINDER

S REFUSE PROG VERS, 157, SPLITR, SPLITC
EMPLIB $SS COPYING THE FOLLOWING -SWW- HALFWAY INTO XOT FROM THE 4TH FILE (BINARY PROG VERS 10/29/71) ON EMPLIB.

SELECTED REFUSED
157 PLOT
158 LLL
SPLITR QUBIT
SPLITC FINDER

- END OF COPY -
LINES

END OF COPY

EXECFILE RD
EMPLIB $$$ EXECFILE RD FILE NAMED XOT

EXECINO RD
EMPLIB $$$ REbound RD FILE NAMED NEW

SFILB RD BGO
EMPLIB $$$ MADE BINARY OUTPUT FILE BGO INSTEAD OF XOT

SFIND PROG VERS
EMPLIB $$$ 1ST FILE FOUND (BINARY PROG VERS 10/29/71 ) ON ARCHIV.
EMPLIB $$$ COPIED FILE FOUND TO BGO

SFILES RD GGO, BI BGO
EMPLIB $$$ MADE BINARY OUTPUT FILE GGO INSTEAD OF BGO
EMPLIB $$$ MADE BINARY INPUT FILE GGO INSTEAD OF AGO

EXECINO BI
EMPLIB $$$ REbound BI FILE NAMED GGO

SELECTC ,ISO:SPLITR:SPLITL
EMPLIB $$$ COPYING THE FOLLOWING BINARY RECORDS ONTO GGO FROM BGO

SELECTED

REFUSED

ISO

PLOT

LLL

SPLITR

SPLITL

OUPFIT

FINGER

LOGVAL

REDICT

RITECT

RITRAN

RITRAN

REDRAN

BLANKS

ESPON

LINES

END OF COPY

EMPLIB $$$ MAXIMUM RECORD LENGTH PROCESSED FOR SELECT-REFUSE WAS 2038. 9999 IS MAXIMUM ALLOWED.
EMPLIB $$$ FINISHED $$$
Appendix B. PROGRAM LISTING
PROGRAM EMPLIB

PROGRAM EMPLIB(XIT,XLPL,LGO,EMPLIB=40006,ARCHIV,INPUT=10008,OUTPUT
*FO=10008,NEWL,TAPE4,L,TAPE1=XIT,TAPE2=XLPL,TAPE3=LGO,TAPE4=EMPLIB,
*TAPE5=ARCHIV,TAPE6=INPUT,TAPE7=OUTPUT,TAPE8=NEWL)

COMMON //LMAX,L(6000)
COMMON //ARGS/NAMC,INCR,NEWL,IBOLO,INCR,NAMEC(1000),JTOC,LASTF,
*IARCH
DIMENSION CARD(45),T0C(4,50),MODE(2)
COMMON /FILES/FILNAM(9),FETS(9),XCl)
DIMENSION CARO(8),T0C(4,50),MODE(2)
COMMON /FILES/FILNAM(9),FETS(9),XCl)

INTEGER TOC,Todate,DATE,A,CHAR,OLDATE,DELETE,FILNAM
DATA CHRNAM,,CPHCP6,CANETHD,4ADN
DATA SHREADP6M,4HKLPY,5HKEEPB,6HCREATGE,4HHFING,3D,SHIDB7HASORY4
15 *,6HRENAMETHRENA"EP,6HREFUSE,6HSELECT,SHFILES,6HREWIND,7HENDFILE#
EIHNOREWIND,LISKIP,5SHSKIPB,iOHCREATEARCH, ?HREPLACEOHREPLACEB/,NCHA
R/29/
DATA DOL/IHS/,NOOE/S2HSOURCE,6HOINARY/,LASTH/4HLAST/

MX=0
IMOD=0
JTOC=0
LASTF=0
IARCH=0
IRS=0
JCV=0
IFIRST=3
CALL FTDNIN(1,0)
CALL DATE(QDATE)
CALL GETFIL
PRINT 1,DOIT,FILNAM(1),FILNAM(2),FILNAM(3),FILNAM(4),FILNAM(5)
1 FORMAT(*EMPLIB $IS THE DATE IS *,A10,* AND THE WORK FILES ARE*/
10X,BINARY OUTPUT = *,A7/10X,BINARY INPUT = *,A7/10X,SOURCE O
10X,ARCHIVE KEEP = *,A7/10X,EMPLIB OUTPUT = *,A7/10X,EMPLIB INP
10X = *,A7)
CONTINUE
READ 2,CARD
2 FORMAT*(D44)
IF(EOF(6).NE.0) GO TO 100
IF(CARD(.NE.0) GO TO 20
PRINT 3
3 FORMAT(*EMPLIB $IS INVALID CONTROL CARD FOLLOWS, JOB WILL BE ABOR
10X AFTER READING INPUT FILE.*)
IFLAG=1
2) PRINT 4,CARD
4 FORMAT*(NO,80A1)
IF(IFLAG.NE.0) GO TO 10
IF(IFIRST.NE.0) GO TO 2500
I=1
CALL NEXT2(CARD(2),I,J,K)
IF(J,K+.OR.J,K+CHAR(2)) GO TO 2500
55 IFIRST=1
PROGRAM E4IPLIB

