## MODIFY DIRECTIVES INDEX

<table>
<thead>
<tr>
<th>Name</th>
<th>Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>*BKSP</td>
<td>6-2</td>
</tr>
<tr>
<td>*CALL</td>
<td>7-2</td>
</tr>
<tr>
<td>*CALLALL</td>
<td>7-2</td>
</tr>
<tr>
<td>*CALLC</td>
<td>7-3</td>
</tr>
<tr>
<td>*COMMENT</td>
<td>7-3</td>
</tr>
<tr>
<td>*COPY</td>
<td>4-1</td>
</tr>
<tr>
<td>*COPYPL</td>
<td>4-2</td>
</tr>
<tr>
<td>*CREATE</td>
<td>4-3</td>
</tr>
<tr>
<td>*CSET</td>
<td>7-4</td>
</tr>
<tr>
<td>*CHBOR</td>
<td>7-4</td>
</tr>
<tr>
<td>*DECLARE</td>
<td>5-2</td>
</tr>
<tr>
<td>*DEFINE</td>
<td>8-1</td>
</tr>
<tr>
<td>*DELETE</td>
<td>5-2</td>
</tr>
<tr>
<td>*D</td>
<td>5-2</td>
</tr>
<tr>
<td>*EDIT</td>
<td>5-3</td>
</tr>
<tr>
<td>*ELSE</td>
<td>6-2</td>
</tr>
<tr>
<td>*ENDIF</td>
<td>7-4</td>
</tr>
<tr>
<td>*IDENT</td>
<td>5-3</td>
</tr>
<tr>
<td>*IF</td>
<td>7-5</td>
</tr>
<tr>
<td>*IFDEF</td>
<td>7-5</td>
</tr>
<tr>
<td>*IGNORE</td>
<td>5-4</td>
</tr>
<tr>
<td>*INSERT</td>
<td>5-5</td>
</tr>
<tr>
<td>*I</td>
<td>5-5</td>
</tr>
<tr>
<td>*INWIDTH</td>
<td>8-2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>*MODNAME</td>
<td>5-5</td>
</tr>
<tr>
<td>*MOVE</td>
<td>8-2</td>
</tr>
<tr>
<td>*NIFCALL</td>
<td>7-6</td>
</tr>
<tr>
<td>*NOSEQ</td>
<td>7-6</td>
</tr>
<tr>
<td>*OPLFILE</td>
<td>4-3</td>
</tr>
<tr>
<td>*PREFIX</td>
<td>8-3</td>
</tr>
<tr>
<td>*PREFIXC</td>
<td>8-3</td>
</tr>
<tr>
<td>*PURDECK</td>
<td>5-6</td>
</tr>
<tr>
<td>*READ</td>
<td>6-2</td>
</tr>
<tr>
<td>*READPL</td>
<td>6-2</td>
</tr>
<tr>
<td>*RESTORE</td>
<td>5-7</td>
</tr>
<tr>
<td>*RETURN</td>
<td>6-3</td>
</tr>
<tr>
<td>*REWIND</td>
<td>6-4</td>
</tr>
<tr>
<td>*SEQ</td>
<td>7-7</td>
</tr>
<tr>
<td>*SORSEQ</td>
<td>7-7</td>
</tr>
<tr>
<td>*SKIP</td>
<td>6-4</td>
</tr>
<tr>
<td>*SKIR</td>
<td>6-4</td>
</tr>
<tr>
<td>*UNYANK</td>
<td>5-7</td>
</tr>
<tr>
<td>*UPDATE</td>
<td>8-3</td>
</tr>
<tr>
<td>*WHO</td>
<td>7-7</td>
</tr>
<tr>
<td>*WHOIF</td>
<td>7-7</td>
</tr>
<tr>
<td>*WIDTH</td>
<td>7-8</td>
</tr>
<tr>
<td>*YANK</td>
<td>5-8</td>
</tr>
<tr>
<td>*/</td>
<td>8-4</td>
</tr>
</tbody>
</table>

## OPLEDIT DIRECTIVES INDEX

<table>
<thead>
<tr>
<th>Name</th>
<th>Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>*CSSET</td>
<td>A-4</td>
</tr>
<tr>
<td>*EDIT</td>
<td>A-4</td>
</tr>
<tr>
<td>*OPLEDIT</td>
<td>A-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>*PULLALL</td>
<td>A-5</td>
</tr>
<tr>
<td>*PULLMOD</td>
<td>A-6</td>
</tr>
<tr>
<td>*PURGE</td>
<td>A-6</td>
</tr>
</tbody>
</table>
MODIFY
REFERENCE MANUAL

CDC® OPERATING SYSTEM:
NOS 2
<table>
<thead>
<tr>
<th>REVISION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (03-06-76)</td>
<td>Manual released. This manual reflects NOS 1.1 at PSR level 419.</td>
</tr>
<tr>
<td>B (12-03-76)</td>
<td>Revised to update the manual to NOS 1.2 at PSR level 439, and to make typographical and technical corrections. New directives IF, ELSE, ENDIF, and IFNCALL are added. The previous DEFINE directive has a new parameter added that allows a value to be associated with a defined name. This edition obsoletes the previous edition.</td>
</tr>
<tr>
<td>C (07-15-77)</td>
<td>Revised to update the manual to NOS 1.2 at PSR level 452, to reformat error messages, and to make typographical and technical corrections. Support of CDC CYBER 170 Series, Model 171 is also included.</td>
</tr>
<tr>
<td>D (02-07-78)</td>
<td>Revised to update the manual to NOS 1.3 at PSR level 472; to add new information regarding common decks; to add examples of the IF, ELSE, ENDIF, and NIPCALL directives; to change the type font of the terminal sessions; and to make typographical and technical corrections. This edition obsoletes all previous editions.</td>
</tr>
<tr>
<td>E (06-22-79)</td>
<td>Revised to update the manual to NOS 1.4 at PSR level 498, and to make typographical and technical corrections. New error messages INITIALIZATION DIRECTIVE OUT OF ORDER and INVALID CS ON INPUT are added. Support of CDC CYBER 170 Series, Model 170 is also included. This edition obsoletes all previous editions.</td>
</tr>
<tr>
<td>F (12-05-80)</td>
<td>Revised to update the manual to NOS 1.4 at PSR level 530, and to make typographical and technical corrections. Sections 1, 2, and 8 have been reorganized. List Option feature is enhanced and expanded. This edition obsoletes all previous editions.</td>
</tr>
<tr>
<td>G (09-30-85)</td>
<td>Revised to update the manual to NOS 2.4.2 at PSR level 642 and to incorporate extensive usability changes. Sections 1, 3, 4, 5, 6, 7, 8, and appendix A have been reorganized and rewritten. Section 2 is new. Support of the CALLC, CEET (for both Modify and OPLEVIT), and S00SEQ directives has been added. Other new features include support of mixed ASCII and DISPLAY code on OPLA, maximum 150-character line length for source files, and nested common deck calls. This edition obsoletes all previous editions.</td>
</tr>
</tbody>
</table>

Publication No.
60450100

All rights reserved
Printed on the United States of America

Address comments concerning this manual to:

Control Data
Technical Publications
4201 North Lexington Avenue
St. Paul, Minnesota 55126-6198
or use Comment Sheet in the back of this manual.
New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

<table>
<thead>
<tr>
<th>PAGE</th>
<th>REV</th>
<th>PAGE</th>
<th>REV</th>
<th>PAGE</th>
<th>REV</th>
<th>PAGE</th>
<th>REV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Front Cover</td>
<td>G</td>
<td>5-4</td>
<td>G</td>
<td>B-4</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inside Front Cover</td>
<td>G</td>
<td>6-5</td>
<td>G</td>
<td>Index-1</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>G</td>
<td>6-6</td>
<td>G</td>
<td>Index-2</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3/4</td>
<td>G</td>
<td>7-1</td>
<td>G</td>
<td>Comment Sheet</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>G</td>
<td>7-2</td>
<td>G</td>
<td>Inside Back Cover</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>G</td>
<td>7-3</td>
<td>G</td>
<td>Back Cover</td>
<td>-</td>
<td>7/8</td>
<td>G</td>
</tr>
<tr>
<td>9</td>
<td>G</td>
<td>7-5</td>
<td>G</td>
<td>7-6</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>G</td>
<td>7-7</td>
<td>G</td>
<td>7-7</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-1</td>
<td>G</td>
<td>7-8</td>
<td>G</td>
<td>7-8</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-2</td>
<td>G</td>
<td>7-9</td>
<td>G</td>
<td>7-9</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-3</td>
<td>G</td>
<td>7-10</td>
<td>G</td>
<td>1-10</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-4</td>
<td>G</td>
<td>7-11</td>
<td>G</td>
<td>1-11</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-5</td>
<td>G</td>
<td>7-12</td>
<td>G</td>
<td>1-12</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-6</td>
<td>G</td>
<td>7-13</td>
<td>G</td>
<td>1-13</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-1</td>
<td>G</td>
<td>7-14</td>
<td>G</td>
<td>2-1</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-2</td>
<td>G</td>
<td>8-1</td>
<td>G</td>
<td>2-2</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-3</td>
<td>G</td>
<td>8-2</td>
<td>G</td>
<td>2-3</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-4</td>
<td>G</td>
<td>8-3</td>
<td>G</td>
<td>2-4</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-5</td>
<td>G</td>
<td>8-4</td>
<td>G</td>
<td>2-5</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-6</td>
<td>G</td>
<td>8-5</td>
<td>G</td>
<td>2-6</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-7</td>
<td>G</td>
<td>8-6</td>
<td>G</td>
<td>2-7</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-8</td>
<td>G</td>
<td>9-1</td>
<td>G</td>
<td>2-8</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-9</td>
<td>G</td>
<td>9-2</td>
<td>G</td>
<td>2-9</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-10</td>
<td>G</td>
<td>9-3</td>
<td>G</td>
<td>2-10</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-11</td>
<td>G</td>
<td>9-4</td>
<td>G</td>
<td>2-11</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-12</td>
<td>G</td>
<td>9-5</td>
<td>G</td>
<td>2-12</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-13</td>
<td>G</td>
<td>9-6</td>
<td>G</td>
<td>2-13</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-14</td>
<td>G</td>
<td>9-7</td>
<td>G</td>
<td>2-14</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-15</td>
<td>G</td>
<td>9-8</td>
<td>G</td>
<td>2-15</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-16</td>
<td>G</td>
<td>9-9</td>
<td>G</td>
<td>2-16</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-1</td>
<td>G</td>
<td>9-10</td>
<td>G</td>
<td>3-1</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-2</td>
<td>G</td>
<td>10-1</td>
<td>G</td>
<td>3-2</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-3</td>
<td>G</td>
<td>10-2</td>
<td>G</td>
<td>3-3</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-4</td>
<td>G</td>
<td>10-3</td>
<td>G</td>
<td>3-4</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-5</td>
<td>G</td>
<td>10-4</td>
<td>G</td>
<td>3-5</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-6</td>
<td>G</td>
<td>10-5</td>
<td>G</td>
<td>3-6</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-1</td>
<td>G</td>
<td>10-6</td>
<td>G</td>
<td>4-1</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-2</td>
<td>G</td>
<td>10-7</td>
<td>G</td>
<td>4-2</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-3</td>
<td>G</td>
<td>10-8</td>
<td>G</td>
<td>4-3</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-4</td>
<td>G</td>
<td>10-9</td>
<td>G</td>
<td>4-4</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-5</td>
<td>G</td>
<td>10-10</td>
<td>G</td>
<td>4-5</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-6</td>
<td>G</td>
<td>10-11</td>
<td>G</td>
<td>4-6</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-1</td>
<td>G</td>
<td>10-12</td>
<td>G</td>
<td>5-1</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-2</td>
<td>G</td>
<td>10-13</td>
<td>G</td>
<td>5-2</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-3</td>
<td>G</td>
<td>10-14</td>
<td>G</td>
<td>5-3</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-4</td>
<td>G</td>
<td>10-15</td>
<td>G</td>
<td>5-4</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-5</td>
<td>G</td>
<td>A-1</td>
<td>G</td>
<td>5-5</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-6</td>
<td>G</td>
<td>A-2</td>
<td>G</td>
<td>5-6</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-7</td>
<td>G</td>
<td>A-3</td>
<td>G</td>
<td>5-7</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-8</td>
<td>G</td>
<td>A-4</td>
<td>G</td>
<td>5-8</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-9</td>
<td>G</td>
<td>A-5</td>
<td>G</td>
<td>5-9</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-10</td>
<td>G</td>
<td>A-6</td>
<td>G</td>
<td>5-10</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6-1</td>
<td>G</td>
<td>A-7</td>
<td>G</td>
<td>6-1</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6-2</td>
<td>G</td>
<td>A-8</td>
<td>G</td>
<td>6-2</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6-3</td>
<td>G</td>
<td>A-9</td>
<td>G</td>
<td>6-3</td>
<td>G</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This manual describes the program library maintenance utility Modify. Modify is part of the Network Operating System (NOS) Version 2. Modify is used to maintain and update source files that are on libraries in a compressed format. NOS can operate on the following computer systems:

- CDC® CYBER 180 Computer Systems
  Models 810, 830, 835, 840, 845, 850, 855, 860, 990

- CDC CYBER 170 Computer Systems
  Models 171, 172, 173, 174, 175, 176, 720, 730, 740, 750, 760, 815, 825, 835, 855, 865, and 875

- CDC CYBER 70 Computer Systems
  Models 71, 72, 73, and 74

- 6000 Computer Systems

AUDIENCE

Because the advantages of Modify are best utilized by a programmer with a large volume of source program text or symbolic data, the manual is written for the experienced NOS applications or systems programmer.

ORGANIZATION

Section 1 describes the purposes and features of the Modify utility, and Section 2 provides a tutorial overview of basic Modify concepts. Section 3 describes the MODIFY command. The remaining sections describe Modify directives and file formats. Appendix A describes the OPEDIT utility, which is used to manage modification sets created by Modify.

SUBMITTING COMMENTS

The last page of this manual is a comment sheet. Please use it to give your opinion on the manual's usability, to suggest specific improvements, and to report any errors. If the comment sheet has already been used, you can mail your comments to:

Control Data
Technical Publications
4201 North Lexington Avenue
St. Paul, Minnesota 55126-6198

60450100 G
Additionally, if you have access to SOLVER, which is an online facility for reporting problems, you can use it to submit comments about the manual. Use NS2 as the product identifier.

CONVENTIONS

CONTROL STATEMENT

The manuals of many NOS products use the term control statement instead of the term command. This manual uses the term command almost exclusively. You can consider the two synonymous.

EXAMPLES

The following conventions apply to examples that appear in this manual:

- Examples of actual terminal sessions that appear in this manual were produced on a display terminal in NORMAL character mode unless otherwise specified. Uppercase characters represent terminal output; lowercase characters represent user input unless otherwise noted. (However, user input that is displayed within the text of this manual is shown in uppercase characters). The vertical spacing in examples does not necessarily coincide with the spacing that appears on your terminal.

- Program examples are written either in FORTRAN 5 or in COMPASS.

COMMAND FORMAT

Interpret uppercase characters within command formats literally. Lowercase characters indicate variable values which are described immediately following the line that shows the command format.

RELATED PUBLICATIONS

The following is a list of NOS operating system manuals and NOS product set reference manuals.

Control Data publishes a Software Publications Release History of all software manuals and revision packets it has issued. This history lists the revision level of a particular manual that corresponds to the level of software installed at the site.

These manuals are available through Control Data sales offices or Control Data Literature Distribution Services (308 North Dale, St. Paul, Minnesota 55103).
<table>
<thead>
<tr>
<th>Control Data Publication</th>
<th>Publication Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOS Version 2 Reference Set, Volume 3, System Commands</td>
<td>60459680</td>
</tr>
<tr>
<td>NOS Version 2 Reference Set, Volume 4, Program Interface</td>
<td>60459690</td>
</tr>
<tr>
<td>FORTRAN Version 5 Reference Manual</td>
<td>60481300</td>
</tr>
<tr>
<td>COMPASS Version 3 Reference Manual</td>
<td>60492600</td>
</tr>
</tbody>
</table>

You might also want to consult the NOS System Information Manual. This is an online manual that includes brief descriptions of all NOS operating system and NOS product set manuals. You can access this manual by logging into NOS and entering the command EXPLAIN.

**DISCLAIMER**

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or undefined parameters.
CONTENTS

1. INTRODUCTION
   Additional Features
   Input and Output Files
   Modify Input Files
   Source Files
   Program Libraries
   Directives File
   Output Files
   Compile File
   New Program Library
   Source Output File
   Statistical List File
   1-1

2. MODIFY EXECUTION
   Modify Directives
   Directive Placement
   Directive Format
   Modify Execution
   Initialization Phase
   Old Program Libraries
   Source Files
   Modification Phase
   Line Identifiers
   Modification Sets
   Line Modifications
   Insertion Lines
   Editing Decks
   Using Alternate Directives Files
   Compile Phase
   Compile File Expansion
   IF Directive
   Common Deck Calls
   2-1

3. MODIFY COMMAND
   3-1

4. INITIALIZATION DIRECTIVES
   COPY Directive
   CUPFP Directive
   CREATE Directive
   OPLFILE Directive
   Examples of Initialization Directives
   4-1

5. MODIFICATION DIRECTIVES
   DECK Directive
   DELETE (or D) Directive
   EDIT Directive
   IDENT Directive
   IGNORE Directive
   INSERT (or I) Directive
   MODNAME Directive
   PURDECK Directive
   RESTORE Directive
   UNYANK Directive
   YANK Directive
   Examples of Modification Directives
   5-1

6. FILE MANIPULATION DIRECTIVES
   BKSP Directive
   READ Directive
   READPL Directive
   RETURN Directive
   REWIND Directive
   SKIP Directive
   SKIPR Directive
   Examples of File Manipulation Directives
   6-1

7. COMPIL FILE DIRECTIVES
   CALL Directive
   CALLALL Directive
   CALLC Directive
   COMMENT Directive
   CSET Directive
   CWROR Directive
   ELSE Directive
   ENDIF Directive
   IF Directive
   IFCALL Directive
   NIFCALL Directive
   NOSEQ Directive
   SEQ Directive
   SORSEQ Directive
   WOR Directive
   WEOR Directive
   WIDTH Directive
   Compile File Directive Examples
   7-1

8. SPECIAL DIRECTIVES
   DEFINE Directive
   INWIDTH Directive
   MOVE Directive
   PREFIX Directive
   PREFIXC Directive
   UPDATE Directive
   / Directive
   Examples of Special Directives
   8-1

9. MODIFY FILE FORMATS
   Source Decks and Files
   User-Prepared Input Source File
   Modify-Generated Source Files
   Program Library Files
   Deck Records
   Prefix Table Format
   Modification Table Format
   Text Format
   Directory Record
   Prefix Table Format
   Directory Table Format
   Directives File
   Compile File
   Compressed Compile File Format
   9-1
1-1 Modify Execution Flow
1-2 Modify Execution from Batch Job
1-3 Modify Execution from Interactive Job
2-1 Modify Source Deck
4-1 Initialization Directives Examples
4-2 Batch Job Creating Program Libraries

1-2 5-1 Modification Directive Examples
1-6 6-1 File Manipulation Directives Examples
1-6 7-1 Compile File Directives Examples
2-5 7-9 NOS Procedure File With ASCII Deck
4-4 8-1 Special Directive Example
4-6 9-1 Library File Format

2-1 Initialization Directives
2-2 Modification Directives
2-3 File Manipulation Directives

2-14 2-4 Compile File Directives
2-14 2-5 Special Directives
2-15
INTRODUCTION

Modify is a NOS system utility used to maintain large source program files. Modify's primary functions are as follows:

- To store program source files in a special, compressed library format. These library files are called program libraries.
- To maintain status information and modification history information for each line of code or text in a program library.
- To allow direct modification of program libraries through the use of Modify directives.
- To output various types of files including program libraries, compiler or assembler input files, and statistical listing files.

The Modify utility is called by the MODIFY command, described in section 3. On the MODIFY command, you specify the old program library you want to modify, the output files you want Modify to generate, and the processing options for the Modify run.

You can run Modify as a batch or interactive job. When you enter an interactive MODIFY command and do not specify an input directives file, Modify prompts you to enter directives interactively.

ADDITIONAL FEATURES

Other features of Modify include:

- Formatting of lines to facilitate line-by-line modification of the file.
- Insertion, deletion, deactivation, or reactivation of individual lines within the file.
- Deactivation or reactivation of one or more groups of changes (called modification sets) previously made to the text.
- Replacement of often-used groups of lines (called common decks) by one-line calls for their insertion.
- Inclusion of both DISPLAY code (uppercase only) and ASCII code (upper and lowercase) decks on the same program library.
- Generation of an output file in a format suitable for input to an assembler or compiler.
- Limiting the range of modifications to specified records (decks).
• Output of statistical information regarding program library status and lines modified.
• Support of 63- and 64-character sets.

