UPDATE Reference Manual
SR–0013 K

Cray Research, Inc.
NEW FEATURES

This version of the product introduces another difference between NUPDATE and UPDATE. NUPDATE offers the following ID keywords: %G%, %H%, %I%, %M%, %T%, %V%, %Z%.

This new version also introduces option OS on the control statement which assigns new sequence numbers using the UPDATE 4.0 scheme.

This revision describes how to undefine the default name, UNICOS, under DEFINE in UPDATE and NUPDATE. In UPDATE, use the -d option to undefine the default. In NUPDATE, specify -d new_define_name, -U UNICOS, or *UNDEF UNICOS to undefine the default.
Each time this manual is revised and reprinted, all changes issued against the previous version are incorporated into the new version and the new version is assigned an alphabetic level.

Every page changed by a reprint with revision has the revision level in the lower righthand corner. Changes to part of a page are noted by a change bar in the margin directly opposite the change. A change bar in the margin opposite the page number indicates that the entire page is new. If the manual is rewritten, the revision level changes but the manual does not contain change bars.

Requests for copies of Cray Research, Inc. publications should be directed to the Distribution Center and comments about these publications should be directed to:

CRAY RESEARCH, INC.
655F Lone Oak Drive
Eagan, Minnesota 55121

<table>
<thead>
<tr>
<th>Revision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>June 1979 - This printing represents a complete rewrite of the manual and brings it into agreement with release 1.06. Changes are not noted by change bars.</td>
</tr>
<tr>
<td>A-01</td>
<td>December 1979 - This change packet brings the manual into agreement with release version 1.07. Changes are noted by change bars.</td>
</tr>
<tr>
<td>B</td>
<td>December 1979 - This reprint includes change packet A-01. It contains no other changes.</td>
</tr>
<tr>
<td>B-01</td>
<td>April 1980 - This change packet brings the manual into agreement with release version 1.08. Changes are noted by change bars.</td>
</tr>
<tr>
<td>C</td>
<td>November 1980 - This reprint incorporates change packet B-01. It contains no technical changes. With this printing, the publication number has been changed from 2240013 to SR-0013.</td>
</tr>
<tr>
<td>D</td>
<td>June 1981 - This rewrite brings the documentation into agreement with the 1.10 version of the released software. Major changes are the addition of the SQ control statement option, the NOSEQ directive, and the SEQ directive. Changes are not noted by change bars. This printing obsoletes all previous editions.</td>
</tr>
<tr>
<td>D-01</td>
<td>May 1982 - This change packet brings the manual into agreement with release version 1.11. Major changes include the addition of the declared modifications option parameter (DC) on the UPDATE control statement, the DECLARE directive, the DC parameter on the IDENT directive, and UPDATE messages. This change packet also includes miscellaneous technical and editorial changes.</td>
</tr>
</tbody>
</table>
D-02  May 1983 - This change packet brings the manual into agreement with release version 1.12. Major changes include adding the YANK and UNYANK directives, UPDATE messages, and the Y and C fields to the identifier table format of the random PL structure. This change packet also includes miscellaneous technical and editorial changes.

E  January 1984 - This rewrite brings the manual into agreement with release version 1.13. New directives are COPY, DEFINE, ELSE, ELSEIF, ENDIF, IF, MASTER, PURGE, RESTORE, REWIND, SKIPF, and WIDTH. New control statement parameters are ML and IF; and parameters I, DW, and * were changed. Other changes include reorganization of section 1 and new examples in section 4.

E-01  November 1984 - This change packet brings the manual into agreement with release version 1.14. New additions to the manual include section 5, and appendixes D and E. AUDPL messages have been appended to the end of appendix B changing the name from UPDATE MESSAGES to MESSAGES. NS was added to the output options in section 2 and changes were made to the following directives: REWIND, RESTORE, COPY, and DEFINE.

F  February 1986 - This rewrite brings the manual into agreement with UPDATE releases 2.0 and 2.1 operating under the Cray operating systems COS 1.15 release and UNICOS 1.0 release, respectively. The COS release added COMDECKS external references, the DEF control statement parameter, the conditional directives SKIP and ENDSKIP, and a new program library format (variable length record). UNICOS information was added to sections 1, 2, 3, and 4, and appendixes B and C. Information on the PLCOPY utility program is in appendix D. (Under UNICOS, the SKIPF directive does not function.) Binary Identifier Dataset Format information (formerly appendix D) has been incorporated into section 5. This printing also includes miscellaneous technical and editorial changes. All previous versions are obsolete.

G  October 1986 - This reprint with revision brings the manual into agreement with UPDATE releases 3.1 and 4.0 operating under the COS 1.16 release and UNICOS 2.0 release, respectively. References to a "set of files in a directory" have been changed to "filenames." References to IDENT.PL and INFO.PL have been changed to Identifier Table and PL Information Table, respectively. Three UNICOS UPDATE error messages were added, and one was deleted. All trademarks are now documented in the record of revision. All previous versions are obsolete.
June 1987 - This reprint with revision brings the manual into agreement with UPDATE release 5.0, AUDPL releases 1.16 and 2.0, and MODECKS release 1.0, all operating under the COS 1.16 and UNICOS 3.0 releases. Major changes include the addition of the MODECKS utility and the FILE directive. Miscellaneous technical and editorial changes have also been included.

June 1988 - This reprint with revision brings the manual into agreement with UPDATE release 5.1, AUDPL releases 1.16 and 2.0, and MODECKS release 1.1, all operating under the COS 1.17 and UNICOS 4.0 releases. Major changes include the addition of several new options to the MODECKS utility. This reprint also includes miscellaneous technical and editorial changes.

November 1989 - This reprint with revision brings the manual into agreement with UPDATE release 5.0 for COS. It adds information for using NUPDATE under COS and for the new sequence numbering scheme, and it describes the new COS listing extraction program, UPIC.

November 1990 - This reprint with revision brings the manual into agreement with UPDATE 6.0 release for UNICOS. It adds new options to the NUPDATE command.
This manual describes UPDATE, a program from Cray Research, Inc. (CRI). UPDATE modifies, edits, and updates source language programs operating under the Cray Research operating systems UNICOS and COS on CRAY Y-MP†, CRAY X-MP EA†, and CRAY X-MP computer systems, and under UNICOS on CRAY-2 systems. UPDATE lets you manage and track software changes. UPDATE also allows repeated results and simplifies the integration of separately produced changes into one program.

Readers should be familiar with the general features of COS or UNICOS; the manual does not teach the basics of either operating system.

The following CRI publications provide related information:

<table>
<thead>
<tr>
<th>Publication Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR-0009</td>
<td>Fortran (CFT) Reference Manual</td>
</tr>
<tr>
<td>SR-0060</td>
<td>Pascal Reference Manual</td>
</tr>
<tr>
<td>SR-0066</td>
<td>Segment Loader (SEGLDR) and ld Reference Manual</td>
</tr>
<tr>
<td>SR-2024</td>
<td>Cray C Reference Manual (COS only)</td>
</tr>
<tr>
<td>SR-2074</td>
<td>Cray Standard C Programmer's Reference Manual (UNICOS only)</td>
</tr>
<tr>
<td>SG-2087</td>
<td>Cray Allegro CL User's Guide</td>
</tr>
</tbody>
</table>

† In X-mode only (24-bit addressing)
The following list defines architecture terms:

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CX/1 systems</td>
<td>This group includes all models of the CRAY X-MP and CRAY-1 computer systems. It is characterized by 24-bit addressing capabilities.</td>
</tr>
<tr>
<td>CEA systems</td>
<td>This group includes all models of the Extended Architecture (EA) series, which are the CRAY Y-MP and CRAY X-MP EA computer systems. It is characterized by 32-bit addressing capabilities.</td>
</tr>
<tr>
<td>CRAY-2 systems</td>
<td>This group includes all models of the CRAY-2 computer systems. It is characterized by 32-bit addressing capabilities, large common memories, and immersion cooling.</td>
</tr>
<tr>
<td>CX/CEA systems</td>
<td>This group designates all models of CRAY X-MP computer systems plus all models of the CRAY Y-MP and CRAY X-MP EA computer systems. It does not include CRAY-1 computer systems.</td>
</tr>
<tr>
<td>EAM bit (hardware)</td>
<td>In CX/1 systems, the EAM bit is the Enhanced Addressing Mode bit in the Flag register. When set, it sign-extends certain instructions for memory addressing in 8- and 16-Mword systems. In CEA systems, the EAM bit is the Extended Addressing Mode bit in the Flag register. It is set by the operating system to select either 24- or 32-bit addressing.</td>
</tr>
<tr>
<td>EMA feature (software)</td>
<td>In CX/1 systems, EMA is the Extended Memory Addressing feature for 8- or 16-Mword systems.</td>
</tr>
<tr>
<td>X-mode</td>
<td>This term refers to the 24-bit addressing mode in CEA systems. The operating systems select this mode with the EAM bit in the Exchange Package.</td>
</tr>
<tr>
<td>Y-mode</td>
<td>This term refers to the 32-bit addressing mode in CEA systems. The operating systems select this mode with the EAM bit in the Exchange Package.</td>
</tr>
</tbody>
</table>
CONTENTS

PREFACE ................................................................. vii

1. INTRODUCTION ......................................................... 1-1
   1.1 OVERVIEW ......................................................... 1-2
   1.2 CONVENTIONS ...................................................... 1-3
   1.3 READER COMMENTS ............................................... 1-4
   1.4 DEFINITIONS ...................................................... 1-5
      1.4.1 Decks ....................................................... 1-5
      1.4.2 Source decks ............................................... 1-6
      1.4.3 Directives ................................................... 1-6
      1.4.4 Modification sets ......................................... 1-7
      1.4.5 Source datasets ............................................ 1-7
      1.4.6 Compile datasets .......................................... 1-7
      1.4.7 Input datasets ............................................. 1-8
   1.5 PROGRAM LIBRARIES .............................................. 1-8
      1.5.1 Program library restrictions .............................. 1-9
      1.5.2 Creating a program library ................................ 1-9
      1.5.3 Modifying a program library ................................ 1-10
         1.5.3.1 Procedure for modifying a PL ......................... 1-10
         1.5.3.2 Processing PL modifications ......................... 1-11
   1.6 COMMON DECK PROGRAM LIBRARIES ................................ 1-11
   1.7 UPDATE MODES .................................................. 1-11
   1.8 PROGRAM LIBRARY TYPES ........................................ 1-13
   1.9 ORGANIZING UPDATE INPUT ...................................... 1-13
      1.9.1 Associativity of input ..................................... 1-14
      1.9.2 Overlapping modifications ................................ 1-14
      1.9.3 Declared modifications .................................... 1-14
   1.10 LISTABLE OUTPUT ............................................... 1-15
      1.10.1 Page header lines ....................................... 1-15
      1.10.2 Messages .................................................. 1-15
      1.10.3 Listing options .......................................... 1-16

2. INVOKING UPDATE AND NUPDATE .................................... 2-1
   2.1 UPDATE CONTROL STATEMENT FOR COS ............................ 2-1
      2.1.1 Examples ................................................... 2-8
   2.2 NUPDATE CONTROL STATEMENT FOR COS ........................... 2-9
   2.3 UNICOS COMMAND LINE .......................................... 2-16
      2.3.1 Examples ................................................... 2-22
3. UPDATE DIRECTIVES

3.1 CATEGORIES OF UPDATE DIRECTIVES

3.1.1 Modification directives
3.1.2 Input edit directives
3.1.3 Run option directives
3.1.4 Compile dataset directives
3.1.5 Deck definition directives

3.2 DIRECTIVE FORMAT

3.2.1 Line identification
3.2.2 Identifier names
3.2.3 Directive format examples

3.3 DIRECTIVES

3.3.1 / - Comment
3.3.2 BEFORE - Insert before a line
3.3.3 CALL - Call common deck
3.3.4 COMDECK - Introduce a common deck
3.3.5 COMPIL - Specify compile or source datasets
3.3.6 COPY - Copy text
3.3.7 CWEOF - Conditionally write end-of-file
3.3.8 DECK - Introduce a deck
3.3.9 DECLARE - Declare deck for modifications
3.3.10 DEFINE - Define names
3.3.11 DELETE - Delete lines
3.3.12 EDIT - Edit decks
3.3.13 ELSE - Reverse condition
3.3.14 ELSEIF - Test condition
3.3.15 ENDIF - End conditional text
3.3.16 FILE - Close file (UNICOS only)
3.3.17 IDENT - Identify modification set
3.3.18 IF - Begin conditional text
3.3.19 INSERT - Insert after a line
3.3.20 LIST and NOLIST - Resume or stop listing
3.3.21 MASTER - Change input master character
3.3.22 MOVEDK - Move a deck
3.3.23 PURGE - Remove modification set
3.3.24 PURGEDK - Remove deck
3.3.25 READ - Read alternative input
3.3.26 RESTORE - Reactivate lines
3.3.27 REWIND
3.3.28 SEQ and NOSEQ - Start or stop sequence number writing
3.3.29 SKIP and ENDSKIP - Conditionally skip a block of directives
3.3.30 SKIPF - Skip dataset files
3.3.31 WEOF - Write end-of-file
3.3.32 WIDTH - Change line width in compile dataset
3.3.33 YANK and UNYANK - Delete or restore decks and modification sets
4. EXAMPLES ............................................................................. 4-1

4.1 CREATING A PROGRAM LIBRARY ...................................... 4-1
  4.1.1 PL creation .................................................................. 4-1
  4.1.2 Creating a PL and a compile file .................................. 4-2
  4.1.3 Reading input from an alternate dataset ...................... 4-2
  4.1.4 Full UPDATE mode ..................................................... 4-3
  4.1.5 Calling common deck into a user program ................... 4-4

4.2 MODIFYING A PROGRAM LIBRARY .................................... 4-4
  4.2.1 Generating a modified PL .......................................... 4-5
  4.2.2 Testing a modification set .......................................... 4-5

4.3 GENERATING AND USING COMMON DECK PROGRAM LIBRARIES 4-7
  4.3.1 Generating a PL with common decks ......................... 4-7
  4.3.2 Generating a PL with external common decks .............. 4-7
  4.3.3 Modifying a PL with external common decks ............... 4-8

4.4 READ FROM ALTERNATIVE DATASETS ................................ 4-9

4.5 INPUT DATASET NOT $IN ................................................. 4-10

4.6 MULTIPLE INPUT DATASETS ............................................ 4-10
  4.6.1 COS input dataset .................................................... 4-10
  4.6.2 UNICOS input dataset .............................................. 4-11

4.7 EXTRACTING DECKS FOR A SOURCE DATASET ................. 4-12

4.8 GENERATING A COMPILE DATASET FROM SOURCE ............ 4-12

4.9 COMPILe DATASET FROM A COMMON DECK .................... 4-13

4.10 EXTRACTING DECKS FOR COMPILATION (NO SOURCE) ..... 4-13

4.11 USING *FILE (UNICOS only) .......................................... 4-14

4.12 RESEQUENCING A PL ................................................... 4-14

4.13 DECK REMOVAL AND POSITIONING ............................... 4-15

4.14 PL EDITING................................................................. 4-16

4.15 CHANGING THE DATA WIDTH ....................................... 4-16

4.16 CONDITIONAL TEXT ..................................................... 4-17

4.17 EXAMPLES SHOWING DATASET CONTENT ......................... 4-19
  4.17.1 Create a new PL from an input file ......................... 4-19
  4.17.2 PL modification ..................................................... 4-20
  4.17.3 Generating an executable program ......................... 4-22
  4.17.4 PL resequenced version ......................................... 4-23

5. AUDPL - PROGRAM LIBRARY AUDIT UTILITY .......................... 5-1

5.1 RESTRICTIONS ............................................................. 5-1

5.2 AUDPL CONTROL STATEMENT (COS) ............................. 5-1

5.3 AUDPL COMMAND LINE (UNICOS) ................................ 5-6

5.4 INPUT DIRECTIVES ....................................................... 5-9
  5.4.1 / - Comment ........................................................ 5-9
  5.4.2 ACTIVE - Active lines .................................... 5-10
  5.4.3 COND - Conditional text directives ...................... 5-10
  5.4.4 DIR - Dataset or file directives ......................... 5-11
  5.4.5 HISTORY - Modification history ......................... 5-11
  5.4.6 INACTIV - Inactive lines .................................. 5-12
  5.4.7 PULLMOD - Pulled modification sets or decks........... 5-12
5. AUDPL - PROGRAM LIBRARY AUDIT UTILITY (continued)

5.5 OUTPUT ........................................... 5-13
  5.5.1 Listing file or dataset .......................... 5-13
    5.5.1.1 Text line listing options .................. 5-14
    5.5.1.2 Summary listing options ................... 5-15
  5.5.2 Reconstructed modification sets ................ 5-17
  5.5.3 Modifications file or dataset ................ 5-19
  5.5.4 Binary identifier list file or dataset and
    identifier list ................................... 5-19

5.6 AUDPL SAMPLE LISTING ............................ 5-20

6. MODECKS ........................................... 6-1

  6.1 MODECKS CONTROL STATEMENT (COS) ................. 6-1
  6.2 MODECKS COMMAND LINE (UNICOS) .................... 6-5
  6.3 USING MODECKS UNDER COS ........................ 6-9
  6.4 USING MODECKS UNDER UNICOS ...................... 6-10

7. UPIC - COS LISTING EXTRACTION PROGRAM .............. 7-1

  7.1 UPIC INPUT .................................... 7-1
  7.2 UPIC OUTPUT ................................... 7-2
    7.2.1 UPDATE listings ............................... 7-2
    7.2.2 Compilation and assembly listings ............ 7-2
  7.3 UPIC CONTROL STATEMENT .......................... 7-2
  7.4 UPIC EXAMPLES .................................. 7-5

APPENDIX SECTION

A. CHARACTER SET ................................... A-1

B. MESSAGES ......................................... B-1

  B.1 UPDATE MESSAGES ................................ B-1
  B.2 AUDPL LOG FILE MESSAGES ......................... B-7
  B.3 MODECKS ERROR MESSAGES ........................ B-9
  B.4 UPIC LOG FILE MESSAGES ........................ B-11
    B.4.1 General messages .............................. B-11
    B.4.2 Control statement error messages ............ B-12
    B.4.3 UPIC internal error messages ................. B-12
C. UPDATE PROGRAM LIBRARY FORMATS .............................................. C-1
   C.1 FORMAT 1 - SEQUENTIAL PL STRUCTURE (COS ONLY) ................. C-1
   C.2 FORMAT 2 - RANDOM PL STRUCTURE ........................................ C-3

D. PLCOPY UTILITY PROGRAM ....................................................... D-1

E. UPDATE DIRECTIVE SUMMARY .................................................. E-1

FIGURES
   1-1 UPDATE Data Flow ............................................................. 1-3
   1-2 Program Library Sequence ................................................ 1-8
   3-1 UPDATE Directives ............................................................. 3-2
   C-1 PL Format 1 ................................................................. C-1
   C-2 PL Format 2 ................................................................. C-4

TABLE
   1-1 Dataset Contents for Full, Quick, and Normal Modes ............... 1-12

INDEX
UPDATE is a line-oriented text editor for maintaining programs in the form of source code, as well as other types of text data. UPDATE creates and modifies program libraries (PLs) and produces output that you can use as input to other programs, particularly compilers and assemblers.

UPDATE executes on all CRAY Y-MP†, CRAY X-MP EA†, and CRAY X-MP computer systems under control of COS or UNICOS, and under UNICOS on the CRAY-2 computer system. You can invoke UPDATE with the UPDATE COS control statement or UNICOS command line; both are described in section 2, Invoking UPDATE and NUPDATE.

NUPDATE is a derivative of UPDATE with improved functionality and performance. Unless otherwise noted, all UPDATE information also applies to NUPDATE. The principal differences are as follows:

- NUPDATE permits identifiers consisting of up to 240 characters; UPDATE identifiers are limited to 8 characters.

- NUPDATE does not support the COS CL= parameter and the UNICOS -u option. CL= and -u let you specify called common decks not found in the PL being processed.

- NUPDATE contains two additional directives, OLDSEQ and OLDPL, which provide compatibility with the UPDATE sequence number generation scheme used in UNICOS 4.0, COS 1.17, and earlier releases, and with the UPDATE PL format, respectively. When using a variable length PL, NUPDATE lets you have up to 500 characters per line while UPDATE limits you to 250.

- NUPDATE contains other options that apply only to UNICOS. See nupdate(1) in the UNICOS User Commands Reference Manual, publication SR-2011.

- NUPDATE also offers the following ID keywords:

<table>
<thead>
<tr>
<th>ID Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%G%</td>
<td>Date the version was created; in the form mm/dd/yy.</td>
</tr>
<tr>
<td>%H%</td>
<td>Current date; in the form mm/dd/yy.</td>
</tr>
</tbody>
</table>

† In X-mode only
### ID Keyword Description

- `%I%` USM version identification string (VID) of the retrieved version.
- `%M%` Module (file) name.
- `%T%` Current time; in the form hh:mm:ss.
- `%V%` Time the version was created; in the form hh:mm:ss.
- `%Z%` A 4-character string @(#) that is used by the what(l) command to retrieve information on the file. (See the what(l) man page in the UNICOS User Commands Reference Manual, publication SR-2011, for more information.)

Three other products, AUDPL, MODECKS, and UPIC, are also described in this manual. AUDPL, the PL audit utility, provides information about UPDATE PLs. MODECKS compares a modified UPDATE deck and an original deck, and it produces a modification file reflecting the differences. UPIC extracts portions of program listings when you do not want to see the entire program. AUDPL, MODECKS, and UPIC are intended to be used in conjunction with UPDATE.

### 1.1 OVERVIEW

UPDATE performs three main functions: it can create a new PL, modify an existing PL, and extract source code from an existing PL. Figure 1-1 illustrates two of these functions, showing the data flow that occurs in creation and modification runs.

For a creation run, your input must include source decks, and it can include a source dataset from an earlier UPDATE run, modification sets, and input directives. Output from a creation run can include a new PL, listings, a compile dataset, and/or a source dataset.

For a modification run, your input must include a PL, and it can include new decks, modification sets, and input directives. Output from a modification run can be a selected listing, a compile dataset, source decks, and/or a new PL.

UPDATE also lets you extract source code from an existing PL without creating or modifying one. Examples of UPDATE's extracting capabilities are given in subsections 4.7, Extracting Decks for a Source Dataset, and 4.10, Extracting Decks for Compilation (No Source).
1.2 CONVENTIONS

This manual uses the following conventions to describe statement and directive syntax:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPPERCASE</td>
<td>This identifies a COS command verb, and an UPDATE or AUDPL directive. With UPDATE 5.0, all UPDATE directives can be given in either uppercase or lowercase, but not in mixed case. This applies to both COS and UNICOS.</td>
</tr>
<tr>
<td><strong>Bold</strong></td>
<td>Bold identifies UNICOS command names, options, or file names.</td>
</tr>
<tr>
<td><strong>Underscore</strong></td>
<td>Underscoring specifies the accepted abbreviation of a verb or parameter.</td>
</tr>
<tr>
<td><em>Italic</em></td>
<td>Italic identifies generic terms representing the words or symbols to be supplied by the user; all variable entities are presented in italics.</td>
</tr>
<tr>
<td>[ ]</td>
<td>Brackets enclose optional portions of a command format.</td>
</tr>
<tr>
<td><strong>Choice 1</strong></td>
<td>Stacked items indicate two or more literal parameters when only one choice can be used.</td>
</tr>
<tr>
<td><strong>Choice 2</strong></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1-1. UPDATE Data Flow
dataset/file  Though the terms dataset and file should be reserved for use with, respectively, COS or UNICOS, this manual occasionally uses them interchangeably for convenience.

... Ellipses indicate optional use of a preceding item one or more times in succession.

command(1) The designation (1) following a command name indicates that the command is documented in the UNICOS User Commands Reference Manual, publication SR-2011.

1.3 READER COMMENTS

If you have any comments about the technical accuracy, content, or organization of this manual, please tell us. You have several options that you can use to notify us:

- Call our Technical Publications department at (612) 683-5729.
- Send us electronic mail from a UNICOS or UNIX system, using the following UUCP mail address:
  
  uunet!cray!publications

- Send us electronic mail from a UNICOS or UNIX system, using one of the following Internet addresses:
  
  Pubs0013@timbuk.cray.com  (comments specific to this manual)
  
  publications@timbuk.cray.com (general comments)

- Send a facsimile of your comments to the attention of "Publications" at fax number (612) 683-5599.

- Use the postage-paid Reader's Comment form at the back of this manual.

- Write to us at the following address:

  Cray Research, Inc.
  Software Publications and Training Department
  655F Lone Oak Drive
  Eagan, MN  55121

We value your comments and will respond to them promptly.

1-4
1.4 DEFINITIONS

This subsection presents brief definitions for terms used later in this manual. The terms dataset and file are used interchangeably in these definitions for convenience; strictly speaking, these terms should be reserved for use with, respectively, COS or UNICOS. Unless otherwise noted, these terms are defined in greater detail in subsections 1.4.1 through 1.4.7.

- A **deck** is a contiguous ordered set of lines that can be referenced with a single name.

- A **program library** (PL) is created by UPDATE and contains one or more decks. Under COS, a PL is a dataset, and under UNICOS, it is a file. (Subsection 1.5 discusses PLs.)

- A **source deck** includes all text and directives that were or will become a deck in a PL; it can be derived from a source dataset. A source deck can be input to or output from UPDATE.

- A **directive** is a command to UPDATE from the input or embedded in the PL.

- A **modification set** is a group of changes to be applied to existing decks in a PL.

- A **source dataset** is a file that is a current, edited copy of one or more source decks from a PL. UPDATE writes the source dataset. You can use decks from a source dataset as input to UPDATE.

- A **compile dataset** is produced by UPDATE for use as input to an assembler, compiler, or other program. Under COS, it is a dataset containing one or more files. Under UNICOS, it is one or more distinguishable files.

- An **input dataset** can include source decks, modification sets, and directives.

- A **listing dataset** is output by UPDATE; it contains messages and requested information for you to read. (See subsection 1.10, Listable Output.)

1.4.1 DECKS

A **deck** is a contiguous ordered set of lines that can be referenced with a single name. It can be a program, part of a program, or other data. It serves as input to UPDATE to become part of a PL. UPDATE supports two types of decks: regular and common.
• A **regular deck** begins with the DECK directive; it is placed in the PL (at one location) and in compile and source datasets.

• A **common deck** begins with the COMDECK directive; it is placed in the PL and in the source dataset, but you can copy its contents to any number of locations in the compile dataset. A common deck that is defined in one PL can be called from decks in another PL if the COS CL control statement parameter or the UNICOS -u command-line option is used. A common deck call can appear anywhere in a regular or common deck. (See subsection 3.3.3, CALL - Call Common Deck.) A common deck call is replaced by the text of the common deck during compile dataset generation.

### 1.4.2 SOURCE DECKS

A **source deck** includes all text and directives that will become a new deck in a PL, or lines and embedded directives from a single deck in the source dataset. It begins with a DECK or COMDECK directive and ends with the last line before the next DECK, COMDECK, IDENT, modification directive, or the end of input.

### 1.4.3 DIRECTIVES

Directives--that is, commands to UPDATE--are in the input to UPDATE or are embedded in the PL. The individual directives are described in section 3. The UPDATE directives are of the following types:

• **Modification** directives change a PL; they are entries to modification sets. The effects of modification directives are saved, and you can yank (undo) them.

• **Input edit** directives change a PL, but they are not saved and cannot be yanked.

• **Run option** directives do not change a PL, and they are not saved; most select I/O options for a run.

• **Compile dataset** directives determine the contents of compile datasets, and they are embedded in the PL.

• **DECK and COMDECK** directives define decks.
1.4.4 MODIFICATION SETS

A modification set is a group of UPDATE modification directives to be applied to existing decks in a PL in either a creation or modification run. You can use the INSERT, BEFORE, DELETE, and RESTORE directives in a modification set. Source decks, input edit directives, and run option directives can be in a modification set, but they are not associated with it. You can remove changes associated with a modification set from the PL, either temporarily or permanently, using the YANK, UNYANK, and PURGE directives.

A modification set begins with a modification identifier, specified by an IDENT directive. It ends with the next IDENT, DECK, or COMDECK directive, or the end of input.

1.4.5 SOURCE DATASETS

A source dataset is a file written by UPDATE that is a current, edited copy of one or more decks from a PL. You can use it as input to UPDATE to create a new PL or to add new decks and common decks to an existing PL. It consists of a single file of active text lines, compile dataset directives, and DECK and COMDECK directives from selected decks in the PL. See table 1-1 for the contents of a source dataset.

The DW parameter on the COS control statement or the -w option on the UNICOS command line determines the length of each line in the source dataset. If you specify the COS SQ or UNICOS -o sq option, you will also receive sequencing information.

Because the source dataset contains UPDATE directives, it is normally not used as input to language processors. (The source dataset always begins with a DECK or COMDECK directive.)

1.4.6 COMPILE DATASETS

A compile dataset contains one or more files of active text lines from selected decks in the PL. Compile dataset directives are not written to the compile dataset, but they are processed as the compile dataset is written. Common decks are expanded, conditional text directives are evaluated, and the dataset is broken into files by the WEOF, CWEOF, and FILE directives. See table 1-1 for the contents of a compile dataset.