GO TO 30

2500 CONTINUE
IF(JTOC.NE.0) GO TO 30
CALL GETTOC(TOC,NFILES,JCR)
60 IF(JCR.EQ.1 AND JTOC.EQ.0) PRINT 17,FILNAM(4)
17 FORMAT* EMPLIB $S$ TOC MISSING ON *,A7,*,*)
JTOC=1
IF(JCR.EQ.1) JTOC=-1
REWRIND 4
65 LASTF=S
10 CALL ISI1(CHAR,CARD2),NCHAR,JUMP,IFILE,TOC,DDATE,JCR,FILNAM)
IF(JUMP.NE.0) GO TO 40
PRINT 19
19 FORMAT* EMPLIB $S$ WILL ABORT AFTER READING INPUT FILE.*)
IFLAG=1
GO TO 10
40 CONTINUE
GO TO (60,60,60,70,70,80,80,90,90,100,110,110,110,110,150,160,160,210,230,
70 CONTINUE
CALL POSFIL(4,LASTF,IFILE)
K=1
IF(JUMP.EQ.2) K=2
CALL CPFIL4(K,0)
80 LASTF=IFILE
J=TOC(4,IFILE)
PRINT 6,IFILE,MODE(J),TOCII,IFILE,J=1,3),FILNAM(4),FILNAM(K)
6 FORMAT* EMPLIB FILE (*,A7,3A10,*), FROM*,A7,*
+ TO *,A7,* FILE.*)
90 CONTINUE
79 CONTINUE
C CHANGE AND CHANGE
L=NCNOCHANGED
NAMOLD=TOC(1,IFILE)
90 IVOLD=TOC(2,IFILE)
75 CONTINUE
OLDATE=TOC(3,IFILE)
TOC(4,IFILE)=ODATE
REWRIND 4
95 REWRIND 40
CALL CPFIL4(K,40,NFILES)
J=9
IF(JUMP.EQ.5 OR JUMP.EQ.29) J=3
IF(NOREW.EQ.0) REWRIND J
103 NOREW=0
CALL NEWFIL(NFILES,TOC,IFILE,J)
LASTF=S
K=TOC(4,IFILE)
PRINT 7,L,IFILE,MODE(K),NAMOLD,IVOLD,OLDATE,MODE(K),K
105 TOCII,IFILE,J=1,3),FILNAM(4),FILNAM(K)
7 FORMAT* EMPLIB $S$ *,A7,*,*TH FILE (WAS *,A7,3A10,*), IS NOW
+ *,A7,3A10,*)*/20K,ON *,A7,* USING CONTENTS OF *,A7,* FILE.*)
GO TO 10
80 CONTINUE
79 CONTINUE
C ADD AND ADD
J=8
IF(JUMP.EQ.7) J=3
REWIND 4
REWIND 40
IF(NOREM.EQ.J) READING J
MOREM=0
CALL CFPYLFILE(4,40,NFILES)
NFILES=NFILES
CALL NEWFIL(NFILES,TAG,NFILES,J)
LASTF=0
K=TOC(4,JFILE)
PRINT 'FILE,MODE(K),TOC(J,IFILE),I=1,3),FILNAME,J)
FORMAT(* EPLIB $$$ ADDED *,I2,*TH FILE (*.A7,3A10,* ) TO *,A7,* FR
ON *,A7,* FILE,*)
GO TO 10
90 CONTINUE
C DROP AND DROPB
REWIND 4
REWIND 40
CALL CFPYFIL(4,40,NFILES)
J=TOC(4,JFILE)
PRINT 9,JFILE,MODE(J),TOC(J,IFILE),I=1,3),FILNAME,J)
J=TOC(4,2)-1
TOC(4,2)=J
IF(J.JEQ.IFILE) GO TO 96
DO 95 I=IFILE,J
DO 95 K=1,4
TOC(K,1)=TOC(K,I+1)
95 CONTINUE
96 CONTINUE
CALL NEWFIL(NFILES,TAG,IFILE,0)
LASTF=0
NFILES=NFILES-1
9 FORMAT(* EPLIB $$$ DROPPED *,I2,*TH FILE (*.A7,3A10,* ) FROM *,A7,
*,*)
GO TO 10
100 CONTINUE
C TOC
PRINT 11,FILNAME,J
11 FORMAT(* EPLIB $$$ TABLE OF CONTENTS OF *,A7)
DO 105 I=1,NFILES
IF(I.EQ.2) GO TO 104
K=TOC(4,I)
PRINT 31,J,TAG(J,I),I=1,3),MODE(K)
104 CONTINUE
PRINT 32,TAG(J,I),J=1,4)
110 CONTINUE
PRINT 32,TAG(J,I),J=1,4)
FORMAT(* EPLIB $$$ TABLE OF CONTENTS OF *,A7)
DO 105 I=1,NFILES
IF(I.EQ.2) GO TO 104
K=TOC(4,I)
PRINT 31,J,TAG(J,I),I=1,3),MODE(K)
104 CONTINUE
PRINT 32,TAG(J,I),J=1,4)
110 CONTINUE
PRINT 32,TAG(J,I),J=1,4)
FORMAT(* EPLIB $$$ TABLE OF CONTENTS OF *,A7)
DO 105 I=1,NFILES
IF(I.EQ.2) GO TO 104
K=TOC(4,I)
PRINT 31,J,TAG(J,I),I=1,3),MODE(K)
104 CONTINUE
PRINT 32,TAG(J,I),J=1,4)
110 CONTINUE
PRINT 32,TAG(J,I),J=1,4)
FORMAT(* EPLIB $$$ TABLE OF CONTENTS OF *,A7)
DO 105 I=1,NFILES
IF(I.EQ.2) GO TO 104
K=TOC(4,I)
PRINT 31,J,TAG(J,I),I=1,3),MODE(K)
104 CONTINUE
PRINT 32,TAG(J,I),J=1,4)
110 CONTINUE
PRINT 32,TAG(J,I),J=1,4)
PROGRAM EMLIB