INPUT AND OUTPUT FILES

Figure 1-1 is a block diagram showing the execution flow of a Modify run in terms of the files used during the run.

Figure 1-1. Modify Execution Flow
MODIFY INPUT FILES

As shown in figure 1-1, there are three types of Modify input files:

- Source files
- Program libraries
- Directives file

Source Files

Input source files for Modify are program files, NOS procedure files, or text files to be converted to program library format. A file must be converted to program library format before Modify can perform any other operations on it.

In program library format, a logical record is referred to as a deck. In creating a source input file for Modify, it is usually advantageous to place each program, subroutine, or procedure in a separate deck. This practice allows you to modify, reassemble, or recompile the program library in smaller, individual units.

The only special preparation required for source files before being input to Modify are the addition of one or more header lines in each deck. The first line is required and specifies the name of the deck. You can include other lines to designate a deck as a common deck or to define the code set (lowercase only or uppercase and lowercase) of the deck. Section 2 describes source file preparation in greater detail.

Program Libraries

One of the key features of Modify is the use of program libraries for storage and modification of source files. A program library is a source file that has been converted to program library format during a previous Modify run. A program library has the following special characteristics:

- It is compressed: Modify has replaced three or more consecutive blanks within a line with special compression codes.
- It is sequenced: Modify has assigned a sequence number and modification name to each line of the file. This information is printed on the compile output file.
- It is indexed: Modify has built a directory of the decks (records) on the program library file.

Modify uses program libraries for both input and output. By convention, input program libraries are referred to as old program libraries (OPLs). The primary OPL file is defined on the MODIFY command P parameter. Additional OPLs can be specified using one of the Modify directives, the OPLFILE directive.
A deck containing a frequently used group of source statements, such as a group of DATA or COMMON statements in a FORTRAN program, can be designated as a common deck. Modify has special compiler directives that allow you to call a common deck from anywhere in the program library. A call to a common deck causes Modify to insert the common deck into the compile output file at the point from which it was called. This eliminates the need for duplicate sequences of lines at multiple points within a library.

Section 9 describes the file structure of a program library file.

Directives File

You initiate the Modify utility by entering a MODIFY command. Most of the operations that Modify performs, however, are controlled by Modify directives that you enter from a directives file.

The directives file is a file containing the directives that define the operations to be performed in a particular Modify run. You have a number of options in creating the directives file.

For batch jobs, you have three options:

- You can place directives and insertion text in a NOS file. The name of the directives file is then specified on the MODIFY command I parameter (I=filename format).
- The Z parameter of the MODIFY command allows you to append directives to the MODIFY command following the command terminator.
- If both the I and Z parameters are omitted from the MODIFY command, Modify looks for the directives file in the next record following the job from which Modify was called.

For interactive jobs, you can use the I and Z parameters in the same way they are used in batch jobs. A third option is to enter directives interactively. If you do not specify either the I or Z parameter on the MODIFY command, Modify prompts you to enter directives from file INPUT.

There are numerous examples of directive files contained in the examples in sections 4 through 9 of this manual.
OUTPUT FILES

Modify produces four types of output files:

- Compile file
- New program library file
- Source output file
- Statistical list file

Compile File

The compile file is a program text file that contains the edited OPL decks with modifications incorporated by Modify. Each line of the compile file has line-sequencing information used to identify individual lines for modification. You can use the compile file as input to the COMPASS assembler or to a compiler. You can also send the compile file to an output device for printing. The MODIFY command C parameter selects compile file output. Other MODIFY parameters control various processing options that you can select when the compile file is to be input directly to a language processor.

New Program Library

The new program library file contains the same updated information as the compile file except in program library format. You can use the new program library as input to a subsequent Modify run.

Source Output File

This file contains the same updated text as the compile file except that compile file directives have not been processed or removed from the file. Line-sequencing information does not appear on the source output file unless you explicitly request it using the SORSEQ directive.

Statistical List File

The statistical list file lists information on the Modify run. This information includes details of program text changes, file status, errors, and other significant events that occur during the run.

Figures 1-2 and 1-3 provide examples of a batch job and an interactive job executing a Modify run.

60450100 G 1-5
JOBMOD.
USER,USERNAME,PASSWRD,FAMILY.
GET,MAINP.
MODIFY,p=0,f,N.
SAVE,NPL=MAINPL.
--EOR--
*REWIND MAINP)
*CREATE MAINP
---EOI--
Input directives for MODIFY command.
End-of-information.

Figure 1-2. Modify Execution from Batch Job

/old,mainp
/scopy,mainp
DECK1
*** MAIN PROGRAM
PROGRAM MAIN
PRINT*,’BEGIN MAIN PROGRAM.’
CALL SUB1
PRINT*,’END MAIN PROGRAM.’
STOP
END
--EOR--
DECK3
*** EMPTY DECK
--EOR--
--EOF--
/modify,p=0,f,n,l=0
? *create mainp
?
MODIFICATION COMPLETE.
/replace,npl=mainpl

l=0 specifies no Modify output.
Input directives are requested and entered immediately following MODIFY command. Null input line (carriage return only) terminates input.
Modify lists a completion message.

Figure 1-3. Modify Execution from Interactive Job
Section 2 provides an overview of Modify concepts and the basic information you need to know to run a Modify job. Topics discussed include: how to create and use Modify input files, how to use Modify directives, how to create and use modification sets, and how to control expansion of the compile file.

MODIFY DIRECTIVES

Most of the control you have over Modify execution is through the use of Modify directives. There are five kinds of Modify directives:

- Initialization Directives
  These directives specify additional program libraries or source files to input to Modify. The MODIFY command P parameter specifies the primary OPL to be modified. If additional files are to be merged with the primary OPL, these are specified by initialization directives.

- Modification Directives
  These directives define the actual changes to be made to individual lines or groups of lines in the OPL.

- File Manipulation Directives
  These directives perform file positioning functions and specify alternate files from which Modify directives and insertion text are to be read.

- Compile File Directives
  These directives allow you to control which decks of the modified OPL are written to the compile file. Common deck calls and conditional expansion directives control the selection and ordering of decks written to the compile file.

- Special Directives
  These directives perform miscellaneous functions such as redefining the directive prefix character, moving decks within a program library, and defining the line length of input files.

Tables 2-1 through 2-5 at the end of this section provide a brief description of each of the Modify directives. You may want to refer to these tables as you read through this section.
DIRECTIVE PLACEMENT

As stated in the previous section, there are a number of ways you can enter directives for a
Modify run: interactively from file INPUT; from a directives file; from the command line,
using the Z parameter; or from a record following a batch job file.

The following rules describe the sequence and placement of Modify directives within the
directives file:

- All initialization directives must precede all modification directives.
- File manipulation directives can be placed anywhere within the directives file.
- File manipulation directives cannot be placed in an alternate directives file (as
  specified by a READ or READPL directive).
- Compile file directives are not processed until the compile phase when Modify
  expands the compile output file. When encountered in the directives file, Modify
  does not recognize the compile file directives, but simply processes them as text
  lines. (The WIDTH, SEQ, SORSEQ, and NOSEQ compile file directives are exceptions to
  this rule.)
- The WIDTH, SEQ, SORSEQ, and NOSEQ compile file directives can be placed anywhere in
  the directives file, or they can be inserted as text lines as part of a modification
  set.
- Special directives can be placed anywhere in the directives file.

DIRECTIVE FORMAT

A Modify directive has the following format:

*dirname p1,p2,....,pn

Where:

* The directive prefix character is in column 1. The asterisk (*) is the
  default. The PREFIX and PREFIXC directives are used to change the
  prefix character.

dirname The directive name,

pi A directive parameter.

The directive name and parameters can be separated by any display code character having a
value of 55g or greater; this includes all alphanumeric characters and any of the
following special characters: : + - * / ( ) $ =

Some directives require specific separators, as noted in the directive descriptions. No
embedded blanks are permitted within a parameter. However, any number of blanks can be
placed between the directive name and the first parameter or between two parameters,
provided the directive does not exceed the maximum line length.
MODIFY EXECUTION

Modify execution takes place in three phases:

- **Initialization phase**
  Modify reads the initialization directives, converts source files to program library format, and merges all input program libraries onto a single file.

- **Modification phase**
  Modify reads the modification directives and insertion text from the directives file.

- **Compile phase**
  Modify incorporates modifications, expands the compile file, and writes edited decks to the selected output files.

INITIALIZATION PHASE

During the initialization phase, Modify reads the initialization directives and merges all the specified input files into a single program library. This library is the internal file referred to in figure 1-1 as the merged old program library. The merged old program library consists of all decks written from the input program libraries and source files and a directory to those decks. It is the merged old program library that is later modified by the modification directives.

Figure 1-1 shows two types of input files: old program libraries and source files.

Old Program Libraries

An old program library (OPL) is a program library created by a previous Modify run. The primary OPL is the file specified on the MODIFY command P parameter. In many cases, this is the only file input to Modify, but you can specify up to 50 additional OPLs. These additional OPL input files are specified on an OPLFILE directive placed in the input directive file.

OPL files on tape or without a directory require special handling. The COPYPL directive is used to copy an OPL from tape to a local NOS file. The COPY command copies decks from one program library to another, creating a new directory for the copied decks. Thus, COPY can be used to create a directory for an OPL that does not have a directory, to create a new directory to replace an old one, or to copy selected decks from one program library to another.

Source Files

The input file marked source file in figure 1-1 refers to source program or data files that are not yet converted to program library format. Source files to be converted must be specified in a CREATE initialization directive included in the directive file. If no OPL files are specified for a Modify run, the CREATE directive essentially creates a new program library from the original source file. If one or more OPLs are specified for the run, CREATE converts the source file and merges it with the existing program libraries.
Modify allows line lengths up to 150 characters. The 150 characters may be all uppercase, all lowercase, or mixed uppercase and lowercase. Modify assumes a default input line length of 72 characters. If you have a source file containing text lines longer than 72 characters, you should enter an INWIDTH directive specifying the maximum line length. Modify requires this information when writing files in compressed format.

Note that compile file sequence numbers, by default, are written starting in column 73 of the compile file. You can use the WIDTH compile directive to change the location of line sequence numbers on the compile and source output files.

For decks containing NOS procedures, line sequence numbers must be inhibited on the compile file for the procedures to execute properly. A NOSSEQ compile file directive entered for the deck accomplishes this purpose. To obtain a listing that includes the line sequence numbers, you can perform a separate Modify run that includes the line information on the compile file. You can also include a SORSEQ directive which causes the line sequence information to be included on the output source file.

Source files must be properly formatted before they are input to Modify. Formatting of source files may include the following steps:

- A deck name must be assigned to each record on the file.
- Common decks must be marked as such.
- Decks may be marked for expansion as DISPLAY code (uppercase only) or ASCII code (uppercase and lowercase) decks.
- Compile file directives may be inserted in the source file.

Figure 2-1 illustrates the organization of a source file containing two decks.

Except for assigning deck names, all of the listed steps are optional. Each record of the source file must have a deck name that begins in column 1 of the first line of the record. Deck names can be from one to seven characters long and can consist of alphanumeric characters and any of the following special characters:

```
+ - * / ( ) $ =
```

Common decks must have the word COMMON written on the second line beginning in column 1.

If the deck is to be marked as an ASCII or DISPLAY code deck, the code designation must appear either on the line following the deck name or on the line following the COMMON header line. The word ASCII written from column 1 of this line indicates that the deck will be expanded in uppercase and lowercase characters (6/12-bit display code) on the compile file. For uppercase characters only (6-bit display code), write the word DISPLAY on this line. If a code set is not specified, DISPLAY is assumed.

The source file may contain embedded compile file directives, or you may insert compile file directives later as part of a modification set. Generally, the compile file directives are ignored (that is, treated as text lines) during the initialization and modification phases of execution. However, there are four exceptions to this rule: the WIDTH, SEQ, NOSSEQ, and SORSEQ compile file directives. When Modify encounters one of these directives during the initialization phase, the directive applies to all decks output to the compile file. When encountered during the modification phase or when embedded within program library decks, however, these directives affect only the decks with which they are explicitly associated.
MODIFICATION PHASE

Entry of the first modification directive ends the initialization phase and begins the modification phase of execution. During the modification phase, Modify reads the rest of the directives file and makes changes to the internal new program library (NPL) as specified by the modification directives.

During the modification phase of a Modify run, there are three major tasks to be performed:

- Defining a modification set.
- Making individual line modifications as part of a modification set.
- Activating or deactivating a modification set or sets.

It is not necessary to perform all three of these major tasks during every Modify run. In fact, it is not necessary to perform any of them. For example, if you just want to extract certain decks from the OPL without making any changes to the OPL, you can enter an EDIT directive specifying the selected decks. Those decks are then written to the output files with no changes made to the original OPL.

The following paragraphs tell you how to define and manipulate modification sets and how to modify individual lines of the OPL. Also described are the use of the EDIT directive to select decks for editing and the use of the READ and READFL directives, which allow you to define alternate files from which to read Modify directives.
Line Identifiers

Modify assigns a line identifier to each line of code or text in a program library. In sequenced output files, the line identifier comprises the line-sequencing information printed beginning in column 73 of the file. (You can change this position using the WIDTH directive.) The modification directives also use the line identifier to specify the point within a deck at which a line is to be inserted, deleted, or replaced.

The complete form of the line identifier is as follows:

\texttt{modname.number}

Where:

- \texttt{modname} is the name of the modification set to which the line belongs or, if the line is not associated with a modification set, the name of the deck containing the line.
- \texttt{number} is the sequence number of the line within the modification set or deck; must be a decimal ordinal from 1 to 162143. A blank or any character other than 0 through 9 terminates the sequence number. Normally, the numbering of lines belonging to a modification set is reset to 1 at the beginning of each deck; however, you can select continuous sequencing of line numbers by specifying the UPDATE directive.

In practice, you will probably use the abbreviated form of the line identifier more often than the complete form. The abbreviated form uses a default modname determined by a DECK or MODNAME directive. Default modnames are discussed under Line Modifications. The abbreviated form of the line identifier is as follows:

\texttt{number}

where number is the same as above.

Example:

\begin{verbatim}
IF(XVALUE.LE.0) GO TO 84
XVALUE=XVALUE-1
CALL XRTN (XVAL)
84 CONTINUE
\end{verbatim}

This example gives you an idea of what sequenced compile file output looks like. The example represents four lines extracted from the compile file listing for a deck called PROC. Three of the lines use the deck name for the modname portion of the line identifier. These lines belonged to the original source deck. Since lines PROC.132 through PROC.133 are missing, we can assume that they were deleted from the deck by a subsequent modification set.

The line marked MOD12.18 was inserted as part of a modification set called MOD12. Normally, the sequencing of lines inserted by a modification set is reset at the beginning of each deck; therefore, we might assume that deck PROC contains 17 prior lines inserted by modification set MOD12. This is not always the case, however, since the UPDATE directive allows you to specify continuous sequencing of line identifiers without regard to deck residency.
Modification Sets

The modification directives control source file modifications at two different levels: individual lines and modification sets. Modification of individual lines is self-explanatory; the INSERT, DELETE, and RESTORE directives insert new lines of program code and delete or replace old lines of code.

The term modification set refers to a group of individual line modifications considered as a single set. The Modify utility allows you to group individual line changes together into modification sets and to assign a unique name to each set. For each line affected by a modification set, Modify enters a modification history byte in the modification table of the deck containing the line (Refer to section 9 for more information on the program library format). Using modification sets is advantageous for at least two reasons:

- Modification sets simplify the task of keeping a historical record of changes made to a program. At any time, you can obtain an output listing identifying the lines affected by a particular modification set.

- Modification sets can be manipulated as a whole. You can activate, deactivate, or purge an entire modification set using a single modification directive.

You define a new modification set by entering an IDENT directive specifying the name of the modification set. For example, the following IDENT directive establishes a modification set with the name NEWMOD:

*IDENT NEWMOD

Once you have entered an IDENT directive, any modifications you make to the file belong to that modification set until you enter another IDENT directive. Lines inserted or changed as part of the modification set are identified as such on the output files; the modification name appears as the modname portion of the line identifier.

The IDENT directive is not required. If you do not enter an IDENT directive, the default name ******** is used for the modification set. It is recommended, however, that you define a unique name for each modification set you create.

Once you have created a modification set, you can manipulate lines within it as a set, using the YANK and UNYANK directives. By specifying a particular modification set in a YANK directive, you remove the effects of the entire modification set from the program library. Any lines inserted or activated by the modification set are deactivated, and any lines deactivated by the modification set are reactivated by the YANK directive. Similarly, the UNYANK directive restores the effects of an entire modification set.

The OPLEDIT utility, described in appendix A, provides some additional tools for manipulating modification sets.
Line Modifications

Modification of lines in a program library is done on a deck-by-deck basis. Before you can enter any modifications for a particular deck, you must enter a DECK directive to identify the deck. The entry of a deck directive also establishes the default modname for line identifiers in subsequent modification directives.

Example:

*DECK OPT/134
*DELETE 6

The first line of this example opens for modification a deck called OPT/134. The first line also establishes that deck name as the default modname for subsequent modification directives. Since OPT/134 is the default modname, the second line of the example is equivalent to line:

*DELETE OPT/134.6

The default modname established by a DECK directive remains in effect until another DECK directive is entered or until a MODNAME directive is entered.

The MODNAME directive is used to explicitly change the default modname for subsequent line identifiers. The modname specified by MODNAME can be the name of any modification set previously in effect for the deck being modified. Once the default modname has been changed in this way, MODNAME can also be used to reestablish the current deck name as the default modname.

Example:

*DECK OPT/134
*DELETE 6
*MODNAME MOD3A
*DELETE 27
*INSERT 33
  Insert this line after line MOD3A.33
*MODNAME OPT/134

The third line of this example establishes the default modname MOD3A. MOD3A is the name of a modification set created during a previous Modify run. The following DELETE directive then deletes line MOD3A.27 from deck OPT/134. The fifth line instructs Modify to add an insertion line following line MOD3A.33 of the deck; the INSERT directive is followed by the line to be inserted. The last line of the example reestablishes deck name OPT/134 as the default modname.
The use of the MODNAME directive is convenient if you have several lines to be changed for the named modification set. However, since there were only two lines to be changed for MOD3A in the above example, it probably would have been more efficient to omit both MODNAME directives and simply specify the full form of the line identifier for the MOD3A lines to be changed as follows:

*DELETE MOD3A.27
*INSERT MOD3A.33

Do not confuse the default modname specified by DECK or MODNAME with the name of the current modification set as specified by the IDENT directive. DECK and MODNAME affect only the line identifiers used by the modification directives to locate changes in the deck to be modified. When the changed lines are printed in the compile file at the end of the Modify run, the modname portion of their line identifiers reflects the modification set name defined by the IDENT directive in effect at the time the changes were made.

After you have defined a DECK using the DECK directive, you can begin making modifications to individual lines in the deck. Line modifications are made using the INSERT, DELETE, and RESTORE directives.

The INSERT directive inserts one or more lines of text into a program library. The lines to be inserted are placed in the directive file following the INSERT directive.

The DELETE and RESTORE directives deactivate and reactivate, respectively, one or more lines of the program library. DELETE and RESTORE can also insert lines of text. Any text lines following DELETE or RESTORE in the directive file are inserted following the deactivated or reactivated lines in the program library.

Example:

*DELETE OPT/134.84
  c Insert two
  c comment lines.
*DECK NEWDECK

The DELETE directive in this example deletes line 84 of deck OPT/134 and inserts two comment lines immediately following the deleted line.

Insertion Lines

Modify continues reading insertion lines following an INSERT, DELETE, or RESTORE directive until it encounters an end-of-record or another modification directive. Modify does not process compile file directives embedded in insertion lines; they are treated as insertion lines. File manipulation directives embedded in insertion lines are processed and may change the source of insertion lines (See Alternate Directive Files below), but do not terminate insertion. In other words, if a file manipulation directive specifies an alternate directive file, Modify immediately begins reading the alternate file, but continues reading insertion text from the current file once the alternate file is exhausted.
Editing Decks

To be written to any of the output files, an OPL deck must be selected for editing, either on the MODIFY command itself or on an EDIT directive placed in the directive file. If you include directives to modify a deck in a Modify run but do not specify the deck as an edited deck, the modifications are not reflected in the output files. Even if a deck is not modified during a Modify run, you must select the deck as an edited deck if you want to include it in the output files.