The COS DW parameter, UNICOS -w option, or the WIDTH directive control the length of each line. Each text line is associated with an identifier and sequence number. The identifier is the name of the deck or common deck for an original line, or the name of the modification set that added the line. Under COS, sequence information appears in the compile dataset.
by default. Under UNICOS, you must explicitly request sequence information. You will find the identifier and sequence number printed to the right of the text line, unless the PL has been identified as a variable-length record PL. In this case, no sequence information appears in the compile dataset. You can also turn off sequence information with the COS NS option, UNICOS -0 ns option, or the NOSEQ directive.

1.4.7 INPUT DATASETS

An input dataset can include source decks, modification sets, and input directives. You can use a source dataset from one UPDATE run as an input dataset for a later UPDATE run. Specify an input dataset with the COS I parameter, UNICOS -i option, or the READ directive. Under COS, input datasets containing more than one file must be specified once for each file to be read.

1.5 PROGRAM LIBRARIES

A PL is a dataset (in COS) or a file (in UNICOS) containing one or more decks created by UPDATE. Each deck is specially formatted for use as input to future UPDATE runs. Following the last deck in a PL, UPDATE supplies a directory consisting of tables describing each deck, each modification set identifier, and the entire PL. (See figure 1-2.) These tables differ, depending on whether the PL has random or sequential organization. (See appendix C for information on PL formats.)

```
| Deck 1 |
| Deck 2 |
| ... |
| Deck n |
| UPDATE |
| Directory |
```

Figure 1-2. Program Library Sequence
A deck begins with a DECK or COMDECK directive; it contains lines of text and can contain compile dataset directives. UPDATE assigns an identifier and sequence number to each text line or directive. The identifier of an original line from the deck is the deck or common deck name; a line added later has as its identifier the name of the modification set that added it. The DECK or COMDECK directive has sequence number 1. UPDATE sequences the remaining lines in the deck, beginning with number 2. Lines added by modification sets begin with sequence number 1.

Deleted lines remain in the PL with an inactive status. A descriptor precedes each text line or directive in the PL, giving the line's identifier, sequence number, and a correction history recording modifications to the line.

1.5.1 PROGRAM LIBRARY RESTRICTIONS

The number of lines within one modification set or one deck cannot exceed 131,071. The number of identifiers (that is, modification set identifiers or deck names) in one PL must not exceed 16,383.

UPDATE cannot read a PL from a magnetic tape dataset.

1.5.2 CREATING A PROGRAM LIBRARY

Create a PL by supplying the following:

- An UPDATE control statement or command line
- UPDATE directives
- Input text

The input text for creation of a new PL consists of one or more source decks you have prepared, or the source dataset from an earlier UPDATE run. Modification sets can also be applied in a creation run. Designate a creation run by specifying P=0 on the COS UPDATE control statement, or by not specifying the UNICOS -p option on the command line (see section 2, Invoking UPDATE and NUPDATE, for details on the control statements).

The DECK and COMDECK directives specify whether a new deck is a regular deck or common deck. Other directives within the source decks affect the output listing, call common decks, or write COS end-of-file (EOF) records. (See subsection 3.1.4, Compile Dataset Directives.)
1.5.3 MODIFYING A PROGRAM LIBRARY

You can modify the PL by adding or purging decks or by adding or deleting (deactivating) lines from existing decks. You can apply modifications against an existing PL to produce either a new PL or a current compile or source dataset.

1.5.3.1 Procedure for modifying a PL

Modify a PL by supplying the following:

- An UPDATE control statement or command line directing the computer system to modify that PL
- UPDATE directives specifying the modification set identifier or the deck identifier, the lines to be deleted, and locations of insertions
- Any new lines to be added

Input for a modification run can include new decks or common decks, modification sets, input edit directives, and run option directives. When a PL has been modified, the newly generated PL is known as a new PL.

A modification set begins with an IDENT directive containing a modification set identifier; a new deck begins with a DECK directive containing a deck name. If you wish to insert text at a location specified in the directive, that text must immediately follow the directive. Decks can also contain certain directives such as CALL, CWEOF, and WEOF. (See subsection 3.1.4, Compile Dataset Directives, for a complete listing.)

Following a modification run, the new PL, if generated, consists of modified decks, an updated identifier table, and an updated PL information file. (See subsection C.2, Format 2, Random PL Structure, for details on the format.) The sequence of decks and tables in the new PL is the same as that of the old PL.

Directives can cause lines to be inserted or deleted from the PL. A deletion does not physically remove the line but deactivates it; that is, the line is logically removed and does not appear in compile or source output. UPDATE maintains a record of active and inactive lines (see subsection C.2, Format 2, Random PL Structure). If the line status bit indicates an inactive line, the line is effectively deleted until restored. If the line status is active, the line remains in the deck.
1.5.3.2 Processing PL modifications

UPDATE processes modifications in two passes:

- During the first pass, UPDATE scans the directives and new text and constructs tables for use during the second pass.
- During the second pass, UPDATE modifies each deck on a deck-by-deck basis, deleting lines and sequencing and identifying each new line according to its modification set identifier.

If you want to receive compile output from a modification run, UPDATE creates it during the second pass. At this time, calls for common decks are replaced with current common deck text of the common deck. The calls can be in the original decks, or you can insert them during a modification run. Other embedded directives are also executed at this time.

1.6 COMMON DECK PROGRAM LIBRARIES

Common decks defined with the COMDECK directive in one PL may be called from decks in other PLs. For documentation purposes, this manual labels the PL in which a common deck is defined as a Common Deck Program Library. Called common decks defined in PLs other than the one currently being processed are labeled Externally Defined Common Decks.

For UPDATE to resolve references to externally defined common decks, the COS CL parameter or UNICOS -u option must be used in the control statement (see section 2, Invoking UPDATE and NUPDATE). Externally defined common decks are processed only in a Read mode. You cannot modify them or specify them with the Q parameter or -q option or a COMPIL DIRECTIVE, and they do not appear in a source dataset or new PL. The externally defined common deck text is copied to the compile dataset.

1.7 UPDATE MODES

The UPDATE control statement specifies either Full, Quick, or Normal mode. The mode determines the contents of the compile dataset, the source dataset, and the new PL. (See table 1-1.)
<table>
<thead>
<tr>
<th>Mode</th>
<th>Compile Dataset Contents</th>
<th>Source Dataset Contents</th>
<th>New PL Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>All decks and called common decks</td>
<td>All decks and all common decks defined in the PL being processed</td>
<td>All decks and all common decks defined in the PL being processed</td>
</tr>
<tr>
<td>Quick</td>
<td>Only decks specified by either COMPILE directives, or the UPDATE Q parameter or -q option, and called common decks</td>
<td>Decks and common decks specified by COMPILE directives, and decks specified by the UPDATE Q parameter or -q option</td>
<td>Only decks specified by either COMPILE directives or the UPDATE Q parameter or -q option, and all common decks defined in the PL being processed</td>
</tr>
<tr>
<td>Normal</td>
<td>Decks specified by COMPILE directives, modified decks, and decks calling modified common decks</td>
<td>Decks and common decks specified by COMPILE directives, modified decks, and decks calling modified common decks</td>
<td>Same as Full mode</td>
</tr>
</tbody>
</table>

In Full mode (F or -f), UPDATE writes all active lines in the PL and externally defined common deck texts to the compile dataset. Only the active lines in a PL are written to the source dataset. The PL Identifier table determines the sequence, and no COMPILE directive is necessary.

In Quick mode, UPDATE writes decks specified with the Q parameter or -q option and decks specified by a COMPILE directive to the compile dataset or the source dataset, and to the new PL. You cannot name externally defined common decks with the Q parameter, -q option, or the COMPILE directive. The PL Identifier table determines the sequence unless the K or -o k option is used. Modifications to decks that you did not specify with the Q parameter or -q option or by a COMPILE directive are not processed.
In Normal mode (Full and Quick mode parameters omitted), UPDATE writes decks specified by COMPILER directives, modified decks, and decks calling modified common decks to compile or source datasets, and all decks are written to the new PL. Under UNICOS, the -m option, naming the new PL file name, must be used. You cannot name externally defined common decks with the COMPILER directive, and UPDATE does not write them to the new PL.

1.8 PROGRAM LIBRARY TYPES

UPDATE distinguishes two types of program libraries: regular and variable-length record formats.

Under COS, UPDATE stores fixed-length "cards" to the PL and writes fixed-length lines of text to the compile and source datasets. Under UNICOS, UPDATE writes fixed-length lines of text to the compile and source datasets. You can specify them with the DW parameter or -w option.

Variable-length record PLs store and write variable-length lines of text. You can create these PLs by specifying DW=* or -w '.*' at the time the PL is created; however, you cannot create new PLs of one type from old PLs of the other type. Variable-length record PLs let you preserve the end of line locations in the original input text, instead of removing trailing blanks from the end of the line. The maximum line length is 256 characters.

Under COS, the storage space required for regular and variable-length record PLs is not significantly different, because blank compression is performed on all UPDATE PLs. Under UNICOS, trailing blanks are trimmed off of input datasets and PLs to save space, and the storage space required for regular and variable-length record PLs is not significantly different. If trailing blanks are significant, you must use a variable-length record format. If the PL is of variable-length record format, UPDATE does not write sequence information to the compile dataset.

1.9 ORGANIZING UPDATE INPUT

This subsection describes aspects of input to UPDATE that must be considered if you are to obtain correct output.
1.9.1 ASSOCIATIVITY OF INPUT

The compile and source files resulting from an UPDATE run will be the same whether input is processed in one UPDATE run or in several. That is, you can group modifications in different ways in a series of runs without affecting the result, as in the associative principle of mathematics. (This does not imply, however, that the order of modifications can be changed without affecting the result.) Directives EDIT, COPY, and DEFINE are exceptions to this principle. In addition, your results will be unpredictable if your input does not begin with an IDENT directive, or if you have used the MASTER directive but not at the beginning of each separate section of input. PURGEDK can follow modifications to the purged deck only if you declare those modifications.

1.9.2 OVERLAPPING MODIFICATIONS

UPDATE can handle modifications to text added earlier in the same UPDATE run, but overlapping or conflicting modifications generate caution and note messages. Messages about overlapping modifications are written to the listing or error datasets if the value given for the ML parameter or -m option is less than the default of 3. Caution messages indicate conflicts between directives; these conflicts occur when more than one directive inserts new text in the same place, and for implicit overlaps, when newly inserted text is deleted by a later modification set. Notes appear when a directive explicitly references a line added by an earlier modification set in the same UPDATE run.

1.9.3 DECLARED MODIFICATIONS

A modification declaration specifies the deck to be modified by subsequent directives and is required when the declared modifications option is specified (DC=ON on the COS control statement or the -o dc option specified in the UNICOS command line). For a modification declaration, you can use a DECLARE directive or the DC parameter on the IDENT directive. If modification declarations are required, one of these kinds of modification declarations must precede modification of a deck.

Modifications following the DECLARE directive affect only the specified deck or common deck. An UPDATE error is the result for each modification directive that affects any deck other than the declared deck.

If declared modifications are optional (DC=OFF or DC omitted on the COS control statement, or the -o dc option omitted from the UNICOS command line), DECLARE directives are optional and honored when they appear. Using an IDENT directive clears the previous modification declaration.
1.10 LISTABLE OUTPUT

UPDATE can produce a listing dataset and an error dataset, as specified on the UPDATE control statement. The error dataset contains only messages. The listing dataset contains these messages plus information requested by the control statement or command-line listing options.

Listable output is divided into pages. Under COS, the LPP parameter on the OPTION control statement controls the number of lines per page (see the COS Reference Manual, publication SR-0011). Under UNICOS, there is no page control.

1.10.1 PAGE HEADER LINES

Each page of output in the listing and error datasets contains two header lines with the following information:

- Job name (COS only)
- UPDATE revision level and compilation date
- Current date and time
- Page number for this UPDATE listing
- A description of the output on this page. For the input listing, this includes the name of the deck, common deck, or modification set being input. For the modification summary, this includes the name of the deck or common deck currently being processed.
- Date when the PL was created
- Name of the last identifier added to the PL

1.10.2 MESSAGES

Listing messages have five levels of severity: COMMENT, NOTE, CAUTION, WARNING, and ERROR. The COS ML parameter or UNICOS -m option specifies the highest level to be omitted from a listing or error dataset. The default is to write only error and warning messages. This parameter does not affect log file messages (stderr under UNICOS).
1.10.3 LISTING OPTIONS

The UPDATE control statement allows you the following set of options for listable output. Under UNICOS, your first option-argument must be preceded by the -o option and a space. Additional options-arguments must be separated by one space.

<table>
<thead>
<tr>
<th>COS</th>
<th>UNICOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option</td>
<td>Option-argument</td>
</tr>
<tr>
<td>CD</td>
<td>cd</td>
</tr>
<tr>
<td>ED</td>
<td>ed</td>
</tr>
<tr>
<td>ID</td>
<td>id</td>
</tr>
<tr>
<td>IF</td>
<td>if</td>
</tr>
</tbody>
</table>
### COS Option

<table>
<thead>
<tr>
<th>Option</th>
<th>Option-argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>in</td>
<td>Writes an echo of the input to UPDATE to the listing dataset. The input is divided according to decks, common decks, and modification sets.</td>
</tr>
<tr>
<td>UM</td>
<td>um</td>
<td>Writes to the listing and error datasets a list of modifications that remain unprocessed at the end of the UPDATE run, either because they refer to nonexistent lines or because they modify decks that were not specified when using Quick mode.</td>
</tr>
<tr>
<td>K</td>
<td>k</td>
<td>Orders all decks that are written to the compile dataset and to new PL datasets, as directed by the Q parameter values on the control statement and the COMPILe directives. This option is ignored in Full or Normal mode.</td>
</tr>
</tbody>
</table>

---

**NOTE**

If a modification set affects two or more decks and the K option is in effect, the sequence numbers of inserted lines can be inconsistent with sequencing that has occurred without the K option.

---

<table>
<thead>
<tr>
<th>Option</th>
<th>Option-argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>na</td>
<td>Does not abort if directive errors or modification errors occur. All requested datasets are generated.</td>
</tr>
<tr>
<td>NR</td>
<td>na</td>
<td>Does not rewind the source dataset or compile dataset at the beginning or end of UPDATE execution.</td>
</tr>
<tr>
<td>NS</td>
<td>ns</td>
<td>Suppresses line sequence information in the source and compile datasets. SEQ and NOSEQ directives are ignored when this option is used. NS is ignored for variable-length record PLs.</td>
</tr>
</tbody>
</table>
OLDSEQ oldseq Assigns sequence numbers using the old scheme rather than the new (default). See subsection 3.2.1, Line Identification, for a description of the differences between the two. (Applies to NUPDATE only).

OS os Assigns new sequence numbers using the UPDATE 4.0 scheme.

SQ sq Provides sequence numbers for source and compile output. The defaults are sequence numbers on the compile output and no sequence numbers on the source output.

UM um Writes unprocessed modifications to the listing dataset and/or the error dataset.

SSQ ssq Provides sequence numbers for source output only, not compile output. (Applies to NUPDATE only).
2. INVOKING UPDATE AND NUPDATE

This section describes the COS control statements to invoke UPDATE and NUPDATE and the UNICOS command line to invoke UPDATE. (Section 1 describes the conventions used in the control statements.)

2.1 UPDATE CONTROL STATEMENT FOR COS

The COS UPDATE control statement loads the UPDATE program into the user field and begins execution. The UPDATE statement is read from the control statement file of a user job deck. Use the parameters on the UPDATE statement to specify the datasets to be used, the contents of the UPDATE listing, and other features of the run.

Format:

```
UPDATE[,P=pdn][,I=ldn1:ldn2:...:ldnn][,C=cdn]
[,CL=cln1:cln2:...:clnn][,DEF=def1:def2:...:defn][,N=ndn]
[,L=ldn][,E=edn][,S=sdn][,*=m][,/=c][,DW=dw][,DC=dc][,ML=n]
,F
,Q=d1:d2:...:dn]
,[,options].
,Q='d1,d2,···,dj,dk,···,dn'
```

P=pdn  Dataset name of the program library (PL)

If omitted or P, the input PL is $PL.
If P=pdn, the input PL is pdn.
If P=0, no PL is used; this is for only a creation run.
I=idn₁:idn₂:···:idnₙ
Names of input datasets; these datasets contain the directives and text for the UPDATE run. UPDATE reads them in the order given. You can specify up to 64.

If I or omitted, the input dataset is the next file of $IN.

If I=idn, the input dataset is a dataset with the name idn.

If I=O, no input dataset is read (invalid for a creation run).

If I=idn₁:idn₂:···:idnₙ, the first input dataset to be read is idn₁, the second is idn₂, and so on.

C=cdn
Name of the compile dataset; decks that are determined by the UPDATE mode (F, Q, or Normal; see table 1-1) are written on the specified dataset.

If C is omitted, the compile output is written to $CPL.
If C=cdn, the compile output is written to dataset cdn.
If C=O, no compile dataset is generated.

CL=cln₁:cln₂:···:clnₙ
Called common decks that cannot be found in the PL being processed are searched for in the common deck PLs (clnₙ) specified on the CL parameter. The search order is the order in which the libraries are specified. No modifications can be made to a common deck PL, and externally defined common deck text is not written to the source dataset or to the new PL. The common deck text appears in the compile dataset. The maximum number of common deck libraries is 64.

DEF=def₁:def₂:···:defₙ
The DEF parameter defines names to be used with an IF directive. You can use names consisting of up to 8 characters; if they are longer, only the first 8 characters are recognized. Defined names need not be different from deck names, common deck names, or modification identifiers, because they are known only in the UPDATE run being processed. You may define up to 64 names with the control statement DEF parameter.
N=ndn  Name of the new PL dataset. The UPDATE mode determines the contents of the new PL (see table 1-1).

If omitted in a modification run, no new PL is written.

If omitted in a creation run, the new PL is written to $NPL.

If N, the new PL is written to $NPL.

If N=ndn, the new PL is ndn.

If N=0, no new PL is written.

L=ldn  Name of the listing dataset; this dataset receives the UPDATE list output.

If omitted or L, the list output is written to $OUT.

If L=ldn, the list output is written to the dataset named ldn.

If L=0, no list output is generated.

E=edn  Name of the error listing dataset; this dataset contains ERRORS, WARNINGS, CAUTIONS, NOTES, and COMMENTS as requested by the ML parameter.

If omitted or E, the output is written to $OUT.
If E=edn, the output is written to edn.
If E=0, errors are written only to the listing dataset.

NOTE

If E and L specify the same dataset, L is honored and E is set to 0.

S=sdn  Source dataset name; the UPDATE mode determines the contents of this dataset (see table 1-1). You can use this dataset as the input for a subsequent creation run.

If omitted or S=0, no source output is generated.
If S=sdn, the source output is written to sdn.
If S, the source output is written to $SR.
**=m

Master character; the first character of directives read from the input file or written to the source file. Invalid master characters are comma, period, colon, equal sign, and space. The keyword alone is invalid.

If omitted in a creation run, the master character for directives is **.

If omitted in a modification run, the master character is that used in the creation run for the PL.

If **=m, the master character for directives is m.

*/=c

Comment character; indicates a comment. The keyword alone is invalid.

If omitted, the comment character is /.
If */=c, the comment character is c.

**w=dw

Data width value; the number of characters of data written to each line in the compile and source datasets. The dw range is 1 through 256.

If omitted, or if a DW with no value is specified in a creation run, columns 1 through 72 contain data; otherwise, columns 1 through dw contain data.

If **w=* in a creation run, variable-length records are written to the compile and source datasets.

In a modification run, if the PL was created with **w=*, only **w=*, **w, or no DW parameter is accepted, and UPDATE continues to process variable-length text lines for the PL. In a modification run, if no value for **w is specified, or **w is unspecified, columns 1 through last**w contain data; last**w is the **w value specified when the PL was written. If **w=dw, columns 1 through dw contain data.

The number of characters per line written to the new PL is a maximum of either dw, pl**w, or 80; pl**w is the number of characters per line in the existing regular PL. For variable-length record PLs, the same number of characters per line appears in the new PL as they did in the old PL.
NOTE

For regular PLs, sequence information is provided in the source and compile datasets as follows:

When you omit the data width value, when DW is specified alone, or when you specify the data width value as \( dW \), the following is true: \( dw+1 \) through \( dw+8 \) contain an identifier, right-justified with leading spaces; \( dw+9 \) contains a period; and \( dw+10 \) through \( dw+15 \) contain a sequence number, left-justified with trailing spaces.

When you specify the data width value as \( Ldw \), the entire sequencing field of the compile output is left-justified.

For variable-length record PLs, no sequence information appears in the compile dataset. Source dataset sequence information, if you request it, appears three spaces to the right of the end of the text line.

Sequence information appears by default in the compile dataset but must be requested in the source dataset.

\textbf{DC=dc} 

Declared modifications option. This ensures that modifications apply to the correct deck or common deck. Declaration of PL modifications may be required (see subsections 1.5.3, Modifying a Program Library, and 3.1.1, Modification Directives).

If DC is omitted or DC=OFF, the mod declaration is not required, but it is enforced if present.

If DC=ON or if DC stands alone, the mod declaration is required.
ML=n Message level: highest level of severity for UPDATE listing messages to be suppressed. For example, ML=2 allows CAUTION, WARNING, and ERROR messages to be printed to the listing or error datasets. The default, used when the parameter is omitted or the keyword alone is specified, is 3. (The ML parameter does not affect UPDATE log file messages.) The following levels are available:

<table>
<thead>
<tr>
<th>Level</th>
<th>Severity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>COMMENT</td>
<td>Currently unused</td>
</tr>
<tr>
<td>2</td>
<td>NOTE</td>
<td>Information not related to errors</td>
</tr>
<tr>
<td>3</td>
<td>CAUTION</td>
<td>Possible error</td>
</tr>
<tr>
<td>4</td>
<td>WARNING</td>
<td>Probable error</td>
</tr>
<tr>
<td>5</td>
<td>ERROR</td>
<td>Fatal error</td>
</tr>
</tbody>
</table>

F, Q, or omitted
Full, Quick, or Normal UPDATE run. This determines the contents of the compile dataset, the source dataset, and the new PL (see table 1-1).

F  Full UPDATE mode; all active lines are processed. The PL Identifier table determines the sequence. No COMPILE directive is necessary.

Q=[d_1:d_2:::d_n]  
Q='d_1,d_2,:::d_j,d_k,:::d_n'

Quick UPDATE mode. Decks that are specified with the Q parameter and decks specified by a COMPILE directive are written to the compile dataset and/or the source dataset, and to the new PL. You cannot name externally defined common decks with the Q parameter or the COMPILE directive. The PL Identifier Table determines the sequence unless the K option is used. Corrections to decks that are not specified with the Q parameter or by a COMPILE directive are not included.

In the first method shown, up to 64 decks can be specified. After all the input has been entered, unknown deck names are errors.
Q[d1:d2:::dn]
Q'='d1,d2,:::dj,dk,:::dn'

(continued) In the second method shown, single decks are
separated by commas, and ranges of decks are
separated by periods. After all the input
has been entered, unknown deck names are
errors. The maximum size of the string used
with the second method is 96 characters.
The two methods cannot be combined.

omitted Normal UPDATE mode. Decks specified by
COMPILE directives, modified decks, and
decks calling modified common decks are
written to compile and/or source datasets.
You cannot name externally defined common
decks with the COMPILE directive, and they
are not written to the source dataset.

options The following output options are available on the control
(keyword only) statement. (See subsection 1.10.3, Listing Options, for a
more detailed description of CD, ED, ID, IF, IN and UM.)

CD Writes the generation directives for the compile
dataset to ldn

ED Writes the edited line summary to ldn

ID Writes the identifier summary to ldn

IF Writes a conditional text summary to ldn

IN Lists the input to ldn

K Orders all decks that are written to the compile
dataset and to new PL datasets, as directed by the
Q parameter values on the control statement and the
COMPILE directives. This option is ignored in Full
or Normal mode.

NOTE

If a modification set affects two or more
decks and the K option is in effect, the
sequence numbers of inserted lines can be
inconsistent with sequencing that has
occurred without the K option.
options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>Does not abort if directive errors or modification errors occur. All requested datasets are generated.</td>
</tr>
<tr>
<td>NR</td>
<td>Does not rewind the source dataset or compile dataset at the beginning or end of UPDATE execution.</td>
</tr>
<tr>
<td>NS</td>
<td>Suppresses line sequence information in the source and compile datasets. SEQ and NOSEQ directives are ignored when this option is used. NS is ignored for variable-length record PLs.</td>
</tr>
<tr>
<td>OLDPL</td>
<td>Produces a PL in the old UPDATE format. (Applies only to NUPDATE).</td>
</tr>
<tr>
<td>OS</td>
<td>Assigns new sequence numbers using the UPDATE 4.0 scheme. See subsection 3.2.1, Line Identification, for a description of the differences between the two.</td>
</tr>
<tr>
<td>SQ</td>
<td>Provides sequence numbers for source and compile output. The defaults are sequence numbers on the compile output and no sequence numbers on the source output.</td>
</tr>
<tr>
<td>UM</td>
<td>Writes unprocessed modifications to the listing dataset and/or the error dataset.</td>
</tr>
<tr>
<td>SSQ</td>
<td>Provides sequence numbers for source output only, not compile output. (Applies only to NUPDATE).</td>
</tr>
</tbody>
</table>

2.1.1 EXAMPLES

Following are two examples illustrating the use of the COS control statement. Example 1 shows how a PL is created:

```
UPDATE,P=O,N=NEWPL,I=INPUT.
```

When P=O, no existing PL is specified, and the new PL is written to NEWPL. The input is read from INPUT. The compile output is written to $CPL; all decks are selected.

Example 2 shows how a PL can be modified:

```
UPDATE,P,I=MODS,Q=DECK3:DECK2:DECK4,K,NR,NA.
```

The parameters on this control statement indicate the following:

- P=$PL is implied.
- The input is read from MODS.
• The statement specifies Quick mode (using the Q parameter) with the K output option. If a single COMPILE directive is used (*COMPILE DECK1.DEC4), DECK1 through DECK4 are written to $CPL in the following order:

DECK3
DECK2
DECK4
DECK1

• UPDATE does not rewind $CPL before or after execution.

• If directive or modification errors occur, UPDATE does not abort.

2.2 NUPDATE CONTROL STATEMENT FOR COS

The COS NUPDATE control statement loads the NUPDATE program into the user field and begins execution. The NUPDATE statement is read from the control statement file of a user job deck. Use the parameters on the NUPDATE statement to specify the datasets to be used, the contents of the NUPDATE listing, and other features of the run.

Format:

```
NUPDATE[,P=pdn][,I=idx1:idx2:::idxn][,C=cdn]
[,DEF=def1:def2:::defn][,N=ndx1][,L=ldk][,E=edk][,S=sdn]
[,=n][,C=cw][,Dw=dw][,DC=dc][,ML=n]
[,F]
,Q='[d1],[d2],...,[dn]' [,options].
,Q='[d1],[d2],...,[dj],[dk]'...
```

P=pdn    Dataset name of the program library (PL)

If omitted or P, the input PL is $PL.
If P=pln, the input PL is pln.
If P=0, no PL is used; this is for only a creation run.

I=idx1:idx2:::idxn

Names of input datasets; these datasets contain the directives and text for the NUPDATE run. NUPDATE reads them in the order given. You can specify up to 64.
I=\text{idn}_1:\text{idn}_2:\cdots:\text{idn}_n \\
(continued) If I or omitted, the input dataset is the next file of $\text{SIN}$. 

If I=\text{idn}, the input dataset is a dataset with the name \text{idn}. 

If I=0, no input dataset is read (invalid for a creation run). 

If I=\text{idn}_1:\text{idn}_2:\cdots:\text{idn}_n, the first input dataset to be read is \text{idn}_1, the second is \text{idn}_2, and so on. 

C=cdn Name of the compile dataset; decks that are determined by the NUPDATE mode (F, Q, or Normal; see table 1-1) are written on the specified dataset. 

If C is omitted, the compile output is written to $\text{S CPL}$. 
If C=cdn, the compile output is written to dataset \text{cdn}. 
If C=0, no compile dataset is generated. 

DEF=def_1:def_2:\cdots:def_n The DEF parameter defines names to be used with an IF directive. You can use names consisting of up to 240 characters; if they are longer, only the first 240 characters are recognized. Defined names need not be different from deck names, common deck names, or modification identifiers, because they are known only in the NUPDATE run being processed. You may define up to 64 names with the control statement DEF parameter. 

N=ndn Name of the new PL dataset. The NUPDATE mode determines the contents of the new PL (see table 1-1). 

If omitted in a modification run, no new PL is written. 

If omitted in a creation run, the new PL is written to $\text{SNPL}$. 

If N, the new PL is written to $\text{SNPL}$. 

If N=ndn, the new PL is \text{ndn}. 

If N=0, no new PL is written. 

L=l\text{dn} Name of the listing dataset; this dataset receives the NUPDATE list output. 

If omitted or L, the list output is written to $\text{S OUT}$. 