COC 6600 FTM V3.0-P292 OPT=1 04

I=0
REWIND 5
BUFFER IN(5,1) (A,A(2))
I=I+1
170 IFUNIT(5) 130,125,2000
120 I=0
J=J+1
GO TO 120
130 IF(LENGTH(5).NE.1) GO TO 120
IF(A(1).NE.LASTH) GO TO 120
IARCH=J/2
140 CALL SKIPB(X(K),1)
CALL POSFIL(4,LASTF,IFILE)
BUFFER OUT(5,1) (TOC(1,IFILE),TOC(4,IFILE))
160 IFUNIT(5).GE.0) GO TO 210
ENDIFIL 5
CALL CPYFIL(4,5,1)
LASTF=IFILE
BUFFER OUT(5,1) (LASTH,LASTH)
180 IF(UNIT(5).GE.0) GO TO 210
BACKSPACE 5
IARCH=IARCH+1
I=IARCH
J=TOC(4,IFILE)
190 PRINT 12,IFILE,MODE(J),TOC(K,IFILE),K=1,3,FILNAM(4),FILNAM(5),I,
"FILNAME(5)
11 FORMAT* EMLIB $S$ KEPT *,12,*TH FILE *(A7,3A10,*) FROM *,A7,* G
* N *,A7,* FILE.*/120,* SOURCE AND BINARY FILES NOW KEPT ON *,A7,* F
*ILE.*
195 GO TO 110
150 CONTINUE
C CREATE
REWIND 4
CALL CPYFIL(4,40,1)
210 CALL NENFIL(1,TOC,O,0)
REWIND 4
LASTF=0
PRINT 16,FILNAM(4) -
16 FORMAT* EMLIB $S$ CREATED EMLIB ON FILE NAMED *,A7,* *
*OC=0
GO TO 10
160 CONTINUE
C FIND AND FINDB
IARCH=0
REWIND 5
I=0
M=1
IF(JUMP.EQ.15) M=2
J=1
215 BUFFER IN(5,1) (A,A(4))
I=I+1
IFUNIT(5) 190,160,2020
190 I=0
J=J+1
GO TO 170
190  L=LENGTH(5)
      IF(A(1).NE.LASTH. OR .L.NE.1) GO TO 200
      PRINT 214,MODE(H),NAME,IVERS,FILNAM(5)
21   FORMAT(* EMLIB $S$ FILE SOUGHT (*,A7,2A10,* ) IS NOT ON *,A7,* *)
      IFLAG=1
      GO TO 10
200  IF(L.NE.4) GO TO 170
      IF(M.NE.A(4)) OR NAME.NE.A(1),OR.IVERS.NE.A(2) GO TO 170
      I=I+1
      PRINT 222,I,MODE(H),A(1),A(2),A(3),FILNAM(5)
22   FORMAT(* EMLIB $S$,$I$,I$TH$ FILE FOUND (*,A7,3A10,* ) ON *,A7,* *)
      BUFFER IN(5,1) (A,A)
      IF(UNIT(5).NE.0) GO TO 2020
      M=1-M
      CALL CPYFIL(5,M,0)
23   FORMAT(* EMLIB $S$ COPYING FILE FOUND TO *,A7,* *)
      IARCH=I
      GO TO 10
240  CONTINUE
C HISTORY
25   PRINT 25,FILNAM(5)
26   FORMAT(* EMLIB $S$ HISTORY OF *,A7)
      REMIND 5
      I=0
220  I=I+1
      IARCH=I-1
      BUFFER IN(5,1) (A,A(4))
      IF(UNIT(5).GE.0) GO TO 2030
      IF(LENGTH(5).NE.1 OR A(1).NE.LASTH) GO TO 225
      BACKSPACE 5
      GO TO 10
225  CONTINUE
      K=A(4)
255  PRINT 26,I,(A(J),J=1,3),MODE(K)
26   FORMAT(21X,*KEEP VOL.*,14,1X,4A10)
      BUFFER IN(5,1) (A,A)
      IF(UNIT(5).NE.0) GO TO 2030
      CALL CPYFIL(5,0,1)
250  GO TO 220
236  CONTINUE
C RENAME AND RENAMED
265  REMIND 4
      REMIND 40
      CALL CPYFIL(4,40,NFILES)
      CALL NEWSFIL(NFILES,TOC(0),0)
      LASTF=0
      K=TOC(4,IFILE)
270  PRINT 28,IFILE,MODE(K),NAHOLD,IVOLD,TOC(3,IFILE),MODE(K),TOC(1,IFILE),I=1,3
28   FORMAT(* EMLIB $S$ RENAMED *,I2,*TH FILE (WAS *,A7,3A10,* ) IS NOW *,A7,3A10,* )
      GO TO 10
240  CONTINUE
C REFUSE AND SELECT
275
PROGRAM EMLIB

IRS=1
IF(IFILE.EQ.0) GO TO 245
CALL POSFILE(4,LASTF,IFILE)
K=TOC(4,IFILE)

280 PRINT 29,FILNAM(1),IFILE,MORDER(K),TOKI(I,IFILE),I=1,3,FILNAM(4)
FORMAT(* EMLIB $$$ COPYING THE FOLLOWING BINARY RECORDS ONTO *.*,7*
* FROM THE *.*,2.*,7*TH FILE (*.*,7*,3A10,*3A7,*,7*/*45X,*,7*SELECTED
*.*,7*/*REFUSED*)
K=NREC
IF(JUMP.EQ.20) K=-K
CALL CPYREC(4,1,NAMRECK)
LASTF=IFILE
GO TO 10

245 CONTINUE

290 PRINT 36,FILNAM(1),FILNAM(3)
FORMAT(* EMLIB $$$ COPYING THE FOLLOWING BINARY RECORDS ONTO *.*,7*
* FROM *.*,7*/*45X,*,7*SELECTED,7*X,*,7*REFUSED*)
K=NREC
IF(JUMP.EQ.20) K=-K
CALL CPYREC(3,1,NAMRECK,K)
GO TO 10