Modify has three edit modes:

- **Full Edit**: Edits all decks on the OPL; selected by specifying the F parameter on the MODIFY command.
- **Update Edit**: Edits only the OPL decks named in DECK directives in the directives file; selected by specifying the U parameter on the MODIFY command.
- **Selective Edit**: Edits only the OPL decks named in EDIT directives in the directives file; selected if the F and U parameters are omitted from the MODIFY command.

The full and update edits ignore any EDIT directives in the directives file. The full edit creates a new program library by editing all decks on the OPL. The update edit is normally used to generate a compile file listing of only those decks that were modified during a Modify run.

For both the full edit and update edit, decks are edited in the order they occur in the OPL. However, if any decks are added or replaced, the new or replacement decks are written to the end of the program library. If you want to restore the original deck order, you can do so by using the MOVE special directive in a followup Modify run. The MOVE directive moves a specified deck from one position in a program library to another.

When using a full or update edit, the placement of common decks in the program library is significant since decks are edited in the order they appear. If a common deck is modified during a Modify run and is also called by another deck, the calling deck receives the modified version of the common deck only if the common deck precedes the calling deck in the program library. If the calling deck precedes the common deck, then the common deck will not have been modified at the time the call is processed. For this reason, it is usually best to place the common decks at the beginning of the program library.

In a selective edit, decks are edited in the order in which the EDIT directives are encountered in the directive file unless you specify otherwise using the UPDATE special directive. The UPDATE directive tells Modify to ignore the order of the EDIT directives and edit decks in the order they appear on the OPL.

---

† If a common deck is called from another deck, the compile file output for the calling deck will include the common deck lines regardless of whether or not the common deck was selected for editing. However, for common deck itself to be written to the source file or new program library, the common deck must be selected for editing.
Using Alternate Directives Files

There are two file manipulation directives, READ and READPL, that direct Modify to temporarily stop reading directives from the primary directives file (specified by the I parameter of the MODIFY command) and begin reading directives from an alternate file. When Modify finishes reading directives from the alternate file, it returns to the primary directives file to read the remainder of it.

The READ directive directs Modify to begin reading directives from a specified local file. The READ directive has an option that allows you to read directives either from a single record on the file or to read the entire file.

The READPL directive directs Modify to begin reading directives from a specified deck in the OPL. This directive allows you to copy lines from one position to another within the OPL.

Modify directives placed in an alternate directives file are processed as they would be in the primary directives file with one exception: file manipulation directives may not be placed in an alternative directive file and cause Modify to abort if they are encountered there. Modification and special directives placed in an alternate file are processed normally, and compile file directives are treated as text lines.

Note that if Modify is reading insertion lines (that is, text lines following an INSERT, DELETE, or RESTORE modification directive) from the primary directives file when READ or READPL is encountered, the READ or READPL directive does not terminate insertion. When Modify finishes reading the alternate file and returns to the primary file, insertion continues until Modify encounters another modification directive.

COMPILE PHASE

During the compile phase of execution, Modify incorporates changes specified during the modification phase and writes the selected output files. Any compile file directives encountered in the OPL or in the directives file are processed at this time, and the expanded decks are written to the compile file.

Compile File Expansion

The compile file directives provide a number of ways of controlling the composition and format of the compile file output.

- Various types of call directives call common decks to be written to the compile file.
- The IF directive performs a true or false test for a specified condition to determine whether a block of lines is included or deleted from the compile file.
- You can insert file marks in the compile file using the WEOR, CWEOR, and WEOF directives.
- WIDTH, NOSEQ, and SEQ control the line length and inclusion of line sequence information.
The process of removing compile file directives and replacing them with the specified insertion or deletion operations is referred to as expansion of the compile file. Most of these expansion options are self-explanatory; however, a word about common deck calls and the IF directive might be helpful.

First of all, the IFCALL, NIFCALL, and IF directives are dependent on the previous specification of a symbolic name using the DEFINE special directive. The DEFINE directive defines a symbolic name to Modify and can also associate a numeric value with the name. Defining a symbolic name in this way is like setting a software flag that Modify checks when processing an IFCALL, NIFCALL, or IF directive.

IFCALL calls a common deck if a specified name was previously defined by a DEFINE directive, while NIFCALL calls a common deck only if the specified name was not defined.

IF Directive

The IF directive works in conjunction with ENDFIF and ELSE to set off a block of lines for conditional inclusion in the compile file. The IF directive is always followed by a block of lines terminated by an ENDFIF or ELSE directive. If the condition tested by IF is true, the block of lines is included in the compile file, and if the condition is false, the lines are not included.

There are two ways the IF directive can use the symbolic name defined by DEFINE. If the IF directive specifies only a symbolic name, the condition is considered to be true if the name is defined, false if the name has not been defined. The IF directive can also compare the symbolic name to a numeric value. In this case, the IF condition is either true or false depending on whether the symbolic name and the specified value were associated in a previous DEFINE directive.
Common Deck Calls

The common deck calls, CALL, CALLALL, CALLC, IFCALL, and NIFCALL provide a number of different options for calling common decks. The individual directives are described in section 6. The following are some general rules you should keep in mind when using common deck calls.

Common deck calls (except CALLALL, which is ignored when encountered as a nested call) can be nested; that is, a common deck can contain a call to another common deck. The conditional common deck call, CALLC, is useful in nested calls. CALLC performs a common deck call only on the condition that the specified deck has not already been called from the deck being edited. The entry of a new DECK resets this condition.

Common decks can be designated as either ASCII (6/12-bit display code) or DISPLAY code (6-bit display code) decks by including the code set header line as the second or third line of the deck. An ASCII common deck can be called from a DISPLAY code deck and vice versa. Unless overridden by a CSET directive, common decks are expanded according to their ASCII or DISPLAY code designation. In other words, if an ASCII common deck is called by a DISPLAY code deck, the common deck lines are inserted into the calling deck as uppercase and lowercase lines, even though the calling deck is written in uppercase characters only.

The CSET compile file directive allows you to override the code set designation of a common deck. If a CSET ASCII or CSET DISPLAY directive is entered prior to a common deck call, the common deck is expanded in the code set specified by CSET, regardless of the code set defined for the deck. If the common deck contains nested calls to other common decks, all nested calls are also governed by this CSET directive.

The CSET directive remains in effect until another CSET is encountered or until the end of the deck. A CSET encountered within a common deck is ignored, and the line is placed in the compile file as a comment.
### Table 2-1. Initialization Directives

<table>
<thead>
<tr>
<th>Directive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPY</td>
<td>Copies one or more records from one program library file to another; normally used to copy a program library from tape to system file.</td>
</tr>
<tr>
<td>COPYPL</td>
<td>Copies one or more decks from a program library file to the internal OPL scratch file. Ignores the original file directory and creates a new one for the copied decks.</td>
</tr>
<tr>
<td>CREATE</td>
<td>Converts a properly formatted source file to program library format.</td>
</tr>
<tr>
<td>OPLFILE</td>
<td>Declares additional input OPL files (that is, in addition to the file specified by the MODIFY command P parameter).</td>
</tr>
</tbody>
</table>

† The initialization directives are described in section 4.

### Table 2-2. Modification Directives

<table>
<thead>
<tr>
<th>Directive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECK</td>
<td>Identifies the next deck to be modified and establishes the default deck name for subsequent line identifiers.</td>
</tr>
<tr>
<td>DELETE or D</td>
<td>Deactivates lines and optionally inserts replacement lines in their place.</td>
</tr>
<tr>
<td>EDIT</td>
<td>Specifies a deck or decks to be edited and written to the selected output files; valid only if neither the F (Full edit) nor U (Update) parameter was specified on the MODIFY command.</td>
</tr>
<tr>
<td>IDENT</td>
<td>Defines a new modification set.</td>
</tr>
<tr>
<td>IGNORE</td>
<td>Specifies that subsequent modifications for the current deck are to be ignored.</td>
</tr>
<tr>
<td>INSERT or I</td>
<td>Inserts lines after a specified line.</td>
</tr>
<tr>
<td>MODNAME</td>
<td>Specifies the name of a modification set previously defined for this deck. The specified name becomes the default modification set name for subsequent line identifiers.</td>
</tr>
<tr>
<td>PURDECK</td>
<td>Permanently removes a deck from the program library.</td>
</tr>
<tr>
<td>RESTORE</td>
<td>Reactivates lines and optionally inserts lines following the reactivated lines.</td>
</tr>
<tr>
<td>UNYANK</td>
<td>Reactivates a modification set.</td>
</tr>
<tr>
<td>YANK</td>
<td>Deactivates a modification set.</td>
</tr>
</tbody>
</table>

† The modification directives are described in section 5.
### Table 2-3. File Manipulation Directives†

<table>
<thead>
<tr>
<th>Directive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BKSP</td>
<td>Backspaces a specified number of records on a file.</td>
</tr>
<tr>
<td>READ</td>
<td>Reads a record or group of records from a specified file.</td>
</tr>
<tr>
<td>READPL</td>
<td>Reads a deck or portion of a deck from within the program library file.</td>
</tr>
<tr>
<td>RETURN</td>
<td>Returns the named file to the system.</td>
</tr>
<tr>
<td>REWIND</td>
<td>Rewinds the named files.</td>
</tr>
<tr>
<td>SKIP</td>
<td>Skips a specified number of records on a file.</td>
</tr>
<tr>
<td>SKIPR</td>
<td>Skips past a specified record on a file.</td>
</tr>
</tbody>
</table>

† The file manipulation directives are described in section 6.

### Table 2-4. Compile File Directives (Sheet 1 of 2)†

<table>
<thead>
<tr>
<th>Directive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL</td>
<td>Writes a specified common deck to the compile file.</td>
</tr>
<tr>
<td>CALLALL</td>
<td>Writes all decks to the compile file that have a deck name beginning with a specified character string.</td>
</tr>
<tr>
<td>CALLC</td>
<td>Writes a specified common deck to the compile file only if the deck has not already been called.</td>
</tr>
<tr>
<td>COMMENT</td>
<td>Generates a COMMENT pseudo instruction for COMPASS.</td>
</tr>
<tr>
<td>CSET</td>
<td>Specifies the code set (ASCII or DISPLAY) for subsequently called common decks; overrides the code set defined for a common deck.</td>
</tr>
<tr>
<td>GWBDR</td>
<td>Writes an EOR to the compile file only if information has been written since the last EOR was written.</td>
</tr>
<tr>
<td>ELSE</td>
<td>Reverses an IF directive conditional range; that is, it terminates insertion if the IF directive was true and initiates insertion if the IF directive was false.</td>
</tr>
<tr>
<td>ENDIF</td>
<td>Terminates insertion of lines initiated by an IF or ELSE directive.</td>
</tr>
<tr>
<td>IF</td>
<td>Inserts lines in the compile file if a specified attribute is true; insertion continues until an ELSE or ENDIF directive is encountered.</td>
</tr>
<tr>
<td>IFCALL</td>
<td>Writes a specified common deck to the compile file if a specified name has been defined (by a DEFINE special directive).</td>
</tr>
</tbody>
</table>

† The compile file directives are described in section 7.
Table 2-4. Compile file Directives (Sheet 2 of 2)†

<table>
<thead>
<tr>
<th>Directive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIFCALL</td>
<td>Writes a specified common deck to the compile file only if a specified name has not been defined (by a DEFINE special directive).</td>
</tr>
<tr>
<td>NOSEQ</td>
<td>Inhibits inclusion of line sequence information on the compile file.</td>
</tr>
<tr>
<td>SEQ</td>
<td>Specifies inclusion of line sequence information on the compile file.</td>
</tr>
<tr>
<td>SORSEQ</td>
<td>Specifies inclusion of line sequence information on the source output file.</td>
</tr>
<tr>
<td>WEOF</td>
<td>Writes an EOF to the compile file.</td>
</tr>
<tr>
<td>WEOR</td>
<td>Writes an EOR to the compile file.</td>
</tr>
<tr>
<td>WIDTH</td>
<td>Defines the number of columns preceding sequence information on the compile file.</td>
</tr>
</tbody>
</table>

†The compile file directives are described in section 7.

Table 2-5. Special Directives†

<table>
<thead>
<tr>
<th>Directive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFINE</td>
<td>Defines a name to be used by conditional compiler directives (IFCALL, NIFCALL, IF) to determine the appropriate action to take.</td>
</tr>
<tr>
<td>INWIDTH</td>
<td>Defines the line length for Modify input files.</td>
</tr>
<tr>
<td>MOVE</td>
<td>Moves decks within a program library; used to reorder library decks.</td>
</tr>
<tr>
<td>PREFIX</td>
<td>Changes the prefix character for directives other than compile file directives.</td>
</tr>
<tr>
<td>PREFIXC</td>
<td>Changes the prefix character for compile file directives.</td>
</tr>
<tr>
<td>UPDATE</td>
<td>Specifies continuous sequencing of program library decks and forces editing of decks in the order they occur in the program library.</td>
</tr>
<tr>
<td>/</td>
<td>Inserts a comment line in the Modify statistical list file.</td>
</tr>
</tbody>
</table>

†The special directives are described in section 8.
The MODIFY command initiates execution of the Modify system utility. Using Modify parameters, you specify the input files, output files, and directives file for the Modify run, as well as the processing options you want to select. Processing options include a direct call to the COMPASS assembler or to a program compiler. When selected, assembler or compiler call takes place at completion of the Modify run.

For output listing files not connected to your terminal, the MODIFY command honors the page length and print density set for your job using the SET command and the PL and PD symbolic names (Refer to the NOS Reference Set, Volume 3 for more information on the SET command and PL and PD symbolic names.)

For interactive users, if the I and Z parameters are omitted from the MODIFY command, the system prompts you to enter directives interactively. The interactive prompt for Modify directives is the ? character. To terminate prompting for directives, enter a carriage return in response to the Modify prompt.

Format:

\texttt{MODIFY, p1,p2,\ldots,pn.}

You can specify parameters \( p_i \) in any order; valid parameters are as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Specifies whether the compile file (as defined by the C parameter) is in normal or compressed format.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>A</td>
<td>Specifies compressed compile file format.</td>
</tr>
<tr>
<td>omitted</td>
<td>Specifies normal compile file format.</td>
</tr>
<tr>
<td>C</td>
<td>Defines the compile file to be generated.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>C or omitted</td>
<td>Compile output is written on default file COMPILE.</td>
</tr>
<tr>
<td>C=filename</td>
<td>Compile output is written on file filename.</td>
</tr>
<tr>
<td>C=0</td>
<td>No compile file is generated.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>CG</td>
<td>Defines the file from which systems text is to be loaded; valid only if the Q or X parameter is also specified. If both CG and CS are specified, CS is ignored.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CG</td>
<td>Loads systems text from file SYSTEXT (same as G=SYSTEXT parameter on COMPASS or compiler call).</td>
</tr>
<tr>
<td>CG=filename</td>
<td>Loads systems text from file filename (same as G=filename parameter on COMPASS or compiler call).</td>
</tr>
<tr>
<td>CG=0 or omitted</td>
<td>Loads systems text from overlay named in CS option.</td>
</tr>
</tbody>
</table>

| CL         | Defines the COMPASS listing file; valid only if the Q or X parameter is also specified. |

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CL</td>
<td>Lists output on file OUTPUT (same as COMPASS L=OUTPUT parameter).</td>
</tr>
<tr>
<td>CL=filename</td>
<td>Lists output on file filename (same as COMPASS L=filename parameter).</td>
</tr>
<tr>
<td>CL=0</td>
<td>Writes short list, rather than full list, to file OUTPUT (Same as COMPASS L=0 parameter).</td>
</tr>
</tbody>
</table>

| CS         | Defines overlay from which systems text is to be loaded; valid only if the Q or X parameter is also specified. |

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS or omitted</td>
<td>Loads systems text from SYSTEXT overlay (same as S=SYSTEXT parameter on COMPASS or compiler call).</td>
</tr>
<tr>
<td>CS=filename</td>
<td>Loads system text from file filename (same as S=filename parameter on COMPASS or compiler call).</td>
</tr>
<tr>
<td>CS=0</td>
<td>No systems text (same as S=0 parameter on COMPASS or compiler call).</td>
</tr>
</tbody>
</table>

<p>| CV         | Specifies character set conversion required for program library input files. Conversion is recommended if the character set of the OPL is not the same as the character set in use at the time the OPL is modified. Use the CATALOG command to determine the character set of the OPL (Refer to Volume 3 for a description of the CATALOG command). Check with your site administration to determine the character set in use at your site. |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>CV=63</td>
<td>Converts 64-character OPL to 63-character set.</td>
</tr>
<tr>
<td>CV=64</td>
<td>Converts 63-character OPL to 64-character set.</td>
</tr>
<tr>
<td>CV=0 or</td>
<td>No conversion occurs.</td>
</tr>
<tr>
<td>omitted</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE**

When either CV=63 or CV=64 is selected, modify forces the C parameter to C=0 (e.g., no compile file is generated).

**D**

Specifies whether a directive error will abort the job; used for debugging purposes.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Job is aborted on fatal error, but not on directive error.</td>
</tr>
<tr>
<td>omitted</td>
<td>Either a fatal error or a directive error aborts the job.</td>
</tr>
</tbody>
</table>

**F**

Specifies whether a full edit is to be performed (that is, whether all decks on the OPL are to be edited).

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>All decks on the OPL are edited and written to the NFL listing, the compile file, and the source file (if these files are generated).</td>
</tr>
<tr>
<td>omitted</td>
<td>Decks to be edited are determined by the U parameter or by EDIT directives.</td>
</tr>
</tbody>
</table>

**I**

Defines the directive input file.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I or omitted</td>
<td>For interactive jobs, directives are on file INPUT. For batch jobs, directives are in the next record of the job file.</td>
</tr>
<tr>
<td>I=filename</td>
<td>Directives are contained in the next record of file filename.</td>
</tr>
<tr>
<td>I=0</td>
<td>No directive input.</td>
</tr>
</tbody>
</table>
### Parameter L

**Description**

Defines the Modify statistical list file. (List options for this file are specified by the LO parameter.) It is the job's responsibility to save the named file after the Modify run is completed.

**Option**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L or omitted</td>
<td>Modify statistical list is written to file OUTPUT, which is automatically printed.</td>
</tr>
<tr>
<td>L=filename</td>
<td>Modify statistical list is written to file filename.</td>
</tr>
<tr>
<td>L=0</td>
<td>No statistical list file is generated.</td>
</tr>
</tbody>
</table>

### Parameter LO

**Description**

Defines the list options to be selected for the Modify statistical list file (as defined by the L parameter).

**Option**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LO or omitted</td>
<td>For an output file connected to an interactive terminal, list option E is selected; otherwise, options C, D, E, M, S, T, and W are selected.</td>
</tr>
<tr>
<td>LO=c1c2...cn</td>
<td>The list options (c_i); up to seven of the following options can be specified:</td>
</tr>
<tr>
<td></td>
<td>A  List active lines in deck.</td>
</tr>
<tr>
<td></td>
<td>C  List directives other than INSERT, DELETE, RESTORE, MODNAME, I, OR D.</td>
</tr>
<tr>
<td></td>
<td>D  List deck status.</td>
</tr>
<tr>
<td></td>
<td>E  List errors.</td>
</tr>
<tr>
<td></td>
<td>I  List inactive lines in deck.</td>
</tr>
<tr>
<td></td>
<td>M  List modifications performed.</td>
</tr>
<tr>
<td></td>
<td>S  Include statistics on listing.</td>
</tr>
<tr>
<td></td>
<td>T  List text input.</td>
</tr>
<tr>
<td></td>
<td>W  List compile file directives.</td>
</tr>
</tbody>
</table>

**Example:**

LO=ADEMS

### Parameter N

**Description**

Defines the new program library output file. It is the job's responsibility to save the file after the Modify run is completed.

**Option**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>New program library is written on file NPL.</td>
</tr>
<tr>
<td>N=filename</td>
<td>New program library is written on file filename.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>N=0 or omitted</td>
<td>No new program library file is generated.</td>
</tr>
</tbody>
</table>

**NOTE**
If a local file already exists with the same name as the new program library file, an EVICT command is issued for the local file before the new file is written. (Refer to the WOS Reference Set, Volume 3 for a description of the EVICT command.)