2-10 SR-0013 K
L=ldn If L=ldn, the list output is written to the dataset named ldn.

If L=0, no list output is generated.

E=edn Name of the error listing dataset; this dataset contains ERRORS, WARNINGS, CAUTIONS, NOTES, and COMMENTS as requested by the ML parameter.

If omitted or E, the output is written to $OUT.
If E=edn, the output is written to edn.
If E=0, errors are written only to the listing dataset.

---

**NOTE**

If E and L specify the same dataset, L is honored and E is set to 0.

---

S=sdn Source dataset name; the NUPDATE mode determines the contents of this dataset (see table 1-1). You can use this dataset as the input for a subsequent creation run.

If omitted or S=0, no source output is generated.
If S=sdn, the source output is written to sdn.
If S, the source output is written to $SR.

*=m Master character; the first character of directives read from the input file or written to the source file. Invalid master characters are comma, period, colon, equal sign, and space. The keyword alone is invalid.

If omitted in a creation run, the master character for directives is *.

If omitted in a modification run, the master character is that used in the creation run for the PL.

If *=m, the master character for directives is m.

/=c Comment character; indicates a comment. The keyword alone is invalid.

If omitted, the comment character is /.
If /=c, the comment character is c.
**DW=dw**  

Data width value; the number of characters of data written to each line in the compile and source datasets. The \( dw \) range is 1 through 256.

If omitted, or if a DW with no value is specified in a creation run, columns 1 through 72 contain data; otherwise, columns 1 through \( dw \) contain data.

If DW=* in a creation run, variable-length records are written to the compile and source datasets.

In a modification run, if the PL was created with DW=*, only DW=*, DW, or no DW parameter is accepted, and NUPDATE continues to process variable-length text lines for the PL. In a modification run, if no value for DW is specified, or DW is unspecified, columns 1 through last\( dw \) contain data; last\( dw \) is the DW value specified when the PL was written. If DW=\( dw \), columns 1 through \( dw \) contain data.

The number of characters per line written to the new PL is a maximum of either \( dw \), pl\( dw \), or 80; pl\( dw \) is the number of characters per line in the existing regular PL. For variable-length record PLs, the same number of characters per line appears in the new PL as they did in the old PL.

---

**NOTE**

For regular PLs, sequence information is provided in the source and compile datasets, as follows:

When you omit the data width value, when DW is specified alone, or when you specify the data width value as \( dw \), the following is true: \( dw+1 \) through \( dw+8 \) contain an identifier, right-justified with leading spaces; \( dw+9 \) contains a period; and \( dw+10 \) through \( dw+15 \) contain a sequence number, left-justified with trailing spaces.

When you specify the data width value as \( Ldw \), the entire sequencing field of the compile/output is left-justified.
For variable-length record PLs, no sequence information appears in the compile dataset. Source dataset sequence information, if you request it, appears three spaces to the right of the end of the text line.

Sequence information appears by default in the compile dataset, but it must be requested in the source dataset.

**DC=dc**

Declared modifications option. This parameter ensures that modifications apply to the correct deck or common deck. Declaration of PL modifications may be required (see subsections 1.5.3, Modifying a Program Library, and 3.1.1, Modification Directives).

If DC is omitted or DC=OFF, the mod declaration is not required, but it is enforced if present.

If DC=ON or if DC stands alone, the mod declaration is required.

**ML=n**

Message level: highest level of severity for NUPDATE listing messages to be suppressed. For example, ML=2 allows CAUTION, WARNING, and ERROR messages to be printed to the listing or error datasets. If you omit the parameter or specify the keyword alone, the default is 3. (The ML parameter does not affect NUPDATE log file messages.) The following levels are available:

<table>
<thead>
<tr>
<th>Level</th>
<th>Severity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>COMMENT</td>
<td>Currently unused</td>
</tr>
<tr>
<td>2</td>
<td>NOTE</td>
<td>Information not related to errors</td>
</tr>
<tr>
<td>3</td>
<td>CAUTION</td>
<td>Possible error</td>
</tr>
<tr>
<td>4</td>
<td>WARNING</td>
<td>Probable error</td>
</tr>
<tr>
<td>5</td>
<td>ERROR</td>
<td>Fatal error</td>
</tr>
</tbody>
</table>

F, Q, or omitted

Full, Quick, or Normal NUPDATE run. This parameter determines the contents of the compile dataset, the source dataset, and the new PL (see table 1-1).

**F**

Full NUPDATE mode; all active lines are processed. The PL Identifier Table determines the sequence. No COMPIL directive is necessary.
Quick NUPDATE mode. Decks that are specified with the Q parameter and decks specified by a COMPILE directive are written to the compile dataset and/or the source dataset, and to the new PL. You cannot name externally defined common decks with the Q parameter or the COMPILE directive. The PL Identifier Table determines the sequence unless the K option is used. Corrections to decks that are not specified with the Q parameter or by a COMPILE directive are not included.

In the first method shown, up to 64 decks can be specified. After all the input has been entered, unknown deck names are errors.

In the second method shown, single decks are separated by commas, and ranges of decks are separated by periods. After all the input has been entered, unknown deck names are errors. The maximum size of the string used with the second method is 96 characters. The two methods cannot be combined.

When using a deck name longer than 8 characters, the name must be enclosed in quotation marks. Further, if you use a deck name containing a period, you must enclose it in brackets. To do a quick NUPDATE run of deck LONGNAME.C, you would enter the following:

```
Q="[LONGNAME.C]"
```

To do a range of two decks with periods in them, you would enter the following:

```
Q="[LONGNAME1.C].[LONGNAME2.C]"
```

Normal NUPDATE mode. Decks specified by COMPILE directives, modified decks, and decks calling modified common decks are written to compile and/or source datasets. You cannot name externally defined common decks with the COMPILE directive, and they are not written to the source dataset.
The following output options are available on the control statement. (See subsection 1.10.3, Listing Options, for a more detailed description.)

- **CD** Writes the generation directives for the compile dataset to *ldn*
- **ED** Writes the edited line summary to *ldn*
- **ID** Writes the identifier summary to *ldn*
- **IF** Writes a conditional text summary to *ldn*
- **IN** Lists the input to *ldn*
- **K** Orders all decks that are written to the compile dataset and to new PL datasets, as directed by the Q parameter values on the control statement and the COMPILE directives. This option is ignored in Full or Normal mode.

**NOTE**

If a modification set affects two or more decks and the K option is in effect, the sequence numbers of inserted lines may be inconsistent with sequencing that has occurred without the K option.

- **NA** Does not abort if directive errors or modification errors occur. All requested datasets are generated.
- **NR** Does not rewind the source dataset or compile dataset at the beginning or end of NUPDATE execution
- **NS** Suppresses line sequence information in the source and compile datasets. SEQ and NOSEQ directives are ignored when this option is used. NS is ignored for variable-length record PLs.
- **OLDPL** Generates an UPDATE PL rather than an NUPDATE PL
- **OLDSEQ** Assigns sequence numbers using the old scheme rather than the new (default). See subsection 3.2.1, Line Identification, for a description of the differences between the two.
options
(continued)

SQ Provides sequence numbers for source and compile output. The defaults are sequence numbers on the compile output and no sequence numbers on the source output.

UM Writes unprocessed modifications to the listing dataset and/or the error dataset.

2.3 UNICOS COMMAND LINE

Under UNICOS, the following command line executes UPDATE. Options on the UPDATE command line specify the PL file name, the compile file name, the new PL file name, and the source file name. UPDATE error messages go to stderr, and the listings go to stdout.

Format:

```
```

**-p pdf**  
PL file name. This file is the input PL. You must include the file name; `p` without an argument is invalid.

You can omit `p` only for a creation run because, when `p` is missing, the system reads no input PL.

You must specify either the `-p` or the `-i` option or both.

**-i idf**  
Input file name.

You must specify `-i` for creation runs. The `-i` option without an argument is invalid.

You can specify `-i` - (note the space between the `i` and the second hyphen) to have UPDATE read from standard input.

You must specify either the `-p` or the `-i` option or both.
-c cdf  Compile file name. For multiple compile files, the file name is appended with a period and a single letter as defined by default (.f, .s, .p, and .c) or as specified by the -a option. The default suffixes are assigned in the following order:

.f - Fortran  
.s - Assembly  
.p - Pascal  
.c - C

Do not allow the file name and the following period and suffix to exceed 14 characters (the input file name must not exceed 12 characters). The files must not exist because they will be created by UPDATE. If you specify the keyword alone, UPDATE looks for file names in the PL or input files with *FILE directives. See subsection 3.3.16, FILE, for further information.

-a arguments

arguments has the form x1 x2 ... xN, where each argument specifies the appended characters that are to be assigned to multiple compile files. Each character must be unique and must be one of the 62 alphanumeric characters. The -a option without an argument is invalid.

-u udf

User common deck file name. Called common decks that cannot be found in the PL being processed are searched for in the common deck PLs (udf) specified on the -u option. The search order is the order in which the libraries are specified. No modifications can be made to a common deck PL, and externally defined common deck text is not written to the source file or to the new PL. The common deck text appears in the compile file. You can use a maximum of 20 common deck file names; UPDATE ignores any beyond that number. The -u option without an argument is invalid.

-d ifp

Defines names to be used with an IF directive. Names can consist of up to 8 characters; if they are longer, UPDATE recognizes only the first 8 characters. If you specify more than one defined name, a space (rather than a comma or colon) is the separator character. Defined names need not be different from deck names, common deck names, or modification identifiers, because they are known only in the UPDATE run being processed. You can define up to 64 names with the command-line -d option; UPDATE ignores any beyond that number. The -d option without an argument is invalid.
-n ndf  New PL file name. The UPDATE mode determines the contents of the new PL (see table 1-1). If omitted in a modification run or a creation run, no new PL is written. If you specify -n ndf, the new PL is written to file ndf. For a normal UPDATE run, the -f and -q options must be omitted. The specified file must be empty, because UPDATE will create a new PL in it. The -n option without an argument is invalid.

-s sdf  Source file name. The UPDATE mode determines the contents of this file (see table 1-1). The source file can be the input for a subsequent creation run, but it must not exist on input, because it will be created by UPDATE. The -s option without an argument is invalid.

-x chr  Master character; the first character of directives read from the input file or written to the source file. Invalid master characters are comma, period, colon, equal sign, and space. If the character is a metacharacter (for example, an *), you must protect it in one of the following ways: \*, "*", or '('. The -x option without an argument is invalid.

    If omitted in a creation run, the master character for directives is *.

    If omitted in a modification run, the master character is that used in the creation run for the PL.

-y chr  Comment character that indicates a comment. If the character is a metacharacter (for example, *), you must represent it in one of the following ways: \*, "*", or '('. The -y option without an argument is invalid.

    If omitted in a creation run, the comment character is /.

    If omitted in a modification run, the comment character is that used in the creation run for the PL.

-w dwv  Data width value; the number of characters of data written to each line in the compile and source file. The dwv range is 1 through 256. The keyword alone is invalid.

    If omitted in a creation run, columns 1 through 72 contain data; otherwise, columns 1 through dwv contain data.
If -w '✱' is in a creation run, variable-length records are written to the compile and source files. (The asterisk is a special character and must be represented in one of the following ways: \*, "✱", or '✱').

In a modification run, if the PL was created with -w '✱', only -w '✱' or no -w option is accepted, and UPDATE continues to process variable-length text lines for the PL.

In a modification run with -w unspecified, columns 1 through lastdw contain data; lastdw is the -w value specified when the PL was written. If you specify -w dwv, columns 1 through dwv contain data.

The number of characters per line written to the new PL is the maximum of either the dwv, pldw, or 80; pldw is the number of characters per line in the existing regular PL. For variable-length record PLs, the same number of characters per line appears in the new PL as they did in the old PL.

---

NOTE

For regular PLs, sequence information is provided in the source and compile files as follows:

When you omit the data width value or specify it as dwv, dwv+1 through dwv+8 contain an identifier, right-justified with leading spaces; dwv+9 contains a period; and dwv+10 through dwv+15 contain a sequence number, left-justified with trailing spaces.

When you specify the data width value as Ldwv, the entire sequencing field of the compile output is left-justified.

For variable-length record PLs, no sequence information appears in the compile dataset. Source dataset sequence information, if you request it, appears three spaces to the right of the end of the text line.

Under UNICOS, sequence information does not appear by default in either the source or compile files; you must request it.
-m n  Message level: highest level of severity for UPDATE listing messages to be suppressed. For example, -m 2 allows CAUTION, WARNING, and ERROR messages to be printed to stdout. When -m is omitted, the default is 3. The -m option without an argument is invalid. The following levels are available:

<table>
<thead>
<tr>
<th>Level</th>
<th>Severity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>COMMENT</td>
<td>Currently unused</td>
</tr>
<tr>
<td>2</td>
<td>NOTE</td>
<td>Information not related to errors</td>
</tr>
<tr>
<td>3</td>
<td>CAUTION</td>
<td>Possible error</td>
</tr>
<tr>
<td>4</td>
<td>WARNING</td>
<td>Probable error</td>
</tr>
<tr>
<td>5</td>
<td>ERROR</td>
<td>Fatal error</td>
</tr>
</tbody>
</table>

-f, -q decks

Full, Quick, or Normal UPDATE run. This determines the compile file contents, the source file, and the new PL contents (see table 1-1).

-f  Full UPDATE mode; all active lines are processed. The PL Identifier Table determines the sequence. No COMPILE directive is necessary. The -f option must be used without an argument.

-q qf_1,qf_2,...,qf_n
-q' d_1,d_2,...,d_j,d_k,...,d_n'

Quick UPDATE mode. Decks that you specify with the -q option and decks specified by a COMPILE directive are written to the compile file and/or the source file, and to the new PL. These decks must be in the same case as they appear in the PL--whether uppercase, lowercase, or mixed. You cannot specify externally defined common decks with the -q option or the COMPILE directive. The PL Identifier table determines the sequence unless you employ the k option-argument. Corrections to decks that are not specified with the -q option or by a COMPILE directive are not included.

In the first method shown, you can specify up to 100 decks. After all the input has been entered, unknown deck names are errors. The -q option without an argument is invalid.
-q $q_{f1},q_{f2},...q_{fn}$
-q'd$^\prime_1,d_2,...,d_j,d_k,...,d_n'$

(continued) In the second method shown, single decks are separated by commas, and ranges of decks are separated by periods. After all the input has been entered, unknown deck names are errors. The maximum size of the string used with the second method is 96 characters. The two methods cannot be combined.

omitted Normal UPDATE mode. Decks specified by COMPILE directives, modified decks, and decks calling modified common decks are written to compile and/or source files. You cannot specify externally defined common decks with the COMPILE directive, and they are not written to the source file.

-o arguments

The following output arguments are available in the command line. Each argument must be separated by a space. (See subsection 1.10.3, Listing Options, for a detailed description.)

- cd  Writes the generation directives for the compile file to stdout
- dc  Declared modifications option. This argument ensures that modifications apply to the correct deck or common deck. Declaration of PL modifications may be required (see section 1 and subsection 3.3.17, on the IDENT directive).
- ed  Writes the edited line summary to stdout
- id  Writes the identifier summary to stdout
- if  Writes a conditional text summary to stdout
- in  Lists the input to stdout
- k  Orders all decks that are written to the compile file as directed by the -q option values on the command line and the COMPILE directives. This argument is ignored in Full and Normal modes.
-o arguments
(continued)

NOTE
If a modification set affects two or more decks and the \textit{k} argument is in effect, the sequence numbers of inserted lines can be inconsistent with sequencing that has occurred without the \textit{k} argument.

\textbf{na} Does not abort if directive errors or modification errors occur. All requested files are generated.

\textbf{ns} Suppresses line sequence information in the compile files. SEQ and NOSEQ directives are ignored when this argument is used. The \textbf{ns} argument has no effect on the source file output and is ignored for variable-length record PLs.

\textbf{op} Generates an UPDATE PL rather than an NUPDATE PL.

\textbf{os} Assigns sequence numbers using the old scheme rather than the new (default). See subsection 3.2.1, Line Identification, for a description of the differences between the two.

\textbf{sq} Provides sequence numbers for source and compile output. The default is no sequence numbers. For an example of the output, see subsection 4.17.1, Create a New PL from an Input File.

\textbf{um} Writes unprocessed modifications to \texttt{stdout}

2.3.1 EXAMPLES

Following are two examples illustrating the use of the UNICOS command line. Example 1 shows how a PL is created:

\begin{verbatim}
update -n newpl -i input -c cplfile
\end{verbatim}

The omitted \texttt{-p} option indicates that there is no existing PL. The new PL is written to file \texttt{newpl} and the input is read from \texttt{input}. The compile output is written to \texttt{cplfile.f}; all decks are selected.
Example 2 shows how a PL can be modified:

```
update -p ./plname -i mods -q DECK3,DECK2,DECK4 -o k na -c cplfile
```

The parameters indicate the following:

- The PL file is `plname` in the current working directory.
- The input is read from the `mods` file in the current directory.
- Quick mode with the `k` output option-argument. If you use a single `COMPILE` directive (*COMPILE DECK1.DECK4), DECK1 through DECK4 are written to the compile file (-c `cplfile`) in the following order:
  
  DECK3
  DECK2
  DECK4
  DECK1

- Because of the `na` option-argument, UPDATE does not abort if directive or modification errors occur.
- The compile output is written to `cplfile.f`. 

3. UPDATE DIRECTIVES

UPDATE tasks are specified by a set of directives that usually occupy a file of the input dataset, but they can reside on one or more separate datasets.

This section first describes five general categories of UPDATE directives, and then gives the general format and rules for using them. Subsection 3.3 describes the directives in alphabetical order. In this manual, * is used as the master character for directive descriptions (see subsections 2.1 and 2.2 for descriptions of master characters in COS and UNICOS, respectively). See section 1 to review the conventions this manual uses regarding UPDATE directives.

With UPDATE release versions 5.0 and higher, you can issue UPDATE directives in uppercase or lowercase. You cannot, however, issue the directives in mixed case. For example, both of the following are valid:

   *INSERT
   *insert

But using mixed case, as follows, is not valid:

   *Insert

3.1 CATEGORIES OF UPDATE DIRECTIVES

The UPDATE directives are grouped into five categories:

- Modification directives
- Input edit directives
- Run option directives
- Compile dataset directives
- Deck definition directives

See figure 3-1 for a summary grouping of all the UPDATE directives.
3.1.1 MODIFICATION DIRECTIVES

Modification directives modify text in the program library (PL) by adding new lines or changing the active status of existing lines. The changes made by these directives are associated with a modification set, and these are the only changes that a YANK, UNYANK, or PURGE directive will affect. The directives in this category are as follows:

- BEFORE
- COPY (form 1)
- DELETE
- IDENT
- INSERT
- RESTORE

The COPY directive has two forms. Form 1 copies text from a section of the PL to the input stream; and the added text is associated with a modification set if the COPY directive follows an insertion directive (BEFORE, DELETE, INSERT, or RESTORE). Form 2 is a run option directive, described in subsection 3.1.3, Run Option Directives. If a COPY directive follows a DECK or COMDECK directive, the new text is associated with that deck or common deck, but not with a modification set.
3.1.2 INPUT EDIT DIRECTIVES

Input edit directives make changes to a PL dealing with previous modifications and the order of decks. They are processed in the UPDATE run in which they are input, they are not saved in the PL, and they are not associated with a modification set. Directives in this category are as follows:

- EDIT
- ENDSKIP
- MOVEDK
- PURGE
- PURGEDK
- SKIP
- UNYANK
- YANK

3.1.3 RUN OPTION DIRECTIVES

Run option directives are processed in the UPDATE run in which they are input. They do not change the PL, they are not saved in the PL, and they are not associated with a modification set. Most run option directives specify input or output options. The comment is also included in this group. Directives in this category are as follows:

- /comment
- COMPIL
- COPY (form 2)
- DECLARE
- DEFINE
- LIST
- MASTER
- NOLIST
- READ
- REWIND (applicable only under COS)
- SKIPF (applicable only under COS)
- UNDEF (applicable with NUPDATE)

Form 2 of the COPY directive in this group copies text from the PL to a dataset specified by the user.

3.1.4 COMPIL DATASET DIRECTIVES

Compile dataset directives determine the contents and format of the compile dataset. These directives are embedded in the PL as lines that can be added or deleted just the same as text lines. The compile dataset directives are as follows:
3.1.5 DECK DEFINITION DIRECTIVES

The DECK and COMDECK directives specify whether a new deck is a regular or common deck. These two directives do not fit the categories described in the previous paragraphs. They insert new text into the PL and so are modification directives, but they are not associated with a modification set. DECK and COMDECK directives are stored in the PL. Unlike the compile dataset directives, you cannot delete them like text, and you cannot insert text before them.

3.2 DIRECTIVE FORMAT

A directive has the following syntax:

\[ m, d, p_1, p_2, p_3, \ldots \text{ comment} \]

- \( m \): Master character (an asterisk [*] in this manual)
- \( d \): Directive name or abbreviated name (shown only in uppercase in this manual)
- \( p_i \): Parameter, dependent on directive
- \( \text{comment} \): Optional comment

The first comma can be replaced by one or more spaces. If you use a comment, you must precede it with one or more spaces.

The underlined format of the directives specifies the abbreviation of the directive. With UPDATE 5.0, directives and abbreviations can be either uppercase or lowercase, but not mixed case. Parameters in brackets are optional.
3.2.1 LINE IDENTIFICATION

Each modification set and each deck (or common deck) has a unique identifier. This identifier is the name from the corresponding DECK, COMDECK, or IDENT directive.

The sequence number for a line from the original deck is derived from the line's position in the deck. New lines inserted by a modification set are sequenced in the order in which they will appear in the deck (which is not necessarily the same order in which they appeared in the input dataset).

A given line is uniquely referred to by \textit{id.seq}; \textit{id} is the deck or modification set identifier name, and \textit{seq} is the line sequence number.

Once a deck (or common deck) or modification set becomes a part of a new PL, \textit{id.seq} is permanent.

Unless you select the \textit{old} sequence number generation scheme by specifying the \texttt{-0 os} or \texttt{OLDPL} option on the UPDATE command line or control statement, the \textit{new} scheme is used.

The new numbering scheme assigns sequence numbers in the order that the directives are read from the mods file. The old system assigns numbers in the order that the directives are processed.

In the following example, MOD.2 will appear before MOD.1 using the new scheme, but MOD.1 will precede MOD.2 under the old scheme:

```
*IDENT MOD
*I DECK.2
  line2
  ...
*I DECK.1
  linel
```

The text files generated from this example, under both the old scheme and the new scheme, would appear as follows:

<table>
<thead>
<tr>
<th>Old</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECK.1</td>
<td>DECK.1</td>
</tr>
<tr>
<td>MOD.1</td>
<td>MOD.2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>DECK.2</td>
<td>DECK.2</td>
</tr>
<tr>
<td>MOD.2</td>
<td>MOD.1</td>
</tr>
</tbody>
</table>

Which scheme you use could become important if you do subsequent updates to these insertions.
3.2.2 IDENTIFIER NAMES

Each identifier (deck, common deck, defined names, or modification set name) is a 1- to 8-character name assigned when the identifier is first used. Names cannot include commas, periods, blanks, colons, or equal signs, but they can include any other character in the ASCII code range of 41₈ through 176₈ (see appendix A, Character Set).

On some directives, you can specify names as an inclusive range. When you specify a range of names, the parameter consists of the first identifier name in the range, a period, and the final identifier in the range.

Format:

```
|   | decknamefirst.decknamelast   |
```

Do not confuse periods as used in the preceding format line with periods that separate line sequence numbers from deck identifiers or from modification set identifiers (as described in subsection 3.2.1, Line Identification). The DELETE, RESTORE, and COPY directives require a comma to separate names that define a range in order to avoid ambiguity about the periods used for line sequence numbers.

3.2.3 DIRECTIVE FORMAT EXAMPLES

Examples of UPDATE directive syntax follow.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*BEFORE,X.23</td>
<td>The default master character (*) is used; line 23 of X is specified.</td>
</tr>
<tr>
<td>*EDIT DECK1.DECK2</td>
<td>The range from DECK1 through DECK2 is specified.</td>
</tr>
<tr>
<td>@D X.76,Y.79</td>
<td>The range from line 76 of X through line 79 of Y is specified; @ is the master character.</td>
</tr>
<tr>
<td>$/COMMENT</td>
<td>The master character is $, and / is the comment character.</td>
</tr>
<tr>
<td>$$/COMMENT</td>
<td>The $ is both the master and comment character.</td>
</tr>
</tbody>
</table>
3.3 DIRECTIVES

The set of directives that UPDATE recognizes follows. In this manual, * is used as the master character for directive descriptions (see subsections 2.1 and 2.2 for descriptions of master characters in COS and UNICOS, respectively).

3.3.1 / - COMMENT

A comment, indicated by the comment character, is copied to the list output.

Format:

```
| / comment |
```

/ Default comment character; any other comment character is specified on the UPDATE control statement.

3.3.2 BEFORE - INSERT BEFORE A LINE

BEFORE indicates that lines immediately following it are to be inserted before the line specified.

Format:

```
| *BEFORE id.seq |
```

id Deck or modification set identifier name

seq Line sequence number
3.3.3 CALL - CALL COMMON DECK

CALL indicates the location in which the specified common deck is to be placed when the compile dataset is generated. The combination of common decks and CALL directives lets you maintain a single copy of common text and be assured that the most current copy is always used in a deck that calls it.

You can also call common decks defined in a PL other than the one currently being processed if you use the COS CL parameter or UNICOS -u option. By doing so, you can maintain text that is common to more than one PL in one location. A common deck can contain CALL directives for other common decks but must not contain a CALL for itself. Indirect recursion is also prohibited; that is, if common deck A calls common deck B, common deck B is not allowed to call common deck A. The CALL directive is embedded in a deck, common deck, or input text, and it is assigned a sequence number accordingly.

Format:

```
*CALL cmdk
```

`cmdk` Common deck name

3.3.4 COMDECK - INTRODUCE A COMMON DECK

COMDECK introduces a common deck. Lines up to the next DECK, COMDECK, IDENT, INSERT, DELETE, BEFORE, RESTORE, or end of input compose the common deck. Other directives are interpreted but do not terminate the common deck.

The COMDECK directive is the first line of the common deck and is assigned sequence number 1.
Format:

```
| *COMDECK cmdk[,NOPROP] |
```

`cmdk` Common deck name

NOPROP No propagation parameter. If you specify NOPROP, this parameter indicates to UPDATE decks calling this deck that they are not to be automatically considered as modified whenever this common deck is modified. If you omit NOPROP, and the common deck is modified, all decks containing CALLS for this common deck are also considered modified.

3.3.5 COMPILE - SPECIFY COMPILE OR SOURCE DATASETS

COMPILE specifies the contents of the compile and/or source datasets. In selecting decks for compile output, you need not specify called common decks. Externally defined common decks cannot be specified. Generating source output requires all desired common decks to be specified.

COMPILE directives can occur anywhere in the input, but they must not refer to unknown decks—such as decks introduced later in the same run or, of course, misspellings.

Parameter order is significant only when the optional COS K parameter or UNICOS -k option is selected. Parameter order then specifies deck order for the compile and new PL datasets. The decks are otherwise written in the order in which they appear in the Identifier Table.

Format:

```
| *COMPILE p1[,p2,...,pj:pk,...,pn] |
```

`pi` Single deck name or a common deck name

`pj:pk` Range of deck names and/or common deck names
3.3.6 COPY - COPY TEXT

COPY has two forms for performing two different functions.

Form 1 copies text from the old PL for insertion into the new PL, as if it were in the input stream. In this form, COPY must be used with an insertion directive (INSERT, BEFORE, RESTORE, or DELETE) or be included in a new deck or common deck. Text is copied before any modifications are applied, so you can use this form of COPY to move text from one PL location to another if lines are deleted from their original location.

Form 2 copies text from the old PL into a separate dataset, which is specified on the directive. You can also add sequence information to the copied text if you use the SEQ keyword. The line length is the same as that for the source dataset.

Under UNICOS, dn can be a path name. COPY opens but does not remove the file. If dn exists, COPY appends information onto the end of it, and does not re-create it; otherwise, it creates file dn.

---

NOTE

Text copied for the COPY directive is from the old PL and does not include any modifications from the current UPDATE run.

---

Formats:

```
*COPY p, id1.seq1, id2.seq2
```

(form 1)

```
*COPY p, id1.seq1, id2.seq2, dn[,SEQ]
```

(form 2)

\[p\] Name of the deck or common deck from which text is to be copied

\[id_i\] Deck name or modification set identifier name

\[seq_i\] Line sequence number

\[id_1.seq_1\] First line in text range

\[id_2.seq_2\] Last line in text range
NOTE

The starting and ending line have no default values; that is, COPY p with no line numbers specified is invalid.

dn Under COS, the name of the dataset to which text is to be copied. Under UNICOS, the file name to which text is to be copied. This cannot be a file name already being used by UPDATE, as specified on the control statement.

SEQ Specified if sequence information is to be added to the text or copied to the dataset

3.3.7 CWEOF - CONDITIONALLY WRITE END-OF-FILE

Under COS, CWEOF directs UPDATE to write an end-of-file (EOF) to the compile dataset if the compile dataset was not positioned after an EOF or at the beginning of data. Under UNICOS, it directs UPDATE to close one compile file and open a different one if the file is not at the beginning of data. The new compile file has the same compile file name as the previous compile file, but it is appended with a period and character suffix, as described in subsection 2.2 (UNICOS -c and -a options).