295 CONTINUE
C NOREWIND
NOREM=1
GO TO 10

260 CONTINUE
C SKIP AND SKIPB
I=0
IF(JUMP.EQ.26) I=3
CALL CPYFILE(I,0,IFILE)
PRINT 33,FILNAM(1)
FORMAT(* EMLIB $$$ SKIPPED*,I4,* FILES ON *.*,7,*
GO TO 10

270 CONTINUE
C CREATEARCH
REWIND 5
IARCH=0
A(1)=44LAST
BUFFER OUT(5,1) (A,A)
IF(Unit(5).GE.0) GO TO 2010
REWIND 5
PRINT 35,FILNAM(5)
FORMAT(* EMLIB $$$ CREATED ARCHIVE ON FILE NAMED *.*,7*,
GO TO 10

280 CONTINUE
C REPLACE AND REPLACED
L=8REPLACED
GO TO 75

100 CONTINUE

325 H=MAX-1
IF(IFLAG.EQ.0) PRINT 34,H,M
FORMAT(* EMLIB $$$ MAXIMUM RECORD LENGTH PROCESSED FOR SELECT-REF
USE MAX=',I5,*,7*,15,*IS MAXIMUM ALLOWED.*
IF(IFLAG.EQ.0) PRINT 13

330 FORMAT(* EMLIB $$$ FINISHED $$*)
PROGRAM EMPLIB

REWIND 4
DO 1010 J=1,2
  IF(FETE(I).AND.500).NE.20) GO TO 1010
ENDFILE J
CALL SKIP(I)
1010 CONTINUE
IF(IFLAG.EQ.0) STOP
PRINT 10
18 FORMAT( 'EMPLIB $$$ ABORTING $$$' )
CALL ABORT
2000 CONTINUE
PRINT 14,I,J,FILNAM(5)
14 FORMAT( 'EMPIB $$$ KEEP READ PARITY ERROR ON*,IS,*TH RECORD ON*,I *
5,*TH FILE ON *,A7,*,*')
CALL ABORT
2010 CONTINUE
PRINT 15,FILNAM(5)
15 FORMAT( 'EMPIB $$$ KEEP WRITE PARITY ERROR ON *,A7,*,*')
CALL ABORT
2020 CONTINUE
PRINT 24,FILNAM(5)
24 FORMAT( 'EMPIB $$$ FIND READ ERROR ON *,A7,*,*')
CALL ABORT
2030 CONTINUE
PRINT 27,FILNAM(5)
27 FORMAT( 'EMPLIB $$$ HISTORY READ ERROR ON *,A7,*,*')
CALL ABORT
END
SUBROUTINE GETTOC

SUBROUTINE GETTOC(TOC,NFILES,JCR)
DIMENSION TOC(4,50)
RE%IND 4
JCR=0
CALL CPYFILE(4,0,1)
BUFFER 14(4,1) (TOC,TOC(4,50))
IF(UNIT(4)) 10,20,100
10 NFILES=LENGTH(4)/4
RETURN
10 20 NFILES=2
JCR=1
RETURN
100 PRINT 1
FORMAT(* EMPLIG $S$ GETTOC PARITY ERROR*)
15 CALL ABORT
END
SUBROUTINE CPYFIL

SUBROUTINE CPYFIL(INF, OUTF, NF)
COMMON //FLAX, A(1)
COMMON //FILLS/FILNAM(N), FETS(N), X(N)
INTEGER FILLN, FETS, X
IF(INF.GT.0) GO TO 30
DO 20 I=1, NF
10 BUFFER INF(INF,1) (A, A)
IF(INF(INF)) 10, 20, 200
20 CONTINUE
RETURN
10 CONTINUE
DO 20 INF INF, 13 (A, A)
IF(INF(INF)) 20, 200
20 CONTINUE
CALL CPVBUF (A, A(LMX,?), X(J), X(K), IER)
12 IF(IER.NE.1) GO TO 36
IF(NF.GT.0) CALL FNGFIL IOUT .0
CONTINUE
30 RETURN
20 CONTINUE
PRINT 1, FILNAM(JIN)
1 FORMAT (* EMPIE $$$ *, ?7,* INITIALLY PCSITICNEC AT END-OF-INFORMAT
TION, EMPIE ANOTHER.*)
25 CALL ABORT
36 CONTINUE
CALL CPYBUF (A, A(LMX,?), X(J), X(K), IER)
20 IF(IER.NE.1) GO TO 36
IF(NF.GT.0) CALL FNGFIL IOUT .0
CONTINUE
30 RETURN
20 CONTINUE
PRINT 1, FILNAM(JIN)
1 FORMAT (* EMPIE $$$ CPYFIL READ PARITY ERROR IN $$*, I3, *TH FILE FROM
START OF COPY) ON *,$7,* *)
35 CALL ABORT
30 CONTINUE
PRINT 1, FILNAM(JIN), FILNAM(JOUT), I, IER
2 FORMAT (* EMPIE $$$ I/O ERROR IN CPYBUF WHILE COPYING *, A8, *TC, *A
$7,* FILE NUMBER *, I3/20X, *ERROR CODE IN CQITAL IS *, C20)
40 CALL ABORT
40 CONTINUE
PRINT 1, FILNAM(JIN), I, NF
5 FORMAT (* EMPIE $$$ END-OF-INFORMATION ENCOUNTERED READING COPY $*, I3, *TH
FILE FROM START OF COPY ON *,$7,* *)
45 CALL ABORT
END
SUBROUTINE GETFIL

COMMON /FILES/FILENAME(9),FETS(9),X(1)
INTEGER FILENAME,FETS,X
DATA MASK/?????????