**NR**
Specifies whether the compile file is rewound at the beginning and end of the Modify run.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NR</td>
<td>The compile file is not rewound.</td>
</tr>
<tr>
<td>omitted</td>
<td>The compile file is rewound at the beginning and at the end of the Modify run.</td>
</tr>
</tbody>
</table>

**P**
Defines the input (old) program library file (OPL).

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P or omitted</td>
<td>File OPL is the input program library file.</td>
</tr>
<tr>
<td>P=filename</td>
<td>File filename is the input program library file.</td>
</tr>
<tr>
<td>P=0</td>
<td>There is no input program library file.</td>
</tr>
</tbody>
</table>

**Q**
Defines a call to an assembler or compiler at the end of the Modify run. The directives file and statistical list file are not rewound.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>Calls the COMPASS assembler. At the beginning of the Modify run, Modify sets L0=E and sets the A parameter. At the end of the run, Modify calls the COMPASS assembler. Assembler input is assumed to be file COMPILE. All other assembler parameters are set by default. If CL is not specified with Q, comment lines beginning with an asterisk in column 1 are not written to the compile file; however, compile file directives are still processed.</td>
</tr>
<tr>
<td>Q=comp</td>
<td>Calls the specified compiler or assembler. At the beginning of the Modify run, Modify sets L0=E and sets the A parameter. At the end of the run, Modify calls the compiler or assembler specified by comp.</td>
</tr>
<tr>
<td>Q=0 or omitted</td>
<td>Modify makes no assembler or compiler call.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td><strong>Option</strong></td>
</tr>
<tr>
<td>S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>S=filename</td>
</tr>
<tr>
<td></td>
<td>S=0 or omitted</td>
</tr>
<tr>
<td>U</td>
<td></td>
</tr>
<tr>
<td></td>
<td>U</td>
</tr>
<tr>
<td></td>
<td>omitted</td>
</tr>
<tr>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
</tbody>
</table>
Initialization directives define files to be used as input files for the Modify run in addition to the program library specified on the MODIFY command P parameter. All files specified by the P parameter and by the initialization directives are merged into a single program library file for modification by subsequent Modify directives.

In the case where files defined by the initialization directives contain two or more decks by the same name, only the most recently defined deck is retained on the internal OPL. In other words, whenever a deck is written to the OPL, that deck logically replaces any previous deck with the same name.

Initialization directives are placed on the directives file, and all initialization directives must precede all modification directives. The initialization directives are as follows:

- COPY
- COPYPL
- CREATE
- OPLFILE

**COPY Directive**

The COPY directive copies one or more decks from a named file to the file specified by the MODIFY command P parameter. Before the copy operation begins, Modify performs an EVICT operation on the file to receive the copy; therefore, any information previously contained in this file (if it exists) is lost. (Refer to the NOS Reference Set, Volume 3 for a description of the EVICT command.) If P=O was specified on the MODIFY command, you cannot use the COPY directive.

With the exception of file manipulation directives, COPY must be the first directive processed during the Modify run. Only one COPY operation is allowed per Modify run.

COPY can copy all or part of the original file. Any records on the file that are not in program library format are ignored. COPY creates new directory entries for decks copied, ignoring the directory on the original file.

COPY is useful when copying all or part of a program library residing on magnetic tape to a mass storage file. You can then save the copied file without having Modify create a new program library.
Format:

*COPY filename,deckname

Where:

filename Specifies the name of a local file containing records in program library format. The file need not have a directory and may contain records not in program library format.

.deckname Specifies the last deck to be copied. If .deckname is omitted, or if the named deck is not found, Modify copies all records from the current file position to end-of-file, excluding any records not in program library format.

COPYPL DIRECTIVE

The COPYPL directive copies one or more decks from a file already in program library format. The decks are copied to the Modify internal OPL file where they are merged with any other input files you have specified.

COPYPL can copy all or part of the original file. The file may reside on disk or on magnetic tape. Any records on the file that are not in program library format are ignored. COPYPL creates new directory entries for decks copied, ignoring the directory on the original file.

Format:

*COPYPL filename,deckname

Where:

filename Specifies the name of a local file containing records in program library format. The file need not have a directory and may contain records not in program library format.

.deckname Specifies the last deck to be copied. If .deckname is omitted, or if the named deck is not found, Modify copies all records from the current file position to end-of-file, excluding any records not in program library format.
CREATE DIRECTIVE

The CREATE directive creates a program library file, with a directory, from a properly formatted source file. (Refer to section 3 for a description of the source file format.) You can use CREATE to create a new program library, to add decks to an existing program library, or to replace decks on an existing program library.

Format:

*CREATE filename

Where:

filename Specifies the name of the source file to be converted to program library format. filename must be a local file.

OPLFILE DIRECTIVE

The OPLFILE directive specifies one or more program library files to be edited during the Modify run, in addition to the file specified by the MODIFY command P parameter.

The total number of files declared by OPLFILE directives cannot exceed 50 files. If more than 50 files are specified, the excess files are ignored, and a directive message is issued.

Format:

*OPLFILE filename1,filename2,...,filenameN

Where:

filename Specifies the name of a local program library file (with a directory) to be merged onto the internal OPL scratch file.

EXAMPLES OF INITIALIZATION DIRECTIVES

Figure 4-1 shows an interactive Modify run that creates two program libraries and uses several initialization directives. Figure 4-2 shows the same Modify run to be submitted as a batch job.
/old,mainu
/scopy,mainu

DECK1
*** MAIN PROGRAM
PROGRAM MAIN
PRINT*, 'BEGIN MAIN PROGRAM.'
call sub1
PRINT*, 'END MAIN PROGRAM.'
STOP
END
--E0R--
DECK3
*** EMPTY DECK
--E0R--
--E0F--
/modify,p=0,i=0,f,n=mainpl,c=0
? *create mainu
?
MODIFICATION COMPLETE.
/catalog,mainpl,

CATALOG OF MAINPL

<table>
<thead>
<tr>
<th>REC</th>
<th>NAME</th>
<th>TYPE</th>
<th>LENGTH</th>
<th>CKSUM</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DECK1</td>
<td>OPL</td>
<td>(64)</td>
<td>27</td>
<td>1457 85/07/30</td>
</tr>
<tr>
<td>2</td>
<td>DECK3</td>
<td>OPL</td>
<td>(64)</td>
<td>4</td>
<td>1723 85/07/30</td>
</tr>
<tr>
<td>3</td>
<td>OPL</td>
<td>OPLD</td>
<td></td>
<td>5</td>
<td>1510 85/07/30</td>
</tr>
<tr>
<td>4</td>
<td>EOF</td>
<td>SUM</td>
<td></td>
<td>40</td>
<td></td>
</tr>
</tbody>
</table>

CATALOG COMPLETE.

/get,sub1
/copycf,sub1

DECK2
*** SUBROUTINE 1
SUBROUTINE SUB1
PRINT*, 'ENTER SUBROUTINE 1.'
call sub2
PRINT*, 'EXIT SUBROUTINE 1.'
return
end
EOF ENCONTERED...
/rewind,sub1
REWIND,SUB1,
/modify,p=0,i=0,f,n=altp1,c=0
? *create sub1
?
MODIFICATION COMPLETE.
/catalog,altp1,

CATALOG OF ALTP1

<table>
<thead>
<tr>
<th>REC</th>
<th>NAME</th>
<th>TYPE</th>
<th>LENGTH</th>
<th>CKSUM</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DECK2</td>
<td>OPL</td>
<td>(64)</td>
<td>30</td>
<td>3217 85/07/30</td>
</tr>
<tr>
<td>2</td>
<td>OPL</td>
<td>OPLD</td>
<td></td>
<td>3</td>
<td>2117 85/07/30</td>
</tr>
<tr>
<td>3</td>
<td>EOF</td>
<td>SUM</td>
<td></td>
<td>33</td>
<td></td>
</tr>
</tbody>
</table>

CATALOG COMPLETE.

/get,altp12,
/catalog,altp12,

DECK 2 is another source deck maintained on a
separate program library.

MODIFY command to create program library ALTP1.
? *create sub1
?
MODIFICATION COMPLETE.
/catalog,altp2,

CATALOG OF ALTP2

<table>
<thead>
<tr>
<th>REC</th>
<th>NAME</th>
<th>TYPE</th>
<th>LENGTH</th>
<th>CKSUM</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DECK3</td>
<td>OPL</td>
<td>(64)</td>
<td>25</td>
<td>2116 85/06/18</td>
</tr>
<tr>
<td>2</td>
<td>OPL</td>
<td>OPLD</td>
<td></td>
<td>3</td>
<td>2517 85/06/18</td>
</tr>
<tr>
<td>3</td>
<td>EOF</td>
<td>SUM</td>
<td></td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>

CATALOG COMPLETE.

ALTP2 is an alternate program library created
at an earlier session.

Figure 4-1. Initialization Directives Examples (Sheet 1 of 2)
Figure 4-1. Initialization Directives Examples (Sheet 2 of 2)
Figure 4-2. Batch Job Creating Program Libraries

JOB1.
USER, USERNUM, PASSWORD, FAMILY.
GET, MAINPL.
MODIFY, P=1, F, N=MAINPL, C=0.
GET, SUBT.
MODIFY, P=0, F, N=ALTPL1, C=0.
GET, ALTPL2.
RENAME, OPL=MAINPL.
MODIFY, F, N=MAINPL.
REPLACE, MAINPL.
FTN5, I=COMPILE.
REPLACE, LGO.
--EDR--
*CREATE MAINP
--EDR--
*CREATE SUB1
--EDR--
*OPLFILE ALTPL1
*COPYPL ALTPL2, DECK3
--EDR--

[Creates new program library MAINPL. Since no directive file is specified, Modify reads directives from the first record following the job file.]

[Creates new program library ALTPL1. Directives are specified in the second record following the job.]

[Merges MAINPL, ALTPL1, and ALTPL2 through DECK3. Directives are specified in the third record following the job.]
MODIFICATION DIRECTIVES

The modification directives are used to define new modification sets and to specify changes to existing lines or modification sets.

Modification directives are placed on the directives file following the last initialization directive. The appearance of the first modification directive terminates the initialization phase.

The modification directives are as follows:

- DECK
- DELETE or D
- EDIT
- IDENT
- IGNORE
- INSERT or I
- MODNAME
- PURDECK
- RESTORE
- UNYANK
- YANK
DECK DIRECTIVE

The DECK directive identifies a deck to be modified as a part of the current modification set (refer to the IDENT directive description). This directive also establishes the default deckname (that is, the modname portion of the line identifier) for line identifiers in subsequent modification directives.

The DECK directive does not affect line identifiers assigned to insertion lines. The IDENT directive defines the modification set name (modname) for insertion lines.

A DECK directive is required for each deck associated with a modification set. The default deck name established by a DECK directive continues until the next DECK directive is processed. You can override the default deck name by specifying the full form of the line identifier (refer to Line Identifiers in section 2).

Format:

*DECK deckname

Where:

dekename  Specifies the 1- to 7-character name of the deck to be modified; deckname can consist of alphanumeric characters and any of the following special characters: + - * / ( ) $ *

DELETE (OR D) DIRECTIVE

The DELETE or D directive deletes a specified line or block of lines and optionally inserts replacement lines for the deleted lines. Lines named in a DELETE directive are logically deleted (that is, deactivated) from the program library. In other words, they remain in the program library and retain their sequencing, but they do not appear on output compile or source files. You can restore deleted lines using the RESTORE directive.

Insertion of replacement lines following a DELETE operation continues until another modification directive or an end-of-record is encountered. File manipulation directives within the insertion lines are processed and may change the source of insertion lines, but do not terminate insertion and are not inserted into the deck. Compile file directives are treated as insertion lines and are not processed.

Format:

*DELETE c1,c2

or

*D c1,c2

Where:

c1  Specifies a line identifier (refer to Line Identifiers in section 2).
If only c1 is specified, it identifies a single line to be deleted.
If c1 and c2 are both specified, they identify the first and last lines of a range of lines to be deleted.

Any lines occurring within the range of lines specified that are already deactivated are not affected by this directive.
EDIT DIRECTIVE

The EDIT directive identifies decks to be modified and written to the selected output files. A deck named in an EDIT directive is written to the output files regardless of whether or not any lines within that deck were actually modified. EDIT directives are ignored if either the F (full edit) or the U (update edit) parameter was specified on the MODIFY command.

Format:

*EDIT P1,P2,...,Pn

Where:

P1 specifies either a single deck name or a range of deck names to be edited. If a range of deck names is specified, P1 has the following format:

deckname1-deckname2

where deckname1 specifies the first deck and deckname2 specifies the last deck in a series of decks to be edited.

Decks are edited in the sequence encountered on EDIT directives unless an UPDATE directive specifies otherwise. If decks are being replaced or if new decks are added, the new decks are placed at the end of the library, thus changing the sequence of decks on the library. This should be taken into account when specifying an edit sequence.

IDENT DIRECTIVE

The IDENT directive establishes a new modification set and defines the name of the set. Subsequent DECK directives are associated with this modification set until another IDENT directive is encountered. The entry of a PURDECK directive also terminates processing of a modification set established by an IDENT directive.

The IDENT directive is optional; however, it is recommended that you include an IDENT directive each time you modify a program library. If you do not include an IDENT directive, Modify assigns the characters ***/ as a default modification name.

Format:

*IDENT modname

Where:

modname defines the 1- to 7-character name of the modification set. modname can consist of alphanumeric characters and any of the following special characters: + - * / ( ) $ =
You can use one IDENT directive for several decks, or you can use multiple IDENT directives within a deck. Within the directives file, an IDENT directive defining a modification set must precede any other directives that reference that modification set.

For each DECK directive following an IDENT directive, a new entry is created in the modification table for that deck. The entry contains the name and status (active or inactive) of the new modification set. Likewise, for each line affected by the new modification set, a new modification history byte is created. The modification history byte marks the line as being associated with that modification set. (Refer to section 9 for information on program library format.)

**IGNORE DIRECTIVE**

The IGNORE directive suspends processing of modification directives (except IDENT, DECK, and EDIT) for a specified deck. If the specified deck is already selected for editing (that is, if the F or U option is specified on the Modify command or if an EDIT directive is already entered for the deck), the deck is written to the selected output files in its original, unmodified form.

If the deck is not already selected for editing and is named in a subsequent EDIT directive, Modify deletes the deck name from the EDIT directive, and the deck is not written to the output files. If an ignored deck is named in an EDIT directive in the following format, a directive error is issued, and the Modify run is aborted:

```
EDIT deckname_a,deckname_b
```

When Modify encounters an IDENT, DECK, or EDIT directive following an IGNORE directive, the IGNORE condition is terminated. Normal processing of modification directives resumes with the IDENT, DECK or EDIT directive.

**Format:**

```
*IGNORE deckname
```

**Where:**

- **deckname**
  Name of the deck to be ignored.
INSERT (OR I) DIRECTIVE

The INSERT directive inserts new text lines into the program library after a specified line. Lines to be inserted must immediately follow the INSERT directive.

Insertion of lines continues until another modification directive or an end-of-record is encountered. File manipulation directives within the insertion lines are processed and may change the source of insertion lines, but do not terminate insertion and are not inserted into the deck. Compile file directives within the insertion lines are treated as text lines and are not processed.

Format:

*INSERT c
  or
  *I c

Where:

c Specifies the line identifier of the line after which insertion lines are to be inserted.

MODNAME DIRECTIVE

The MODNAME directive changes the default modification set: name (modname) for line identifiers in subsequent modification directives. The modification set name specified by the MODNAME directive must be the name of a modification set previously defined for the program library.

The MODNAME directive does not affect line identifiers assigned to insertion lines. The default modification set name for insertion lines is defined by the IDENT directive.

The default modification set name established by MODNAME remains in effect until another DECK or MODNAME directive is processed. To return to the default name established by the last DECK directive, you can enter another MODNAME directive specifying the name of that deck.

Format:

*MODNAME modname

Where:

modname Defines the 1- to 7-character name of the modification set, modname can consist of alphanumeric characters and any of the following special characters: + - * / ( ) $ =
PURDECK DIRECTIVE

The PURDECK directive permanently removes a deck or group of decks from a program library. Every line of a deck is purged, regardless of the modification set it belongs to. You cannot rescind a PURDECK operation.

You can place a PURDECK directive anywhere in the directives file. It terminates any previous modification set; therefore, PURDECK cannot be followed by an INSERT, RESTORE, or DELETE directive until another IDENT directive is processed.

PURDECK has two formats. The first format defines a list of one or more decks to be purged. The second format defines the first and last decks of a series of consecutive decks to be purged.

Format one:

*PURDECK deckname1,deckname2,...,decknamen

Where:

decknamei specifies the name of a deck to be purged.

Format two:

*PURDECK deckname1,deckname2

Where:

deckname1 specifies the first deck and deckname2 specifies the last deck of a series of decks to be purged.
RESTORE DIRECTIVE

The RESTORE directive reactivates a line or block of lines previously deactivated by a DELETE or YANK directive. RESTORE also inserts text lines into the program library following the reactivated line or lines. Lines to be inserted must immediately follow the RESTORE directive.

Insertion of lines following a RESTORE operation continues until another modification directive is encountered or until EOR. File manipulation directives within the insertion lines are processed and may change the source of insertion lines, but do not terminate insertion and are not inserted into the deck. Compile file directives are treated as insertion lines and are not processed.

Format:

*RESTORE c1,c2

Where:

c1  Specifies a line identifier (refer to Line Identifiers in section 2). If only c1 is specified, it identifies a single line to be restored. If c1 and c2 are both specified, they identify the first and last lines of a range of lines to be restored. If any lines within the specified range of lines are already active, they are not affected by this directive.

UNYANK DIRECTIVE

The UNYANK directive rescinds a previous YANK directive. Like the YANK directive, UNYANK can specify either a single modification set or a series of modification sets beginning with the named set.

Format:

*UNYANK modname,*

Where:

modname  Specifies the name of the modification set to be reactivated.

*  Specifies that modname and all subsequent modification sets are to reactivated; applicable only to decks in which modname appears in the deck's modification table.
YANK DIRECTIVE

The YANK directive deactivates a modification set or sets. You can use this directive to deactivate a single modification set or a series of modification sets. When deactivating a series of sets (by specifying the * option), YANK deactivates all modification sets subsequent to the named set for all decks in which the named modification set is listed in the deck's modification table. (Refer to section 9 for more information on the modification table format.)

A YANK directive causes Modify to search the edited decks for all lines affected by the named modification set or sets. If a line was activated by the modification set, Modify deactivates it. If a line was deactivated by the modification set, Modify reactivates it.

Modify generates a new modification history byte for every line that changes status as a result of the YANK operation. The edited decks are effectively restored to the status they had prior to the original modification.

YANK affects only those decks that are edited by EDIT directives or by specification of the F or U parameter on the MODIFY command. This allows the use of the YANK directive on selected decks.

Format:

*YANK modname,*

Where:

modname Specifies the name of the modification set to be deactivated.

* Specifies that all modification sets subsequent to modname are also to be deactivated; applicable only to decks in which modname appears in the deck's modification table.

EXAMPLES OF MODIFICATION DIRECTIVES

Figure 5-1 contains a number of examples of modification directive usage.
Figure 5-1. Modification Directive Examples (Sheet 1 of 2)
Figure 5-1. Modification Directive Examples (Sheet 2 of 2)
FILE MANIPULATION DIRECTIVES

File manipulation directives provide control of file positioning and the use of alternate input files during the initialization and modification phases of a Modify run. File manipulation directives can be inserted in the directives file defined by the MODIFY command I parameter; they cannot be placed in alternate input file as specified by a READ or READPL directive. The file manipulation directives are as follows:

- BKSP
- READ
- READPL
- RETURN
- REWIND
- SKIP
- SKIPR

Note that the manipulation of the following files by these directives is restricted:

<table>
<thead>
<tr>
<th>INPUT</th>
<th>NPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT</td>
<td>SCR1</td>
</tr>
<tr>
<td>COMPILE</td>
<td>SCR2</td>
</tr>
<tr>
<td>SOURCE</td>
<td>SCR3</td>
</tr>
<tr>
<td>OPL</td>
<td></td>
</tr>
</tbody>
</table>

When specified on the MODIFY command, these file names are reserved and cannot be named in a file manipulation directive. For example, if the S parameter is specified, file name SOURCE is reserved and should not be used. File names SCR1, SCR2, and SCR3 are always reserved.
BKSP DIRECTIVE

The BKSP directive repositions the named file one or more logical records in the reverse direction. It does not backspace beyond the beginning-of-information.

Format:

*BKSP file,n

Where:

file Specifies the name of the file to be repositioned.

n Specifies the number of records to backspace. If n is omitted, a value of 1 is assumed.

READ DIRECTIVE

The READ directive instructs Modify to temporarily stop reading directives and insertion text from the primary directives file and to start reading from a specified alternate input file. The alternate file can be in either directives file format or in source file format. Modify reads the alternate file until an end-of-record or end-of-file is encountered (depending on the READ directive format specified), then returns to the original input file and begins reading the next directive following the READ directive.