CWEOF is embedded in a deck, common deck, or input text, and it is assigned an identifier and a sequence number. The CWEOF directive is ignored if you request no compile output. It is also ignored if you specify the -c option with no argument.

Format:

```
   *CWEOF
```

3.3.8 DECK - INTRODUCE A DECK

DECK introduces a new deck; it is the first line in the deck and has sequence number 1. The deck is composed of lines up to the next DECK, COMDECK, IDENT, INSERT, DELETE, BEFORE, RESTORE, or end of input. Input edit directives and run option directives are interpreted but do not terminate the deck. The new deck is ordered within the new PL following existing decks, common decks, and new common decks. Decks can contain embedded directives (for example, CALL, CWEOF, or WEOF).
3.3.9 DECLARE - DECLARE DECK FOR MODIFICATIONS

DECLARE requires that subsequent modifications be applied to the specified deck. This is one of two methods of declaring modifications (see subsection 1.9.3, Declared Modifications).

Format:

```
*DECLARE p  
```

p Deck or common deck name

3.3.10 DEFINE - DEFINE NAMES

DEFINE defines a name (1 to 8 characters in length) to be used by an IF directive. Names declared with this directive do not need to be unique from deck or common deck names or modification set identifiers. Defined names are known only in the run in which they are defined; they are not stored in the PL. However, under UNICOS the default is the literal name UNICOS.

If you are running UNICOS and are using an UPDATE version released prior to 6.0, use the -d option to undefine UNICOS as the default. This does not apply to *DEFINE.

Format:

```
*DEFINE n1[,n2,...,nn]  
```

n_i Defined name

If you are using NUPDATE, specify -d new_define_name, -U UNICOS, or *UNDEF UNICOS to undefine the default.
3.3.11 DELETE - DELETE LINES

DELETE lets you delete (deactivate) lines or ranges of lines and, optionally, replace them with lines appearing after the DELETE directive.

The DELETE directive copies a deleted line to the new PL. Though the line retains its identification, it is flagged as inactive and is not included in compile and source output. A deletion range must not cross a deck boundary.

Formats:

```
*DELETE id1.seq1, id2.seq2   (range delete)
*DELETE id1.seq1, seq2       (range delete, short form)
*DELETE id1.seq1             (single line delete)
```

\( id_i \)   Deck or modification set identifier name  
\( seq_i \)   Line sequence number

3.3.12 EDIT - EDIT DECKS

EDIT removes both deleted lines and lines made inactive by a YANK directive from the specified decks. When using the EDIT directive, you cannot recover removed lines from the PL. EDIT performs no resequencing. UPDATE edits only those decks noted explicitly on the EDIT directive.

---

**NOTE**

EDIT removes all lines that are inactive after all modifications in the current UPDATE run have been applied; this includes lines deleted in modifications that follow the EDIT directive in the input. EDIT does not remove lines in ranges that are restored by modifications following EDIT in the input.
Format:

```
*EDIT p_1, p_2, ... , p_j \cdot p_k, ... , p_n
```

$p_i$ Single deck or common deck

$p_j \cdot p_k$ Range of decks and/or common decks

### 3.3.13 ELSE - REVERSE CONDITION

ELSE reverses the condition from the previous IF or ELSEIF directive (unless the previous IF or ELSEIF has been skipped), to determine whether the text following it is written to the compile dataset. You cannot use ELSE without an IF. You can only use one ELSE with an IF, and ELSE must follow all ELSEIFs associated with that IF.

Format:

```
*ELSE
```

### 3.3.14 ELSEIF - TEST CONDITION

ELSEIF specifies a condition for evaluation when no previous condition in the same IF group was true. If the condition is evaluated as true, ELSEIF writes the text following the directive to the compile dataset, and skips the text following all further ELSEIF and ELSE directives in this IF group. If the condition is false or is not evaluated, it ignores all directives up to the next IF, ELSEIF, ELSE, or ENDIF. ELSEIF must have a matching IF, and it cannot follow ELSE; otherwise, an error message is issued.

Format:

```
*ELSEIF type, name[, ... , boolean, type, name]
```
**type**

Type of conditional name, either DECK, IDENT, or DEF. If DECK, the name must be a deck or common deck name; if IDENT, the name must be a modification set identifier; and if DEF, the name must have been introduced with the DEFINE directive, or the DEF parameter or -d option. A minus sign before the type negates the condition.

**name**

A deck or common deck name, modification set identifier, or defined name, depending on the value of type. Each clause of the condition is true if a name of the proper type is known or, when negated, if the name is unknown.

**boolean**

A logical operator: AND, OR, or XOR. AND has precedence over OR and is evaluated first. OR has precedence over XOR.

---

**3.3.15 ENDIF - END CONDITIONAL TEXT**

ENDIF ends a conditional text range and an IF group. Each ENDIF must have a matching IF; otherwise, you will get an error message.

Format:

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>*ENDIF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**3.3.16 FILE - CLOSE FILE (UNICOS ONLY)**

Under UNICOS, FILE directs UPDATE to close the current compile file, if there is one, and open a new one.

FILE is embedded in a deck or input text, and it is assigned an identifier and a sequence number. If you specify the -c option alone on the command line, UPDATE looks for file names in the PL or input files with *FILE directives. If you do specify a compile file on the -c option, or if you do not request a compile file, UPDATE ignores the FILE directive. If used, a FILE directive must be found in every deck processed before any code is written out. UPDATE aborts if it does not find one.

Under COS, FILE is accepted as a directive, but it has no function and is ignored.
Format:

```
*FILE path
```

path  A UNICOS path and file name to be opened. All directories in the path must exist and be accessible by the caller.

3.3.17 IDENT - IDENTIFY MODIFICATION SET

IDENT provides the modification set identifier that is to be associated with all of the changes in a modification set. IDENT is the first line in a modification set. When you are generating no new PL, IDENT is optional; the default identifier is *NOID*.

Format:

```
*IDENT [ident][,U=u1:u2:u3:...:un][,K=k1:k2:k3:...:kn][,DC=p]
```

ident  Modification set identifier

U=u_i  Unknown modification identifier; specifies an identifier dependency. This dependency is met if UPDATE cannot find any of the specified identifiers in its list of identifiers in the PL or among identifiers added earlier in the same UPDATE run. An identifier is unknown if it is not the name of a deck, common deck, or modification set that you already added to the PL.

K=k_i  Known modification identifiers; specifies a known identifier dependency. The dependency is met if UPDATE finds all the specified identifiers in its list of identifiers that are already in the PL or were added earlier in the same UPDATE run.

DC=p  Deck or common deck declared to contain lines referenced by subsequent modifications (see section 1). If all dependencies are met, p must be known to UPDATE. DC=. is equivalent to omitting the DC parameter.
The number of dependencies is limited only by what fits on the directive line. If not all dependencies are met, UPDATE skips all input up to the following IDENT directive or end of input, and the modification set identifier remains unknown. UPDATE writes a count of skipped IDENTs to the log file, and writes a warning to the listing and error datasets for each skipped IDENT.

Another way of writing U and K arguments is to include U= or K= for each argument, as follows:

\[ U = u_1, u_2, \ldots, K = k_1, k_2, \ldots \]

The following line provides an example:

\[ *IDENT \ MOD84, K = \MOD83, U = \MOD85 \]

This directive assigns identifier MOD84 to the modification set. UPDATE processes the modification set only if MOD83 is known and MOD85 is unknown.

3.3.18 IF - BEGIN CONDITIONAL TEXT

IF begins a conditional text range and gives the condition under which the range is written to the compile dataset. IF is the beginning of an IF group, which can include ELSEIF and ELSE directives and must end with an ENDIF directive. You can nest IF groups to any level.

If the condition is true, the text following the directive is written to the compile dataset, and the text following all ELSEIF and ELSE directives in the IF group is skipped. If the condition is false, all directives up to the next ELSEIF, ELSE, or ENDIF in the IF group are ignored. Skipped text and directives are written to the new PL and to the source dataset but not to the compile dataset.

Format:

```
*IF type, name[, ... , boolean, type, name]
```

\( type \) Type of conditional name: either DECK, IDENT, or DEF. If DECK, the name must be a deck or common deck name; if IDENT, the name must be a modification set identifier; if DEF, you must have introduced the name with the DEFINE directive, or the DEF parameter or -d option. A minus sign before the type negates the condition.
**name**  Deck or common deck name, modification set identifier, or defined name, depending on the value of *type*. Each condition is true if a name of the proper type is known; a negated condition is true if the name is unknown.

**boolean**  A logical operator: AND, OR, or XOR. AND has precedence over OR and is evaluated first; OR has precedence over XOR. The number of clauses is limited only by the length of the directive line.

### 3.3.19 INSERT - INSERT AFTER A LINE

**INSERT** indicates that the lines immediately following are to be inserted after the line specified.

**Format:**

```
*INSERT id.seq
```

- **id**  Deck or modification set identifier name
- **seq**  Line sequence number

### 3.3.20 LIST AND NOLIST - RESUME OR STOP LISTING

**LIST** and **NOLIST** resume the listing or stop the listing, respectively, of lines in the input stream. These directives can occur anywhere in the input, and they control the input listing, but they are otherwise ignored.

The **L=0 COS UPDATE** statement parameter overrides the **LIST** directive. The **NOLIST** directive overrides the **COS UPDATE** control statement option **IN**.

**Formats:**

```
*LIST
*NOLIST
```
3.3.21 MASTER - CHANGE INPUT MASTER CHARACTER

MASTER changes the master character for directives in the input. Directives stored in the PL use an unprintable code for the master character and are not affected by this directive. It also does not affect the master character for directives written to the source file.

Format:

```
*MASTER m
```

`m` New master character for directives in the input dataset(s)

3.3.22 MOVEDK - MOVE A DECK

MOVEDK causes UPDATE to move an entire deck from its present location to a point immediately following a specified destination deck. The sequencing information within the moved deck remains unchanged. The moved deck resides at the indicated point immediately after UPDATE has successfully processed this directive.

Format:

```
*MOVEDK dk1: dk2
  dk1: .
```

`dk1` Deck or common deck to be moved

`dk2` Destination deck or common deck; the new position of `dk1` is immediately after `dk2`.

`.` Specifies beginning of the PL
3.3.23  PURGE - REMOVE MODIFICATION SET

PURGE removes all text added in a modification set, restores all lines deleted by that set, and deletes all lines restored by it. PURGE starts with the last modification set listed, working in reverse order. PURGE is similar to YANK, but its effects are permanent; the affected lines and the identifier name are not written to the new PL. You can only purge modification sets added with a version of UPDATE from the COS 1.12 release or later.

Format (the two final periods are not ellipses):

```
| *PURGE id₁[,id₂,..,idₖ,..,idₙ..] |
```

- `idᵢ`  Modification set identifier
- `idⱼ.idₖ`  Inclusive range of modification set identifiers
- `idₙ..`  Identifier `idₙ` and all identifiers introduced after `idₙ`

3.3.24  PURGEDK - REMOVE DECK

PURGEDK permanently removes the deck from the PL.

Format:

```
| *PURGEDK dk |
```

- `dk`  Name of deck or common deck to be removed
3.3.25 READ - READ ALTERNATIVE INPUT

READ causes UPDATE to begin reading input from the specified dataset starting at its current position. An end-of-file on an input dataset causes UPDATE to resume reading from the previous dataset. READ directives can appear anywhere, but they must not be recursive. There is a limit of 20 levels of files that can be read.

Format:

```
  |    |
  | *READ dn |
  |    |
```

* dn Name of dataset. (Under UNICOS, it can be a directory path and file name, or just the file name when in the current directory.)

3.3.26 RESTORE - REACTIVATE LINES

RESTORE can add new text and restore (reactivate) lines or ranges of lines that have previously been deleted. The optional new text appears on lines immediately following RESTORE and is inserted following the reactivated line(s). A RESTORE range cannot cross a deck boundary.

Formats:

```
  |    |
  | *RESTORE id_1.seq_1,id_2.seq_2 (range restore) |
  | *RESTORE id_1.seq_1,seq_2 (range restore, short form) |
  | *RESTORE id_1.seq_1 (single line restore) |
```

* id_i Deck or modification set identifier name

* seq_i Line sequence numbers
3.3.27 REWIND - REWIND LOCAL DATASET

REWIND (applicable only under COS) rewinds a local dataset so that the dataset is repositioned at its initial point. If the dataset is already at its initial point, this directive has no effect.

Format:

```
*REWIND dn
```

*dn* Name of dataset to be rewound

3.3.28 SEQ AND NOSEQ - START OR STOP SEQUENCE NUMBER WRITING

SEQ and NOSEQ begin writing and stop writing, respectively, sequence numbers to the compile dataset. The compile dataset output contains *dw* data columns per record without sequence numbers or *dw+1* with sequence numbers. The DW parameter or -w option and the WIDTH directive determine the value of *dw*. The SEQ and NOSEQ directives are embedded in a deck, common deck, or input text, and they are assigned a sequence number. The SEQ and NOSEQ directives are ignored if you have not requested compile output, or if you have specified the NS control statement option (or the ns option-argument under UNICOS). These directives have no effect on variable-length record PLs.

Formats:

```
*SEQ

*NOSEQ
```
3.3.29 SKIP AND ENDSKIP - CONDITIONALLY SKIP A BLOCK OF DIRECTIVES

SKIP and ENDSKIP delimit a block of directives that are to be skipped or not skipped, depending on the conditions that you specified on the SKIP directive. You can nest SKIPS, but each SKIP must have a matching ENDSKIP. If an outer SKIP condition is true, SKIPS nested within that block and all other text and directives are skipped. You must associate SKIPS with a KNOWN and/or UNKNOWN identifier condition. Unconditional SKIPS result in an UPDATE error.

Format:

```
*SKIP [K=k_1:k_2:k_3:\ldots:k_n][,U=u_1:u_2:u_3:\ldots:u_n]
  conditional directives, text
  *ENDSKIP
```

- **K=k_i**  
  Known modification identifiers; specifies a known identifier dependency. The dependency is met if UPDATE finds all of the specified identifiers in its list of PL identifiers (deck names, common deck names, and previously added modification identifiers).

- **U=u_i**  
  Unknown modification identifiers; specifies an unknown identifier dependency. This dependency is met if UPDATE cannot find any of the specified identifiers in its list of PL identifiers.

The number of dependencies is limited only by what will fit on the directive line.

Another way of writing K and U arguments is to indicate K= or U= for each argument, shown as follows:

```
K=k_1,K=k_2,\ldots,U=u_1,U=u_2,\ldots
```
3.3.30 SKIPF - SKIP DATASET FILES

SKIPF (applicable only under COS) skips one or more files in a local dataset.

Format:

```
*SKIPF dn[,n]
```

- `dn` Name of dataset in which to skip a file
- `n` Number of files to skip; default is 1.

3.3.31 WEOF - WRITE END-OF-FILE

Under COS, WEOF causes UPDATE to write an end-of-file to the compile dataset. Under UNICOS, WEOF directs UPDATE to close one compile file and open a different compile file. The new compile file has the same compile directory path and file name as the previous compile file, but it is appended with a period and character suffix, as described in subsection 2.2 (UNICOS Command Line, -c and -a options).

UPDATE embeds WEOF in a deck, common deck, or input text, and assigns it a sequence number. WEOF is ignored if you have not requested compile output. It is also ignored if you use the -c option with no argument.

Format:

```
*WEOF
```
3.3.32 WIDTH - CHANGE LINE WIDTH IN COMPILe DATASET

WIDTH changes the number of data characters written to each line in the compile dataset.

Until a WIDTH directive is encountered in the PL, the data width you specified with the COS DW parameter, UNICOS -w option, or its default is used. The WIDTH directive is ignored if you have requested no compile output, or if a variable-length record PL is being processed.

WIDTH does not affect the number of characters stored for each line in the PL or written to the source dataset. If the value given with the WIDTH directive is greater than the number of characters stored in the PL, each line written to the compile dataset is padded with blanks. If the number is less, each line is truncated at the right.

Format:

|*WIDTH dw|

\[dw\] Data width; the number of characters of data written to each line in the compile dataset. Columns \[dw+1\] through \[dw+15\] contain spaces and sequencing information.
3.3.33 YANK AND UNYANK - DELETE OR RESTORE DECKS AND MODIFICATION SETS

YANK temporarily deletes (deactivates) a deck, common deck, or modification set from a PL, but only if the entity to be yanked or unyanked has been created by a version of UPDATE that supported the YANK directive. YANK deactivates all lines in a deck or common deck whether they are original to the deck or added later by a modification set.

UNYANK restores (activates) a deck, common deck, or modification set previously deactivated.

YANK and UNYANK start with the last modification set listed, and they work in reverse order. You can place both directives anywhere in the input. They can be within a modification set started by an IDENT directive, although they are not associated with any modification set.

Formats (the two final periods are not ellipses):

| *YANK id₁[,id₂,...,idj,idk,...,idₙ..] |
| *UNYANK id₁[,id₂,...,idj.idk,...,idₙ..] |

idᵢ  Deck, common deck, or modification set identifier name

idⱼ.idk  Inclusive range of identifiers

idₙ..  Identifier idₙ and all identifiers introduced after idₙ
4. EXAMPLES

This section presents examples of UPDATE operating under COS and UNICOS. See subsection 1.4, Definitions, for UPDATE definitions, and review subsection 1.2 on the conventions used in this manual.

4.1 CREATING A PROGRAM LIBRARY (PL)

The examples in this subsection show how to create a PL and a compile file. Examples are also presented to cover reading input from an alternate dataset, creating in the Full UPDATE mode, and calling common decks into user programs.

4.1.1 PL CREATION

This example shows the creation of PL PRLIB1. The PL consists of two decks: deck ABC and common deck XYZ. The input file used in the examples follows:

```plaintext
*DK ABC
dec ABC
*CDK XYZ
comdeck XYZ
```

COS JCL:

The P=O entry indicates that there is no existing PL; therefore, the default input file, $IN, is used. The default parameters $IN, $CPL, and $NPL files are assumed (see subsection 2.1, COS Control Statement).

```
UPDATE,P=O.
SAVE,DN=$NPL,PDN=PRLIB1.
```

UNICOS command:

The absence of the -p option indicates that there is no existing PL, and that this is a creation run. The input file is infile1.

```
update -n prlib1 -i infile1 -c cplfile
cft cplfile.f
segldr cplfile.o
cp infile mod2
```
4.1.2 CREATING A PL AND A COMPILe FILE

This example creates PL PRLIB1, and a compile file, which consists of deck ABC. The input file used in this example is as follows:

*DK ABC
dec ABC
*CDK XYZ
comdeck XYZ
*C ABC

COS JCL:

The default compile dataset ($CPL) is used, and input is read from $IN.

UPDATE, P=0.
SAVE, DN=$NPL, PDN=PRLIB1.
CFT, I=$CPL.

UNICOS command:

The compile file is indicated as cplfile. By default, an .f extension is appended to indicate the compile file name. See subsection 2.2, UNICOS Command Line (option -c), for detailed information. UPDATE reads input from infile.

update -n prlib1 -i infile -c cplfile
cft cplfile.f

4.1.3 READING INPUT FROM AN ALTERNATE DATASET

In this example, PL PRLIB3 is created from input that is read from both the regular input file and an alternate dataset. Dataset DS1 (or UNICOS file source/s3) is a previously created source dataset, which can contain any number of decks. The contents of the regular input file are as follows:

*READ DS1
*DK XYZ
dec XYZ

COS JCL:

ACCESS, DN=DS1.
UPDATE, P=0.
SAVE, DN=$NPL, PDN=PRLIB3.
UNICOS command:

The contents of the regular input file (infile) are as follows:

*READ source/s3
*DK XYZ
dek XYZ

update -n prlib3 -i infile -c cplfile

4.1.4 FULL UPDATE MODE

When creating a new PL, you can make UPDATE processing more efficient if you request Full mode processing. In this example, PL PRLIB4 is created with two regular decks and two common decks. The contents of the input file are as follows:

*CDK CA  
common deck CA  
*CDK CB  
common deck CB  
*DK A  
dek A  
*DK B  
dek B

COS JCL:

The F parameter indicates that Full mode processing should be used.

UPDATE,P=0,N,F.
SAVE,DN=$NPL,PDN=PRLIB4.

UNICOS command:

The -f option indicates that Full mode processing should be used.

update -i infile -n prlib4 -f -c cplfile
4.1.5 CALLING COMMON DECK INTO A USER PROGRAM

This example shows how you can call a common deck into a user program. The common input file consists of the following:

*COMDECK     COMXX
  ...
  ...

*DECK        TEMP
  IDENT TSTPROG
  ...
  ...

*CALL        COMXX
  ...
  ...

END

COS JCL:

UPDATE,I,P=0.
CAL,I=$CPL.
SEGLDR,GO.

UNICOS command:

The compile file is an assembly language program; therefore, it requires an "s" appended to it by using the -a option.

update -i infile -n prlib5 -c cplfile -a s
as cplfile.s
segldr cplfile.o

The load command (segldr) has no defaults; that is, you must specify the files to be loaded, or there will be no output. With CRAY X-MP computer systems, file names to be loaded must end in .o.

4.2 MODIFYING A PROGRAM LIBRARY

The examples in this subsection show how to generate a modified PL and how to test a modification set.
4.2.1 GENERATING A MODIFIED PL

The sample job that follows performs three functions:

- First, the job updates a previously generated library.
- Next, it creates a compile dataset containing decks A through C and any common decks they call.
- Finally, it generates an updated version of the PL.

The input file for this example is as follows:

*ID MOD1
*D A.15,A.20
   text lines

*I B.119
   text lines

*DK C
*C A,C
   Introduces new deck

COS JCL:

The highest edition number of PRLIB2 is accessed. The *C directive determines the compile dataset ($CPL) contents. This library is saved as the next edition of permanent dataset PRLIB2.

ACCESS,DN=$PL,PDN=PRLIB2.
UPDATE,N.
SAVE,DN=$NPL,PDN=PRLIB2. Saves as next higher edition number

UNICOS command:

Under UNICOS, the new version of prlib2 is in file name prlib3.

update -p prlib2 -n prlib3 -i infile -c cplfile
4.2.2 TESTING A MODIFICATION SET

The following job tests modification set MOD2. The changes are not permanently incorporated into the library; that is, no new PL is generated and saved. Under COS, the UPDATE list options are turned off. Under UNICOS, the listings go to stdout. The *C directive determines the contents of the compile dataset. The input file used in the examples is as follows:

*ID MOD2
*ID MOD1.2
*ID MOD1.2
*D MOD1.2
text lines
*B C.3
text lines
*C A.C

Replaces line 2 of MOD1 modifications
Inserts lines before line 3 of deck C
Writes decks A, B, and C and any common decks they call to compile dataset

COS JCL:

ACCESS,DN=$PL,PDN=PRLIB2.
UPDATE,L=0,C=COMPILE,F.
CFT,I=COMPILE.
SEGLDR,GO.
REWIND,DN=$IN.
COPYF,I=$IN,O=MOD2.
REWIND,DN=MOD2.
SAVE,DN=MOD2.

Accesses highest edition number

UNICOS command:

update -p prlib2 -c cplfile -i infile -f

Saves modification set MOD2
4.3 GENERATING AND USING COMMON DECK PROGRAM LIBRARIES

The following examples show how to generate a common deck PL, a PL that calls external common decks, and how to modify the PL consisting of the external common decks.

4.3.1 GENERATING A PL WITH COMMON DECKS

The following job generates a PL that contains common deck definitions that are accessed later by other PLs. The input file used in this example is as follows:

*COMDECK COM1
com1 text
*COMDECK COM2
com2 text

COS JCL:

UPDATE,P=O,N=COMPL.
SAVE,DN=COMPL.

UNICOS command:

update -n compl -i infile

4.3.2 GENERATING A PL WITH EXTERNAL COMMON DECKS

The following job creates a PL containing decks that call externally defined common decks from PL COMPL. The input file used in the examples is as follows:

*DECK D1
D1 text
*CALL COM1
D1 text
*DECK D2
D2 text
*CALL PLCOM
D2 text
*COMDECK PLCOM
PLCOM text
*CALL COM2
COS JCL:

ACCESS,DN=COMPL.
UPDATE,P=0,N=PL1,CL=COMPL.
SAVE,DN=PL1.

UNICOS command:

update -n pll -u compl -i infile -c cplfile

4.3.3 MODIFYING A PL WITH EXTERNAL COMMON DECKS

The following job modifies the PL generated in the last example. The input file used in the examples is as follows:

*ID MOD,DC=D2
*D D2.3
new text to be inserted

COS JCL:

ACCESS,DN=COMPL.
ACCESS,DN=PL1.
UPDATE,P=PL1,N=PL2,CL=COMPL.
SAVE,DN=PL2.

UNICOS command:

update -p pl1 -n pl2 -u compl -i infile -c cplfile
4.4 READ FROM ALTERNATIVE DATASETS

The following job reads from alternate datasets MOD2 and DECK (mod2 and deck) as well as the regular input file. The input file for this example is as follows:

*READ MOD2 (mod2)
*READ DECK (deck)

Dataset MOD2 contains modification set MOD2 from the previous example
Contents of dataset DECK:
  *DK D
dock D
  *CDK CC
    common deck CC

*D C.12,C.16
*B B.17

deletes lines 12 through 16 of deck C
inserts lines before line 17 in
desk B

text lines
*CDK CD
  common deck CD

COS JCL:

The Q parameter on the control statement determines the compile dataset contents.

ACCESS,DN=MOD2,PDN=MOD2.
ACCESS,DN=DECK,PDN=DECK.
ACCESS,DN=$PL,PDN=PRLIB1.
UPDATE,Q=A:B:C:D,N.
  CAL,I=$CPL.
  SEGLDR,GO.
  SAVE,DN=$NPL,PDN=PRLIB1.

Accesses latest edition number
Saves next higher edition

UNICOS command:

The -q option on the UPDATE command line determines the compile dataset contents. The -a parameter identifies the compile file contents as assembly language and appends character s (cplfile.s). The compile file is then input to the assembler (as).

update -q A,B,C,D -c cplfile -p plib -n nplib -i infile -a s
as cplfile.s
segldr cplfile.o

Assembles compile file
Loads program

You must follow the load command (segldr) with the name of the program or file to be loaded, or there will be no output. With CRAY X-MP computer systems, the name of the program or file to be loaded must end in .o.

SR-0013 K 4-9
4.5 INPUT DATASET NOT $IN

The following example is specific to COS. The job adds to the PL a common deck named CE and replaces lines of code in an existing deck with a call to CE. The input stream is on dataset UPIN. No compile dataset is generated.

```
ACCESS,DN=$PL,PDN=MYUPDATEPL.
ACCESS,DN=UPIN,PDN=MYUPIN.
UPDATE,N,C=0,I=UPIN.
SAVE,DN=$NPL,PDN=MYUPDATEPL.
```

The following are contents of directives dataset MYUPIN:

```
*ID MOD3
*CDK CE
  common deck CE
*D C.25,C.463
*CALL CE
```

Adds common deck CE

CALL directive inserted as text in place of deleted lines

4.6 MULTIPLE INPUT DATASETS

The following jobs use multiple input datasets. All of the runs have the equivalent effect on the PL.

4.6.1 COS INPUT DATASET

You can use both the COS I control statement parameter and the READ directive to specify multiple input datasets. They can be used alone or together.
1. ACCESS,DN=MOD1.
   ACCESS,DN=MOD2.
   ACCESS,DN=MOD3.
   UPDATE,P=PL,I=MOD1:MOD2:MOD3.

2. ACCESS,DN=MOD1.
   ACCESS,DN=MOD2.
   ACCESS,DN=MOD3.
   UPDATE,P=PL.
   /EOF
   *READ MOD1
   *READ MOD2
   *READ MOD3

3. ACCESS,DN=MOD1.
   ACCESS,DN=MOD2.
   ACCESS,DN=MOD3.
   UPDATE,P=PL,I=$IN:MOD3.
   /EOF
   *READ MOD1
   *READ MOD2

4.6.2 UNICOS INPUT DATASET

You can use both the UNICOS -i command line option and the READ directive to specify multiple input datasets.

1. cat mod1 mod2 mod3 | update -p plfile -i - -n nplfile -c cplfile

2. update -p plfile -i infile2 -n nplfile -c cplfile
   infile2 contains:
   *READ /usr/dir/mod1
   *READ /usr/dir/mod2
   *READ /usr/dir/mod3

3. cat infile3 /usr/dir/mod3 | update -p plfile -i - -n nplfile -c cplfile
   infile3 contains:
   *READ /usr/dir/mod1
   *READ /usr/dir/mod2
4.7 EXTRACTING DECKS FOR A SOURCE DATASET

The following job does not change the PL but extracts selected decks, and any decks they call, for compilation and inclusion on a source dataset. The master character is specified as $, which was used when PL FPL was created.

COS JCL:

ACCESS,DN=FPL.
UPDATE,P=FPL,*=$,S,Q=SQRT:TANH:SIN,I=0.
CAL,I=$CPL.
SAVE,DN=$SR,PDN=SRFILE.