L=LOCF(X)
DO 10 I=1,9
J=J+1
FILENAME(I)=X(J)
J=FILENAME(I), AND MASK
10 CONTINUE
RETURN
END
SUBROUTINE SWAPFIL

SUBROUTINE SWAPFIL(IUNIT,NAM)
DIMENSION WORD2(IUNIT),WORD2(IUNIT)
COMMON /FILES/FILNAM(9),FETS(9),X(1)

INTEGER FILNAM,FETS,X
DATA WORD1,10HDSOURCE OUT,10HDSOURCE IN,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT,10HDSOURCE OUT

10 I=FETS(IUNIT)
II=I(I+1)
KFLAG=1
IF(II.NEQ.x MASK(I).AND.(X(I+4).AND..NOT.MASK).EQO) GO TO 10
KFLAG=2

C CHECK TO SEL IF LAST OPERATION WAS AN FOR WRITE.
IF(IN.ME.28) GO TO 5
ENDFILE IUNIT
CALL SKIP3(II,0)

20 5 CONTINUE
I(I+1)=I(I+1).AND..NOT.MASK
I(I+4)=I(I+4).AND.MASK
CONTINUE

25 CONTINUE
IN=NAME
DO 10 J=1,10
M=SHIFT(MASKCH,E*(J-1))
IF(IN.AND.M).NE.(I8.AND.M)) GO TO 30
IN=IN.AND..NOT.M
CONTINUE
30 CONTINUE
I(I)=IN.AND..NOT.MASK.IR.3
I=FILNAM(IUNIT)
FILNAM(IUNIT)=IN
IF(KFLAG.GT.0) GO TO 40

35 PRINT 1,WORD2(IUNIT),WORD2(IUNIT),FILNAM(IUNIT)
1 FORMAT(* EMPLIL $H$ MAKE *A,19+3, FILE * , , * )
RETURN

40 PRINT 2,WORD2(IUNIT),WORD2(IUNIT),FILNAM(IUNIT)
2 FORMAT(* EMPLIL $H$ MADE *A,19+3, FILE * , , * INSTEAD OF *, , . *)
RETURN
END
IDENT CPYBUF
ENTRY CPYBUF
WFO 36/GMCPYBUF.26/5

CPYBUF
655 1
68F 1
SGA B7+07
SA1 A1+07
SA2 A1+06
SA3 A1+06
SA4 A1+06
SA5 A1+06
SA6 A0
SA7 AZER0
SGF X5
SA7 IER
MIF 0
SA7 X5
SA7 FLAG
BX6 X1
SAB BOUNDS
BX7 X2
S7 A6+07
SA1 X3+07
SA2 A1+07
BX6 X1
BX7 X2
SA1 A2+07
SA2 A1+07
SA6 SAVE
SA7 A6+07
BX6 X1
BX7 X2
SA6 A7+07
SA7 A6+07
SA1 BOUNDS
SR2 X1
SA2 A1+07
SB1 X2
SA1 X3+07
MIF 42
SA7 02
BX6 X1*0
BX6 X6+7
SA6 X3+07
SA7 A6+07
SA7 A7+07

X3 TO CONTAIN ADDRESS OF FILEIN
X4 SAME FOR FILEOUT

LOOP
IN
CCHPASS - vER 2. 01/10/72 11.12.50.

DX6 -XQ*X1
DX7 X0*X2
DX7 X0+X7
SA7 A2
SX6 B7
SA7 FLAG
SA1 X3
SA2 X4
SA3 X1
SX6 B3+3
BX7 X0*X2
DX7 X0+X7
SA7 X4
SA1 CIOWORD
DX6 X1+X4
SA6 B7
RECALLA SA8 B7
M2 X5,RECALLA
SA1 X4
SX0 3P008B
BX2 X1*X0
ZQ X2,OKA
SV5 IER
BX6 X1
SA6 X5
ED RETURN
OKA SB1 -378
SB2 X1+81
NE BZ,LOOP
RETURN SA1 SAVE
SA2 A1+B7
MX8 A2
SA5 X3+B7
BX6 -XQ*X1
BX5 X0*X5
BX6 X0+X5
BX7 X2
SA1 A2+B7
SA2 A1+B7
SA5 X3+4
SA6 X3+97
SA7 A5+B7
BX6 X1
BX7 -XQ*X2
BX5 X0*X5
BX7 X0+X5
SA6 A7+B7
SA7 A6+B7
SA1 A2+B7
SA2 A1+B7
SA5 X4+B7
BX6 -XQ*X1
BX5 X0*X5
BX6 X0+X5
BX7 X2
SA1 A2+B7
SA2 A1+B7
COMPASS - VER 2.  01/10/72  11.12.50.

SA5 X4+4
SA6 X4+87
SA7 A6+87
BX6 X1
BX7 -X8+X2
BX5 X0+X5
BX7 X7+X5
SA6 A7+87
SA7 A6+87
SA1 AZERO
SA8 X1
ED CPYBUF
IER BSS 1
AZERO BSS 1
BOUNDS BSS 2
FLAG BSS 1
SAVE BSS 8
CIONORD VFO 18/3NCIO;2/1,4G/0
END
STORAGE USED 190 STATEMENTS 14 SYMBOLS
6600 ASSEMBLY 0.536 SECONDS 42 REFERENCES
SUBROUTINE NEWFIL