The READ directive has three formats. The first format reads an entire file in directives file format. The second format reads a specified record (until end-of-record) in source file format. The third file reads an entire file (until end-of-file or a zero-byte record is encountered) in source file format. Note that the second and third formats of READ assume that the first line of each record is the record name. This line is discarded without processing.

Format 1:

*READ filename

Where:

filename Specifies the name of a file in directive file format.

Format 2:

*READ filename,deckname

Where:

filename Specifies the name of a file in source file format.

deckname Specifies the name of the deck to be read from file filename.
Format 3:
  *READ filename,*

Where:

filename    Specifies the name of a file in source file format.

*    Instructs Modify to read all records of file filename.

READPL DIRECTIVE

The READPL directive instructs Modify to temporarily stop reading the directives file and to start reading from a specified deck within the OPL. This directive allows you to copy text lines from one point in the OPL to another.

Note that READPL cannot read the new version of a deck that has been modified during the same Modify run. If you want to modify a deck and copy it to another part of the program library, these two operations must be done in separate Modify runs; otherwise, READPL copies the old version of the deck.

Format:
  *READPL deckname,c1,c2

Where:

    deckname    Specifies the name of the OPL deck from which text is to be read.
    c1,c2      Specifies the portion of deckname to be read. If omitted, the entire
deck is read. c1 and c2 specify the first and last lines of a range of lines to be read. If c1 is specified, c2 must also be specified (that is, at least two lines must be read).

RETURN DIRECTIVE

The RETURN directive returns the specified file or files to the system.

Format:
  *RETURN file1, file2, ..., filen

Where:

    filei    Specifies the name of a file to be returned.
REWIND DIRECTIVE

The REWIND directive repositions one or more files to beginning-of-information.

Format:

*REWIND file1, file2, ... , filen

Where:

file1 Specifies the name of a file to be rewound.

SKIP DIRECTIVE

The SKIP directive repositions a file forward a specified number of records. If an end-of-information is encountered before the specified number of records has been skipped, the file is positioned at end-of-information.

Format:

*SKIP filename, n

Where:

filename Specifies the name of the file to be repositioned.

n Specifies the number of records to be skipped. If omitted, a value of 1 is assumed.

SKIPR DIRECTIVE

The SKIPR directive repositions the named file forward past a specified record. It does not position the file past end-of-information. If Modify is unable to locate the specified record in the forward search, it positions the file at end-of-information and issues an error message.

Format:

*SKIPR filename, recname

Where:

filename Specifies the name of the file to be repositioned.

recname Specifies the name of the record after which the file is to be positioned.

EXAMPLES OF FILE MANIPULATION DIRECTIVES

Figure 6-1 shows several examples of the file manipulation directives.
Alternate directives file.


MODIFICATION COMPLETE.

Figure 6-1. File Manipulation Directives Examples (Sheet 1 of 2)
/catalog,newpl,r

CATALOG OF NEWPL

<table>
<thead>
<tr>
<th>REC</th>
<th>NAME</th>
<th>TYPE</th>
<th>LENGTH</th>
<th>CKSUM</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DECK1</td>
<td>OPL</td>
<td>46</td>
<td>1441</td>
<td>85/07/30</td>
</tr>
<tr>
<td></td>
<td>MOD1</td>
<td>MODX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>DECK2</td>
<td>OPL</td>
<td>65</td>
<td>7103</td>
<td>85/07/30</td>
</tr>
<tr>
<td></td>
<td>MOD1</td>
<td>MOD2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MOD3</td>
<td>MODX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>DECK3</td>
<td>OPL</td>
<td>44</td>
<td>0054</td>
<td>85/06/18</td>
</tr>
<tr>
<td></td>
<td>MOD1</td>
<td>MODX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>OPL</td>
<td>OPLD</td>
<td>7</td>
<td>7403</td>
<td>85/07/30</td>
</tr>
<tr>
<td>5</td>
<td>EOF</td>
<td></td>
<td>SUM</td>
<td>206</td>
<td></td>
</tr>
</tbody>
</table>

CATALOG COMPLETE.
/rewind,comx
REWIND,COMX.
/ftn5,i=cinx,l=0
0.040 CP SECONDS COMPILATION TIME.
/lgo
BEGIN MAIN PROGRAM.
LINE 3 ADDED BY MODIFICATION SET MOD1.
ENTER SUBROUTINE 1.
ENTER SUBROUTINE 2.
LINE 2 ADDED BY MODIFICATION SET MOD1.
EXIT SUBROUTINE 2.
EXIT SUBROUTINE 1.
LINE 1 ADDED BY MODIFICATION SET MOD1.
END MAIN PROGRAM.
STOP

0.016 CP SECONDS EXECUTION TIME.

Figure 6-1. File Manipulation Directives Examples (Sheet 2 of 2)
The compile file directives provide control over the formatting of the compile file, including the insertion of common decks into the compile file. You can embed compile file directives in the source deck when you originally create a program library, or you can insert compile file directives into a deck as part of a modification set. When included in the directives file, compile file directives are not recognized as directives during the modification phase of Modify processing; they are simply treated as text lines to be inserted.

The compile file directives are as follows:

- CALL
- CALLALL
- CALLC
- COMMENT
- CSET
- CWEOR
- ELSE
- ENDF
- IF
- IFDCALL
- NIFCALL
- NOSEQ
- SEQ
- SORSEQ
- WEOF
- WERO
- WIDTH
CALL DIRECTIVE

The CALL directive writes a copy of a common deck to the compile file. The CALL directive itself is not written to the compile file, but it is written to the new program library and source output files. Common deck calls may be nested; that is, a common deck may itself contain a call to another common deck. However, a common deck may not call itself.

Format:

*CALL, deckname

Where:

  deckname  Specifies the name of the common deck to be written to the compile file.

CALLALL DIRECTIVE

The CALLALL directive writes all decks to the compile file that have a deck name beginning with a specified character string. CALLALL cannot be used in a common deck. If encountered in a common deck, it is ignored.

Format:

*CALLALL string

Where:

  string  A 1- to 7-character string that specifies the initial characters of decks to be called.

Example:

The following call calls all common decks whose names begin with the characters CMD:

*CALLALL CMD
CALLC DIRECTIVE

The CALLC directive performs a conditional call to a common deck. The deck called by CALLC is written to the compile file only if that deck has not already been called from the deck being edited. Entry of a new DECK directive resets this condition. The use of CALLC for nested common deck calls inhibits the inclusion of duplicate copies of a common deck within the same compile file deck.

Format:

*CALLC deckname

Where:

deckname Specifies the name of the common deck to be written to the compile file.

COMMENT DIRECTIVE

The COMMENT directive creates a COMPASS language COMMENT pseudo instruction in the following format:

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>OPERATION</th>
<th>VARIABLE SUBFIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMENT</td>
<td>crdate</td>
<td>moddate</td>
</tr>
</tbody>
</table>

Where:

crdate       Creation date in the format yy/mm/dd.
moddate      Modification date in the format yy/mm/dd.
comments      Comment string up to 70 characters long.

The comment begins in column 3. Modify obtains the dates from the operating system.

Format:

*COMMENT comments  .

Where:

comments      Comment string up to 70 characters long.
CSET DIRECTIVE

The CSET directive overrides the code set designation for subsequently called common decks. When a CSET directive precedes a common deck call in the directive file, the common deck is expanded in the compile file in the code set specified by CSET, regardless of the code set defined for the common deck.

When in effect, the CSET directive also governs the code set for nested common deck calls. CSET remains in effect until the end of the deck containing CSET or until another CSET directive is encountered. A CSET directive encountered in a common deck is ignored and is written to the compile file as a comment.

Format:

* CSET charset

Where:

charset Can be either of the following:

ASCII Specifies expansion of common decks in 6/12-bit display code (uppercase and lowercase characters).

DISPLAY Specifies expansion of common decks in 6-bit display code (uppercase characters only).

CWEOR DIRECTIVE

The CWEOR directive conditionally writes an end-of-record (EOR) to the compile file. The EOR is written on the condition that information has been written since the last EOR. The CWEOR directive inhibits writing of successive EORs to the compile file.

Format:

* CWEOR

ELSE DIRECTIVE

The ELSE directive reverses the effect of the preceding IF directive. When the IF directive causes the following lines to be written to the compile file, ELSE terminates the insertion. If the lines following IF were skipped, ELSE terminates the skip and begins inserting the lines immediately following the ELSE directive. ENDIF terminates the insertion of lines initiated by ELSE.

Format:

* ELSE
ENDIF DIRECTIVE

The ENDIF directive marks the end of the conditional insertion lines controlled by an IF directive. If lines are being skipped as a result of the IF or ELSE directive, ENDIF terminates skipping and resumes normal processing with the line following ENDIF. If lines are not being skipped when ENDIF is encountered, ENDIF has no effect and processing continues with the next line.

Format:

*ENDIF

IF DIRECTIVE

The IF directive provides for the conditional insertion of lines into the compile file. The IF directive defines a true or false condition to be tested by Modify. When Modify processes an IF directive and finds the condition to be true, all text lines following the IF directive are written to the compile file until an ELSE or ENDIF directive terminates the insertion. If the condition is false, all lines following the IF directive are skipped until an ELSE or ENDIF directive terminates skipping and resumes normal processing.

Format:

*IF attr,tname,tvalue

Where:

attr      Defines the test attribute; must be one of the following:
          DEF       tname is defined by a DEFINE directive.
          UNDEF    tname is not defined by a DEFINE directive.
          EQ       tname equals tvalue.
          NE       tname does not equal tvalue.

tname     Specifies a 1- to 7-character name to be compared to names previously defined by a DEFINE directive.

tvalue    Specifies a numeric value to be compared to the value set by a DEFINE directive. This parameter is not valid when the DEF or UNDEF attribute is specified.
IFCALL DIRECTIVE

The IFCALL directive conditionally writes a common deck to the compile file. The common deck is called only if a specified name has been defined by a DEFINE directive during the modification phase.

Format:

*IFCALL cname,deckname

Where:

cname       Specifies the name to be checked for definition by a DEFINE directive.

deckname    Specifies the name of the common deck to be written to the compile file.

NIFCALL DIRECTIVE

The NIFCALL directive conditionally writes a common deck to the compile file. The common deck is called only if a specified name has not been defined by a DEFINE directive during the modification phase. The NIFCALL directive itself is not written to the compile file but is written to the new program library and source output files.

Format:

*NIFCALL cname,deckname

Where:

cname       Specifies the name to be checked for definition by a DEFINE directive.

deckname    Specifies the name of the common deck to be written to the compile file.

NOSEQ DIRECTIVE

The NOSEQ directive inhibits the inclusion of line sequencing information on the compile file.

Format:

*NOSEQ
SEQ DIRECTIVE

The SEQ directive instructs Modify to write line sequencing information to the compile file. SEQ is usually used to reverse the effect of a previous NOSEQ directive.

Format:

*SEQ

SORSEQ DIRECTIVE

The SORSEQ directive specifies that compile file line sequencing information is to be printed on the source output file (specified by the MODIFY command S parameter). The sequence information appears in columns 81 through 95 of the source output file.

Format:

*SORSEQ

WEOF DIRECTIVE

The WEOF directive writes an end-of-file (EOF) to the compile file.

Format:

*WEOF

WEOR DIRECTIVE

The WEOR directive writes an end-of-record (EOR) to the compile file.

Format:

*WEOR
WIDTH DIRECTIVE

The WIDTH directive sets the maximum line length for lines written to the compile file. The line length set by this directive refers to the length of the line up to, and excluding, the line sequencing information written to the compile file.

A WIDTH directive can be entered either during the initialization phase or during the compile phase. A WIDTH directive entered during the compile phase applies only for the deck containing the WIDTH directive. A WIDTH directive entered during the initialization phase is applicable for all decks except those containing a compile phase WIDTH directive. An initialization WIDTH directive is overridden by a WIDTH directive entered for a particular deck during the compile phase.

If no WIDTH directive is entered, Modify uses a default value of 72 columns for compile file output.

Format:

*WIDTH n

Where:

n Specifies the length of compile file lines excluding line sequencing information. The maximum value for n is 150 columns.

COMPILE FILE DIRECTIVE EXAMPLES

Figure 7-1 shows several examples of compile file directive usage.
Copy of source file to be incorporated into program library.

```
IDENT SUB3
ENTRY SUB3

*COMMENT CALL DECK DECKS
*** CALL COMMON DECK.
*IFDEF DECKS
** MYTEXT, DECKS =-
SUB3 DATA 0 ENTRY/EXIT ORIGIN JOT
EG SUB3 RETURN
JOT BSS 1 END

COPY COMPLETE.
"COPYR","csub"
DECKS COMMON ORIGIN MACRO A SA1 668 GET JOB ORIGIN
MAC 24
BX6 =X0*X1
AX6 24
SAB A STORE JOB ORIGIN
ENDM

COPY COMPLETE.
"MODIFY", "DECK =0", "n=mainpl", "c=com1", "s=mainp"
? *ofil file opl
? *rewind csub
? *create csub
? *ident mod4
? *deck deck1
? = 2
? common jot
? call sub3
? if(jot.eq.3)printe,'Interactive job.'
? if(jot.ne.3)printe,'Batch job.'
? *deck deck4
? = 0
? *wear
? *deck deck3
? = 0
? *wear
? *deck deck2
? = mod1.4
? = 0
? *wear

MODIFICATION COMPLETE.
"CATALOG", "mainpl",

<table>
<thead>
<tr>
<th>CATALOG OF MAINPL</th>
<th>TYPE</th>
<th>FILE</th>
<th>CKSUM</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECORD</td>
<td>NAME</td>
<td>LENGTH</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DECK1</td>
<td>60</td>
<td>4621</td>
<td>85/08/14</td>
</tr>
<tr>
<td></td>
<td>MOD1</td>
<td>MOD4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DECK3</td>
<td>37</td>
<td>4355</td>
<td>85/06/18</td>
</tr>
<tr>
<td></td>
<td>MOD1</td>
<td>MOD4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DECK2</td>
<td>61</td>
<td>7616</td>
<td>85/08/14</td>
</tr>
<tr>
<td></td>
<td>MOD1</td>
<td>MOD2</td>
<td>MOD3</td>
<td>MOD4</td>
</tr>
<tr>
<td></td>
<td>DECK4</td>
<td>50</td>
<td>1677</td>
<td>85/08/14</td>
</tr>
<tr>
<td></td>
<td>MOD4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DECKS</td>
<td>27</td>
<td>6354</td>
<td>85/08/14</td>
</tr>
<tr>
<td></td>
<td>OPL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DECKS</td>
<td>13</td>
<td>4046</td>
<td>85/08/14</td>
</tr>
<tr>
<td></td>
<td>OPLD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EOF</td>
<td>SUM</td>
<td>312</td>
<td></td>
</tr>
</tbody>
</table>

CATALOG COMPLETE.
```

Figure 7-1. Compile File Directives Examples (Sheet 1 of 3)
Figure 7-1. Compile File Directives Examples (Sheet 2 of 3)
/scopy,mainp
DECK1
*** MAIN PROGRAM. DECK DECK1.
  PROGRAM MAIN
  COMMON JOT
  PRINT*, 'BEGIN MAIN PROGRAM.'
  CALL SUB3
  IF(JOT.EQ.3)PRINT*, 'INTERACTIVE JOB.'
  IF(JOT.NE.3)PRINT*, 'BATCH JOB.'
  CALL SUB1
  PRINT*, 'END MAIN PROGRAM.'
  STOP
END
--EOR--
DECK3
DECK2
*** SUBROUTINE 2, DECK DECK3.
  SUBROUTINE SUB2
  PRINT*, 'ENTER SUBROUTINE 2.'
  PRINT*, 'EXIT SUBROUTINE 2.'
  RETURN
END
--EOR--
DECK2
*** SUBROUTINE 1, DECK DECK2.
  SUBROUTINE SUB1
  PRINT*, 'ENTER SUBROUTINE 1.'
  CALL SUBROUTINE SUB2
  IN DECK2.
  CALL SUB2
  PRINT*, 'EXIT SUBROUTINE 1.'
  RETURN
END
--EOR--
DECK4
**SUB3
IDENT SUB3
ENTRY SUB3
*COMMENT CALL DECK DECK3
*** PRINT ALL COMMON DECK.
*WIFCALL MYTEXT,DECK5
  SUB3
  DATA 0 ENTRY/EXIT ORIGIN JOT
  EQ SUB3 RETURN
  USE //
  JOT BSS 1
  END
--EOR--
DECK5
COMMON ORIGIN MACRO A
  SA1 660 SET JOB ORIGIN
  MX0 Z4
  BX6 -X0*X1
  AX6 Z4
  SA6 A STORE JOB ORIGIN
ENDM
--EOR--
--EOR--

Figure 7-1. Compile File Directives Examples (Sheet 3 of 3)
COMMMSG is an ASCII common deck called from common deck CPARMS.

CPARMS contains a set of parameters common to both procedures stored in deck PRCDECK. CPARMS also contains a call to deck COMMMSG. A conditional call (CALLC) is used to ensure that only one copy of the message is written to the compile file.

PRCDECK contains two procedures, TPROC1 and TPROC2. TPROC1 contains a nested procedure call to TPROC2.

Figure 7-2. NOS Procedure File With ASCII Deck (Sheet 1 of 3)
CATALOG OF NPL

<table>
<thead>
<tr>
<th>REC</th>
<th>NAME</th>
<th>TYPE</th>
<th>FILE LENGTH</th>
<th>CHKSUM</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>COMMBS</td>
<td>OPLC</td>
<td>67</td>
<td>2546</td>
<td>85/08/12</td>
</tr>
<tr>
<td>2</td>
<td>CPAPMS</td>
<td>OPLC</td>
<td>62</td>
<td>1316</td>
<td>85/08/12</td>
</tr>
<tr>
<td>3</td>
<td>PRCDEC</td>
<td>OPL</td>
<td>67</td>
<td>7332</td>
<td>85/08/12</td>
</tr>
<tr>
<td>4</td>
<td>OPL</td>
<td>OPLD</td>
<td>7</td>
<td>1714</td>
<td>85/08/12</td>
</tr>
</tbody>
</table>

CATALOG COMPLETE.

/COPY,compile
/PROC,TPROC1=
/P1"PARAMETER 1"=(*N=T01)
/P2"PARAMETER 2"=(*N=T02)
/P3"PARAMETER 3"=(*N=T03)

NOTE:/This common deck could be used to store an informative message regarding the NOTE:/common parameters in deck CPARMS. Because NOTE:/this deck is called by the CALLC directive, NOTE:/only one copy of the message will be NOTE:/included in the calling deck.

NOTE:/TPROC1 PARAMETERS = #P1=#P1,#P2=#P2,#P3=#P3./
BEGIN,TPROC2,TPROC3.
REVERT,NOLIST,TPROC1
EXIT.
REVERT,ABORT,TPROC1
PROC,TPROC2=
/P1"PARAMETER 1"=(*N=T01)
/P2"PARAMETER 2"=(*N=T02)
/P3"PARAMETER 3"=(*N=T03)

NOTE:/TPROC2 PARAMETERS = #P1=#P1,#P2=#P2,#P3=#P3./
REVERT,NOLIST,TPROC2
EXIT.
REVERT,ABORT,TPROC2

This common deck could be used to store an informative message regarding the common parameters in deck CPARMS. Because this deck is called by the CALLC directive, only one copy of the message will be included in the calling deck.

TPROC1 PARAMETERS = P1=T01,P2=T02,P3=T03.

TPROC2 PARAMETERS = P1=T01,P2=T02,P3=T03.

The compile file is renamed TPROC3 and is followed by a call to the procedure file. Output from execution of file TPROC3.

Figure 7-2. NOS Procedure File With ASCII Deck (Sheet 2 of 3)
/rename, onl=npl
RENAME, ONL=NPL.
/modify,f,c,n
? =ident cmod
? =deck prcdeck
? *i 2
? *cset display
? MODIFICATION COMPLETE.
/rename, tprocs=compile
RENAME, TPROCS=COMPILE.
/tprocs
THIS COMMON DECK COULD BE USED TO STORE
AN INFORMATIVE MESSAGE REGARDING THE
COMMON PARAMETERS IN DECK CPARMS. BECAUSE
THIS DECK IS CALLED BY THE CALLC DIRECTIVE,
ONLY ONE COPY OF THE MESSAGE WILL BE
INCLUDED IN THE CALLING DECK.

TPROC1 PARAMETERS - P1=T01, P2=T02, P3=T03.

TPROC2 PARAMETERS - P1=T01, P2=T02, P3=T03.

The CSET DISPLAY directive causes contents of COMMARG to
be printed in uppercase characters.