UNICOS command:

The source file is indicated as srfile. You must specify the PL master character within single quotation marks because it is a UNICOS special character. The assembly language compile file is created in cplfile.s.

update -p plfile -x '$' -s srfile -q SQRT,TANH,SIN -c cplfile -a s as cplfile.s

4.8 GENERATING A COMPILE DATASET FROM SOURCE

UPDATE generates compile datasets from source datasets without writing a PL. This can be useful when you need common information in several places and UPDATE is being used to expand common decks.

COS JCL:

ACCESS,DN=SRFILE.
UPDATE,I=SRFILE,C=COMPILE,P=0,N=0,F.
SAVE,DN=COMPILE.

UNICOS command:

Because the -n option is omitted, no new PL is written.

update -i srfile -c cplfile -f
4.9 **COMPILE DATASET FROM A COMMON DECK**

A common deck is not written to the compile dataset unless it is called by a deck written to the compile dataset. If you add a dummy deck that calls the common deck, you can include a common deck that would not otherwise be written to the compile dataset. The input file used in this example is as follows:

```
*DECK DUMMY
*CALL COM01
```

**COS JCL:**

```
ACCESS,DN=MYPL.
UPDATE,P=MYPL,C=COM01,F.
SAVE,DN=COM01.
```

**UNICOS command:**

```
update -p mypl -c com01 -i infile -f
```

4.10 **EXTRACTING DECKS FOR COMPILATION (NO SOURCE)**

Generating a compile dataset that is a subset of the PL decks is a common task. You do not need an input dataset. The following jobs show the easiest way to perform the task. Decks ST, CT, E, J, and S are written to the compile file ($CPL with COS, or cplfile.s with UNICOS) and are ready for assembly.

**COS JCL:**

```
ACCESS,DN=$PL,PDN=COSPL.
UPDATE,I=0,Q=ST:CT:E:J:S.
```

**UNICOS command:**

```
update -p plfile -q ST,CT,E,J,S -c cplfile -a s
```
4.11 USING *FILE (UNICOS ONLY)

FILE can operate only under UNICOS. The following example shows the use of FILE. Note that on the UNICOS command line, the -c option is not given a file name. Because a file was not specified, UPDATE assumes that the input contains file directives, and it proceeds to open file comp1.f. (If there had been a compile file specified on the -c option, UPDATE would have ignored the FILE directive.) It closes file comp1.f when it encounters a new one—in this case, file comp2.s.

input:
*dk comp1
*file comp1.f
deq comp1
*DK comp2
*FILE comp2.s
deq comp2

UNICOS command:
update -i input -c -f
cft comp1.f
as comp2.s

4.12 RESEQUENCING A PL

You can resequence a PL by first generating a source dataset from a full update of the old PL, and then using it as input to an UPDATE creation run. The new PL has the same decks and common decks as the old PL, but all modification history information is gone.

COS JCL:

ACCESS,DN=$PL,PDN=OLDPL.
UPDATE,F,S,I=0.
UPDATE,P=0,I=$SR.
SAVE,DN=$NPL,PDN=NEWPL.

UNICOS command:

update -f -s srfile -p plfile
update -i srfile -n nplfile
4.13 DECK REMOVAL AND POSITIONING

The following job shows the PURGEDK and MOVEDK directives in a typical application. The jobs replace deck B with a new version in the same position relative to decks A and C. The COS PL MYUPDATEPL and the UNICOS PL myupdatepl currently consist of the following decks:

CDK CA
CDK CB
DK A
DK B
DK C
CDK CC
CDK CD
DK D
CDK CE

The contents of the input file consist of the following:

*PURGEDK B
*DK B
text lines
*MOVEDK B:A

COS JCL:

ACCESS,DN=$PL,PDN=MYUPDATEPL.
UPDATE,L=0,C=0,N,F.
SAVE,DN=$NPL,PDN=MYUPDATEPL.

UNICOS command:

update -p myupdatepl -n nplfile -i infile -f
4.14 PL EDITING

As modifications to a PL accumulate, you can reduce the size of the PL by permanently removing deactivated lines from one or more decks. This process is known as editing. The EDIT directive specifies a deck or group of decks to be edited. The input file used in this example follows:

*EDIT A.D
*EDIT CA.CD

Edits decks A through D
Edits common decks CA through CD

COS JCL:

ACCESS,DN=$PL,PDN=MYUPDATEPL.
UPDATE,C=0,N.
SAVE,DN=$NPL,PDN=MYUPDATEPL.

UNICOS command:

update -p myupdatepl -n nplfile -i infile

4.15 CHANGING THE DATA WIDTH

The following UPDATE runs first create a PL and then build several new versions of that PL with varying data widths. The number of characters stored for each line in the new PL is never less than the number of characters per line in the old PL. Therefore, if the data width is decreased later, no characters are lost. You can change the data width of the compile dataset by using the WIDTH directive, the COS DW parameter, or the UNICOS -w option.

<table>
<thead>
<tr>
<th>COS Control Statement</th>
<th>Characters per Line</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Compile Dataset:</td>
</tr>
<tr>
<td>UPDATE,P=0,N=PL1.</td>
<td>72</td>
</tr>
<tr>
<td>UPDATE,P=PL1,N=PL2,DW=80.</td>
<td>80</td>
</tr>
<tr>
<td>UPDATE,P=PL2,N=0.</td>
<td>80</td>
</tr>
<tr>
<td>UPDATE,P=PL2,N=PL3,DW=116.</td>
<td>116</td>
</tr>
<tr>
<td>UPDATE,P=PL3,N=PL4,DW=112</td>
<td>112</td>
</tr>
<tr>
<td>UPDATE,P=PL4,N=PL5.</td>
<td>112</td>
</tr>
</tbody>
</table>
### 4.16 CONDITIONAL TEXT

Conditional text directives are used for text and directives that are always in the PL, but they are not always used to generate the compile dataset. The conditions that are tested can be either permanent or temporary. They are permanent if they test the existence of decks or modification set identifiers in the PL, and they are temporary if they check for defined names that are known only in the UPDATE run in which they are defined.

In the following example, deck SUBX contains text that is written to the compile dataset only if one of the names STACK and MULTI has been defined in that UPDATE run.

```plaintext
*DECK SUBX
unconditional text
*IF DEF,STACK,OR,DEF,MULTI
conditional text if either STACK or MULTI is defined
*IF DEF,MULTI
conditional text only if MULTI is defined
*ENDIF
conditional text if either STACK or MULTI is defined
*ELSE
conditional text if neither STACK nor MULTI is defined
*ENDIF
unconditional text
```

When neither STACK nor MULTI is defined, the following compile dataset is written:

```plaintext
unconditional text 
SUBX.2
conditional text if neither STACK nor MULTI is defined 
SUBX.10
unconditional text 
SUBX.12
```

SR-0013 K
When STACK is defined, the following compile dataset is written:

- unconditional text
- conditional text if either STACK or MULTI is defined
- conditional text if either STACK or MULTI is defined
- unconditional text

When MULTI is defined, the following compile dataset is written:

- unconditional text
- conditional text if either STACK or MULTI is defined
- conditional text only if MULTI is defined
- conditional text if either STACK or MULTI is defined
- unconditional text

You can use conditional text directives to surround compile dataset directives whose use is conditional. For example, if you do not want sequencing information for some decks in the compile datasets for a PL, you can use directives in conditional text ranges to turn off the sequencing.

The following is the source dataset for such a PL:

```
*DECK A
*IF -DEF,SEQ
*NOSEQ
*ENDIF
  text for deck A
*SEQ
*DECK B
  text for deck B
*DECK C
*IF -DEF,SEQ
*NOSEQ
*ENDIF
  text for deck C
*SEQ
```

If the name SEQ is not defined in an UPDATE run, the following compile dataset is written:

```
  text for deck A
  text for deck B
  text for deck C
```

The name SEQ is defined by adding directive *DEFINE SEQ to the input or specifying DEF=SEQ on the COS control statement, or -d SEQ on the UNICOS command line. When the name SEQ is defined, the following compile dataset is written:

```
  text for deck A
  text for deck B
  text for deck C
```

4-18
4.17  EXAMPLES SHOWING DATASET CONTENT

The following examples show the contents of the input, compile, and source datasets for several typical UPDATE runs.

4.17.1  CREATE A NEW PL FROM AN INPUT FILE

This example creates a new PL from text and directives in $IN or infile.

COS JCL:

UPDATE,P=0,N=PL1.

UNICOS command:

update -i infile -n pl1 -c cplfile1 -o sq

The compile dataset ($CPL under COS, and cplfile1.f under UNICOS) consists of the following:

<table>
<thead>
<tr>
<th>PROGRAM EXAMPLE</th>
<th>EXAMPLE.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>REAL A(10),B(10)</td>
<td>BLOCK1.2</td>
</tr>
<tr>
<td>COMMON /BLOCK1/ A,B</td>
<td>BLOCK1.3</td>
</tr>
<tr>
<td>READ *,A,B</td>
<td>EXAMPLE.4</td>
</tr>
<tr>
<td>CALL DIVIDE</td>
<td>EXAMPLE.5</td>
</tr>
<tr>
<td>WRITE *,A,B</td>
<td>EXAMPLE.6</td>
</tr>
<tr>
<td>STOP</td>
<td>EXAMPLE.7</td>
</tr>
<tr>
<td>END</td>
<td>EXAMPLE.8</td>
</tr>
<tr>
<td>SUBROUTINE DIVIDE</td>
<td>DIVIDE.2</td>
</tr>
<tr>
<td>REAL A(10),B(10)</td>
<td>BLOCK1.2</td>
</tr>
<tr>
<td>COMMON /BLOCK1/ A,B</td>
<td>BLOCK1.3</td>
</tr>
<tr>
<td>DO 100 I=1,10</td>
<td>DIVIDE.4</td>
</tr>
<tr>
<td>A(I)=A(I)/B(I)</td>
<td>DIVIDE.5</td>
</tr>
<tr>
<td>100 CONTINUE</td>
<td>DIVIDE.6</td>
</tr>
<tr>
<td>RETURN</td>
<td>DIVIDE.7</td>
</tr>
<tr>
<td>END</td>
<td>DIVIDE.8</td>
</tr>
</tbody>
</table>
4.17.2 PL MODIFICATION

You can modify the PL in dataset PL1 with the following statements:

**COS JCL:**

```
UPDATE,P=PL1,N=PL2,F.
```

**UNICOS command:**

```
update -p pl1 -n pl2 -f -c cplfile2 -i infile2
```

The following input dataset is read from the next COS file of $IN or from the UNICOS file `infile2`:

```
*IDENT MOD1,DC=DIVIDE
*    - Modifies subroutine DIVIDE in deck DIVIDE
*/
*BEBEFORE DIVIDE.5
   IF (B(I).NE.0) THEN
*INSERT DIVIDE.5
   ELSE
      A(I)=0
   ENDIF
*/
*IDENT MOD2A,DC=BLOCK1
*    - Modifies common deck BLOCK1 in common deck BLOCK1
*/
*DELETE BLOCK1.2
   PARAMETER (SIZE=5)
   REAL A(SIZE),B(SIZE)
*/
*IDENT MOD2B,DC=DIVIDE
*    - Modifies subroutine DIVIDE in deck DIVIDE
*/
*DELETE DIVIDE.4
   DO 100 I=1,SIZE
*IDENT MOD3A,DC=EXAMPLE
*I EXAMPLE.2
   LOGICAL IA
```

```
```
*B EXAMPLE.4

IF (IA()) WRITE *, 'Enter ', SIZE, ' values for X and Y:
*IDENT MOD3B, DC=.
*/
*/ - Adds a new subroutine written in CAL
*/
*DECK EOF1
*CWEOF
*DECK IA

IDENT IA
*
* Returns true if interactive, false if batch
*
IA ENTER NP=0
GET, S1 S6&S7, JCIA, A0
S1 S1<D'63
EXIT
END

Under COS, the following compile dataset is again in $CPL. Under UNICOS, there are two new compile files. The first file generated (cplfile2.f) contains the Fortran source code that has the default suffix of .f. The second file generated is appended with suffix .s for assembly language.

PROGRAM EXAMPLE
LOGICAL IA
PARAMETER (SIZE=5)
REAL A(SIZE), B(SIZE)
COMMON /BLOCK1/ A, B
IF (IA()) WRITE *, 'Enter ', SIZE, ' values for X and Y:
READ *, A, B
CALL DIVIDE
WRITE *, A, B
STOP
END
SUBROUTINE DIVIDE
PARAMETER (SIZE=5)
REAL A(SIZE), B(SIZE)
COMMON /BLOCK1/ A, B
DO 100 I=1, SIZE
IF (B(I).NE.0) THEN
   A(I)=A(I)/B(I)
100 CONTINUE
RETURN
END

eof (COS) or start of file cplfile2.s (UNICOS)
Having the compile dataset divided into more than one file is useful when a single PL contains subroutines written in more than one language. You can use a single UPDATE to generate input to more than one language processor.

**COS JCL:**

UPDATE,P=PL2,I=0,F.
CFT,I=$CPL,L=0.
CAL,I=$CPL,L=0.
SEGDR,GO.

**UNICOS command:**

The first compile file would contain Cray Fortran (CFT) subroutines and be named `cplfile3.f`. The second file would contain assembly subroutines and be named `cplfile3.s`. The UNICOS generation job for program EXAMPLE follows:

```
update -p plfile2 -f -c cplfile3
cft cplfile3.f
as cplfile3.s
segldr cplfile3.o
```
You can create a resequenced version of the PL by generating a source dataset from PL2 and using that as input to UPDATE in a creation run.

The contents of the source dataset used in this example are as follows:

*COMDECK BLOCK1
PARAMETER (SIZE=5)
REAL A(SIZE),B(SIZE)
COMMON /BLOCK1/ A,B

*DECK EXAMPLE
PROGRAM EXAMPLE
LOGICAL IA

*CALL BLOCK1
  IF (IA()) WRITE *,'Enter ','SIZE,' values for X and Y:'
  READ *,A,B
  CALL DIVIDE
  WRITE *,A,B
  STOP
END

*DECK DIVIDE
SUBROUTINE DIVIDE

*CALL BLOCK1
  DO 100 I=1,SIZE
  IF (B(I).NE.0) THEN
    A(I)=A(I)/B(I)
  ELSE
    A(I)=0
  ENDIF
  100 CONTINUE
RETURN
END

*DECK EOF1
*CWEOF
*DECK IA

IDENT IA

* Returns true if interactive, false if batch
*
IA ENTER NP=0
GET,S1 S66S7,Jcia,A0
S1 S1<63
EXIT
END
COS JCL:

UPDATE,P=PL2,F,I=0,S=SOURCE.
UPDATE,P=0,N=PL3,I=SOURCE.

UNICOS command:

update -p plfile2 -f -s source
update -n plfile3 -c cplfile4 -i source

The compile dataset from the resequenced PL is as follows:

```plaintext
PROGRAM EXAMPLE
LOGICAL IA
PARAMETER (SIZE=5)
REAL A(SIZE),B(SIZE)
COMMON /BLOCK1/ A,B
IF (IA()) WRITE *, 'Enter ',SIZE,' values for X and Y:'
READ *,A,B
CALL DIVIDE
WRITE *,A,B
STOP
END
SUBROUTINE DIVIDE
PARAMETER (SIZE=5)
REAL A(SIZE),B(SIZE)
COMMON /BLOCK1/ A,B
DO 100 I=1,SIZE
IF (B(I).NE.0) THEN
   A(I)=A(I)/B(I)
ELSE
   A(I)=0
ENDIF
100 CONTINUE
RETURN
END
```

eof  start of cplfile4.s

IDENT  IA
* Returns true if interactive, false if batch
* IA
  ENTER   NP=0
  GET,S1  S6&S7,JCIA,A0
  S1      S1<D'63
  EXIT
  END
5. AUDPL - PROGRAM LIBRARY AUDIT UTILITY

If you want information about program libraries (PLs) written by UPDATE, use the program library audit utility (AUDPL). From an input PL dataset, AUDPL lists specified text lines from decks and common decks, lists changes made to text lines, gives a summary of specified aspects of a PL, and writes reconstructed modification sets to a modification sets dataset. AUDPL makes no changes to a PL.

5.1 RESTRICTIONS

You can use AUDPL to process PLs written by a version of UPDATE from release 1.13 or higher. PLs written by an earlier UPDATE must be rewritten by a new version of UPDATE before they can be processed by AUDPL.

The PULLMOD directive and PM control statement parameter work correctly only for modification sets added by an UPDATE from release 1.12 or higher. Earlier modification sets do not have valid modification histories in the PL. Insertions can be reconstructed for these modification sets, but not deletions.

AUDPL cannot read a PL from a magnetic tape dataset.

If an UPDATE EDIT directive has removed lines deleted by the modification set, reconstructed modification sets will not be complete.

Modification sets that you added before a PL was resequenced cannot be reconstructed.

5.2 AUDPL CONTROL STATEMENT (COS)

The AUDPL control statement loads the AUDPL program into the user field and begins execution. The AUDPL statement is put in the control statement file of a user job. Parameters on the AUDPL statement specify the datasets to be used, contents of the AUDPL listing, and other features of the run.
Format:

```
AUDPL [,P=pln][,I=ldn][,L=ldn][,M=mdn] [,*,=m] [,/=c]
[DW=dlw],[LW=ldw],[JU=ju]
,DK=dk1:dk2:...:dkn
,DK='dk1,dk2,...,dkj,dkk,...,dkn'
,PM=pl1:pl2:...:pln
,PM='pl1,pl2,...,plj,plk,...,pln'

P=pln    Program library dataset name
        If omitted or P, the PL is $PL.
        If P=pln, the PL is pln.
        P=0 is invalid.

I=ldn    Input dataset name; this dataset contains the directives
        for the AUDPL run.
        If I, the input dataset is the next file of $IN.
        If I=ldn, the input dataset is ldn.
        If omitted or I=0, no input dataset is read.

L=ldn    Listing dataset name; this dataset receives the AUDPL list
        output.
        If omitted or L, the list output is written to $OUT.
        If L=ldn, the list output is written to ldn.
        If L=0, no list output is generated.

M=mdn    Modifications dataset name; this dataset receives
        reconstructed modification sets as selected by the PM
        control statement option or the PULLMOD directive.
        If omitted or M, the modifications dataset is $MODS.
        If M=mdn, the modification sets are written to mdn.
        If M=0, no modifications dataset is written.

B=bdn    Binary identifier list dataset name; this dataset receives
        a list of identifier names from the PL.
        If B, the identifier dataset is $BID.
        If B=bdn, the identifier dataset is bdn.
        If omitted or B=0, no binary identifier list dataset is
        written.
```
*\(=m\)  Master character for directives written to the listing and modifications datasets. Invalid master characters are comma, period, colon, equal sign, and space.

If omitted, the master character is read from the PL.

If \(*\=m\), \(m\) is used as the master character. The keyword alone is invalid.

\(/=c\)  Comment character for comments written to the modifications dataset. (The comment character for AUDPL comments is always \(\/=\).)

If the option is omitted, the comment character for the comment directive is \(\=/\).

If \(\/=c\), the comment character for the comment directive is \(c\). The keyword alone is invalid.

\(\text{DW=}dw\)  Data width value; the number of characters of data written to each line in the modifications dataset.

If \(\text{DW=}dw\), columns 1 through \(dw\) contain data. The \(\text{DW}\) range is 1 through 256.

If omitted or if the \(\text{DW}\) keyword stands alone, columns 1 through \(\text{last}dw\) contain data; \(\text{last}dw\) was the \(\text{DW}\) value on the \text{UPDATE} statement when the PL was written.

If it is a variable-length record PL, only \(\text{DW=}\ast\), \(\text{DW}\) alone, or \(\text{DW}\) omitted, are allowed.

\(\text{LW=}lw\)  Listing width; the length of each line written to the listing dataset.

If \(\text{LW=}lw\), the width of the listing dataset is \(lw\) characters. Valid values are 80 and 132.

When the listing width is specified as \(\text{Clw}\), the listing is continuous; that is, it is not divided into pages.

If omitted or \(\text{LW}\), the width of the listing dataset is 132 characters and the listing is divided into pages.
JU=ju Justification; how the identifier name and sequence number of each line are justified.

The identifier and sequence number are printed at the right of the AUDPL listing, regardless of the value of DW.

If the option is omitted, the JU keyword stands alone, or JU=C is employed, the identifier name is right-justified, and the sequence number is preceded by a period and left-justified.

If JU=N, the identifier name is left-justified and the sequence number right-justified, with no period in between. If JU=L, the entire sequencing field is left-justified, with a period between the identifier and sequence number.

DK=dk1:dk2:···:dkn  
DK='dk1,dk2,···,dkj,dkk,···,dkn'  
Decks to which text line list options A, C, D, H, and I and the PM parameter apply. The DK control statement parameter has no effect on directives.

In the first method shown, you can specify up to 64 decks.

In the second method shown, single decks are separated by commas, and ranges of decks are separated by periods. The maximum size of the string is 87 characters. You cannot, however, combine the two methods.

If DK is omitted, the text line list options apply to all decks in the PL. The keyword alone is invalid.

PM=id1:id2:···:idn  
PM='id1,id2,···,idj,idk,···,idn'  
Pulled modification sets; reconstructs modification sets for the identifiers in the list. The scope of the search for text modified by the specified identifiers consists of the decks specified with the DK control statement parameter.

In the first method shown, you can specify up to 64 identifiers.

In the second method shown, single identifiers are separated by commas, and ranges of identifiers are separated by periods. The maximum size of the string is 87 characters. You cannot, however, combine the two methods. The keyword alone is invalid.
LO=string

Listing options. The string contains characters representing options; the default is no list options. The following summarizes the options. See subsection 5.5.1, Listing File or Dataset, for descriptions.

Options A, C, D, H, and I are text line options and, if a DK parameter has been given, apply only to the decks specified with the DK parameter.

A Writes active lines to ldn
C Writes conditional text directives to ldn; the output is a subset of the output of option D.
D Writes dataset directives to ldn; the output is a subset of the output of option A.
H Writes modification histories to ldn, along with active and inactive text lines
I Writes inactive lines to ldn

Options K, L, M, N, O, P, S, and X are summary options and apply to the entire PL.

K Writes deck line counts to ldn
L Writes identifier list to ldn
M Writes modification set cross-reference to ldn
N Writes identifier list in ASCII collation order to ldn
O Writes overlapping modification set list to ldn
P Writes a short summary of the PL to ldn
S Writes the status of modification sets
X Writes a common deck cross-reference to ldn

CM Copy modifications; writes the reconstructed modification sets to ldn as well as to mdn. Keyword only.

NA No abort; if nonfatal errors are detected, AUDPL does not abort until all processing has been completed. Keyword only.

NR No rewind; does not rewind modifications dataset or binary identifier list dataset at beginning or end of AUDPL execution. Keyword only.
5.3 AUDPL COMMAND LINE (UNICOS)

Under UNICOS, the following command line executes AUDPL. Use the parameters on the command line to specify the files to be used, the contents of the AUDPL listing, and other features of the run.

Format:

```
audpl -p plfile [-i infile] [-l lstfile] [-m modfile]
[-b [idfile]] [-x mc] [-y cc] [-j ju] [-d decks]
[-P modsets] [-O arguments]
```

- `p plfile`
  Program library file name; required there is no default.

- `i infile`
  Input file name. If you specify -i -, the input file is stdin. If you specify -i in, the input file is in. If omitted, there is no input file.

- `l lstfile`
  The listing file name. If -l is omitted, the default listing file is stdout. If you specify -l list, the listing file is list.

- `m modfile`
  The modifications file name. This file receives reconstructed modification sets as selected by the -P control statement option. If omitted, the modification file is stdout. If you specify -m modf, the modification file is modf.

- `b [idfile]`
  This file receives a list of identifier names from the PL. If you specify -b with no argument, the identifier file is stdout. If you specify -b idn, the identifier file is idn. If you omit the -b option, there is no identifier file.

- `x mc`
  The master character for directives written to the listing and modification files. Invalid master characters are the comma, period, colon, equal sign, and space.

  If omitted, the master character is read from the PL. If you specify -x mc, mc is used as the master character. The -x option alone is invalid.
-y cc  The comment character for comments written to the
modifications file. (The comment character for AUDPL
comments is always \/.)

If you omit this option, the comment character is \\.
If you specify -y cc, the comment character is cc. The
-y option alone is invalid.

-j ju  Justification; how the identifier name and sequence number
of each line are justified.

The identifier and sequence number are printed to the
right of the AUDPL listing.

If you omit the -j option, or specify -j c, the
identifier name is right-justified, and the sequence
number is preceded by a period and is left-justified.

If you specify -j a, the identifier name is
left-justified and the sequence number right-justified,
with no period in between. If you specify -j l, the
entire sequencing field is left-justified, with a period
between the identifier and sequence number.

The -j option alone is invalid.

-d decks  Decks to which text line list option-arguments a, c, d, h,
and i and the -P option apply. The -d option has no effect
on directives. The two valid formats are as follows (where
dk represents a deck):

\[ dk_1, dk_2, ..., dk_n \]
\[ 'dk_1, dk_2, ..., dk_j, dk_k, ..., dk_n'. \]

In the first method shown, you can specify up to 64 decks.

In the second method shown, single decks are separated by
commas, and ranges of decks are separated by periods. The
maximum size of the string is 96 characters. You cannot,
however, combine the two methods.

If -d is omitted, the text line list options apply to all
decks in the PL. The -d option alone is invalid.

The syntax using the -d option is nonstandard.
-P modsets

Pulled modification sets; reconstructs modification sets
for the identifiers in the list. The scope of the search
for text modified by the named identifiers consists of the
decks specified with the -d option.

'\id_1,\id_2,\ldots,\id_j,\id_k,\ldots,\id_n'  

Single identifiers are separated by commas, and ranges of
identifiers are separated by periods. The maximum size of
the string is 96 characters.

The -P option alone is invalid.

The syntax using the -P option is nonstandard.

-O arguments

Listing arguments. The arguments contain characters
representing options; the default is no list arguments.
The following summarizes the arguments that are
available. The arguments must be separated by a space.
See subsection 5.5.1, Listing File or Dataset, for
descriptions.

Arguments a, c, d, h, and i are text line options and,
if a -d option has been given, apply only to the decks named
with the -d option.

- a Writes active lines to 1f
- c Writes conditional text directives to 1f; the output
  is a subset of the output of option d.
- d Writes file directives to 1f; the output is a subset
  of the output of option a.
- h Writes modification histories to 1f, along with
  active and inactive text lines
- i Writes inactive lines to 1f

Arguments k, l, m, n, o, p, s, and x are summary arguments
and apply to the entire PL.

- k Writes deck line counts to 1f
- l Writes identifier list to 1f
- m Writes modification set cross-reference to 1f
- n Writes identifier list in ASCII collation order to 1f
- o Writes overlapping modification set list to 1f
- p Writes a short summary of the PL to 1f
- s Writes the status of modification sets
- x Writes a common deck cross-reference to 1f
5.4 INPUT DIRECTIVES

In addition to responding to the list options and other parameters in the control statement, AUDPL recognizes a set of directives read from the input dataset or file. These directives specify a more limited part of the PL than do the list options in the control statement.

The format of the AUDPL directives is the same as that for the UPDATE directives described in subsection 3.2, Directive Format (see this subsection for format description).

With AUDPL versions 2.0 and higher, all input directives can be issued in either uppercase or lowercase; however, you cannot issue the directives in mixed case.

The following subsections describe the AUDPL directives.

5.4.1 / - COMMENT

A comment appears only in the input dataset or file and is ignored by AUDPL.

Format:

```
*/comment
```

// Comment character followed by desired comments
5.4.2 ACTIVE - ACTIVE LINES

ACTIVE writes to the listing file or dataset all active text lines in a text range or an entire deck.

Format:

*ACTIVE id1.seq1,id2.seq2[,DK=deck]

id1.seq1 First line in text range
id2.seq2 Last line in text range
DK=deck Deck or common deck containing the text range
deck Deck or common deck name; specifies entire deck.

5.4.3 COND - CONDITIONAL TEXT DIRECTIVES

You can use COND to write to the listing file or dataset all active text lines that contain the conditional text directives IF, ELSEIF, ELSE, andENDIF.

Format:

*COND id1.seq1,id2.seq2[,DK=deck]

id1.seq1 First line in the text range
id2.seq2 Last line in the text range
DK=deck Deck or common deck containing the text range
deck Deck or common deck name; specifies entire deck.
5.4.4 DIR - DATASET OR FILE DIRECTIVES

Use DIR to write to the listing file or dataset all active text lines that contain dataset directives.

Format:

*DIR id₁.seq₁, id₂.seq₂[, DK=deck]
*DIR deck

id₁.seq₁  First line in the text range
id₂.seq₂  Last line in the text range
DK=deck   Deck or common deck containing the text range
deck      Deck or common deck name; specifies entire deck.

5.4.5 HISTORY - MODIFICATION HISTORY

HISTORY writes the modification history for a single line or a text range. The modification history for a line identifies the modification sets that have deleted and restored the line. Yanked modification sets that have affected the line are flagged. All active and inactive lines in the range are listed.