SUBROUTINE NEWFIL(NFILES,TOC,IFILE,J)
DIMENSION TOC(4,NFILES)
REWIND 4
REWIND 40
L=MAX0(2,NFILES)
K=L
IF(24IFILE .NE.0.AND.J.EQ.6) K=K-1
DO 30 I=1,K
IF(I.NE.2) GO TO 10
BUFFER OUT(4,J) TOC,TOC(4,K))
IFUNIT(4).GE.(0) GO TO 100
ENDFILE 4
IF(NFILES.NE.1) CALL CPYFIL(40,0,1)
GO TO 30
10 IF(I.NE.IFILE) GO TO 20
IF(NFILES.NE.IFILE) CALL CPYFIL(40,0,1)
IF(J.NE.0) CALL CPYFIL(J,4,1)
GO TO 30
20 CALL CPYFIL(40,4,1)
CONTINUE
REWIND 4
RETURN
100 CONTINUE
PRINT 1
1 FORMAT(* EMPLIB $3$ TOC WRITE PARITY ERROR IN NEWFIL.*)
CALL ABORT
END
SUBROUTINE POSFIL(NLASTF,IFILE)
COMPGN /FILES/FILNM(9),FETS(9),X(1)
INTEGER FILNM,FETS,X
IF(IFILE,GT,LASTF) GO TO 10
5 I=FETS(N)
CALL SKIPB(X(I),LASTF-IFILE+1)
GO TO 30
10 JFILES=IFILE-LASTF-1
IF(JFILES.EQ.0) GO TO 30
CALL CPYFIL(1,0,JFILES)
10 LASTF=IFILE-1
RETURN
END
SUBROUTINE ISIT

COMMON /ARGS/NAME, IVERS, NAMOLD, IVOLD, NREC, NAMREC(100), JTOC, LASTF, ICARCH
INTEGER FILTYP(8), NTYP(8)
DIMENSION CARO(79), TOC(4,1), CHAR
DIMENSION NUNS(10)
INTEGER CARD, TOC, ODATE, CHAR
DATA FILTYP/2H0, 2H01, 2H02, 2H03, 1H4, 1H0, 1H7, NTYP/1, 3-2, 8, 4, 5,
17, 6/
DATA IB/1H /* MASK/* 7B/
DATA MODE/6HSOURCE, 6HNARROW,/
DATA NUNS/1H0, 1H1, 1H2, 1H3, 1H4, 1H5, 1H6, 1H7, 1H8, 1H9/
JUMP=0
NAME=10
IST=1
CALL NEXTHD(CARD, IST, NAME, JFLAG)
IF(JFLAG.EQ.0) GO TO 10
CONTINUE
1 FORMAT(* EMLIB *** IMPROPER DIRECTIVE ON CARD.*,**79A1)
10 CONTINUE
DO 40 JUMP=1, NCHAR
IF(CHCAR(JUMP).EQ.27) GO TO 90
CONTINUE
40 PRINT 1, CARD
1 FORMAT(* EMLIB *** UNRECOGNIZABLE DIRECTIVE.*,**A1O,**79A1)
JUMP=0
RETURN
50 CONTINUE
IF(JUMP.EQ.27) RETURN
IF(JUMP.EQ.13) GO TO 330
IF(JUMP.EQ.21) GO TO 500
IF(JUMP.EQ.0) GO TO 55
IF(JUMP.GE.13.AND.JUMP.LE.17.AND.JUMP.NE.19) GO TO 55
JUMP=0
PRINT 7, NAME
7 FORMAT(* EMLIB *** DIRECTIVE REQUIRES TABLE OF CONTENTS, WHICH HAS NOT BEEN CREATED.*,**A10)
RETURN
55 IF(JUMP.EQ.10.0R.JUMP.EQ.16) RETURN
IF(JUMP.EQ.24) RETURN
IF(JUMP.EQ.25.0R.JUMP.EQ.26) GO TO 600
IF(JUMP.EQ.22.0R.JUMP.EQ.23) GO TO 700
NAME=1B
CALL NEXTHD(CARD, IST, NAME, JFLAG)
IF(JFLAG.NE.11.0R.(JUMP.NE.19.AND.JUMP.NE.20)) GO TO 56
IFILE=0
GO TO 630
60 CONTINUE
IF(JFLAG) 70, 80, 63
63 CONTINUE
PRINT 3, NAME, CARD
3 FORMAT(* EMLIB *** CANNOT FI'D PROGRAM NAME ON CARD.*,**79A1)
JUMP=3
SUBROUTINE ISIT

RETURN
70 CONTINUE
PRINT 4, NAME, CARD
4 FORMAT(* EMLIB $S$ PROGRAM NAME TOO LONG.*A10,*A9,F7.1)
JUMP=0
RETURN
60 CONTINUE
CALL NEXTWO(CARD,IST,IVERS,JFLAG)
65 IF(JFLAG.GE.9) GO TO 100
PRINT 5,IVERS,CARD
5 FORMAT(* EMLIB $S$ VERSION NAME TOO LONG.*A10,*A9,F7.1)
JUMP=0
RETURN
70 CONTINUE
IF(JFLAG.EQ.1) IST=IST-1
IF(JUMP.EQ.14.OR.JUMP.EQ.15) RETURN
H=TOC(4,2)
MODE=1
75 IF(JUMP.EQ.29) MODE=2
IF(JUMP.EQ.3.OR.JUMP.EQ.5.OR.JUMP.EQ.9) MODE=2
IF(JUMP.EQ.1.OR.JUMP.EQ.7.OR.JUMP.EQ.12.OR.(JUMP.GE.18.AND.JUMP.LE
',.20)) MODE=2
GO 160 IFILE=14
IF(TOC(4,IFILE).EQ.MODE.AND.TOC(1,IFILE).EQ.NAME.AND.TOC(2,IFILE).EQ.
'FLS.VERS) GO TO 170
IF(JUMP.EQ.1.OR.TOC(4,IFILE).NE.MODE.OR.TOC(1,IFILE).NE.NAME) GO T
O 160
C RUN
85 IVERs=TOC(2,IFILE)
RETURN
160 CONTINUE
IF(JUMP.EQ.6.OR.JUMP.EQ.7) GO TO 180
PRINT 6, NAME, IVERs
6 FORMAT(* EMLIB $S$ *.2A10,* NOT IN TOC.*)
JUMP=0
RETURN
170 CONTINUE
IF(JUMP.EQ.6.OR.JUMP.EQ.7) GO TO 185
C RUN, COPY, KEEP, AND KEEPB
95 IF(JUMP.EQ.1.OR.JUMP.EQ.2.OR.JUMP.EQ.3.OR.JUMP.EQ.11.OR.JUMP.EQ.12
'T) RETURN
IF(JUMP.EQ.4.AND.JUMP.NE.5) GO TO 190
C CHANGE AND CHANGES
100 CONTINUE
C ADD AND ADDB
180 CONTINUE
FILE=TOC(4,2)+1
TOC(4,2)=IFILE
TOC(1,IFILE)=NAME
TOC(2,IFILE)=IVERs
TOC(3,IFILE)=DATE
TOC(4,IFILE)=JUMP=5
RETURN
105 165 CONTINUE
SUBROUTINE ISIT