Inserts a CSET DISPLAY
directive as the third line
of deck PRCECK.

Figure 7-2. NOS Procedure File With ASCII Deck (Sheet 3 of 3)
The special directives control miscellaneous operating features of Modify. You can place special directives anywhere in the directives file.

The special directives are as follows:

- DEFINE
- INWIDTH
- MOVE
- PREFIX
- PREFIXC
- UPDATE
- /

**DEFINE DIRECTIVE**

The DEFINE directive defines a name and associated value to be used by conditional compiler directives, IF, IFCALL, and NIFCALL. The IF directive uses a defined name to test for a true or false condition. IFCALL calls a common deck if a specified name has been defined by a DEFINE directive. NIFCALL calls a common deck if a specified name has not been defined by a DEFINE directive.

Format:

```
*DEFINE dname, dvalue
```

Where:

- **dname** Defines a 1- to 7-character name consisting of alphanumeric characters and special characters: + - * / ( ) $ =

- **dvalue** Defines a numeric value assigned to dname. The maximum value for dvalue is 1777777. If dvalue is omitted, a value of 0 is assumed.
INWIDTH DIRECTIVE

The INWIDTH directive defines the maximum line length of Modify input files. Since Modify program libraries use a compressed format, Modify must know the maximum line lengths of files to be compressed. The INWIDTH directive can be placed anywhere in the directives file.

Format:

*INWIDTH n

Where:

n  Number of characters per line on input files. The maximum value that can be specified for n is 150 characters; 72 characters is the default line length.

MOVE DIRECTIVE

The MOVE directive repositions (reorders) one or more decks within a program library. For a deck to be repositioned by a MOVE directive, it must have been selected for editing by selection of the F or U parameter on the MODIFY command or by specification of an EDIT directive.

The MOVE directive repositions decks by placing them after a specified deck within the program library. The first parameter of the MOVE directive (deckname_r) defines the deck after which the repositioned decks are to be inserted. The remaining parameters define the decks to be moved and the order in which they are to be positioned.

Format:

*MOVE deckname_r,deckname_1,deckname_2,...,deckname_n

Where:

deqname_r  Specifies the deck following which the repositioned decks are to be placed.

deqname_i  Specifies the decks to be moved in the order they are to be reinserted following deckname_r.
PREFIX DIRECTIVE

The PREFIX directive changes the prefix character for subsequent Modify directives except compile file directives. (To change the prefix character for compile file directives, use PREFIXC.) The asterisk (*) is used for the prefix character if no PREFIX directive is entered.

Format:

*PREFIX c

Where:

c Defines the prefix character; c can be any graphic character except a blank.

PREFIXC DIRECTIVE

The PREFIXC directive changes the prefix character for compile file directives. The asterisk (*) is used for the prefix character if no PREFIXC directive is entered.

Format:

*PREFIXC c

Where:

c Defines the prefix character; c can be any graphic character except a blank.

UPDATE DIRECTIVE

The UPDATE directive affects the sequencing of lines in the compile file and the order in which Modify edits decks.

With respect to line sequencing, UPDATE instructs Modify to sequence lines continuously from one deck to the next in the edit sequence. Normally, Modify resets the line sequence number to 1 at the beginning of each deck.

With respect to the editing sequence, UPDATE instructs Modify to edit decks in the order they appear in the OPL. This directive overrides the sequence determined by the EDIT directives if the EDIT directives occur in some other order.

The UPDATE directive remains in effect until the end of the Modify run.

Format:

*UPDATE
/ DIRECTIVE

The / directive inserts a comment line into the Modify statistical list file; otherwise, this directive is ignored. The / directive can occur anywhere in the directives file.

Format:

/* comment

Where:

comment Defines a comment string of up to the maximum line length.

Example:

/* *****MODIFICATIONS*****

EXAMPLES OF SPECIAL DIRECTIVES

Figure 8-1 shows several examples of special directives usage. Note that compile file directives can be ignored (depending on the language processor) by changing the compile file directive prefix character.

/old,opl=mainpl
/modify,1,c=com1,n=mainpl,l=0
? *# change prefix character to #
? *prefix #
? #indent mod6
? #deck deck4
? #1 4
? #space 4
? #prefix #
? #move deck5,deck1,deck2,deck3,deck4
? #MODIFICATION COMPLETE.
/catalog,mainpl,r

<table>
<thead>
<tr>
<th>REC</th>
<th>NAME</th>
<th>TYPE</th>
<th>FILE</th>
<th>LENGTH</th>
<th>CKSUM</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DECK5</td>
<td>OPL64</td>
<td>27</td>
<td>6354</td>
<td>85/08/14.1000106</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>DECK1</td>
<td>OPL64</td>
<td>60</td>
<td>4621</td>
<td>85/08/14.1000106</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>DECK2</td>
<td>OPL64</td>
<td>61</td>
<td>7616</td>
<td>85/08/14.1000106</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>DECK3</td>
<td>OPL64</td>
<td>37</td>
<td>4355</td>
<td>85/06/18.1000106</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>DECK4</td>
<td>OPL64</td>
<td>54</td>
<td>3334</td>
<td>85/08/14.1000106</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>OPL</td>
<td>OPL0</td>
<td>13</td>
<td>1535</td>
<td>85/08/15.1000106</td>
<td></td>
</tr>
<tr>
<td>7 * EOF *</td>
<td>SUM</td>
<td></td>
<td></td>
<td></td>
<td>316</td>
<td></td>
</tr>
</tbody>
</table>

The common deck (DECK5) now comes before any deck that might call it.

Figure 8-1. Special Directive Examples (Sheet 1 of 3)
Figure 8-1. Special Directive Examples (Sheet 2 of 3)
EXAMPLE is not defined during this modification run. The "ELSE" path in module MOD7 will be taken.

MODIFICATION COMPLETE.

/*
 * com3, l=0, p=mainpl
 * edit deck1
 */

/*
 * copycf.com3
 */

+++ MAIN PROGRAM, DECK DECK1.

PROGRAM MAIN
COMMON JOT
PRINT*, "BEGIN MAIN PROGRAM."
CALL SUB1
PRINT*, "EXAMPLE HAS NOT BEEN DEFINED."
IF(JOT.EQ.3)PRINT*, "INTERACTIVE JOB."
IF(JOT.NE.3)PRINT*, "BATCH JOB."
CALL SUB1
PRINT*, "END MAIN PROGRAM."
STOP
END
*/

EOI ENCOUNTERED.

Figure 8-1. Special Directive Examples (Sheet 3 of 3)
Types of Modify files significant to Modify execution are as follows:

- Source files
- Program library files
- Directives file
- Compile file
- Statistical list file

SOURCE DECKS AND FILES

A source file is a collection of information either prepared by the user or generated by Modify (MODIFY command S option).

USER-PREPARED INPUT SOURCE FILES

You prepare a source deck for input to Modify by inserting the appropriate header lines at the beginning of the deck. The first line of a deck must always contain the deck name. If the deck is to be designated as a common deck or as an ASCII or DISPLAY code deck, this information is placed on the lines following the deck name. Section 2 gives detailed information on header line formats. You may also insert compile file directives into the source language deck to control compile file output from Modify. An end-of-record terminates each source deck. An end-of-file or end-of-information terminates a group of decks. The deck name, COMMON statements, and code set designator are not placed on the program library.

Modify source decks should not be confused with a compiler or assembler program. A Modify source deck can contain any number of FORTRAN programs, subroutines, or functions; COMPASS assembler IDENT statements; or sets of data. Typically, each Modify deck contains one program for the assembler or compiler or one set of data.
MODIFY-GENERATED SOURCE FILES

The source file generated as output by Modify contains a copy of all active lines within decks written on the compile file and new program library. The source file is optional output from Modify and is controlled through use of the S option on the MODIFY command. Once generated, the source file can be used as source input on a subsequent Modify run. The file is a coded file that contains 150-column images. Any sequencing information beyond the 150th column is truncated unless the SORSEQ directive has been entered. When the F parameter is selected on the MODIFY command, the source file contains all lines needed to recreate the latest copy of the program library.

When the U parameter is selected, the source file contains only those decks named on DECK directives; that is, only the decks updated during the current Modify run.

When neither F nor U is selected, the source file contains only those decks explicitly requested on EDIT directives.

PROGRAM LIBRARY FILES

Program library files (figure 9-1) provide the primary form of input to Modify. When a program library file is input, it is an old program library and has a default name of OPL. When it is output, it is a new program library and has a default name of NPL. During execution of Modify, the program library files must reside on disk.

Before writing the new program library, an EVICT is performed on the file. Refer to NOS Reference Set, volume 3, for a description of the EVICT command.

A program library consists of a record for each deck on the library. The last deck record is followed by a record containing the library directory. EDIT directives and the MODIFY command options determine the contents of the new program library. Only edited decks are written on the new program library.
Figure 9-1. Library File Format
DECK RECORDS

Each deck record consists of a prefix table, a modification table, and text.

Prefix Table Format

<table>
<thead>
<tr>
<th>ID</th>
<th>Bits</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>59-48</td>
<td>Table type</td>
<td>Identifies table as prefix table.</td>
</tr>
<tr>
<td></td>
<td>47-36</td>
<td>wc</td>
<td>Word count; length of table is 168 words.</td>
</tr>
<tr>
<td></td>
<td>35-00</td>
<td>none</td>
<td>Reserved for future system use.</td>
</tr>
<tr>
<td>1</td>
<td>59-18</td>
<td>deckname</td>
<td>Name of deck obtained for source deck identification line; 1 to 7 characters.</td>
</tr>
<tr>
<td></td>
<td>17-00</td>
<td>none</td>
<td>Reserved for future system use.</td>
</tr>
<tr>
<td>2</td>
<td>59-00</td>
<td>creation date</td>
<td>Date that deck was created.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Format of date is as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>yy/mm/dd</td>
</tr>
<tr>
<td>3</td>
<td>59-00</td>
<td>latest modification date</td>
<td>Date of most recent entry in modification table. Format of the date is the same as for creation date.</td>
</tr>
<tr>
<td>168</td>
<td>11-06</td>
<td>code set</td>
<td>Identifies code set used to create this deck.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>01</td>
<td>ASCII (6/12-bit display code)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00</td>
<td>DISPLAY (6-bit display code)</td>
</tr>
<tr>
<td>168</td>
<td>05-00</td>
<td>char set</td>
<td>Identifies character set used to create this deck.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00g</td>
<td>63-character set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64g</td>
<td>64-character set</td>
</tr>
</tbody>
</table>
### Modification Table Format

<table>
<thead>
<tr>
<th>ID Word</th>
<th>Bits</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>59-48</td>
<td>Table type</td>
<td>Identifies table as modification table. The least significant digit indicates whether the deck is common or not as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 Deck is not common.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 Deck is common.</td>
</tr>
<tr>
<td>47-12</td>
<td>none</td>
<td>Reserved for future system use.</td>
<td></td>
</tr>
<tr>
<td>11-00</td>
<td>$f$</td>
<td>Number of modification names in table.</td>
<td></td>
</tr>
<tr>
<td>word$_i$</td>
<td>59-18</td>
<td>modname$_i$</td>
<td>1- to 7-character modification set name. Each modification to a deck causes a new entry in this table.</td>
</tr>
<tr>
<td>16</td>
<td>$y_i$</td>
<td>YANK flag is as follows:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 Modifier is not yanked.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 Modifier is yanked.</td>
<td></td>
</tr>
</tbody>
</table>
Text Format

Text is an indefinite number of words that contain a modification history and the compressed image of each line in the deck. Text for each line is in the following format:

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>59</td>
<td>a</td>
<td>Activity bit is as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0   Line is inactive.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1   Line is active.</td>
</tr>
<tr>
<td>58-54</td>
<td>wc</td>
<td>Number of words of compressed text.</td>
</tr>
<tr>
<td>53-36</td>
<td>seq no</td>
<td>Sequence number (octal) of line according to position in deck or modification set.</td>
</tr>
<tr>
<td>35-18</td>
<td>mhb₄</td>
<td>Modification history byte. Modify creates a byte for each modification set that changes the status of the line. Modification history bytes continue to a zero byte. Since this zero byte could be the first byte of a word and the compressed line image begins a new word, the modification history portion of the text could terminate with a zero word. The format of mhb₄ is as follows:</td>
</tr>
<tr>
<td></td>
<td>and</td>
<td>Activity bit is as follows:</td>
</tr>
<tr>
<td></td>
<td>subsequent</td>
<td>0   Modification set deactivated the line.</td>
</tr>
<tr>
<td></td>
<td>18-bit</td>
<td>1   Modification set activated the line.</td>
</tr>
<tr>
<td></td>
<td>bytes</td>
<td>Index to the entry in the modification table that contains the name of the modification set that changes the line status. A modification number of zero indicates the deck name.</td>
</tr>
</tbody>
</table>
The compressed image of the line is display code. One or two spaces are each represented by 558; they are not compressed. Three or more embedded spaces are replaced in the image as follows:

3 spaces replaced by 0002
4 spaces replaced by 0003
... ... ...
64 spaces replaced by 0077g
65 spaces replaced by 007755g
66 spaces replaced by 0077355g
67 spaces replaced by 00770002g

Trailing spaces are not considered as embedded and are not included in the line image. On a 64-character set program library or compressed compile file, a 00 character (colon) is represented as a 0001 byte. A 12-bit zero byte marks the end of the line.

DIRECTORY RECORD

The library file directory contains a prefix table followed by a table containing a two-word entry for each deck in the library. Directory entries are in the same sequence as the decks on the library.

Prefix Table Format

A Modify-generated directory has the name OPL. However, if the name of the directory is changed (by LIBEDIT, for example), that name is retained on new program libraries then generated.
Directory Table Format

<table>
<thead>
<tr>
<th>ID Word</th>
<th>Bits</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>59-48</td>
<td>Table type</td>
<td>Identifies table as program library directory.</td>
</tr>
<tr>
<td></td>
<td>17-00</td>
<td>1</td>
<td>Directory length excluding ID word.</td>
</tr>
<tr>
<td>1, 3,</td>
<td>59-18</td>
<td>deckname1</td>
<td>Name of program library deck; 1 to 7 characters, left-justified.</td>
</tr>
<tr>
<td></td>
<td>17-00</td>
<td>type1</td>
<td>Type of record:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6 Old program library deck (OPL).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7 Old program library common deck (OPLC).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10 Old program library directory (OPLD).</td>
</tr>
</tbody>
</table>

**NOTE**

Other record types are defined for NOS but are ignored by Modify (refer to NOS Reference Set, volume 3, for a complete description of record types).

2, 4, 29-00 | random address1 | Address of deck relative to beginning of file. |
DIRECTIVES FILE

The directives file contains the Modify directives record. This record consists of initialization, file manipulation, and modification directives, and any source lines (including compile directives) to be inserted into the program library decks. An option of the MODIFY command designates the file from which Modify reads directives. Normally, the directive file is the job INPUT file. READ and READFL directives cause Modify to stop reading directives from the directives file named on the MODIFY command and to begin reading from some other file containing directives or insertion lines.

COMPILE FILE

The compile file is the primary form of output for Modify. It can be suppressed by the C=0 parameter of the MODIFY command if compile file output is not required.

If a compile file is specified on the MODIFY command, Modify writes the edited programs on it in a format acceptable as source input to an assembler, compiler, or other data processor. Through MODIFY command parameters and directives, you can specify whether the text on the file is to be compressed or expanded, sequenced or unsequenced. If the text is expanded, you can also specify the width of each line of text preceding the sequence information.

Expanded compile file format for each line consists of x columns of the expanded line (where x is the width requested), followed by 14 columns of sequence information, if sequencing information is requested, and terminated by a zero byte. An end-of-record terminates the decks written on the compile file.

COMPRESSED COMPILe FILE FORMAT

<table>
<thead>
<tr>
<th>ID Word</th>
<th>7700</th>
<th>0016</th>
<th>char set</th>
<th>17</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>modname₁</td>
<td>seq. no.₁</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>compressed line₁</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>modname₂</td>
<td>seq. no.₂</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>compressed line₂</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>char set</td>
<td>Character set of record. 0000g signifies 63-character set. 0064g signifies 64-character set.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>seq no₁</td>
<td>Sequence number of the line relative to the modification set identified by modname.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>compressed line</td>
<td>A line in compressed form. Refer to the compressed text description on the previous page for text formats of deck records.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
STATISTICAL LIST FILE

Depending on list options selected on the MODIFY command, statistical list output for Modify contains the following items:

- Input directives
- Status of each deck

Modifiers are listed first, followed by a list of activated lines, deactivated lines, active lines, and inactive lines as they are encountered. To the left of each line are two flags: a status flag and an activity flag. The status flag can be I (inactive) or A (active). The activity flag can be D (deleted) or A (activated). Following these lines are the unprocessed modifications and errors, if any. The last line contains a count of active lines, inactive lines, and inserted lines.

- Statistics

This includes lists of the following:
- Decks on program library
- Common decks on program library
- Decks added by initialization directives
- Decks on new program library
- Decks written on compile file

A replaced deck is enclosed by parentheses. Completing the statistics is a line containing counts of the number of lines on the compile file and the amount of storage used during the Modify run.

- Errors

Modify prints the line in error, if any, above the diagnostic message. Error messages other than those identified as fatal can be overridden through selection of the MODIFY command D (debug) option.

SCRATCH FILES

Modify uses scratch files in the following situations:

- Scratch File 1 (SCR1) Used when common decks are modified and no new program library is requested.
- Scratch File 2 (SCR2) Used when insertions overflow memory.
- Scratch File 3 (SCR3) Used when a CREATE or COPYPL directive is processed. This file is in program library format.

These files are returned by Modify at the end of the Modify run.
CREATE PROGRAM LIBRARY

EXAMPLE 1

This example illustrates how Modify can be used to construct a file in program library format from source decks. This example contains only one source deck (PROG), consisting of a FORTRAN 5 program. The deck is terminated by an end-of-file mark. The next record on the job file contains the directives. It is the job's responsibility to save the newly created program library (TAPE) for use in future Modify runs.

Unless C=0 is specified, a compile file is generated. This example shows the compile file (COMPILE) being used as input to the compiler. The compiler places the compiled program on file LGO. The LGO command calls for loading and execution of the compiled program.

MODJOB.
USER,USERNUM,PASSWRD,FAMILY.
COPYBF,INPUT,SOURCE.
MODIFY,P=1,N=TAPE,F.
FTN5,I=COMPILE.

: File-related commands.

LGO.
--EOR--
PROG

(SOURCE DECK)

--EOF--
*REWIND SOURCE
*CREATE SOURCE \hline Directives Input.

--EOI--
EXAMPLE 2

This example illustrates creation of a library from source decks on a source file other than the job file. After the library has been created, it can be modified, edited, and written on a compile file for use by an assembler or compiler.

Contents of File SALLY:
TOM
COMMON
  
  
  
(SOURCE DECK FOR TOM)
  
  
  
  --EOR--
JACK
COMMON
  
  
  
(SOURCE DECK FOR JACK)
  
  
  
  --EOR--
RON
  
  
(SOURCE DECK FOR RON)
  
  
  
*CALL TOM
*IF CALL REQ,JACK
--EOI--

Job Deck:

Job command

MODIFY,N,F,P=0.

File-related commands.

--EOR--
*REWIND SALLY
*CREATE SALLY

Directives Input.

*DEFINE REQ

--EOR--

10-2

60430100 G
MODIFY PROGRAM LIBRARY

EXAMPLE 1

In this example, Modify uses all default parameters. The sequencing information shown for inserted lines is assigned during modification.

```
MODIFY.

--EOR--
*IDENT MOD10
*DECK BOTTLE
*/ *****MODIFICATIONS
*D 10
*D 4
(TEXT LINE TO BE INSERTED IS ASSIGNED MOD10.1)
*D 20,22
(TEXT LINES TO BE INSERTED ARE ASSIGNED TO MOD10.2 THROUGH MOD10.4)
*I MOD9.30
(TEXT LINE TO BE INSERTED IS ASSIGNED MOD10.5)
*EDIT BOTTLE
--EOI--
```

Modification set MOD10.
EXAMPLE 2

This job modifies deck EDNA for replacement on the program library. No compile file is produced.