Format:

*HISTORY id₁.seq₁, id₂.seq₂[, DK=deck]
*HISTORY deck

id₁.seq₁  First line in the text range
id₂.seq₂  Last line in the text range
DK=deck   Deck or common deck containing the text range
deck      Deck or common deck name; specifies entire deck.
5.4.6 INACTIV - INACTIVE LINES

With INACTIV, you can write inactive text lines in a text range or an entire deck.

Format:

- \*INACTIV id₁.seq₁, id₂.seq₂[,DK=deck]

\*INACTIV deck

id₁.seq₁ First line in the text range
id₂.seq₂ Last line in the text range
DK=deck Deck or common deck containing the text range
deck Deck or common deck name; specifies entire deck.

5.4.7 PULLMOD - PULLED MODIFICATION SETS OR DECKS

PULLMOD allows you to reconstruct one or more modification sets or decks. The output is written to the modifications file or dataset, which has the same format as an UPDATE input file. It is then echoed to the listing file or dataset if you use the CM control statement option.

PULLMOD works correctly only for modification sets added with a version of UPDATE from release 1.12 or higher. You cannot reconstruct deletions from earlier modification sets, because those modification sets do not have valid modification histories in the PL.

Format (all on one line):

- \*PULLMOD

id₁,id₂,...,idj,idk,...,p,...,idn[,DK=deck]

idᵢ Single identifier
idⱼ,idₖ Range of identifiers
Prefix representing all modification sets or decks whose identifiers begin with the specified characters. The prefix is followed by a colon.

The deck for which the modification set is to be reconstructed.

5.5 OUTPUT

You can use AUDPL to generate one, two, or three forms of output datasets: (1) a listing file or dataset, (2) a modifications file or dataset, and/or (3) a binary-identifier list file or dataset. The following subsections describe the contents of the files and the commands that control their content and format.

5.5.1 LISTING FILE OR DATASET

The listing file or dataset contains PL text lines, reconstructed modification sets, and other information written in response to parameters in the AUDPL control statement and directives in the input file.

Under COS, the listing dataset is divided into pages by default. The LPP parameter in the OPTION control statement controls the number of lines per page (see the COS Reference Manual, publication SR-0011). Use the LW control statement parameter to control the width of the listing. It can also be used to inhibit the division of the listing dataset into pages.

Under UNICOS, the listing file comes out as raw text with no headers and no page breaks. You can make your listing output more readable by using a pipe together with the pg command. (For information on using pipes and on pg, see the UNICOS Primer, publication SG-2010.) The output for a text line is preceded by the name of the deck to which it belongs and a flag that gives special attributes of the line. It is also followed by the line's identifier and sequence number.

A text line continues for as many output lines as are needed to write all significant characters. Text lines are written according to their positions in the PL, with active and inactive lines interspersed. The order of decks and common decks in the listing is determined by their order in the PL, which is the same as their order in the identifier list.
### 5.5.1.1 Text line listing options

The AUDPL control statement allows the following set of options for text-line listable output:

<table>
<thead>
<tr>
<th>COS Option</th>
<th>UNICOS Option-argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>a</td>
<td>Writes active lines from decks and common decks specified with the DK parameter. The ACTIVE directive also writes active lines from decks and common decks to the listing.</td>
</tr>
<tr>
<td>I</td>
<td>i</td>
<td>Writes inactive lines from the decks and common decks specified with the DK parameter. The INACTIV directive also writes inactive lines to the listing. Inactive lines are flagged with '&amp;i;' between the deck name and the text for the line.</td>
</tr>
<tr>
<td>D</td>
<td>d</td>
<td>Lists all active text lines containing file or dataset directives for each deck and common deck specified with the DK parameter. The DIR directive writes these lines for directives in decks or ranges named in the DIR directive. Directives are flagged with '&amp;d;' in the flag field of the output for the line.</td>
</tr>
<tr>
<td>C</td>
<td>c</td>
<td>Writes all active text lines containing the conditional text directives IF, ELSEIF, ELSE, and ENDIF for each deck and common deck specified with the DK parameter. The COND directive writes these lines for directives in decks or ranges specified in the COND directive. The nesting level of the directive appears in the flag field of the output for the line (for example, '&amp;0;' for the outer level).</td>
</tr>
<tr>
<td>H</td>
<td>h</td>
<td>Along with the HISTORY directive, writes modification histories. The listing of each text line, either active or inactive, is followed by a list of the changes made to the line. This list tells you which modifications have deleted and restored the line, and whether the modifications that made the changes have been yanked.</td>
</tr>
</tbody>
</table>
5.5.1.2 Summary listing options

The AUDPL control statement allows you the following output options in the listing dataset:

<table>
<thead>
<tr>
<th>COS Option</th>
<th>UNICOS Option-argument</th>
<th>Description</th>
</tr>
</thead>
</table>
| P          | p                      | Writes general information about the PL. The summary consists of the following:  
• The name of the dataset containing the PL (under COS only)  
• The date the PL was written  
• The last identifier added to the PL  
• The default master character  
• The default data width for the compile and source files or datasets written by UPDATE. The data width is determined by the UPDATE DW parameter that was specified when the PL was built.  
• The number of characters of data stored for each line in the PL. This may be more than the number written to the compile and source files or datasets.  
• The number of decks, common decks, and modification set identifiers in the PL. |
<p>| L          | l                      | Lists all identifiers in the PL (the same function as the ID option in UPDATE). In the output, a deck name is preceded by a single asterisk (*), a common deck by two asterisks (**), and a yanked identifier by a minus sign (-). Purged identifiers are not included in this list, because they are no longer in the PL. Identifiers that have been unyanked are the same as identifiers that were never yanked. |</p>
<table>
<thead>
<tr>
<th>COS Option</th>
<th>UNICOS Option-argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>n</td>
<td>Writes an ASCII collation order list of all identifiers in the PL to the listing file or dataset. The list is divided into separate sections for common decks, decks, and modification sets.</td>
</tr>
<tr>
<td>K</td>
<td>k</td>
<td>Writes the number of active and inactive text lines in each deck and common deck in the PL to the listing file or dataset.</td>
</tr>
<tr>
<td>M</td>
<td>m</td>
<td>Writes a modification set cross-reference to the listing file or dataset. The cross-reference consists of the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A list of decks changed by each modification set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A list of modification sets that changed each deck</td>
</tr>
<tr>
<td>O</td>
<td>o</td>
<td>Writes a list of overlapping modification sets in each deck. The list shows, for each modification set, the modification sets that overlap it and the modification sets it overlaps. A modification set overlaps an earlier modification set if it changes the status of text lines changed by the earlier modification set.</td>
</tr>
<tr>
<td>S</td>
<td>s</td>
<td>Separates the modification set identifiers into the following groups:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Active modification sets; active if at least one change remains unaffected by subsequent modifications.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Inactive (or dormant) modification sets; inactive if every change has been superseded by a subsequent modification.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Dead modification sets; dead if no longer listed in the modification history of any line, active or inactive, in the PL.</td>
</tr>
</tbody>
</table>
COS UNICOS
Option Option-argument Description
X x Writes a common deck cross-reference. The cross-reference lists the following:

- Common decks called from each deck
- Decks calling each common deck
- Uncalled common decks

The output indicates the number of times a common deck is called from each deck. Common decks using the no-propagation option on the COMDECK directive are flagged with a plus sign (+) in the cross-reference listing. Common decks defined in another PL and referenced in the PL being processed are flagged with a pound sign (#) in the cross-reference listing.

5.5.2 RECONSTRUCTED MODIFICATION SETS

You can use the CM option to generate a copy of the reconstructed modification sets written to the modifications dataset for the PULLMOD directive and PM parameter. If the data width for the reconstructed modification sets is greater than the listing width, the reconstructed modifications are truncated.

Comments written to the reconstructed modification set give the following:

- The name(s) of the deck(s) to which the modification set applies
- A list of earlier modification sets upon which the reconstructed one depends; that is, modification sets directly referenced by one or more directives.
- A list of earlier modification sets that the reconstructed modification set overlaps—that is, modifications sets that had already modified one or more lines that were deleted or restored by the reconstructed modification set.
- A list of later modification sets that overlap the reconstructed one.
A reconstructed modification set may differ from the original set. It produces the same results, however, when applied to a PL from which the reconstructed modification set and all later modification sets have been purged. It also produces the same results when applied to the PL before the original modification set has been added.

The following examples, one each for COS and UNICOS, show the use of AUDPL to reconstruct a modification set:

Under COS, the procedure is as follows:

```
UPDATE,P=PL1,N=PL2,F,I=MOD1,S=S1,C=C1.
AUDPL,P=PL2,PM=MOD1,M=MOD1P.
UPDATE,P=PL1,N=PL2P,F,I=MOD1P,S=S1P,C=C1P.
```

Under UNICOS, the same procedure looks as follows:

```
update -p PL1 -n PL2 -f -i mod1 -s S1 -c C1
audpl -p PL2 -P mod1 -m mod1P
update -p PL1 -n PL2P -f -i mod1P -s S1P -c C1P
```

The following discussion applies to both the COS and the UNICOS examples.

In line 1 of these examples, UPDATE modifies the program library PL1 with the modification set MOD1 (mod1 under UNICOS) and writes a new program library PL2, a source file/dataset S1, and a compile file/dataset C1 (cl.f under UNICOS).

In line 2, AUDPL uses PL2 to reconstruct MOD1 on the file/dataset MOD1P (mod1P and mod1P under UNICOS, respectively, under UNICOS).

In line 3, UPDATE modifies the original program library, PL1, with the reconstructed modification set MOD1P (mod1P under UNICOS) and writes PL2P, S1P, and C1P (clp.f under UNICOS).

As long as modifications from program library PL1 have not been purged from PL2, and as long as no EDIT directive has been used after MOD1 was added, the following relationships will hold:

- Modifications dataset MOD1P (mod1P under UNICOS) is equivalent to input file/dataset MOD1 (mod1 in UNICOS) in that it inserts and deletes the same text lines (although it may use different sequences of directives to do so).
- Program library PL2P is identical to program library PL2.
- Source file/dataset S1P is identical to source file/dataset S1.
- Compile file/dataset C1P is identical to compile file/dataset C1.
5.5.3 MODIFICATIONS FILE OR DATASET

UPDATE writes the modification sets reconstructed by the PULLMOD directive and the PM control statement parameter to the modifications file or dataset. You can use the modifications file or dataset as input to a subsequent UPDATE run.

The DW parameter on the control statement determines the length of each line in the modifications file or dataset. No end-of-file (EOF) is written between modification sets, so the modifications file is a single file. If you specify the NR control statement option, no EOF is written at the end of the modifications file or dataset.

5.5.4 BINARY-IDENTIFIER LIST FILE OR DATASET AND IDENTIFIER LIST

The file or dataset that receives a list of identifiers from the PL is called a binary-identifier list dataset under COS and an identifier list under UNICOS. Under COS, this dataset includes the name of each identifier and information about whether the identifier is a deck, common deck, or modification set identifier, and whether it has been yanked.

The COS binary identifier list dataset has one file. Each 3-word record in the dataset has information about one identifier in the PL.

Record format for the COS binary-identifier list dataset:

<table>
<thead>
<tr>
<th>0</th>
<th>Identifier Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Identifier Type</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Yank Flag</td>
</tr>
</tbody>
</table>

The identifier name is left-justified and zero-filled.

The identifier type is as follows:

0 For modification set identifiers
1 For decks
2 For regular common decks
3 For common decks with the NOPROP option

The Yank flag is 1 for identifiers that are yanked; otherwise, it is 0.
Under UNICOS, the identifier list contains the name of each identifier and information about whether the identifier is a deck, common deck, or modification set identifier, and whether it has been yanked.

The format of the identifier list is as follows:

```
| NAME. TYPE.   | .NOPROP | .YANKED |
```

The types possible are deck, comdeck, and modification. If the type is comdeck, and if the common deck is a NOPROP common deck, .NOPROP is listed on that line. You will receive one of the following results:

- NAME.DECK
- NAME.MODIFICATION
- NAME.COMDECK
- NAME.COMDECK.NOPROP
- NAME.MODIFICATION.YANKED

5.6 AUDPL SAMPLE LISTING

This subsection provides an example of an AUDPL program listing. See subsection 4.17, Examples Showing Dataset Content, for the generation of program library PL2, which is used in this example. Under COS, the job control statement that generates this example is as follows:

```
AUDPL,P=PL2,PM=MOD2A:MOD2B:EXAMPLE,LO=AIKLMNOPSX.
COPYF, I=$MOD.
```

Under UNICOS, the command line that generates this example is as follows:

```
audpl -p PL2 -P mod2a mod2b example -O a i k l m n o p s x
```

The AUDPL sample listing is as follows:

```
***** Program library summary

Dataset containing the program library
(COS only): PL2

Date the dataset was written: 07/16/84

Last identifier added to the PL: IA
```
Default master character for directives read from the input and written to a source dataset: *

Default data width for compile and source datasets: 72

Data width of text stored in the PL: 80

Number of decks: 4

Number of common decks: 1

Number of modification set identifiers: 5

***** Identifier list

*EXAMPLE **BLOCK1 **DIVIDE MOD1 MOD2A MOD2B MOD3A MOD3B *EOF1 *IA

*** 4 decks 1 common deck 5 modification set identifiers

***** Common decks

BLOCK1

***** Decks

DIVIDE EOF1 EXAMPLE IA

***** Modification set identifiers

MOD1 MOD2A MOD2B MOD3A MOD3B

*** 4 decks 1 common deck 5 modification set identifiers

***** Text line listings

EXAMPLE <d> *DECK EXAMPLE EXAMPLE.1
EXAMPLE PROGRAM EXAMPLE EXAMPLE.2
EXAMPLE LOGICAL IA MOD3A.1
EXAMPLE <d> *CALL BLOCK1 EXAMPLE.3
EXAMPLE IF (IA())WRITE*, 'Enter', SIZE, 'values for X and Y:' MOD3A.2
EXAMPLE READ *, A, B EXAMPLE.4
EXAMPLE CALL DIVIDE EXAMPLE.5
EXAMPLE WRITE *, A, B EXAMPLE.6
EXAMPLE STOP EXAMPLE.7
EXAMPLE END EXAMPLE.8

SR-0013 K 5-21
Deck EXAMPLE has 10 active lines, 0 inactive lines

BLOCK1 <d> *COMDECK BLOCK1
BLOCK1 <i> REAL A(10), B(10)
    deleted by MOD2A
BLOCK1 PARAMETER (SIZE=100)
BLOCK1 REAL A(SIZE), B(SIZE)
BLOCK1 COMMON /BLOCK/ A, B

Common deck BLOCK1 has 4 active lines, 1 inactive lines

DIVIDE <d> *DECK DIVIDE
DIVIDE SUBROUTINE DIVIDE
DIVIDE <d> *CALL BLOCK1
DIVIDE <i> DO 100 I = 1, 10
    deleted by MOD2B
DIVIDE DO 100 I = 1, SIZE
DIVIDE IF (B(I) .NE. 0) THEN
DIVIDE A(I) = A(I)/B(I)
DIVIDE ELSE
DIVIDE A(I) = 0
DIVIDE ENDIF
DIVIDE 100 CONTINUE
DIVIDE RETURN
DIVIDE END

Deck DIVIDE has 12 active lines, 1 inactive line

EOF1 <d> *DECK EOF1
EOF1 <d> *CWEOF

Deck EOF1 has 2 active lines, 0 inactive lines

IA <d> *DECK IA
IA IDENT IA
IA * Return true if interactive, false if batch
IA * NP=0
IA GET, S1 S6&S7, JCIA, A0
IA S1 S1<D'63
IA EXIT
IA END

Deck IA has 10 active lines, 0 inactive lines

The entire PL has 38 active lines, 2 inactive lines
### Deck type | deck name | active lines | inactive lines
---|---|---|---
common deck | BLOCK1 | 4 | 1
deck | DIVIDE | 12 | 1
deck | EOF1 | 2 | 0
deck | EXAMPLE | 10 | 0
deck | IA | 10 | 0

The entire PL has 38 active lines, 2 inactive lines, 4 decks and 1 common deck.

***** Common decks called by each deck

Deck DIVIDE calls: BLOCK1

Deck EXAMPLE calls: BLOCK1

***** Decks calling each common deck

BLOCK1 is called by: DIVIDE EXAMPLE

***** Active modification sets (last modification to at least one line)

MOD1 MOD2A MOD2B MOD3A

***** Dead modification sets (not in the modification history of any line)

MOD3B

***** Modification sets that changed each deck and common deck

BLOCK1 is modified by: MOD2A

DIVIDE is modified by: MOD1 MOD2B

EXAMPLE is modified by: MOD3A

***** Decks and common decks changed by each modification set

MOD1 modifies decks: DIVIDE

MOD2A modifies decks: BLOCK1

MOD2B modifies decks: DIVIDE

MOD3A modifies decks: EXAMPLE
**** Overlapping mods in deck EXAMPLE

EXAMPLE is overlapped by: EXAMPLE

EXAMPLE overlaps mod(s): EXAMPLE

/EOF

ID MOD 2A
*/
*/ - MOD2A modifies deck(s): BLOCK1
*/
*DC BLOCK1
*D BLOCK1.2

PARAMETER (SIZE=100)
REAL A(SIZE),B(SIZE)
*/
*/ - End of MOD2A
*/

*ID MOD2B
*/
*/ - MOD2B modifies deck(s): DIVIDE
*/
*DC DIVIDE
*D DIVIDE.4

DO 100 I = 1,SIZE
*/
*/ - End of MOD2B
*/
/EOF

*DECK EXAMPLE

PROGRAM EXAMPLE

*CALL BLOCK1
READ *,A,B
CALL DIVIDE
WRITE *,A,B
STOP
END
The MODECKS utility compares two sets of UPDATE decks and creates a modification file (mod). MODECKS compares a modified deck (an edit file or dataset) and an original deck (a source file or dataset) and creates the modification file necessary to create the edit file from the source file.

MODECKS currently works on all PLs of any data width, including variable length PLs. However, the maximum line length of any card in the PL is 128 for space and speed reasons.

This section provides, first, the COS control statement and its parameters, then the UNICOS command line and parameters, and then gives an example illustrating the use of MODECKS with each operating system.

6.1 MODECKS CONTROL STATEMENT (COS)

You can load and execute MODECKS with the control statement that follows:

Format:

```
MODECKS[E=edn],S=sdn[,M=mdn][,LOCATE][,Q]
[,IDENTS=id1:id2:...idn]
[,ORDER='dk1.dk2,dk3.dk4,...,dk-1.dkn'][,NP][,NN][,NM]
[,NH][,PREFIX='x1,x2,...xn'][,DESC=ddn].
```

- **E=edn**  
  Edit file dataset name. This is a file that has been edited and changed by the user. If omitted or E, the edit file is $IN. If E=edn, the edit file is edn. E=O is invalid.

- **S=sdn**  
  Source file dataset name. This is a file that has been created by UPDATE with sequence numbers attached. If omitted or S, the source file is $SR. If S=sdn, the source file is sdn. S=O is invalid.
M=mdn  Mod file dataset name. This file receives the modification file that MODECKS creates. If omitted or M, the mod file is $OUT. If M=mdn, the mod file is mdn. M=0 is invalid.

LOCATE  Tells MODECKS to generate additional comments for certain INSERT and DELETE directives. If the line being referenced by the INSERT or DELETE directive does not have the deck name as part of its sequence number, MODECKS identifies the nearest line that does contain the deck name, and generates a comment on the INSERT or DELETE directive.

The following is an example of using LOCATE with MODECKS. Assume that your source file (SDN) reads as follows:

```
*DECK SAMPLE
Line 1  SAMPLE.1
Line 2  SAMPLE.2
Line 2a MOD1.1
Line 2b MOD1.2
Line 2c MOD1.3
Line 2d MOD1.4
Line 3  SAMPLE.4
Line 4  SAMPLE.5
Line 4a MOD1.5
Line 4b MOD1.6
```

Now, assume that your edit dataset (EDN) reads as follows:

```
*DECK SAMPLE
Line 0a
Line 1
Line 2
Line 2b
Line 2c
Line 2d
Line 3
Line 4
Line 4a
Line 5
```
Your edit dataset calls for the deletion of two lines (lines 2a and 4b in the SDN) and the insertion of two lines (new line 0a and line 5 in the EDN). To perform this operation, you would enter the following control statement:

MODECKS, E=EDN,S=SDN,LOCATE.

The resulting output from MODECKS would look like this:

```
  | *I SAMPLE.1
  | Line0a
  | *D MOD1.1  1 LINE BELOW: SAMPLE.3
  | *D MOD1.6  2 LINES BELOW: SAMPLE.5
  | Line 5
```

The use of LOCATE lets MODECKS tell you where the change was made in relation to the other identifiers (column 2 in the output). When you add a new line, MODECKS prints out the insertion text (the last line of the preceding sample output).

Q

Tells MODECKS to compare only the decks that are present in both the edit and the source files. The default is to purge decks in the source file but not in the edit file, to add decks that are in the edit file but not in the source file, and to produce MOVEDK directives needed to put the decks in the PL in the same order as they appear in the Edit file. Keyword only.

IDENTS=id1:id2:...idn

These are identifiers (from 1 to 7 characters) that appear in the *ID directives of the modifications. You may specify up to 64 identifiers. For the first 26 *ID directives, id1 is used with the characters "A-Z" appended, one for each *ID directive. For the 27th to the 52nd *ID directives, id2 is used, and so on. If IDENTS is omitted, no *ID directives is produced. If you specify the keyword alone, the identifier defaults to MODIDNT.
ORDER='dk1.dk2,dk3.dk4,...,dk-l.dkn'
These are the ranges of decks that are to be alphabetized. You can provide up to 32 ranges. MODECKS produces MOVEDK directives needed to put the edit decks into alphabetical order within each range of decks specified. The range of decks is noninclusive by default. You can enclose the decks by brackets or parentheses, mixed freely, to aid in visual clarity; however, the system ignores parentheses. A left bracket, [, means that the first deck in the range is included in the alphabetizing scheme; a right bracket, ], means that the second deck in the range is included in the alphabetizing scheme.

There are two decks predefined in MODECKS: $$FIRST and $$LAST that match the first and last decks in the file, respectively. The default is to generate MOVEDK directives that will move the source file decks into the same order as the edit file decks. The keyword alone alphabetizes the entire range of decks; that is, ORDER='[$$FIRST.$$LAST]'.

NP Tells MODECKS not to generate PURGEDK directives. The default is to generate PURGEDK directives for decks that are found in the source dataset but not in the edit dataset. NM is implied.

NN Tells MODECKS not to generate any *DECK directives. The default is to generate *DECK directives for decks that are found in the edit dataset but not in the source dataset. NM is implied.

NM Tells MODECKS not to generate any *MOVEDK directives. The default is to generate the *MOVEDK directives needed to put the decks in the ordering scheme specified.

NH Tells MODECKS not to generate a mod header. The default is to generate a mod header for each identifier dataset as supplied with the IDENTS option.

PREFIX= 'xl,x2,...,xn'
Tells MODECKS the prefixes that are to be used when computing mod dependencies, which is any line being inserted or deleted and having a standard MODID format; that is, one consisting of 8 characters and not matching the deck name. Then MODECKS determines any UPDATE dependencies that the mod has. The default prefix is the same as the one specified in the IDENT directives.

DESC=ddn
Specifies the file containing the mod description to be inserted into the mod header. If omitted, no description file is read. If DESC=ddn, the description file is ddn.
6.2 MODECKS COMMAND LINE (UNICOS)

Under UNICOS, you can load and execute MODECKS with the following command line:

Format:

```
modecks [-e ef] [-s sf] [-m mf] [-1] [-q] [-i id1 id2 ... idn]
[[-o range] [-p] [-n] [-v] [-h] [-x x1 x2 ... xn] [-d df]]
```

- `e ef` The edit file name. This is a file that has been edited and changed by the user. If omitted, the edit file is stdin. If you specify `e page`, the edit file is page.

- `s sf` The source file name. This is a file that has been created by UPDATE with sequence numbers attached. If you specify `s skb`, the source file is skb. This is a required option.

- `m mf` The mod file name. This file receives the modification file that MODECKS creates. If you do not specify a file, the mod file is stdout. If you specify `m modf`, the mod file is modf.

- `-l` The `-l` option tells MODECKS to generate additional comments for certain INSERT and DELETE directives. If the line being referenced by the INSERT or DELETE directive does not have the deck name as part of its sequence number, MODECKS identifies the nearest line that does contain a deck name, and generates a comment on the INSERT or DELETE directive.
The following is an example of using the -l option with MODECKS. Assume that your source file reads as follows:

```
*deck sample
Line 1 sample.2
Line 2 sample.3
Line 2a mod1.1
Line 2b mod1.2
Line 2c mod1.3
Line 2d mod1.4
Line 3 sample.4
Line 4 sample.5
Line 4a mod1.5
Line 4b mod1.6
```

Now, assume that your edit file (page) reads as follows:

```
*deck sample
Line 0a
Line 1
Line 2
Line 2b
Line 2c
Line 2d
Line 3
Line 4
Line 4a
Line 5
```

Your edit file calls for the deletion of two lines (lines 2a and 4b in skb) and the insertion of another two lines (new line 0a and line 5 in page). To perform this operation, you would enter the following command line:

```
modecks -e page -s skb -l
```
The resulting output from modecks looks like this:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>*I sample.1</td>
<td></td>
</tr>
<tr>
<td>Line 0a</td>
<td></td>
</tr>
<tr>
<td>*D mod1.1</td>
<td>1 LINE BELOW: sample.3</td>
</tr>
<tr>
<td>*D mod1.6</td>
<td>2 LINES BELOW: sample.5</td>
</tr>
<tr>
<td>Line 5</td>
<td></td>
</tr>
</tbody>
</table>

The use of the -l option lets MODECKS tell you where the change was made in relation to the other identifiers (column 2 in the output). When you add a new line, MODECKS prints out the insertion text (the last line is the preceding sample output).

-q

The -q option tells MODECKS to compare only the decks that are present in both the edit file and the source file. The default is to purge decks in the source file but not in the edit file, to add decks that are in the edit file but not in the source file, and to produce MOVEDK directives needed to put the decks in the PL in the same order as they appear in the edit file. The -q option takes no arguments.

-i id1 id2 ... idn

These are identifiers will appear in the *ID directives of the modifications. You may specify up to 64 identifiers. For the first 26 *IDENT directives, id1 is used with the characters "A-Z" appended, one for each *IDENT directive. For the 27th to the 52nd *IDENT directives, id2 is used, and so on. If you omit -i, no *ID directives are produced. If you specify the -i option alone, the identifier defaults to MODIDENT.

-o range

The -o option defines the ranges of decks that are to be alphabetized. The format is as follows (rng has the format deck1.deckj):

rng1,rng2, ... rngn
You can provide up to 32 ranges, and MODECKS will produce MOVEDK directives needed to put the edit decks into alphabetical order within each range of decks specified. The range of decks is noninclusive by default. You can surround the decks by brackets or parentheses, mixed freely, to aid in visual clarity; however, the system ignores parentheses. A left bracket, [, means that the first deck in the range is included in the alphabetizing scheme; a right bracket, ], means that the second deck in the range is included in the alphabetizing scheme.

There are two decks predefined in MODECKS: $$FIRST and $$LAST to match the first and last decks in the file, respectively. The default is to generate MOVEDK directives that will move the source file decks into the same order as the edit file decks. The -o option alone alphabetizes the entire range of decks; that is, -o'[$$FIRST.$$LAST]'.

The -p option tells MODECKS not to generate PURGEDK directives. The default is to generate PURGEDK directives for decks that are found in the source file but not in the edit file. The -v option is implied.

The -n option tells MODECKS not to generate any *DECK directives. The default is to generate *DECK directives for decks that are found in the edit file but not in the source file. The -v option is implied.

The -v option tells MODECKS not to generate any *MOVEDK directives. The default is to generate the *MOVEDK directives needed to put the decks in the ordering scheme given.

The -h option tells MODECKS not to generate a mod header. The default is to generate a mod header for each identifier file as supplied with the -i option.

The -x option tells MODECKS the prefixes that are to be used when computing mod dependencies. A mod dependency is defined as any line that is being inserted or deleted and has a standard MODID format; that is, it has 8 characters and does not match the deck name. Then MODECKS determines any UPDATE dependencies that the mod has. The default prefix is the same as the one specified in the -i directives.

The -d option tells MODECKS which file contains the mod description that will be inserted into the mod header.
6.3 USING MODECKS UNDER COS

The following example illustrates the use of MODECKS under COS. A MODECKS session can be broken down into three stages. First you transfer the original deck from COS to the front end, as follows:

```
ACCESS,DN=PL,ID=MYPL.
UPDATE,P=PL,Q=DECK1,NS,S=DECK1,C=O,N=0.
DISPOSE,DN=DECK1,TEXT='-mydir/tmp/d.edit'.
```

This example assumes you are running a UNIX front-end system; if you are not using a UNIX system, change only the TEXT entry. For a full discussion of the UPDATE parameters, see section 2, Invoking Update. In this example, the original deck is accessed without sequence numbers (the NS on the UPDATE line).