CDC 6600 FTM V3.0-P292 OPT=1 04

J=TOC(4,IFILE)
PRINT 11,IFILE,MODEM(J)
TOC(J,IFILE),I=1,3

FORMAT(* EMPLIB $$$ ADDING FILE ALREADY IN TOC IS NOT PERMITTED. *
** FILE IS *.EM* FILE (*.EM.P,*.EM,*.*)
JUMP=0
RETURN

190 IF(JUMP.NE.5.AND.JUMP.NE.9) GO TO 210
C DROPO AND DROP
IF(IMAGE.GT.2) RETURN

5 FORMAT(* EMPLIB $$$ DROPPING LIBRARY OR TABLE OF CONTENTS IS NOT
* PERMITTED. *)
JUMP=0
RETURN

125 IF(JUMP.NE.17.AND.JUMP.NE.18) GO TO 210
C RENAME AND RENAME
CALL NEXTWDCARD,JST,JNAME,FLAG
IF(FLAG.EQ.1) GO TO 220

12 FORMAT(* EMPLIB $$$ MISSING COMMA, **,?,9A1)
JUMP=0
RETURN

220 RETURN

135 IF(FLAG.GE.0) GO TO 230
PRINT 12,CARD

13 FORMAT(* EMPLIB $$$ TABLE OF CONTENTS IS NOT
PERMITTED. *)
JUMP=0
RETURN

230 RETURN

149 RETURN

153 RETURN

155 RETURN

160 CONTINUE
NAHOL=TOC(1,IFILE)
IVOLD=TOC(2,IFILE)
TOC1,JFILE)=NAME
TOC2,JFILE)=IVERS
RETURN

300 CONTINUE
C CREATE
TOC(1,2)=3MTOC
TOC(2,3)=IB
TOC(3,2)=DDATE
TOC(4,3)=2
TOC(1,1)=6MEMPLIB
TOC(21)=IB
TOC(1,1)=DDATE
TOC(4,1)=2
JCR=0
RETURN
SUBROUTINE ISIT

C
500 CONTINUE
C FILES
510 CALL NEXTWO(CARD,IST,NAME,JFLAG)
      IF(JFLAG.EQ.2) RETURN
      IF(JFLAG) 520,530,510
520 PRINT 13,NAME,CARD
13 FORMAT(* EMPLIB $$$ WORD IS TOO LONG,**,A10,***,79A1)
      JUMP=0
      RETURN
530 CONTINUE
      DO 540 I=1,8
      IF(NAME.EQ.FILTYP(I)) GO TO 550
      CONTINUE
      PRINT 14,NAMExFILTP
14 FORMAT(# EMPLIB $$$ FILE TYPE *,A10,* IS NOT ONE OF THE ALLOWED FO
RMS *,8A3,* *)
      JUMP=0
      RETURN
550 IFILE=NTYP(I)
560 CALL NEXTWO(CARD,IST,NAME,JFLAG)
      IF(JFLAG) 520,560,540
570 CALL SWAPFIL(IFILE,NAME)
      IF(IFILE.NE.4.AND.IFILE.NE.5) GO TO 510
      IF(IFILE.EQ.4) GO TO 570
580 IARCH=0
      GO TO 510
590 JANOC=0
      GO TO 510
600 CONTINUE
C REFUSE AND SELECT
      IF(JUMP.NE.19.AND.JUMP.NE.20) GO TO 700
605 NREC=0
510 CALL NEXTWO(CARD,IST,NAME,JFLAG)
      IF(JFLAG.EQ.2) GO TO 640
      IF(JFLAG) 520,620,630
620 NREC=NREC+1
      IF(NREC.GT.100) GO TO 650
      NAMEC=NREC+NAME
KFLAG=0
      GO TO 610
630 KFLAG=1
      GO TO 610
640 IF(KFLAG.EQ.0) RETURN
      READ 15,CARD
15 FORMAT(79A1)
650 PRINT 16,CARD
      RETURN
660 READ 15,CARD
16 FORMAT(1X,79A1)
      KFLAG=0
      IST=1
      GO TO 610
680 PRINT 17
SUBROUTINE ISIT

C CDC 6630 FTN V3.0-P292 OPT=1 04

17 FORMAT(* EMPLIB $$ MORE THAN 100 RECORD NAMES GIVEN -- TOO MANY.*
) JUMP=5
RETURN

225 CONTINUE IF(JUMP.NE.22.AND.JUMP.NE.23) GO TO 930
C REWIND AND ENDFILE
NAME=IB
CALL NEXTWO(CARD,IST,NAME,JFLAG)
IF(JFLAG.EQ.0) GO TO 710
PRINT 14 NAME,JFLAG
19 FORMAT(* EMPLIB $$ ILLEGAL FILE TYPE.*,A10,**,79AI)
JUMP=5
RETURN