- MODIFY,N,C=0.  
  File-related commands.
- 
--EOR--
*IDENT  A2
*DECK  EDNA
*MODNAME  A1
*/ *****MODIFICATIONS
*D 30
  TAG  RJ  CHECK
*MODNAME  EDNA
  *I 7011
  ERR  SA1  LIST1
       ZR  X1,ABORT
       PRINT  (D*** ERROR 131 ***)
       EQ  ABORT
  
  *D 7644,7650  
  *EDIT  EDNA  
  --EOI--  

Modification set A2.
Delete line A1.30.
Insert line A2.1.
Delete lines A2.2 through A2.5 after EDNA.7011.
Delete lines EDNA.7644 through EDNA.7650.
EXAMPLE 1

The following job calls Modify twice. On the first call, Modify deactivates all but lines 32 through 54 and writes the source for these lines on source file FRANK. On the second call, Modify deletes the remainder of the lines and reinserts the saved lines at the beginning of KEN.

```
.*
.MODIFY,S=FRANK,C=0.
.MODIFY,N,C=CAL.

.--EOR--
*IDENT MOV1
*DECK KEN
*D 1,31
*D 55,63
*EDIT KEN
.--EOR--
*IDENT MOV2
*REWARD FRANK
*DECK KEN
*D 32,54
*I 0
*READ FRANK,KEN
*EDIT KEN
.--EOI--
```

File-related commands.

Modification set MOV1.

Delete lines before line KEN.32.
Delete lines KEN.55 through KEN.63.
Transfer remaining lines (KEN.32 through KEN.54) to source file FRANK.
Modification set MOV2.

Delete remainder of lines in KEN.
Insert lines at beginning of KEN.
Read insertion text from deck KEN on file FRANK.
EXAMPLE 2

This job moves text lines from one deck to another. On the first call to Modify, lines 32 through 54 of deck KEN on file OPL are saved on source file FRANK. On the second call, the saved lines are inserted into deck WILL.

```
* *
MODIFY,S=FRANK,C=O.
MODIFY,N,C=MEL.
* *
* EOR--
* IDENT F1
* DECK KEN
*D 1,31
*D 55,63
*EDIT KEN
* EOR--
*REWIND FRANK
*IDENT F2
*DECK WILL
*I 25
*READ FRANK,KEN
*EDIT WILL
* EOI--
```

File-related commands.

Modification set F1.

Delete lines KEN.1 through KEN.31.
Save lines KEN.32 through KEN.54 on source file FRANK.

Insert text after line WILL.25.

Insertion text taken from deck KEN on file FRANK.
Deck WILL is written on NFL and compile file MEL.
READ DIRECTIVES FROM AN ALTERNATE FILE

This job illustrates how the READ directive can be used to change the source of directives and correction text from the primary input file (in this case INPUT) to some other file.

... File-related commands...

MODIFY.
COMPASS, I=COMPILE.
LGO.
--EOR--
*IDENT JAN
*READ DIR
*DECK C

Read contents of DIR.

*DECK A
... Corrections for A.

*DECK B
... Corrections for B.

--EOR--

--EOI--

Return to INPUT file
YANK AND UNYANK MODIFICATION SETS

This example illustrates a job that deactivates modification set JULY and all subsequent modification sets. The change is not incorporated into the library; it is for the purposes of this run only.

```
*                   File-related commands.
* MODIFY,P=LIB,F.
COMPASS,I=COMPILE.
LGO.
--EOR--
*IDENT NEGATE
*DECK MASTER
*YANK JULY,*
--EOI--
```

To incorporate the preceding change on a new program library, add the N parameter to the MODIFY command.

The effects of a YANK can be nullified in future runs and, consequently, the effects of the yanked modification sets can be restored through the UNYANK directive. Such a modification might appear as follows:

```
*IDENT RESTORE
*DECK MASTER
*UNYANK JULY,*
```

PURGE DECKS

Decks BAD, WORSE, and WORST are no longer needed. The following job removes them from the library. They could also be removed through a selective edit using EDIT directives. In either case, the removal is permanent.

```
*                   File-related commands.
* MODIFY,N,C=Q,F.
*                   File-related commands.
--EOR--
*PURDECK BAD,WORSE,WORST
--EOI--
```
CHANGE THE DIRECTIVES PREFIX CHARACTER

EXAMPLE 1

This example illustrates how to maintain directives input on a library. Because * is the prefix used on the library, a different prefix is required when modifying the library. In this case, / becomes the prefix character.

```
ATTACH,OPL.
GET,FIX.
MODIFY,P=FIX,C=Z,N=FIX2.
REIND,Z.
COPYSBF,Z,OUTPUT.
REIND,Z.
MODIFY,I=Z.
COMPASS,I,S,B=LT01.
  
--EOR--
*PREFIX /  
/WIDTH 58  
/IDENT F1  
/DECK CORR  
/I 873  
*I 1007  
   LDC 7777B  
   STM STMA+1  
/D 880  
/EDIT CORR  
--EOI--
```

The contents of deck CORR on compile file Z are as follows:

```
*IDENT NIX       CORR 1  
*DECK GRM1TD     CORR 2  
*I MHD2.19       CORR 3  

*D 997,1000       CORR 873  
*I 1007  
   LDC 7777B       F1 1  
   STM STMA+1      F1 2  
} Inserted lines.

*LJM STM       CORR 879* Instruction CORR.880  

*D 980,984       CORR 881* has been deleted.
```

After file Z is produced, the deck GRM1TD is modified by the contents of Z. The resulting compile file (COMPILE) contains COMPASS language PP code and is assembled using COMPASS.
The job produces a new program library (FIX2) incorporating the changes made to deck CORR. The resulting COMPASS listing appears as follows:

<table>
<thead>
<tr>
<th>STD</th>
<th>SM</th>
<th>Corrections on File Z (Correction IDs)</th>
<th>Contents of COMPILE (Deck IDs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDC</td>
<td>7777B</td>
<td>F1</td>
<td>GRM1TD 1007</td>
</tr>
<tr>
<td>STM</td>
<td>STMA+1</td>
<td>F1</td>
<td>2 NIX 11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 NIX 12</td>
</tr>
</tbody>
</table>

Since the comments go through the correction identification, the INWIDTH directive must be deleted if a new program library is generated. However, for maintenance purposes, there is an advantage of seeing the correction identifiers with the deck identifiers.

**EXAMPLE 2**

This example illustrates changing the compile file prefix character so that when Modify produces the compile file, it recognizes only directives using the specified prefix. The directives prefix, in this case, is unaltered.

```
ATTACH,OPL.
MODIFY.
COMPASS,I,S,B.
--EOR--
*IDENT TEST1
*DECK TEST
*PREFIXC /
*EDIT TEST
--EOI--
```
Deck TEST contains the following:

```
  
  LDM  TILT
  STD  CM

  *CALL PCC
  /CALL PPCA
```

Modify ignores the common deck call to PPC. COMPASS interprets it as a comment. Modify acts on the common deck call to PPCA and replaces the /CALL directive with a copy of common deck PPCA.

**USE OF THE Z PARAMETER**

**EXAMPLE 1**

The following MODIFY command creates a compile file from an alternate OPL:

```
MODIFY,Z./*OPLFILE,OPLZ/*EDIT,DECK1
```

**EXAMPLE 2**

This MODIFY command write two decks to the compile file:

```
MODIFY,Z./*EDIT,DECK1,DECK2
```

**SAMPLE FORTRAN 5 PROGRAM**

This set of Modify examples illustrates how you can use Modify to maintain a FORTRAN 5 program in program library format. The FORTRAN 5 program calculates the area of a triangle from the base and height read from the data record.
EXAMPLE 1

The following job places the FORTRAN 5 program and subroutine as a single deck (ONE) on the new program library (NPL) and on the compile file (COMPILE). Following Modify execution, FORTRAN 5 is called to compile the program. The LGO command calls for execution of the compiled program. This program does not execute because of an error in the SUBROUTINE statement. The name of the subroutine should be MSG, not MSA.

```
COPYBF,INPUT,S.
MODIFY,P=0,N,F.
FTNS,I=COMPILE.
LGO.

--EOR--
ONE

PROGRAM ONE (TAPE1)
PRINT 5
5 FORMAT (1H1)
10 READ 100,BASE,HEIGHT,I
100 FORMAT(2F10.2, I1)
   IF (I.GT.0) GO TO 120
   IF (BASE.LE.0) GO TO 105
   IF (HEIGHT.LE.0) GO TO 105
   GO TO 106
105 CALL MSG
106 AREA = .5*BASE*HEIGHT
PRINT 110,BASE,HEIGHT,AREA
110 FORMAT(//,' BASE=',F20.5,' HEIGHT=',
       +F18.5,' AREA=',F20.5)
   WRITE(1) AREA
   GO TO 10
120 STOP
END
SUBROUTINE MSG
PRINT 400
400 FORMAT (//,'FOLLOWING INPUT DATA NEGATIVE OR ZERO')
RETURN
END
--EOF--
*REWIND S
*CREATE S
--EOR--
   200.24  500.76
   300.24  600.76
   400.00  700.00
   326.32  425.36
   500.00  600.00
   000.00  150.00
   700.43  800.00
   100.00  300.00
   050.00  100.00
   150.00  200.00

--EOI--
```

End of source deck.

Directives input.

Data record.
EXAMPLE 2

Examination of Modify output from the creation job reveals that the erroneous SUBROUTINE statement has line identifier ONE.20. The following job corrects the error and generates a new program library.

```
MODIFY,N,F.
FTNS,I=COMPILE.
LGO.
--EOR--
*IDENT MOD1
*DECK ONE
*DELETE 20
--EOR--
SUBROUTINE MSG  Identified as MOD1.1 on NPL.
    200.24   500.76
    300.24   600.76
    400.00   700.00
    326.32   425.36
    500.00   600.00
    000.00   150.00
    700.43   800.00
    100.00   300.00
    050.00   100.00
    150.00   200.00
--EOI--
```

Data record.

60430100 G

10-13
EXAMPLE 3

This job uses the same input as the first job but divides the program into two decks: ONE and MSG. DECK MSG is a common deck. A CALL MSG directive is inserted into deck ONE to ensure that MSG is written on the compile file whenever deck ONE is edited.

```
COPYBF, INPUT, S.
MODIFY, P=0, N, F.
FTN5, I=COMPILE.
LG0.
.
--EOR--

MSG
COMMON

SUBROUTINE MSG
PRINT 400
400 FORMAT (///,'FOLLOWING INPUT DATA NEGATIVE OR ZERO')
RETURN
END
--EOR--

ONE

PROGRAM ONE (TAPE1)
PRINT 5
5 FORMAT (1H1)
10 READ 100,BASE,HEIGHT,I
100 FORMAT (2F10.2, I1)
    IF (I.GT.0) GO TO 120
    IF (BASE.LE.0) GO TO 105
    IF (HEIGHT.LE.0) GO TO 105
    GO TO 106
105 CALL MSG
106 AREA= .5*BASE*HEIGHT
    PRINT 110,BASE,HEIGHT,AREA
110 FORMAT (///,' AREA=', F18.5)
    WRITE(1) AREA
    GO TO 10
120 STOP
END

*CALL MSG
--EOF--

*REWIND S
*CREATE S
--EOR--

200.24  500.76
300.24  600.76
400.00  700.00
326.32  425.36
500.00  600.00
000.00  150.00
700.43  800.00
100.00  300.00
050.00  100.00
150.00  200.00

--EOI--
```

10-14  60450100 G
EXAMPLE 4

This example adds a deck to the library created in the previous example. With no new program library generated (N is omitted from MODIFY command), the addition is temporary.

COPYBF, INPUT, S.
MODIFY.
FTN5, I=COMPILE.
LGO.

--EOR--
TWO

PROGRAM TWO

--EOF--

END

*CALL MSG

--EOF--

*REWIND S

*CREATE S

*IDENT MOD2

*DECK MSG

*DELETE MSG.3

400 FORMAT ('///,'FOLLOWING INPUT DATA POSITIVE')

*EDIT TWO

--EOR--

--EOR--

data record

--E01--
OPLEDIT Utility

OPLEDIT is a NOS utility used to manage modification set entries for decks in Modify-formatted old program libraries (OPLs). Using OPLEDIT, you can purge a modification set, reconstruct an earlier modification set, or consolidate a number of modification sets into a single set.

The OPLEDIT utility consists of the OPLEDIT command and the following directives:

- **CSET** Specifies the code set (ASCII or DISPLAY) used to create one or more specified Modify decks.
- **EDIT** Defines the decks to be edited and written to the new program library.
- **PULLALL** Consolidates two or more modification sets into a single modification set.
- **PULLMOD** Reconstructs the effects of one or more modification sets incorporated prior to the most recent modification set.
- **PURGE** Eliminates one or more modification sets from the OPL.

**OPLEDIT Command**

The OPLEDIT command initiates execution of the OPLEDIT utility.

Format:

```
OPLEDIT, P1, P2, ..., Pn
```

**Parameter (P1)**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies the directive input file for the OPLEDIT run.</td>
</tr>
</tbody>
</table>

**Option**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I or omitted</td>
</tr>
<tr>
<td>I=filename</td>
</tr>
<tr>
<td>I=0</td>
</tr>
<tr>
<td>Parameter (p₁)</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>P</td>
</tr>
<tr>
<td>P or omitted</td>
</tr>
<tr>
<td>P=filename</td>
</tr>
<tr>
<td>P=0</td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>N=filename</td>
</tr>
<tr>
<td>N=0 or omitted</td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td>L or omitted</td>
</tr>
<tr>
<td>L=filename</td>
</tr>
<tr>
<td>L=0</td>
</tr>
<tr>
<td>M</td>
</tr>
<tr>
<td>M or omitted</td>
</tr>
<tr>
<td>M=filename</td>
</tr>
<tr>
<td>Parameter (p₁)</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>LO</td>
</tr>
</tbody>
</table>

**Option**  **Description**

LO or omitted  List option E is selected if the list output file is assigned to an interactive terminal. Otherwise, options C, D, E, M, and S are selected.

List Option  Description

| C | List input directives |
| D | List deck status |
| E | List errors |
| M | List modifications |
| S | List directory |

LO=c₁c₂...c₅  c₁ selects one of the above options. Up to five options can be listed. Options must not be separated.

F  Specifies that all decks on the OPL are to be written to the selected output files.

D  Specifies debug mode; errors are listed on the output file but do not abort the OPLEDIT run.

U  Instructs OPLEDIT to automatically insert a Modify EDIT directive into all decks included in a consolidated modification set generated by PULLALL or a reconstructed modification set generated by PULLMOD. This parameter ensures that all decks associated with a PULLALL- or PULLMOD-generated modification set an edited during a subsequent Modify run (that is, if the OPLEDIT new program library is used as input to Modify).

**Option**  **Description**

| U | Generate EDIT directives for all decks. |
| omitted | Generate EDIT directives for common decks only. |
| U=0 | Generate no EDIT directives. |

Z  Specifies that OPLEDIT directives are included on the OPLEDIT command following the command terminator. If Z is specified, the I parameter is invalid.

Use of the Z parameter eliminates the need to use a separate input file when only a few directives are needed. The first character following the command terminator is the separator character for all directives on the command. Any display code character (including the space) not used in any of the directives can be used as the separator character.
CSET DIRECTIVE

The CSET directive determines the code set bit setting (bit 2**6 of word 0+16B) in the modification table of all subsequently edited decks. (Refer to section 9 for more information on the modification table format.) CSET works in conjunction with the EDIT directive; only decks specified on EDIT directives following the CSET directive are affected.

The purpose of CSET is to facilitate conversion of program library decks created prior to the addition of the ASCII-DISPLAY code option of Modify. The code set bit of the modification table was not used by previous versions of Modify. The only recommended use of CSET is to set or clear this bit to reflect the code set actually used in the specified deck(s).

Format:

```
CSET,codeset
```

Where:

codeset Specifies the code set used to create the decks to be converted.

Options that can be specified for codeset are as follows:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII</td>
<td>Specifies 6/12-bit DISPLAY code. This option sets bit 2**6 of word 0+16B of the modification table for affected decks.</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>Specifies 6-bit DISPLAY code. This option clears bit 2**6 of word 0+16B of the modification table for affected decks.</td>
</tr>
</tbody>
</table>

EDIT DIRECTIVE

The EDIT directive instructs OPLEDIT to edit a program library deck and transfer it to the new program library. The deck names specified normally are the decks that contain the modification identifiers.

Format:

```
*EDIT p1,p2,...,pn
```

Where:

```
p1 Specifies a deck or range of decks to be edited. If a range of decks is specified, p1 must be in the following format:
```

dekname_a..dekkname_b

```
dekname_a and dekkname_b are the first and last decks, respectively, of the range of decks specified.
```
PULLALL DIRECTIVE

The PULLALL directive consolidates two or more modification sets into a single modification set. PULLALL has two formats, as shown below. Format 1 consolidates all active modification sets into a single set. Format 2 creates a consolidated modification set from all modification sets subsequent to, and including, a specified set.

PULLALL produces two types of output files, a new program library and a directives file. Both of these files include only edited decks; that is, decks selected for editing by an EDIT directive or by the OPLEDIT command F parameter. All decks associated with the consolidated modification set are written to the new program library specified by the OPLEDIT command N parameter. You can use this file as the OPL for a subsequent Modify run.

The directives file is written to the file defined by the OPLEDIT M parameter. This file contains all the directives and insertion lines necessary to produce the net effect of the consolidated modification set. The IDENT directives that defined the original modification sets are deleted from this file and are replaced by a single IDENT ***** directive at the beginning of the file.

The directives file is written in a format (that is, with the name of the record appearing as the first line of the file) suitable to be read by the Modify READ directive in either of the following formats:

*READ, file,*
or
*READ, file, deckname

Format 1:
*PULLALL

Format 2:
*PULLALL modname

Where:

modname Specifies the first (earliest) modification set to be included in the consolidated modification set.
PULLMOD DIRECTIVE

The PULLMOD directive reconstructs one or more modification sets previously incorporated in a program library. Unlike the PULLALL directive, the structure of the original modification sets is not changed; the original IDENT directives are neither changed nor deleted.

PULLMOD produces two types of output files, a new program library and a directives file. Both of these files include only edited decks; that is, decks selected for editing by an EDIT directive or by the OPEDIT command F parameter. All decks associated with the reconstructed modification set are written to the new program library specified by the OPEDIT command N parameter. You can use this file as the OPL for a subsequent Modify run.

The directives file is written to the file defined by the OPEDIT M parameter. This file contains all the directives and insertion lines necessary to produce the net effect of each reconstructed modification set. Each reconstructed set is written to a separate record on the file.

The directives file is written in a format (that is, with the name of the record appearing as the first line of each record) suitable to be read by the Modify READ directive in either of the following formats:

*READ,file,*
or
READ, file, deckname

Format:

*PULLMOD modname1, modname2, ..., modnamen

Where:

modnamei Specifications the name of a modification set to be reconstructed.

PURGE DIRECTIVE

The PURGE directive removes the effects of one or more previously incorporated modification sets from edited decks in the program library. All information pertaining to purged modification sets is removed from the modification table of affected decks (refer to section 9 for a description of the modification table).

Format:

*PURGE modname,*

Where:

modname Specifies the name of the modification set to be purged if the * parameter is not included. If * is included, modname specifies the first of a series of modification sets to be purged.