Second, you use the front end to make changes to the deck—in this case, using a UNIX editor:

```
vi d.edit
```

Third, you run MODECKS under COS to compare decks:

```
ACCESS,DN=PL,ID=MYPL.
UPDATE,P=PL,Q=DECK1,SQ,S=DECK1,C=O,N=0.
FETCH,DN=EDIT,TEXT='-mydir/tmp/d.edit'.
MODECKS,E=EDIT,S=DECK1,IDENTS=MODID,M=MODOUT.
UPDATE,P=PL,N=NEWPL,I=MODOUT,F,S=O,C=O.
DISPOSE,DN=MODOUT,TEXT='-mydir/tmp/modout'.
```

On the MODECKS line, MODECKS compares the EDIT file and the DECK1 file and creates a mod file, MODOUT, to reflect the changes made. UPDATE then applies the mod to a PL and creates a new PL. Finally, the mod is saved to the front end using the DISPOSE control statement. There are, of course, other methods possible. For example, if the deck was on the front end rather than on Cray disk, you could use FETCH rather than ACCESS. And if you wanted the mod to be saved to Cray disk rather than to the front end, you could use SAVE rather than DISPOSE.
6.4 USING MODECKS UNDER UNICOS

The following example illustrates the use of MODECKS under UNICOS. A MODECKS session can be broken down into three stages. First, you transfer the original deck, as follows:

```
update -p PL -s d.edit -o ns -q deck1
```

The source file in this example is named d.edit. The original deck is accessed without sequence numbers (the ns on the command line). (For a full discussion of the UPDATE parameters, see section 2, Invoking UPDATE.)

Second, you make whatever changes are desired using the editor:

```
vi d.edit
```

Third, you run MODECKS, as follows:

```
modecks -e d.edit -s d.src -m modoutput -i MODID
```

In this example, UPDATE gets the original deck with sequence numbers (using sq on the command line). Then, MODECKS compares d.edit and d.src and creates a mod to reflect the changes made. Finally, UPDATE applies the mod to PL and gives a new PL.
APPENDIX SECTION
A. CHARACTER SET

Appendix A contains characters used by UPDATE. Code values are octal. Codes 000 through 037 (NUL through US) and 177 (DEL) are not recognized. Separators are invalid for name, master, and comment characters. The symbol (s) identifies separators in the following table.

<table>
<thead>
<tr>
<th>Character</th>
<th>ASCII Code</th>
<th>Character</th>
<th>ASCII Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>040 (s)</td>
<td>1</td>
<td>061</td>
</tr>
<tr>
<td>!</td>
<td>041</td>
<td>2</td>
<td>062</td>
</tr>
<tr>
<td>&quot;</td>
<td>042</td>
<td>3</td>
<td>063</td>
</tr>
<tr>
<td>#</td>
<td>043</td>
<td>4</td>
<td>064</td>
</tr>
<tr>
<td>$</td>
<td>044</td>
<td>5</td>
<td>065</td>
</tr>
<tr>
<td>%</td>
<td>045</td>
<td>6</td>
<td>066</td>
</tr>
<tr>
<td>&amp;</td>
<td>046</td>
<td>7</td>
<td>067</td>
</tr>
<tr>
<td>'</td>
<td>047</td>
<td>8</td>
<td>070</td>
</tr>
<tr>
<td>(</td>
<td>050</td>
<td>9</td>
<td>071</td>
</tr>
<tr>
<td>)</td>
<td>051</td>
<td>:</td>
<td>072 (s)</td>
</tr>
<tr>
<td>*</td>
<td>052</td>
<td>;</td>
<td>073</td>
</tr>
<tr>
<td>+</td>
<td>053</td>
<td>&lt;</td>
<td>074</td>
</tr>
<tr>
<td>,</td>
<td>054 (s)</td>
<td>=</td>
<td>075 (s)</td>
</tr>
<tr>
<td>-</td>
<td>055</td>
<td>&gt;</td>
<td>076</td>
</tr>
<tr>
<td>.</td>
<td>056 (s)</td>
<td>?</td>
<td>077</td>
</tr>
<tr>
<td>/</td>
<td>057</td>
<td>@</td>
<td>100</td>
</tr>
<tr>
<td>0</td>
<td>060</td>
<td>A</td>
<td>101</td>
</tr>
<tr>
<td>Character</td>
<td>ASCII Code</td>
<td>Character</td>
<td>ASCII Code</td>
</tr>
<tr>
<td>-----------</td>
<td>------------</td>
<td>-----------</td>
<td>------------</td>
</tr>
<tr>
<td>B</td>
<td>102</td>
<td>Y</td>
<td>131</td>
</tr>
<tr>
<td>C</td>
<td>103</td>
<td>Z</td>
<td>132</td>
</tr>
<tr>
<td>D</td>
<td>104</td>
<td>[</td>
<td>133</td>
</tr>
<tr>
<td>E</td>
<td>105</td>
<td>]</td>
<td>135</td>
</tr>
<tr>
<td>F</td>
<td>106</td>
<td>^</td>
<td>136</td>
</tr>
<tr>
<td>G</td>
<td>107</td>
<td>_</td>
<td>137</td>
</tr>
<tr>
<td>H</td>
<td>110</td>
<td>'</td>
<td>140</td>
</tr>
<tr>
<td>I</td>
<td>111</td>
<td>a</td>
<td>141</td>
</tr>
<tr>
<td>J</td>
<td>112</td>
<td>b</td>
<td>142</td>
</tr>
<tr>
<td>K</td>
<td>113</td>
<td>c</td>
<td>143</td>
</tr>
<tr>
<td>L</td>
<td>114</td>
<td>d</td>
<td>144</td>
</tr>
<tr>
<td>M</td>
<td>115</td>
<td>e</td>
<td>145</td>
</tr>
<tr>
<td>N</td>
<td>116</td>
<td>f</td>
<td>146</td>
</tr>
<tr>
<td>O</td>
<td>117</td>
<td>g</td>
<td>147</td>
</tr>
<tr>
<td>P</td>
<td>120</td>
<td>h</td>
<td>150</td>
</tr>
<tr>
<td>Q</td>
<td>121</td>
<td>i</td>
<td>151</td>
</tr>
<tr>
<td>R</td>
<td>122</td>
<td>j</td>
<td>152</td>
</tr>
<tr>
<td>S</td>
<td>123</td>
<td>k</td>
<td>153</td>
</tr>
<tr>
<td>T</td>
<td>124</td>
<td>l</td>
<td>154</td>
</tr>
<tr>
<td>U</td>
<td>125</td>
<td>m</td>
<td>155</td>
</tr>
<tr>
<td>V</td>
<td>126</td>
<td>n</td>
<td>156</td>
</tr>
<tr>
<td>W</td>
<td>127</td>
<td>o</td>
<td>157</td>
</tr>
<tr>
<td>Character</td>
<td>ASCII Code</td>
<td>Character</td>
<td>ASCII Code</td>
</tr>
<tr>
<td>-----------</td>
<td>------------</td>
<td>-----------</td>
<td>------------</td>
</tr>
<tr>
<td>p</td>
<td>160</td>
<td>x</td>
<td>170</td>
</tr>
<tr>
<td>q</td>
<td>161</td>
<td>y</td>
<td>171</td>
</tr>
<tr>
<td>r</td>
<td>162</td>
<td>z</td>
<td>172</td>
</tr>
<tr>
<td>s</td>
<td>163</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>164</td>
<td></td>
<td></td>
</tr>
<tr>
<td>u</td>
<td>165</td>
<td></td>
<td></td>
</tr>
<tr>
<td>v</td>
<td>166</td>
<td></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>167</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This section contains UPDATE, AUDPL, and MODECKS log file messages that are generated by UPDATE. There are three categories: UPDATE, AUDPL, and MODECKS.

B.1 UPDATE MESSAGES

The UPDATE program generates three types of log file messages:

- Informative: No action is taken.
- Error: Job aborts when UPDATE execution is finished, unless the UPDATE NA statement parameter is selected.
- Fatal error: Aborts execution immediately.

Under UNICOS, these messages are written to stderr. A code identifier, as shown, precedes messages, and they are listed numerically by this code. An explanation follows each message.

UD001 - PL: dn PL DATE: m/d/y LAST ID: id
The name of the dataset containing the PL is dn, m/d/y is the creation date of this PL version, and id is the name of the last identifier added to the PL. Class: informative.

UD002 - n UPDATE WARNINGS
UPDATE detected n probable user errors. If ML<4, E<>0, and L=0, warning messages are written to the listing and error datasets. Class: informative.

UD003 - EMPTY INPUT FILE, DN = dn (with COS)
UD003 - EMPTY INPUT FILE, FILENAME = file (with UNICOS)
The next file in COS dataset dn or UNICOS file file, specified as an input dataset/file for UPDATE, is empty. Determine whether the primary input file or READ datasets are nonnull or need to be rewound. Class: informative.

UD004 - DATASET NOT LOCAL, DN = dn (COS only)
The dataset indicated was not accessed before UPDATE execution. It was the PL, an input dataset, or was specified by a READ directive. Class: fatal error.
UD005 - RECURSIVE READ OF DN = dn
An attempt was made to read dataset dn recursively with the READ directive. Class: error.

UD006 - INVALID READ DATASET NAME, DN = dn
A READ directive encountered by UPDATE while reading input contains an invalid dataset name. Class: error.

UD007 - ERROR IN UPDATE CONTROL STATEMENT
One or more of the following control statement errors exist:

- Both the new PL and the old PL datasets have the same name
- Both F and Q were specified
- P=0 and I=0 (PL creation mode and no input)
- Invalid comment and/or master character
- Invalid DW value

Class: fatal error.

UD008 - MODS WITHOUT IDENTIFIER, NEW PL SUPPRESSED
A new PL cannot be generated with modifications that are not identified with an IDENT directive. Generation of a new PL has been suppressed. This occurs only when the N parameter specifies creation of a new PL, or in creation runs. Class: error.

UD010 - INVALID PROGRAM LIBRARY, DN = dn (with COS)
UD010 - INVALID PROGRAM LIBRARY, FILENAME = file (with UNICOS)
The dn/file is not in a PL format recognized by UPDATE. Class: fatal error.

UD011 - n FATAL UPDATE ERRORS
Update detected n fatal errors. Error messages are written to the listing and error datasets. Class: informative.

UD012 - PL FORMAT CONVERSION COMPLETE
A sequential format PL has been internally rewritten as a random format PL. Class: informative.

UD013 - SEQUENCE NUMBER EXCEEDS 131071 FOR ID = id
An attempt was made to add more than 131,071 lines with one identifier. The insertion must be split over two or more identifiers or decks. Class: fatal error.

UD014 - NUMBER OF IDENTIFIERS EXCEEDS 16383
Too many deck, common deck, and modification set identifiers are defined for this PL. Before any new identifiers can be added, you must resequence the PL by creating a new PL from the source dataset. Class: fatal error.
UD015 - DECK SPECIFIED BY Q PARAMETER NOT FOUND, DECK = dkname
The listed value for the Q control statement parameter was dkname, but it is not a deck or common deck in this PL, and it was not introduced by the input datasets. Class: error.

UD016 - PL MASTER CHARACTER IS: m
m is the master character recorded when the PL was created. It is the default for the master character used in the input and source datasets. This is written only if the master character is not the default (*). Class: informative.

UD017 - n MODIFICATION SETS SKIPPED
Because of unsatisfied IDENT directive dependency conditions, n modification sets were skipped. Use ML=1 on the UPDATE control statement to get a NOTE message written to the listing and error datasets for each skipped IDENT. Class: informative.

UD018 - n UNPROCESSED MODIFICATION DIRECTIVES
When UPDATE finished execution, n modification directives were left unprocessed, either because they modified decks and common decks that were not specified in a Quick mode UPDATE run, or because they referenced lines that were not found in the PL. Use the UM option on the UPDATE control statement to get a list of unprocessed modifications. Class: informative.

UD019 - n INPUT LINES TRUNCATED TO pldw CHARACTERS
n input lines longer than pldw characters were truncated to pldw characters. pldw is the number of characters per line stored in the PL, and it is defined by the DW control statement parameter. The minimum value, and default, for pldw is 80. Class: informative.

UD020 - MORE THAN 100 FATAL INPUT ERRORS
More than 100 fatal input errors were detected and UPDATE aborted. A DECK or COMDECK directive may be missing, or the wrong master character may have been specified. Class: error.

UD021 - n OVERLAPPING MODIFICATIONS
There were n directives that either referenced lines that were inserted earlier in the same UPDATE run or deleted a range of text that included newly inserted lines. Use ML=1 on the UPDATE control statement to get NOTE and CAUTION messages about overlaps to determine whether the overlaps were proper and expected. Class: informative.

UD022 - INVALID DC PARAMETER VALUE
The DC parameter on the UPDATE control statement is equated to an invalid value. Valid values are ON and OFF. Class: fatal error.

UD023 - INTERNAL UPDATE ERROR: ID NOT IN SEQUENCE TABLE
An UPDATE internal logic error caused the current identifier name not to be found in the Sequence Table. Class: fatal error.
UD024 - INTERNAL UPDATE ERROR: INVALID DIRECTIVE KEY
An UPDATE internal logic error caused an invalid directive key to be assigned. Class: fatal error.

UD025 - INTERNAL UPDATE ERROR: PL I/O STATUS ERROR
An UPDATE internal logic error caused an I/O status error to occur during a read of the PL. Class: fatal error.

UD026 - INTERNAL UPDATE ERROR: $UDT1 I/O STATUS ERROR
An UPDATE internal logic error caused an I/O status error in a character read of temporary dataset $UDT1. Class: fatal error.

UD027 - INTERNAL UPDATE ERROR: $UDT2 I/O STATUS ERROR
An UPDATE internal logic error caused a count exhaustion during a character read of temporary dataset $UDT2. Class: fatal error.

UD028 - INTERNAL UPDATE ERROR: ID NAME NOT IN IDENTIFIER TABLE
An UPDATE internal logic error caused an identifier name to be omitted from the Identifier Table. Class: fatal error.

UD029 - INTERNAL UPDATE ERROR: DECK NAME NOT IN IDENTIFIER TABLE
An UPDATE internal logic error caused an identifier name to be missing from the Identifier Table; thus, it was not found when UPDATE tried to get the name of the next deck to process. Class: fatal error.

UD030 - INTERNAL UPDATE ERROR: ID NUMBER NOT IN IDENTIFIER TABLE
An UPDATE internal logic error caused an identifier to be missing from the Identifier Table. Class: fatal error.

UD031 - INTERNAL UPDATE ERROR: ERROR IN ID LIST IN OLD FORMAT PL
UPDATE was unable to read the identifier list in an old PL format. Class: fatal error.

UD032 - INTERNAL UPDATE ERROR: OLD FORMAT PL IS UNREADABLE
UPDATE was unable to read an old PL format. Class: fatal error.

UD033 - INTERNAL UPDATE ERROR: PL INFORMATION FILE READ ERROR
UPDATE had a read error on a partial record read of the PL information file. Class: fatal error.

UD034 - INTERNAL UPDATE ERROR: UNKNOWN PROCESS TYPE
An UPDATE internal logic error caused an invalid process type (INSERT, BEFORE, DELETE, or RESTORE) to be returned from a Modification Table entry. Class: fatal error.

UD035 - INTERNAL UPDATE ERROR: DECK DIRECTIVE IN COMDECK
An UPDATE internal logic error caused a DECK directive to be placed in a common deck instead of starting a new deck. Class: fatal error.

UD036 - INTERNAL UPDATE ERROR: COMDECK DIRECTIVE IN DECK
An UPDATE internal logic error caused a COMDECK directive to be placed in a common deck instead of starting a new common deck. Class: fatal error.
UD037 - LIST CONTROL STATEMENT PARAMETER IGNORED
The LIST control statement parameter is not supported and is ignored.
Class: informative.

UD038 - INTERNAL UPDATE ERROR: NEW PL TABLE IS UNSORTED
A call to library routine ORDERS failed; the new PL is ordered as specified by the K option, but PLs built from it will revert to the old ordering. Class: informative.

UD039 - INTERNAL UPDATE ERROR: NO SECOND DELETE ENTRY
An UPDATE internal logic error caused the second entry in the Modification Table for a DELETE or RESTORE range to be missing. Class: fatal error.

UD040 - INTERNAL UPDATE ERROR: BAD HDC IN PL LINE
When handling deleted lines with bad correction histories, an active line was found with a header descriptor count of 0. Class: fatal error.

UD041 - n MULTIPLE INSERTIONS
There were n locations in which new text was inserted by directives in more than one modification set. Use ML=2 on the UPDATE control statement to write CAUTION messages to the listing dataset for each multiple insertion. Class: informative.

UD042 - INVALID ML PARAMETER VALUE; MUST BE 0-4
An invalid value was specified for the ML parameter on the UPDATE control statement. Class: fatal error.

UD043 - INTERNAL UPDATE ERROR: DELETE TABLE ENTRY NOT FOUND
An UPDATE internal logic error caused an entry to be missing from the Delete Table. Class: fatal error.

UD044 - DATASET dname USED FOR MORE THAN ONE PURPOSE
Dataset dname was specified by more than one control statement parameter (for example, both C and S). Class: fatal error.

UD045 - DW=* CANNOT BE APPLIED TO A FIXED LENGTH RECORD PL
A regular PL has been specified with a variable-length record specifier. This conversion is not allowed. Class: fatal error.

UD046 - INVALID DW VALUE FOR A VARIABLE LENGTH RECORD PL
A variable-length record PL was specified with a DW value other than "*". This is not allowed. Class: fatal error.

UD047 - INTERNAL UPDATE ERROR: INCONSISTENT CC AND DW, PLDW
An inconsistency was detected in the PL information file. Class: fatal error.
UD048 - NUMBER OF COMPILe FILES EXCEEDED. APPENDED IS A: numcpl (with UNICOS only)
The number of created compile files exceeds the number specified by the -a option or the default value of 4; numcpl is the number given to the exceeded compile file. This additional file is appended with numcpl (starting with 1 and incrementing until all files are labeled). Check the UNICOS command line options -c and -a. Class: caution.

UD050 - ATTEMPT TO act FILE FAILED. type FILENAME = file (with UNICOS only)
The reason for failure is provided in the following act descriptions:

OPEN UPDATE tried to open either a type PL or NPL (new PL) file. The type specified is either PL FILE or NPL FILE. (If NPL is the type, the new PL must have write access. If PL is the type, the PL must have read access.) This message is followed by the name of the file UPDATE is trying to open. If the type is PL FILE, check the -p option on the command line and ensure that the file name is an existing UPDATE PL. If the type is NPL FILE, UPDATE cannot create the file specified by the -n option. You may not have permission to create this file, or the file name may be invalid.

WRITE UPDATE tried to write to the NPL FILE (type). The file name (file) indicates the specific deck, or comdeck, or UPDATE directory in which this write occurred. TIDENT and PLINFO names mean that these tables, in the UPDATE directory, could not be written to the NPL. The NPL file name may be in error and cannot be used. This error may be caused by an empty input file. There was an attempt to write the UPDATE directory, but no decks or comdecks were written first. (There must be decks or comdecks to which the write is directed.) This message is followed by the name of the file to which UPDATE is writing.

SEEK UPDATE tried to position a PL or NPL file to a specific deck, comdeck, or UPDATE directory. The type indicates the location this seek occurred. TIDENT and PLINFO names mean that UPDATE was unable to locate this directory table. This message is followed by the name of the file to which UPDATE is trying to position.

CLOSE UPDATE tried to close either the PL or NPL file. The type given is either PL FILE or NPL FILE. The name of the file UPDATE is trying to close follows.

UD051 - Expected *FILE or *CFILE AT SEQUENCE: identifier (UNICOS only).
UPDATE is attempting to write output to the compile file, -c was specified with no argument, and no *FILE or *CFILE directive was encountered. Class: fatal.
B.2 AUDPL LOG FILE MESSAGES

AUDPL generates three classes of log file messages:

- **Informative:** No action is taken.
- **Error:** If the NA option was used, AUDPL continues processing, using defaults where necessary, and aborts when it is done. Otherwise, it aborts immediately when the error is detected.
- **Fatal error:** AUDPL aborts immediately when the error is detected.

Log file messages are preceded by a code identifier as shown, and they are listed numerically by this code identifier. An explanation follows each message.

PL001 - PROGRAM LIBRARY REQUIRED; SPECIFY P OR ACCESS $PL
You cannot use AUDPL without a PL for input. P=0 was specified on the control statement, or the P parameter was not used and dataset $PL is not local. Class: fatal error.

PL002 - LISTING OPTION k USED MORE THAN ONCE
One of the listing options for the LO parameter was used twice. Class: error.

PL003 - KEYWORD k MUST BE EQUATED
Keyword k on the control statement was used without being equated to a value. Class: error.

PL004 - INVALID DATA WIDTH, VALID RANGE IS 1-256
The value used with the DW control statement parameter was not in the range 1 through 256. Class: error.

PL005 - INVALID LISTING WIDTH, LW MUST BE 80 OR 132
The value used with the LW control statement parameter was not 80 or 132, optionally preceded by 'C'. Class: error.

PL006 - k IS NOT A VALID LIST OPTION
One of the letters in the string for the LO control statement parameter was not a valid list option. Class: error.

PL007 - INVALID JUSTIFICATION VALUE, MUST BE C, L, OR U
The value specified for the JU control statement option was not C, L, or U. Class: error.

PL008 - DATASET dsname USED FOR MORE THAN ONE PURPOSE
The same dataset name was equated to more than one of the control statement parameters that specify datasets used by AUDPL. Class: fatal error.
PL009 - PROGRAM LIBRARY NOT LOCAL
The dataset specified with the P control statement parameter was not local to the job. Class: fatal error.

PL010 - INPUT DATASET NOT LOCAL
The dataset specified with the I control statement parameter was not local to the job. Class: fatal error.

PL011 - OLD PL; RUN THROUGH UPDATE FIRST
The PL in the dataset specified with the P control statement parameter was written by an UPDATE from before UPDATE release 1.13. Write a new PL from the old one with an UPDATE from release 1.13 or higher, and use the new PL as input to AUDPL. Class: fatal error.

PL012 - PROBLEM READING PL
AUDPL is unable to read the PL in the dataset specified with the P control statement parameter. Check to see that the dataset contains a valid PL, and if so, report the problem to a system programmer. Class: fatal error.

PL013 - NAME IN DK LIST NOT IN PL, ID = idname
One of the names in the list for the DK control statement parameter was not in the identifier directory for the PL. Check the spelling of the name listed. Class: error.

PL014 - IDENTIFIER IN DK LIST IS NOT A DECK, ID = idname
One of the identifiers in the list for the DK control statement parameter is a modification set, not a deck or common deck. Class: error.

PL015 - BACKWARD RANGE IN DK LIST: idname1 TO idname2
In the identifier range specified in the message, the second identifier comes before the first identifier in the identifier directory for the PL. Check the identifier list from the L list option for the order of identifiers in the PL. Class: error.

PL016 - SYNTAX ERROR IN DK LIST
The list of identifiers given for the DK control statement parameter contains a syntax error, possibly because the two list types were combined. Class: error.

PL017 - NAME IN PM LIST NOT IN PL, ID = idname
One of the identifiers in the list for the PM control statement parameter is not in the identifier directory for the PL. Class: error.

PL018 - BACKWARD RANGE IN PM LIST: idname1 TO idname2
In the identifier range specified in the message, the second identifier comes before the first identifier in the identifier directory for the PL. Check the identifier list from the L list option for the order of identifiers in the PL. Class: error.
PL019 - SYNTAX ERROR IN PM LIST
The list of identifiers given for the PM control statement parameter contains a syntax error, possibly because the two list types were combined. Class: error.

PL020 - n INPUT DIRECTIVE ERRORS
There were n errors detected in the AUDPL input directives. Error messages for input errors are written to the listing dataset. Class: error.

PL021 - WARNING, NO ACTION REQUESTED OF AUDPL; USE I, LO, PM, OR B
The AUDPL control statement did not specify any actions to be taken through the LO, PM, or B parameter, and no input dataset was specified. Class: informative.

PL101 - INTERNAL AUDPL ERROR: INVALID IDENTIFIER NUMBER
An AUDPL internal logic error caused an invalid identifier number to be used as the identifier for a text line. Report the problem to a system programmer. Class: fatal error.

PL102 - INTERNAL AUDPL ERROR: ORDERS ROUTINE FAILED
An AUDPL internal logic error caused the return status of $SCILIB routine ORDERS indicating that it was unable to sort the table passed to it by AUDPL. Report the problem to a system programmer. Class: fatal error.

B.3 MODECKS ERROR MESSAGES

MODECKS error messages are preceded by a code identifier as shown, and they are listed numerically by this code identifier. An explanation follows each message.

MD001 - SOURCE DATASET NOT SPECIFIED
The source dataset was not specified. Class: fatal error.

MD002 - EMPTY DATA FILE. DN=__________
The specified file was empty. Class: informative.

MD003 - NO SEQUENCE NUMBERS FOUND. DN=__________
The specified source file did not contain sequence numbers in UPDATE format. Class: fatal error.

MD004 - NO UPDATE DIRECTIVE. DN=__________
The specified file did not contain a recognized UPDATE directive as the first line in the file; that is, a DECK or COMDECK directive. Class: fatal error.
MD005 - DECK TOO LARGE, DN=__________
The specified deck was too large (more than 500,000 lines) to fit into
the tables. Class: fatal error.

MD006 - MODECKS WILL NOT SUPPORT DW GREATER THAN 128
Because of lack of space, a maximum of 128 characters can be on one line
in the edit or source dataset. Class: fatal error.

MD007 - ________ IS A DUPLICATE DECK
Two decks were encountered with the same name. Class: fatal error.

MD008 - INTERNAL MODECKS ERROR, TABLES CORRUPTED
Because of a logic error, the tables have become corrupted. Class: fatal error.

MD009 - INTERNAL MODECKS ERROR ________ NOT FOUND IN TABLES
Because of a logic error, the specified deck name was not found in the
the table. Class: fatal error.

MD010 - NO MORE IDENTIFIERS SPECIFIED, USING ________ AGAIN
The identifiers specified were not enough to cover all decks that have
been changed in this mod. The specified identifier is duplicated.
Class: caution.

MD011 - DECK IN RANGE NOT FOUND; DECK = __________
The deck specified in the ordering range was not found in the edit file.
Class: fatal error.

MD012 - ________ FOLLOWS ________
The decks given in the ordering range were specified in the incorrect
order. Class: fatal error.

MD013 - ERROR IN ORDER SYNTAX
There was an error in the ordering syntax specified. Class: fatal error.

MD014 - UNABLE TO OPEN FILE. DN=__________
An error was encountered when attempting to open the specified file.
Class: fatal error.

MD015 - IDENTIFIER ________ TRUNCATED TO 7 CHARACTERS
The specified identifier was longer than 7 characters; therefore, it is
truncated to 7 characters. Class: caution.

MD016 - MORE THAN 64 IDENTIFIERS SPECIFIED.
Too many identifiers were specified on the command line. The usage is
printed out. Class: fatal error.

MD017 - PREMATURE END OF FILE ON TEMPORARY FILE.
When creating the mod header, the temporary file was incorrect. Class:
internal fatal error.
MD018 - INTERNAL MODECKS ERROR, NO CHANGED DECKS.
While creating the mod header, no changed decks were detected. Class: internal fatal error.

MD019 - MORE THAN 500 DEPENDENCIES. NONE WRITTEN.
There were more than 500 dependencies in the generated mod; none were written. Class: informative.

B.4 UPIC LOG FILE MESSAGES
UPIC messages are coded messages in the range UP000 through UP099. They include general messages, messages on control statement errors, and internal error messages. UPIC runs only under COS.

B.4.1 GENERAL MESSAGES
UP001 - UPIC Version n.nn
This gives the UPIC release version number. Class: informative.

UP002 - Page size is nnnnnn
This gives the page size UPIC is using for the output dataset. Class: informative.

UP003 - nnnnnnn lines read, nnnnnnn lines written to xxxxxxxx
UPIC prints this message when it has completed processing. xxxxxxxx is the output dataset name. Class: informative.

UP004 - nnnnnnn unrecognized lines read
UPIC failed to recognize one or more lines in the input dataset. The lines could be from an unsupported language processor, or they could contain any other text sandwiched between listings. This message is always preceded by one or more UP006/UP007 message pairs. Class: informative.

UP005 - nnnn.nnn seconds processing time
UPIC completed scanning the input dataset in nnnn.nnn seconds. Class: informative.

UP006 - Unknown listing page, begins ...
UPIC failed to recognize one or more lines in the input dataset. The lines could be from an unsupported language processor or any other text sandwiched between listings. This message is always followed by a UP007 message, which contains the first 72 characters of the unrecognized line. Up to 10 UP006/UP007 message pairs may be printed. Once UPIC has completed processing, it gives a count of unrecognized lines in the UP004 message. Class: informative.
UP007 - <first 72 characters of unrecognized line>
This is the companion message to UP006. See UP006 for a complete
discussion. Class: informative.

B.4.2 CONTROL STATEMENT ERROR MESSAGES

UP011 - CL parameter out of range 1 - nnn
Check your UPIC control statement to ensure you have specified a legal
value for this parameter. Class: fatal.