235 DO 720 I=1,9
IF(NAME.EQ.FILTYPI(I)) GO TO 730
720 CONTINUE PRINT 14,NAME,FILTYP
JUMP=0
RETURN

730 J=I NAME=FILTYP(I)
IF(JUMP.EQ.23) GO TO 750
IF(I.NE.6.AND.I.NE.7.AND.I.NE.8) GO TO 740

245 PRINT 19,CHAR(JUMP),FILTYP(I)
19 FORMAT(* EMPLIB **.,A10,** IS AN ILLEGAL DIRECTIVE FOR THE FILE T
**.,A2,**)
JUMP=5
RETURN

250 REMIND I
NAME=7REWIND
745 PRINT 25,NAME,FILTYP(I),FILNAM(I)
20 FORMAT(* EMPLIB $$.*,A5,A2,* FILE NAMED *,A7,**)
RETURN

255 IF(I.NE.1.AND.I.NE.2) GO TO 735
ENDFILE I
NAME=6HENDFILED)
GO TO 745

900 CONTINUE
C SKIP AND SKIPB
CALL NEXTWO(CARD,IST,NAME,JFLAG)
IF(JFLAG.NE.2) GO TO 910
IFILE=1
RETURN

265 IF(JFLAG.EQ.0) GO TO 620
PRINT 21,NAME
21 FORMAT(* EMPLIB $$ ILLEGAL NUMBR.*,A10)
JUMP=0
RETURN

270 FILE=0
I=SHIFT(MASK,54)
DO 820 J=1,3
L=I.AND.NAME
IF.LEQ.(I.AND.ID) RETURN

275 DO 830 K=1,19

46
SUBROUTINE ISIT

CNC 660D FTM V3.0-P292 OPT=1 04

IF(L.EQ.(I.AND.SHIFT(NUMS(K),66-6*N))) GO TO 840
830 CONTINUE
PRINT 21, NAME
JUMP=0
RETURN
840 IFILE=10*IFILE+K-1
850 I=SHIFT(I,54)
RETURN
900 CONTINUE
285 C REPLACE AND REPLACE
HAMOLD=TOPC(1,IFILE)
IVOLD=TOPC(2,IFILE)
CALL NEXTWO(CARO,IST,NANE,JFLAG)
IF(JFLAG.NE.1) GO TO 215
NAME=10
CALL NEXTWO(CARO,IST,NANE,JFLAG)
IF(JFLAG.LT.0) GO TO 255
IVERS=10
CALL NEXTWO(CARO,IST,IVERS,JFLAG)
295 IF(JFLAG.LT.0) GO TO 255
GO TO 260
END
SUBROUTINE NEXTWO

SUBROUTINE NEXTWO(CARD, IST, NAME, JFLAG)
INTEGER CARD(I)
C
JFLAG=-1 IS ERROR, 0 IS NORMAL RETURN, 1 IS COMMA, 2 IS EMPTY CARD
NAME=I
JFLAG=2
IF(IST.GT.70) RETURN
DO 16 I=IST, 79
IF(CARD(I).EQ.1M.) GO TO 40
IF(CARD(I).EQ.1N.) GO TO 15
IF(CARD(I).EQ.1H.) GO TO 20
15 CONTINUE
10 IST=IST+1
RETURN
20 I=I-1
JFLAG=0
DE 36 J=1, 11
IST=J+1
IF(IST.GT.79) RETURN
IF(CARD(IST).EQ.1M.) RETURN
IF(J.LT.11) CALL APPEND(J, CARD(IST), NAME)
30 CONTINUE
JFLAG=-1
RETURN
40 IST=IST+1
JFLAG=1
RETURN
END
SUBROUTINE APPEND

SUBROUTINE APPEND(I, CHAR, X)

DATA MASK/776/
ITEMP=SHIFT(MASK, 60-6*I)
JITEMP=SHIFT(CHAR, 6-6*I)

X=ITEMP.AND..NOT.JITEMP
X=X.OR.JITEMP
RETURN
END
SUBROUTINE CPYREC

COPRCH /MRC/PR
COMMON //MAIX,A(1)

INTEGER A

COMMON //MAIX,A(1)

INTEGER A

INTEGER A

CONTINUE

BUFFER VM(IN,1) (A,A(MAX))

IFUNIT(IN)) 30,20,100

PRINT 1

FORMAT(SX,=*-END OF COPY-*

RETURN

30 L=LENGTH(IN)

IF(L.EQ.0) GO TO 90

M=MAXD(MAX,L)

IF(L.EQ.0) GO TO 90

DO 40 I=1,L

M=SHIFT(IN,65536-L)

IF(I.EQ.N) GO TO 50

IX=(IX.AND.NOT.L).OR.(I*2).AND.M

CONTINUE

40 IX=(IX.AND.NOT.L).OR.(I*2).AND.M

GO TO 70

IF(I.EQ.N) GO TO 70

GO TO 80

GO TO 60

BUFFER OUT(INOUT,1) (A,ATL)

IFUNIT(INOUT,10) GO TO 110

PRINT 2,IX

2 FORMAT(*X*,15)

GO TO 10

PRINT 2,IX

3 FORMAT(*X*,15)

GO TO 10

CONTINUE

CONTINUE

PRINT 6

FORMAT(*X,=*-EMPTY RECORD ENCOUNTERED-*

GO TO 10

PRN 4

FORMAT(* EMPLIE $$$ READ ERROR IN CPYREC.*)

CALL ABORT

100 PRN 4

FORMAT(* EMPLIE $$$ WRITE ERROR IN CPYREC.*)

CALL ABORT

END

45