* Specifies that modname and all modification sets subsequent to modname should be purged.
OPLEDIT EXAMPLES

Figure A-1 illustrates the use of the OPLEDIT directives.

```
/get,mainpl
/catalog,mainpl,r

CATALOG OF MAINPL

<table>
<thead>
<tr>
<th>REC</th>
<th>NAME</th>
<th>TYPE</th>
<th>LENGTH</th>
<th>CKSUM</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DECK5</td>
<td>OPL</td>
<td>27</td>
<td>6354</td>
<td>85/08/14</td>
</tr>
<tr>
<td>2</td>
<td>DECK1</td>
<td>OPL</td>
<td>60</td>
<td>4621</td>
<td>85/08/14</td>
</tr>
<tr>
<td></td>
<td>MOD1</td>
<td>OPL</td>
<td>2</td>
<td>61</td>
<td>85/08/14</td>
</tr>
<tr>
<td></td>
<td>MOD4</td>
<td>OPL</td>
<td>61</td>
<td>7616</td>
<td>85/08/14</td>
</tr>
<tr>
<td>4</td>
<td>DECK3</td>
<td>OPL</td>
<td>37</td>
<td>4355</td>
<td>85/06/18</td>
</tr>
<tr>
<td></td>
<td>MOD1</td>
<td>OPL</td>
<td>37</td>
<td>4355</td>
<td>85/06/18</td>
</tr>
<tr>
<td>5</td>
<td>DECK4</td>
<td>OPL</td>
<td>54</td>
<td>3334</td>
<td>85/08/14</td>
</tr>
<tr>
<td></td>
<td>MOD4</td>
<td>OPL</td>
<td>54</td>
<td>3334</td>
<td>85/08/14</td>
</tr>
<tr>
<td>6</td>
<td>OPL</td>
<td>OPLD</td>
<td>13</td>
<td>1535</td>
<td>85/08/15</td>
</tr>
<tr>
<td>7</td>
<td>* EOF *</td>
<td></td>
<td>SUM = 316</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CATALOG COMPLETE.
```

```
/opledit,p=mainpl,m=mods,n=newpl
? *purge mod4,*
? *pullmod mod2,mod3
? *pullall mod1
? *edit deck1,deck4
?
OPLEDIT COMPLETE.
```

```
/catalog,newpl,r

CATALOG OF NEWPL

<table>
<thead>
<tr>
<th>REC</th>
<th>NAME</th>
<th>TYPE</th>
<th>LENGTH</th>
<th>CKSUM</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DECK1</td>
<td>OPL</td>
<td>35</td>
<td>1221</td>
<td>85/08/14</td>
</tr>
<tr>
<td>2</td>
<td>DECK2</td>
<td>OPL</td>
<td>55</td>
<td>3330</td>
<td>85/08/14</td>
</tr>
<tr>
<td></td>
<td>MOD1</td>
<td>OPL</td>
<td>55</td>
<td>3330</td>
<td>85/08/14</td>
</tr>
<tr>
<td></td>
<td>MOD2</td>
<td>OPL</td>
<td>55</td>
<td>3330</td>
<td>85/08/14</td>
</tr>
<tr>
<td>3</td>
<td>DECK3</td>
<td>OPL</td>
<td>34</td>
<td>1075</td>
<td>85/06/18</td>
</tr>
<tr>
<td></td>
<td>MOD1</td>
<td>OPL</td>
<td>34</td>
<td>1075</td>
<td>85/06/18</td>
</tr>
<tr>
<td>4</td>
<td>DECK4</td>
<td>OPL</td>
<td>45</td>
<td>4105</td>
<td>85/08/14</td>
</tr>
<tr>
<td>5</td>
<td>OPL</td>
<td>OPLD</td>
<td>11</td>
<td>2101</td>
<td>85/08/16</td>
</tr>
<tr>
<td>6</td>
<td>* EOF *</td>
<td></td>
<td>SUM = 223</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CATALOG COMPLETE.
```

Figure A-1. OPLEDIT Examples (Sheet 1 of 2)
Results of the PULLALL directive.

Results of the PULLMOD directive.

These numbers indicate the location of directive affecting a modset. They are the last active sequence number in the deck from which the directive was copied (refer to figure 5-1).
This appendix contains an alphabetical listing of the messages that may appear in your dayfile. Lowercase characters are used to identify variable names or fields. All messages are sorted according to the first nonvariable word or character. Messages beginning with special characters (such as hyphens or asterisks) are sorted as if the special characters were not present.

If you encounter a diagnostic or informative message that does not appear in this appendix, consult the NOS 2 Diagnostic Index. This publication catalogs all messages produced by NOS and its products and specifies the manual or manuals in which each message is fully documented.
MESSAGE | SIGNIFICANCE | ACTION | ROUTINE
--- | --- | --- | ---
COLUMN OUT OF RANGE. | Requested width exceeds maximum allowed (150). | Change width to 150 or less. | MODIFY
COPY FILE EMPTY. | No information on program library being copied. | Verify that copy file exists and is properly positioned at 001. | MODIFY
CREATION FILE EMPTY. | No source decks on file being used for creation. | Verify that creation file contains proper source decks. | MODIFY
CSET - UNKNOWN CHARACTER SET. | The character code set on the CSET directive is missing or an unknown code set was specified. | Specify either ASCII or DISPLAY code. | MODIFY
CV OPTION INVALID. | A CV option other than 63 or 64 was specified. | Specify 63 or 64 for conversion option. | MODIFY
DIRECTIVE ERRORS. | Dayfile message indicating that one or more input directives were in error. | Examine output file to determine reason for error. | MODIFY
DIRECTIVE NOT REACHED. | Sequence number exceeds deck range. | Use correct sequence number. | MODIFY
DUPLICATE MODIFIER NAME. | Modifier on IDENT has been used previously for the deck. | Choose unique name for deck. | MODIFY
ERROR IN ARGUMENTS. | An invalid parameter has been encountered on the OPLEDIT command. | Correct command and retry. | OPLEDIT
ERROR IN DIRECTORY. | The program library contains an error. | Use COPY or COPYPL to create new program library. | MODIFY
ERROR IN MODIFY ARGUMENTS. | Invalid parameter on MODIFY command. | Consult manual for correct command syntax. | MODIFY
FILE NAME CONFLICT. | The same file cannot be used for both applications without conflict. | Use different file name for one of the applications. | MODIFY
FILE NAME CONFLICT. | The same file cannot be used for both applications without conflict. | Use different file name for one of the applications. | MODIFY
FIRST SOURCE LINE IS AFTER SECOND SOURCE LINE. | Either an error in parameters or lines are out of order. | Verify that correct line sequence is used. | MODIFY
FORMAT ERROR IN DIRECTIVE. | This message is returned as a result of a directive format error, a format error in a source input file, or the inclusion of an ignored deck in an EDIT directive specifying a series of deck names. | Consult manual for correct format. | OPLEDIT
IDENT NAME PREVIOUSLY REFERENCED. | A modification directive or a different IDENT directive refer to the current modname. | Choose a different modification name for the IDENT directive. | MODIFY
INCORRECT ATTRIBUTE. | Attribute specified on IF directive is other than EQ, NE, LE, SE, or UNDEF. | Use correct attribute. | MODIFY
INCORRECT CS ON INPUT. | The input data uses the 64-character set, but the PL uses the 65-character set. | Convert either the input data or the PL so both use the same character set. | MODIFY
deckname - INCORRECT CS, 63 ASSUMED. | The lower byte of word 168 of the prefix table for the named deck on the program library does not contain 0000 or 0064. | If 64-character set is desired, the deck must be recreated. | MODIFY
INCORRECT DIRECTIVE. | Directive is out of sequence. An example would be a CREATE directive after a modification directive for Modify. | Use correct sequence. | OPLEDIT
INCORRECT NUMERIC FIELD. | Incorrect parameter on MODIFY or OPLEDIT command. | Verify parameters and retry. | MODIFY
deckname - INCORRECTLY NESTED CALL OR COMMON DECK. | A redundant nested call was found. A CALL, MDEPEND, or IFCALL calls a common deck which has already been called in the current nesting sequence. | Remove the redundant call. | MODIFY
INITIALIZATION DIRECTIVE OUT OF ORDER. | A non-initialization directive was encountered before the initialization directive. | Consult manual for correct directive order. | MODIFY
MESSAGE | SIGNIFICANCE | ACTION | ROUTINE
--- | --- | --- | ---
LINE NOT FOUND. | An EOR was reached in the program library before finding the specified command. | Verify the command exists and remove the extra EORs from the source file. | MODIFY
-LO-ERROR, MUST BE IN -ECTWSIA- | Incorrect list option requested. Fatal error. | Specify $ S, T, M, W, D, B, J, I, or A or a combination of these characters for list option. The characters must not be separated. | MODIFY
MEMORY OVERFLOW. | Insufficient field length has been specified for OPLEDIT to execute. | Increase field length with RFL command and retry. | MODIFY
deckname - MIXED CHARACTER SET DETECTED. | Upon editing the named deck on the program library, the character set (63- or 64-character) was different from the character set of previously edited decks. | Recreate the deck under the desired character set. | MODIFY
MIXED CHARACTER SET OPL. | OPLEDIT detected decks on the program library that are in different character sets (63- and 64-characters). | Use MODIFY to recreate erroneous decks under one character set and retry. | MODIFY
MODIFICATION COMPLETE. | Modify has completed execution of the directives. | None. | MODIFY
MODIFICATION/DIRECTIVE ERRORS. | Modification and/or directive errors are encountered when debug mode is selected. | Consult listing and correct specified errors. | MODIFY
MODIFICATION ERRORS. | Modify has detected errors during the modification phase; fatal if $I option is not selected. | Consult listing and correct specified errors. | MODIFY
NAMES SEPARATED BY $ $ IN WRONG ORDER. | Requested decks not in correct sequence. | Determine correct sequence and retry. | MODIFY
NO DIRECTIVES. | Directives file empty. | Verify that directives file exists and is correctly positioned at BOI. | OPLEDIT
NO *IF IN PROGRESS. | An ELSE or ENDF directive was encountered without a previous IF directive. | Check for omitted IF directive or unnecessary ELSE or ENDF directive. | MODIFY
OPERATION INVALID FROM ALTERNATE INPUT. | File manipulation attempted from other than original directives file. | Move file manipulation directives to original directives file. | MODIFY
OPLEDIT COMPLETE. | Informative message indicating that OPLEDIT has completed processing. | None. | MODIFY
OPLEDIT ERRORS. | Errors were encountered during OPLEDIT execution. | Consult output listing for description of errors. | OPLEDIT
OVERLAPPING MODIFICATION. | More than one modification was attempted for a single line. | Remove redundant line modification. | MODIFY
PL ERROR IN DECK deckname. | An error was detected in the program library format during processing of deck deckname. | Replace or re-create erroneous deck. | OPLEDIT
PROGRAM LIBRARY EMPTY. | No information on file specified as program library. | Verify that program library file exists and is positioned at BOI. | OPLEDIT
RECORD NOT FOUND. | Modify was unable to locate requested record on file specified. | Verify that record exists on specified file. | MODIFY
RECURSIVE *IF,$ IS ILLEGAL. | An IF directive was encountered while a previous IF range was still active (no ELSE or ENDIF encountered). | Check for missing ENDIF or ELSE directive or unnecessary IF directive. | MODIFY
REDUNDANT CONVERSION IGNORED. | An attempt was made to convert the program library file to a like character set (63 to 63 or 64 to 64). Conversion option set to zero. | Verify conversion mode desired. | MODIFY
RESERVED FILE NAME. | A reserved filename was incorrectly used. | Choose a non-reserved file name. | OPLEDIT EDIT DATABASE IAFEX
<table>
<thead>
<tr>
<th>MESSAGE</th>
<th>SIGNIFICANCE</th>
<th>ACTION</th>
<th>ROUTINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>S OPTION ILLEGAL WITH A, X, OR Q.</td>
<td>Source option not legal when A, X, or Q option is selected.</td>
<td>Remove S option from MODIFY command and specify on separate Modify run.</td>
<td>MODIFY</td>
</tr>
<tr>
<td>TOO MANY GPL FILES.</td>
<td>More than 50 program library files declared.</td>
<td>Specify excess program libraries on subsequent Modify runs.</td>
<td>MODIFY</td>
</tr>
<tr>
<td>UNKNOWN DECK.</td>
<td>Unable to locate requested deck on program library.</td>
<td>Verify that deck name is correct.</td>
<td>MODIFY</td>
</tr>
<tr>
<td>UNKNOWN MODIFIER</td>
<td>Modifier not in modification table for deck.</td>
<td>Determine correct modifier.</td>
<td>MODIFY</td>
</tr>
<tr>
<td>VALUE ERROR.</td>
<td>Value specified on IF or DEFINE directive is greater than 17777776.</td>
<td>Select value less than or equal to 17777776.</td>
<td>MODIFY</td>
</tr>
<tr>
<td>X OR Q INCORRECT WITHOUT COMPILE.</td>
<td>Selection of X or Q option requires that a compile file name be selected.</td>
<td>Specify C option on Modify command (not C=0).</td>
<td>MODIFY</td>
</tr>
</tbody>
</table>
INDEX

ASCII code 1-1; 2-4,13; 7-4; 9-1; A-4
Assembler call 1-5; 3-1
Q parameter 3-3
X parameter 3-6
BKSP directive 6-2
CALL directive 2-13; 7-2
CALLALL directive 2-13; 7-2
CALLG directive 2-13; 7-3
Character set conversion 3-2
Code set 2-4,13; 7-4; 9-1; A-4
COMMON directive 7-3
Common decks
Calling 1-4; 2-13; 7-2
Code set 2-4,13; 7-4; 9-1; A-4
Conditional call 2-13; 7-3,6
Definition 1-4
Format 9-1
Nested calls 7-2
Placement in PL 2-10
(See also Decks)
COMMON header line 2-4; 9-1
COMPASS 9-1
Listing file 3-2
Compile file
C parameter 3-1
Compressed 3-1; 8-2
Conditional insertion of lines 7-5
Definition 1-5
Formatted 3-1
Compile file directives 2-1,2,13; 7-1,2; 9-1
Compile phase 2-3,11
Compiler call 1-5; 3-1
Q parameter 3-3
X parameter 3-6
Compressed files 8-2; 9-7
COPY directive 2-3; 4-1
COPYP directive 2-3; 4-2
CREATES directive 2-3; 4-3
CSET directive (Modify) 2-13; 7-4
CSET directive (OPEDIT) A-4
CWDIR directive 7-4
Debugging 3-3; 9-10
Deck
Copying 6-3
Definition 1-3
Moving 8-2
Name conventions 2-4; 5-2
Purging 5-6; 10-8
(See also Common decks)
Status 9-10
DECK directive 2-6,8; 5-2
DEFINE directive 2-12; 8-1
DELETE directive 2-7,9; 5-2,6
Diagnostic messages 9-10; B-1
/ directive 8-4
Directive format 2-2
Directives file
Alternate 2-11; 6-2; 9-9; 10-7
Definition 1-4
Directive sequence 2-2
I parameter 1-4; 3-3
Z parameter 1-4; 3-6; 10-11
Directory, library file 2-3; 4-1,2; 9-7,8
DISPLAY code 1-1; 2-4,13; 7-4; 9-1; A-4
EDIT directive 2-5,10; 5-3
Editing decks 2-10; 3-3,6; 5-3; 8-3
ELSE directive 2-12; 7-4
End-of-file 7-7
End-of-record 7-4,7
ENDIF directive 2-12; 7-5
Error messages 9-10; B-1
EVENT command 9-2
Expansion, compile file 2-12
File formats 9-1
File Manipulation Directives 2-1,2,11,15; 6-1
Files, reserved 6-1
FORTRAN 9-1
Full edit 2-10
IDENT directive 2-7; 5-3,6; A-6
IF directive 2-12; 7-5; 8-1
IFCALL directive 2-12; 7-6; 8-1
IGNORE directive 5-4
Initialisation directives 2-1,2,14; 4-1
Initialisation phase 2-3
INSERT directive 2-7,9; 5-5,6
Insertion lines 11-11
INWIDTH directive 2-4; 8-2; 10-10
Line
Activation 2-9
Deactivation 2-9
Modification 2-8
Line identifier 2-4,6; 5-2,5
Line length, maximum
Compile file 7-8
Input file 2-4; 8-2
Line sequencing 1-5; 2-4,6,8; 7-7; 8-3; 9-9
Modification directives 2-1,2,14; 5-1
Modification history byte 5-4,8; 9-6
Modification phase 2-3,5
Modification set 2-5
Consolidation A-3
Creation 5-3
Deactivation 5-8
Dock modifications 5-2
Definition 2-7; 5-3
Management through OPEDIT A-1
Name conventions 5-3
Purging A-6
Reactivation 5-7
Reconstruction A-6
Modification table format 9-5
MODIFY command
Format 3-1
Input files 4-1
Interactive entry 3-1
MODNAME directive 2-6,8,9; 5-5
MOVE directive 2-10; 8-2
New program library
Definition 1-5; 9-2
File 1-5
N parameter 3-4
(See also Program Library)
NIFCALL directive 2-12; 7-6; 8-1
NOS procedures 2-4
NOSSEQ directive 2-2,4; 7-6
NPL (See New program library)
Old program library
Copying 2-3
Definition 1-3; 2-3; 9-2
P parameter 2-3; 3-5
(See also Program library)
OPE (See Old program library)
OPLEDIT command A-1
OPLEDIT utility A-1
OPFILE directive 2-3; 4-3
PD symbolic name 3-1
PL symbolic name 3-1
Prefix character 2-2; 8-3; 10-9
PREFIX directive 2-2; 8-3; 10-9
Prefix table format 9-4,7
PREFIXC directive 8-3; 10-10
Procedures (See NOS procedures)
Program Library
Characteristics 1-3
Copying 4-1,2
Creation 2-3; 4-3; 10-1,2
Definition 1-3
File format 9-2
PULLALL directive A-5
PULLMOD directive A-6
PURDECK directive 5-3; 10-8
PURGE directive A-8
READ directive 2-11; 6-2; 9-9; 10-7; A-5,6
READPL directive 2-11; 6-3; 9-9
Reserved files 6-1
RESTORE directive 2-7,9; 5-6,7
RETURN directive 6-3
REWIND directive 6-4
Scratch files 9-10
Selective edit 2-10
SEQ directive 2-2,4; 7-7
SET command 3-1
SKIP directive 6-4
SKIPR directive 6-4
SORSEQ directive 2-2,4; 7-7
Source input file
Conversion to PL format 2-3; 4-3
Definition 1-3; 2-3
Format 2-4; 9-1
Source output file
Format 9-2
$ parameter 3-6
Special directives 2-1,2,16; 8-1
Statistical list file
Comment lines 9-4
Definition 1-5
Format 9-10
L parameter 3-4
List options 3-4
Systems text
Loading 3-2
Overlay 3-2
SYSTXT 3-2
Tape files, copying 2-3; 4-1
UNYANK directive 2-7; 5-7; 10-8
UPDATE directive 2-6,10; 9-3
Update edit 2-10
WEOP directive 7-7
WEDR directive 7-7
WIDTH directive 2-2,4,6; 7-8
YANK directive 2-7; 5-7,8; 10-8
6-bit display code 2-4,13; 7-4; A-4
6/12-bit display code 2-4,13; 7-4; A-4
COMMENT SHEET

MANUAL TITLE: CDC® Modify Reference Manual

PUBLICATION NO.: 60450100

NAME:

COMPANY:

STREET ADDRESS:

CITY: __________________________  STATE: ______________  ZIP CODE: __________________________

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

☐ Please Reply  ☐ No Reply Necessary

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.
MODIFY COMMAND PARAMETERS

MODIFY (P1, P2, ..., Pn)

A Presence of A causes compressed compile file.

C Compile file output; COMPILE if C or omitted. No compile file if C=0. Otherwise, output on file name (C=1fn).

CB COMPASS binary output file; used with Q and X options only. Output on LOG if CB. No binary if CB=0. Otherwise, output on the file named (CB=1fn).

CG COMPASS get text option; used with Q and X options only. Systems text on SYSText if CG. No system text if CG=0. Defined by CS option if CG is omitted. Otherwise, systems text on file named (CG=1fn).

CL COMPASS list output; used with Q and X options only. Short list if CL=0 or omitted. Output on file OUTPUT if CL. Otherwise, list output on file named (CL=1fn).

CS COMPASS system text; used with Q and X options only. Systems text on SYSText overlay if omitted or CS. No systems text if CS=0; otherwise, systems text on file named (CS=1fn).

CV Program library character set conversion. None if CV is omitted; 63 to 64 if CV=64; 64 to 63 if CV=63.

D Debug option. Directive error or fatal error causes job step abort if D is omitted. No job step abort for directive errors if D is used.

F Full edit. If omitted, deck editing determined by U option or by EDIt directives. If F is specified, all decks are edited and written on compile file, new program library, and source file.

I Directives input. If omitted, directives and corrections on INPUT. If I=0 there is no input file. Otherwise, on named file (I=1fn).

L List output. Omitted or L, listings on OUTPUT. I=1fn, output to named file. I=0, no list output.

LO List options. Omitted or LO, option E if list output file is assigned to a terminal; options E, C, T, M, W, D, and S not assigned to a terminal. Otherwise, LO=c1c2...cn to a maximum of seven options (ACDELMST or W).

N New program library. Omitted or N=0, no new library. N, output on NPL. N=1fn, output to named file.

NR No rewind on compile file. Omitted, compile file rewound before and after Modify run.

P Program library input. Omitted or P, library on OPL. P=1fn, library on named file. P=0, no program library input file.

Q Execute assembler or compiler; no rewind of directives file or list output file. Omitted or Q=0, assembler or compiler not automatically called. Q, Modify sets A parameter and LOG=E and calls COMPASS. This option enables CB, CG, CL, and CS options. If Q=1fn, Modify calls assembler on 1fn.

S Source output (invalid if A, U, or X selected). Omitted or S=0, no source output. S=1fn, output on SOURCE. S=1fn, output on named file.

U Update edit. Omitted, editing set by F or by EDIt directives. F takes precedence over U. If U, only decks changed (named on DECK directives) are edited and written on compile file, new program library, and source file.

X Execute assembler or compiler; same as Q except directives file and list output are rewound.

Z Directives on MODIFY command. Omitted, directives are next record on INPUT or identified by I option. Z, directives follow the MODIFY command terminator. Each directive must be preceded by a separator character.