UP012 - PD option must be 6 or 8 if specified
Check your UPIC control statement to ensure you have specified a legal
value for this parameter. Class: fatal.

UP013 - PS parameter must be positive
Check your UPIC control statement to ensure you have specified a legal
value for this parameter. Class: fatal.

UP014 - * may only be set to a 1-character value
Check your UPIC control statement to ensure you have specified a legal
value for this parameter. Class: fatal.

B.4.3 UPIC INTERNAL ERROR MESSAGES

UP021 - Unknown processor type: xxxxxxxx
UPIC internal error. Routine PRNT does not recognize the noted type of
language processor. Have your systems programmer verify that the main
program (UPIC), and subroutines CKPROC and PRNT, all support the named
processor type. Class: fatal.

UP022 - UPDATE ID table overflow. Maximum is nnnnn.
Probable error in specifying ID and NOID parameters, too many wildcard
character combinations. If you need more Identifier table space,
recompile UPIC with parameter MAXID set to a larger value. MAXID is
declared in common deck UPICBLK. Class: fatal.

UP023 - Debug (ckproc): Processor type is xxxxxxxx
Routine CKPROC produces this message every time UPIC detects a new
listing page. UPIC issues this message only if you have specified the
DEBUG control statement option. Class: informative.
C. UPDATE PROGRAM LIBRARY FORMATS

UPDATE accepts two program library (PL) formats. Format 1 PLs can be read only. These libraries were created with UPDATE 1.05 or earlier, but they are still available to you. As a preliminary step, UPDATE always internally converts PL format 1 to PL format 2. When new PLs are written, the format 2 structure is always used. UPDATE completely controls the format of the PLs.

C.1 FORMAT 1 - SEQUENTIAL PL STRUCTURE (COS ONLY)

Format 1 (figure C-1) is a sequential PL structure and applies only to COS operation. Each section of a sequential PL (decks and lists) is separated by an end-of-file (EOF). Decks consist of individual line images that, in a sequential PL, are not separated by an end-of-record (EOR). Details of each format follow.

```
<table>
<thead>
<tr>
<th>Deck 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deck 2</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Deck n</td>
</tr>
</tbody>
</table>
| Deck List | One 2-word entry for each DECK/COMDECK
| ID List | One 1-word entry for each identifier name
| Master Word |
```

Figure C-1. PL Format 1
PL line format:

\[
\begin{array}{cccccccccc}
0 & 8 & 16 & 24 & 32 & 40 & 48 & 56 & 63 \\
0 & |S| & SEQ & |NCHAR| & UDC & |UD_1| & \\
1 & | & UD_2 & | UD_3 & | UD_4 & | UD_5 & \\
\frac{(UDC+2)}{4} & UD_{n-3} & | UD_{n-2} & | UD_{n-1} & | UD_n & \\
& n & | & Line Image & \\
\end{array}
\]

<table>
<thead>
<tr>
<th>Field</th>
<th>Word</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>0</td>
<td>0</td>
<td>Line status bit:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 Inactive</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 Active</td>
</tr>
<tr>
<td>SEQ</td>
<td>0</td>
<td>1-24</td>
<td>Sequence number</td>
</tr>
<tr>
<td>NCHAR</td>
<td>0</td>
<td>25-31</td>
<td>Number of characters in line text</td>
</tr>
<tr>
<td>UDC</td>
<td>0</td>
<td>32-47</td>
<td>Descriptor count</td>
</tr>
<tr>
<td>UD_1</td>
<td>0</td>
<td>48-63</td>
<td>First UPDATE descriptor (identifies name that introduces the line)</td>
</tr>
<tr>
<td>UD_i-UD_n</td>
<td>1-(UDC+2)/4</td>
<td></td>
<td>Modification descriptors giving deletion identifier names</td>
</tr>
</tbody>
</table>

The length of the line image is NCHAR bytes.

Descriptor format:

\[
\begin{array}{cccccc}
0 & 1 & 2 & 15 \\
|D|/| | ID & |
\end{array}
\]

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>0</td>
<td>Descriptor status:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 Deactivated line</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 Activated line</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td>ID</td>
<td>2-15</td>
<td>Identifier number or name</td>
</tr>
</tbody>
</table>

C-2
Deck list entry format:

<table>
<thead>
<tr>
<th>Field</th>
<th>Word</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>0</td>
<td>0-63</td>
<td>Name of DECK or COMDECK (left-justified, zero-filled)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>

ID list entry format:

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>0-63</td>
<td>Identifier name (left-justified, zero-filled)</td>
</tr>
</tbody>
</table>

Master word format:

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>0-7</td>
<td>PL check character (141g=lowercase A)</td>
</tr>
<tr>
<td>MC</td>
<td>8-15</td>
<td>PL master character</td>
</tr>
<tr>
<td>DKCNT</td>
<td>16-39</td>
<td>Length of the deck list</td>
</tr>
<tr>
<td>IDCNT</td>
<td>40-63</td>
<td>Length of the ID list</td>
</tr>
</tbody>
</table>

C.2 FORMAT 2 - RANDOM PL STRUCTURE

Format 2 (figure C-2) is a random PL structure. Each section of a random PL is separated by an EOF record, and line images within each deck are separated by EOR. However, the identifier table and PL information contain no EOR. Details of each format follow.
Figure C-2. PL Format 2

PL line format:

<table>
<thead>
<tr>
<th>0</th>
<th>S</th>
<th>0</th>
<th>SEQ</th>
<th>HDC</th>
<th>UD1</th>
<th>UD2</th>
<th>UD3</th>
<th>UD4</th>
<th>UD5</th>
</tr>
</thead>
<tbody>
<tr>
<td>(HDC+2)/4</td>
<td>UD_{n-3}</td>
<td>UD_{n-2}</td>
<td>UD_{n-1}</td>
<td>UD_{n}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Field | Word | Bits | Description
---|------|------|------------------
S | 0 | 0 | Line status:
   |   |    | 0 Inactive
   |   |    | 1 Active
   | 0 | 1-16 | Reserved
<table>
<thead>
<tr>
<th>Field</th>
<th>Word</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEQ</td>
<td>0</td>
<td>17-33</td>
<td>Line sequence number</td>
</tr>
<tr>
<td>HDC</td>
<td>0</td>
<td>34-47</td>
<td>Header descriptor count</td>
</tr>
<tr>
<td>UD₁</td>
<td>0</td>
<td>48-63</td>
<td>First UPDATE descriptor (specifies name introducing the line)</td>
</tr>
<tr>
<td>UDᵢ-UDₙ</td>
<td>1-((HDC+2)/4)</td>
<td></td>
<td>Modification descriptors</td>
</tr>
</tbody>
</table>

The format of each descriptor is identical to the corresponding descriptor fields in the PLs of format 1.

**Identifier Table format:**

<table>
<thead>
<tr>
<th>Field</th>
<th>Word</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>0</td>
<td>0-63</td>
<td>Identifier name (left-justified, zero-filled)</td>
</tr>
<tr>
<td>TYPE</td>
<td>1</td>
<td>0-7</td>
<td>Identifier type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 Modification</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 Deck</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 Common deck</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 Common deck/no propagation</td>
</tr>
<tr>
<td>T</td>
<td>1</td>
<td>8-9</td>
<td>Temporary flag (used internally, always 0 in PL)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>10-15</td>
<td>Reserved</td>
</tr>
<tr>
<td>Y</td>
<td>1</td>
<td>16</td>
<td>Yank flag:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 Mod, deck, or common deck not deactivated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 Mod, deck, or common deck deactivated</td>
</tr>
</tbody>
</table>

† Field is not used under UNICOS.
Field | Word | Bits | Description
--- | --- | --- | ---
C | 1 | 17 | Correction History Good flag:
   0 Correction history information not attached to deleted line
   1 Correction history present in PL for this modification
ID | 1 | 18-31 | Identifier number
POS | 1 | 32-63 | Position of deck within PL (0 if TYPE=0)

PL Information Table format:

| 0 | CC | MC | | IDCNT | | IDPOS |
|---|---|---|---|---|---|
| 1 | DATE |
| 2 | LEVEL |
| 3 | ///////////////// | DW | PLDW |
| 4 | ///////////////// | | |
| 5 | SIGNATURE |

Field | Word | Bits | Description
--- | --- | --- | ---
CC | 0 | 0-7 | PL check character:
   (142g=b - fixed-length record)
   (143g=c - variable-length record)
MC | 0 | 8-15 | Default master character for input and source datasets

NOTE

UPDATE 1.13 and higher use 252g internally to represent the PL master character, thereby allowing the master character for input directives to be changed. Because of this change, previous UPDATE releases cannot process input directives for PLs created by UPDATE 1.13 and higher.
<table>
<thead>
<tr>
<th>Field</th>
<th>Word</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLMC</td>
<td>0</td>
<td>16</td>
<td>Master Character flag. If set, master character on directives in PL is 252; otherwise, MC is used.</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>17</td>
<td>Reserved</td>
</tr>
<tr>
<td>IDCNT</td>
<td>0</td>
<td>18-31</td>
<td>Number of Identifier Table entries</td>
</tr>
<tr>
<td>IDPOS</td>
<td>0</td>
<td>32-63</td>
<td>PL position of start of Identifier Table</td>
</tr>
<tr>
<td>DATE</td>
<td>1</td>
<td>0-63</td>
<td>ASCII date of PL creation</td>
</tr>
<tr>
<td>LEVEL</td>
<td>2</td>
<td>0-63</td>
<td>Name of last identifier added to PL</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0-43</td>
<td>Reserved</td>
</tr>
<tr>
<td>DW</td>
<td>3</td>
<td>44-53</td>
<td>Default data width for compile and source datasets. If 0, 72 is used; if variable-length record PL, all bits are set (1777b).</td>
</tr>
<tr>
<td>PLDW</td>
<td>3</td>
<td>54-63</td>
<td>Number of characters saved for each line in the PL. If 0, 80 is used; if variable length record PL, all bits set (1777b).</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0-63</td>
<td>Reserved</td>
</tr>
<tr>
<td>SIGNATURE</td>
<td>5</td>
<td>0-63</td>
<td>'HIST OK' if bad correction histories were removed</td>
</tr>
</tbody>
</table>
D. PLCOPY UTILITY PROGRAM

You can use the utility program plcopy to convert blocked, single-filed COS program libraries (PLs) into multifiled, unblocked UNICOS PLs. The COS PLs must have been created with PL format 2 (UPDATE version 1.06 or higher). However, if the blank compression character in the PL is changed, plcopy does not work.

Format:

```
plcopy -i inpath [-o outpath]
```

- `-i inpath` Path and file name of the COS PL to be converted
- `-o outpath` Path of an empty UNICOS directory that receives the converted PL. If the directory you specified is not empty, plcopy aborts and no conversion occurs. If you do not specify a directory, the current working directory path is taken.

† UNICOS only
This appendix contains a summary of the UPDATE directives and their accepted abbreviations. A detailed description can be found on the referenced pages of this manual.

<table>
<thead>
<tr>
<th>Directive</th>
<th>Abbreviation</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>/comment</td>
<td>None</td>
<td>Indicates documentation directive</td>
<td>3-7</td>
</tr>
<tr>
<td>BEFORE</td>
<td>B</td>
<td>Inserts text before specified line</td>
<td>3-7</td>
</tr>
<tr>
<td>CALL</td>
<td>CA</td>
<td>Calls common deck</td>
<td>3-8</td>
</tr>
<tr>
<td>COMDECK</td>
<td>CDK</td>
<td>Defines common text sequence</td>
<td>3-8</td>
</tr>
<tr>
<td>COMPILE</td>
<td>C</td>
<td>Specifies compile dataset contents</td>
<td>3-9</td>
</tr>
<tr>
<td>COPY</td>
<td>CY</td>
<td>Copies text into new PL or dataset</td>
<td>3-10</td>
</tr>
<tr>
<td>CWEEOF</td>
<td>None</td>
<td>Conditionally writes end of file to compile dataset</td>
<td>3-11</td>
</tr>
<tr>
<td>DECK</td>
<td>DK</td>
<td>Defines text sequence</td>
<td>3-12</td>
</tr>
<tr>
<td>DECLARE</td>
<td>DC</td>
<td>Declares deck for modifications</td>
<td>3-12</td>
</tr>
<tr>
<td>DEFINE</td>
<td>DEF</td>
<td>Defines name used by IF</td>
<td>3-12</td>
</tr>
<tr>
<td>DELETE</td>
<td>D</td>
<td>Deletes line or text range</td>
<td>3-13</td>
</tr>
<tr>
<td>EDIT</td>
<td>ED</td>
<td>Removes inactive lines from deck</td>
<td>3-13</td>
</tr>
<tr>
<td>ELSE</td>
<td>None</td>
<td>Determines whether following text is written to compile dataset</td>
<td>3-14</td>
</tr>
<tr>
<td>Directive</td>
<td>Abbreviation</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>ELSEIF</td>
<td>None</td>
<td>Specifies a condition for evaluation when no condition in the same IF group was true</td>
<td>3-14</td>
</tr>
<tr>
<td>ENDF</td>
<td>None</td>
<td>Ends an IF group</td>
<td>3-15</td>
</tr>
<tr>
<td>FILE</td>
<td>None</td>
<td>Closes current compile dataset and opens new one (UNICOS only)</td>
<td>3-15</td>
</tr>
<tr>
<td>ENDSKIP</td>
<td>None</td>
<td>Ends a list of directives that are to be skipped</td>
<td>3-23</td>
</tr>
<tr>
<td>IDENT</td>
<td>ID</td>
<td>Defines modification set identifier</td>
<td>3-16</td>
</tr>
<tr>
<td>IF</td>
<td>None</td>
<td>Begins conditional text range and determines whether following text is written to compile dataset</td>
<td>3-17</td>
</tr>
<tr>
<td>INSERT</td>
<td>I</td>
<td>Inserts text after specified line</td>
<td>3-18</td>
</tr>
<tr>
<td>LIST</td>
<td>None</td>
<td>Starts input listing</td>
<td>3-18</td>
</tr>
<tr>
<td>MASTER</td>
<td>None</td>
<td>Changes the master character for input directives</td>
<td>3-19</td>
</tr>
<tr>
<td>MOVEDK</td>
<td>None</td>
<td>Alters deck position</td>
<td>3-19</td>
</tr>
<tr>
<td>NOLIST</td>
<td>None</td>
<td>Stops input listing</td>
<td>3-18</td>
</tr>
<tr>
<td>NOSEQ</td>
<td>None</td>
<td>Stops sequence number writing to compile dataset</td>
<td>3-22</td>
</tr>
<tr>
<td>PURGE</td>
<td>None</td>
<td>Removes a modification set permanently from the PL</td>
<td>3-20</td>
</tr>
<tr>
<td>PURGEDK</td>
<td>None</td>
<td>Removes a deck or common deck permanently from the PL</td>
<td>3-20</td>
</tr>
<tr>
<td>READ</td>
<td>RD</td>
<td>Reads input from alternate dataset</td>
<td>3-21</td>
</tr>
<tr>
<td>RESTORE</td>
<td>R</td>
<td>Reactivates deleted lines</td>
<td>3-21</td>
</tr>
<tr>
<td>Directive</td>
<td>Abbreviation</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------</td>
<td>-------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>REWIND</td>
<td>None</td>
<td>Rewinds a local dataset</td>
<td>3-22</td>
</tr>
<tr>
<td>SEQ</td>
<td>None</td>
<td>Begins sequence number writing to compile dataset</td>
<td>3-22</td>
</tr>
<tr>
<td>SKIP</td>
<td>None</td>
<td>Begins a list of directives that are to be SKIPPED</td>
<td>3-23</td>
</tr>
<tr>
<td>SKIPF</td>
<td>None</td>
<td>Skips over files in dataset (COS only)</td>
<td>3-24</td>
</tr>
<tr>
<td>UNYANK</td>
<td>None</td>
<td>Restores yanked deck or modification set</td>
<td>3-26</td>
</tr>
<tr>
<td>WEOF</td>
<td>None</td>
<td>Writes an EOF to the compile dataset</td>
<td>3-24</td>
</tr>
<tr>
<td>WIDTH</td>
<td>None</td>
<td>Changes line length in compile dataset</td>
<td>3-25</td>
</tr>
<tr>
<td>YANK</td>
<td>None</td>
<td>Temporarily removes a deck or modification set from a PL</td>
<td>3-26</td>
</tr>
</tbody>
</table>
INDEX
INDEX

Abort (AUDPL), 5-5
  none despite error, 2-15
Active lines directive (AUDPL), 5-10
ACTIVE (AUDPL), 5-10
APML with UPIC, 7-1
Assembly
  listings and UPIC, 7-2
  warnings, omit with UPIC, 7-4
Associativity of input, 1-14
AUDPL directives (see separate entry)
  control statement (see separate entry)
  ACTIVE, 5-10
  /comment, 5-9
  COND, 5-10
  DIR, 5-11
  HISTORY, 5-11
  INACTIV, 5-12
  PULLMOD, 5-12
  BEFORE, 1-7
  Binary identifier list dataset name
    (AUDPL), 5-2
  Blank compression, 1-13, D-1
AUDPL control statement (COS), 5-1
  binary identifier list dataset name, 5-2
  comment character, 5-3
  data width value, 5-3
  input dataset name, 5-2
  justification, 5-4
  listing dataset name, 5-2
  listing options, 5-5
  decks, 5-4
  listing width, 5-3
  master character, 5-3
  modifications dataset name, 5-2
  PL dataset name, 5-2
  pulled modification sets, 5-4
AUDPL directives, 5-9 through 5-13
  ACTIVE, 5-10
  /comment, 5-9
  COND, 5-10
  DIR, 5-11
  HISTORY, 5-11
  INACTIV, 5-12
  PULLMOD, 5-12

BEFORE, 1-7, 1-7
Binary identifier list dataset name
  (AUDPL), 5-2
Blank compression, 1-13, D-1

CAL with UPIC, 7-1
CALL, 1-10, 3-8
Called common decks, 2-2, 2-17
CFT and CFT77 with UPIC, 7-1
Changing the data width example, 4-16
Character set, A-1 through A-3
COMDECK, 1-6, 1-9, 1-11, 3-2, 3-8
Command line, 2-16
Command, convention, 1-3
Comment, 3-4
  AUDPL character (COS), 5-3
  AUDPL character (UNICOS), 5-7
  character, 2-4, 2-11, 2-18
Common deck program library, 1-11, 2-17
Common deck, 1-5, 1-6, 2-2, 3-8
Comparing decks with MODECKS, 6-1
Compilation
date, 1-15
listings and UPIC, 7-2
COMPIL, 3-9
Compile dataset, 1-5, 1-7, 2-4, 2-5
  contents, 1-12
  directives, 1-6, 1-7, 3-1
  CALL, 3-8
  CWOEO, 3-11
  ELSE, 3-14
  ELSEIF, 3-14
  ENDIF, 3-15
  FILE, 3-15
  IF, 3-17
  NOSEQ, 3-22
  SEQ, 3-22
  WEOF, 3-24
  WIDTH, 3-25
  from a common deck example, 4-13
  name, 2-2, 2-10
  Compile file name, 2-17
  Compile output, 1-11, 2-2
  Compression character, D-1
  COND (AUDPL), 5-10
  Conditional text directives, 1-7
  AUDPL, 5-10
  Conditional text
  example, 4-17
  summary, 1-16, 2-7, 2-15, 2-21
  Context lines with UPIC, 7-2
Control statement
AUDPL, 5-1
COS, 2-1
OPTION, 1-15, 5-13
UPIC, 7-2

Conventions, 1-3
Copy modification (AUDPL), 5-5
COPY, 1-14, 3-10
Correction history good flag, C-6

COS
command verb, convention, 1-3
execution, 2-1
listing extraction program, UPIC, 7-1
program library sequence, 1-8

COS control statement, 2-1 through 2-9
called common decks, 2-2
called comment character, 2-4
called compile dataset name, 2-2
called data width value, 2-4
called declared modifications option, 2-5
called error listing dataset name, 2-3
called input dataset name, 2-2
called listing dataset name, 2-3
master character, 2-4
message level, 2-6
mode run, 2-6
new PL dataset name, 2-3
output options, 2-7
PL dataset name, 2-1
source dataset name, 2-3

Creating a PL example, 2-8, 2-22, 4-1, 4-2
and a compile file, 4-2
reading input from an alternate
dataset, 4-2
full UPDATE mode, 4-3
calling common deck into user
program, 4-4
Creation run, 1-2, 2-3, 2-18
CWEOF, 1-7, 1-8, 3-11

Data flow, 1-2
Data width value, 2-4, 2-12, 2-18
(AUDPL), 5-3

Dataset
compile, 1-5, 1-7, 1-14
contents, 1-12
definition, 1-5
error, 1-14
input, 1-5
listing, 1-5, 1-15
magnetic tape, 5-1
source, 1-5, 1-6

Dataset contents example, 4-19
creating a new PL from an input file,
4-19
generating an executable program,
4-22
PL modification, 4-20
PL resequenced version, 4-23

Date, 1-15
Deactivate lines, 3-13
DECK, 1-5, 1-6, 1-8, 1-9, 3-4, 3-11

Deck, 1-4, 1-5
common, 1-5, 1-11
identifier, 1-10
name, 1-16
order, 2-7, 2-21
regular, 1-5
removal and positioning example, 4-15
source, 1-4, 1-5
DECLARE, 1-14, 3-12
Declared modifications, 1-15
option, 2-5, 2-13
DEFINE, 1-14, 3-13
Definitions, 1-5
DELETE, 1-6, 3-13
Differences, UPDATE and NUPDATE, 1-1
DIR (AUDPL), 5-11
Directives, 1-5, 3-1 through 3-26
AUDPL (see separate entry)
errors, examples of NA (na), 2-8, 2-23
format, 3-4
summary, E-1 through E-3
two new ones with NUPDATE, 1-1
UPDATE, 3-1

EDIT, 1-14, 3-13
Edited line summary, 2-7, 2-11, 2-21
ELSE, 3-14
ELSEIF, 3-14
ENDIF, 3-15
ENDSKIP, 3-23

Error
dataset name, 1-15, 2-3
directive, 2-9, 2-23
listing dataset name, 2-11
messages, 8-1
Examples, 4-1 through 4-24, 5-20
changing the data width, 4-16
compile dataset from a common deck, 4-13
conditional text, 4-17
creating a PL, 2-8, 2-22, 4-1, 4-2, 4-3
and a compile file, 4-2
reading input from an alternate
dataset, 4-2
full UPDATE mode, 4-3
calling common deck into user
program, 4-4
dataset contents, 4-19
creating a new PL from an input file,
4-19
PL modification, 4-20
generating an executable program,
4-22
PL resequenced version, 4-23
deck removal and positioning, 4-15
directive format examples, 3-6
extracting decks
for a source dataset, 4-11
for compilation (no source), 4-13
generating a compile dataset from
source, 4-12
generating and using common deck
PLs, 4-6
Examples (continued)

with common decks, 4-6
with external common decks, 4-7
modifying a PL with external common decks, 4-8
input dataset not $IN, 4-10
modifying a PL, 2-8, 2-22, 4-4, 4-5
generating a modified PL, 4-5
testing a modification set, 4-5
multiple input datasets, 4-10
PL editing, 4-16
read from alternative datasets, 4-9
resequencing a PL, 4-14
UNICOS input dataset, 4-11
UPIC, 7-1, 7-5
using *FILE, 4-14

Execution
COS, 2-1
UNICOS, 2-16
Externally defined common decks, 1-11
Extracting decks for
a source dataset example, 4-11
compilation (no source) example, 4-13

File, 1-4
FILE, 3-15
example, 4-11
Fixed-length record. C-6
Full mode, 1-12, 2-6, 2-13, 2-20

Generating a compile dataset from
source example, 4-12
Generating and using common deck PLs example, 4-6
with common decks, 4-6
with external common decks, 4-8
modifying a PL with external common decks, 4-8
Generation directives, 2-7, 2-15, 2-21

HISTORY (AUDPL), 5-11

IDENT, 1-5, 1-10, 1-14, 3-16
Identifier, 1-7, 1-8
names, 3-6
purged, 1-16
summary, 2-7, 2-15, 2-21
table, 1-11, 2-6, C-3
up to 240 with NUPDATE, 1-1
UPDATE and UPIC, 7-3
yanked, 1-16
IF, 3-17
IF directive names, 2-2, 2-10, 2-17
IN parameter and UPIC, 7-1
INACTIV (AUDPL), 5-12
Inactive lines directive (AUDPL), 5-8
IDENT.PL file, C-4
INFO.PL file, C-4
Input, 1-14
AUDPL dataset name (COS), 5-2
AUDPL file name (UNICOS), 5-6

Input (continued)
COS dataset name, 2-1
dataset, 1-5, 1-7
names, 2-2, 2-9
not $IN, example, 4-10
UNICOS input dataset example, 4-11
directives, 1-5, 1-6, 1-7
echo, 1-17
edit directives, 1-6, 3-3
EDIT, 3-13
ENDSKIP, 3-23
MOVEDK, 3-19
PURGE, 3-20
PURGEDK, 3-20
SKIP, 3-23
UNYANK, 3-26
YANK, 3-26
file name, 2-16
listing, 2-7, 2-21
UPIC, 7-1
INSERT, 1-7, 3-18
Insertion directive, 3-10
Invoking UPDATE, 2-1
under COS, 2-1
under UNICOS, 2-16

Job deck, 2-1
Job name, 1-15
Justification (AUDPL), 5-4, 5-7

LDR
maps with UPIC, 7-4
with UPIC, 7-1
Line identification, 3-5
sequence information, suppress, 2-15
LIST, 3-18
Listable output, 1-15
Listing, 2-1
dataset, 1-5, 1-15, 1-14
AUDPL, 5-9
dataset name, 2-2, 2-10
AUDPL 5-2
decks (AUDPL), 5-4, 5-7
extraction program, UPIC, 7-1
options, 1-16
AUDPL, 5-5, 5-8
UPIC, 7-4
width (AUDPL), 5-3
LOCATE (MODECKS), 6-2
Locate (-l option), 8-5
Log file messages, UPIC, B-11
Logical operator, 3-16, 3-18
LPP parameter, 1-15, 5-13

Magnetic tape dataset, 5-1
Maps, load, with UPIC, 7-4
MASTER, 1-14, 3-19
Master character, 2-4, 3-1, C-6
AUDPL, 5-3, 5-6
changing, 2-11
changing for UPIC, 7-3
Messages, 1-15, B-1 through B-12
level, 2-6, 2-11, 2-20
MODECKS, 6-1 through 6-10
COS example, 6-9
messages, B-9
UNICOS example, 6-10
MODECKS control statement (COS), 6-1
MODECKS command line (UNICOS), 6-5
Modes, 1-12, 2-6, 2-20
full, 1-12, 2-2, 2-6
normal, 1-12, 2-2, 2-6, 2-21
quick, 1-12, 2-2, 2-6, 2-20
Modification
commentary, omit with UPIC, 7-4
datasets (AUDPL), 5-2, 5-6, 5-19
declared, 1-14
directives, 1-6, 3-2
BEFORE, 3-7
COPY, 3-10
DELETE, 3-13
IDENT, 3-16
INSERT, 3-18
RESTORE, 3-21
errors, example of NA (na), 2-8, 2-23
history directive (AUDPL), 5-11
identifier, 1-7
overlapping, 1-14
run, 1-2, 2-3, 2-18
set, 1-5, 1-6, 1-8
set identifier, 1-6, 1-8, 3-5
write unprocessed, 2-8, 2-22
Modification file (mod), 6-1
Modifying a PL example, 2-8, 2-22, 4-4
generating a modified PL, 4-5
testing a modification set, 4-5
MOVEDK, 3-19
Multiple input datasets example, 4-11
COS input dataset, 4-10
UNICOS input dataset, 4-11
New PL
contents, 1-12
dataset name, 2-2
file name, 2-18
NOID parameter to UPIC, example, 7-5
NOLIST, 3-18
Normal mode, 1-12, 1-13, 2-2, 2-6, 2-14, 2-21
NOSEQ, 1-7, 3-22
NUPDATE
COS control statement, 2-9
introduction to, 1-1
mode, 2-13
OPTION control statement, 1-15, 5-13
Options (listing), 1-16, 2-7, 2-21
Ordering all decks, 2-15
Organizing UPDATE input, 1-14
Output, 1-15, 2-2
AUDPL, 5-13
options, 2-7, 2-21
UPIC, 7-2
Overlapping modifications, 1-14
Overview, 1-2
Page header lines, 1-15
Page number, 1-15
Parameter order, with COMPILE, 3-9
PL (see Program library)
PLOCOPY utility program, D-1
Procedure for PL modification, 1-10
Processing PL modifications, 1-11
Program library (PL), 1-5, 1-8
audit utility (AUDPL), 5-1 through 5-24
common deck, 1-11
creation, 1-9
COS example, 2-8
UNICOS example, 2-22
dataset name, 2-1, 2-9
AUDPL, 5-2, 5-6
new name, 2-10
editing example, 4-16
modification, 1-10
COS example, 2-8
UNICOS example, 2-22
restrictions, 1-9
organization
random, C-3
sequential, C-1
types, 1-13
Propagation parameter, 3-9
PULLMOD (AUDPL), 5-12
Pulled modification sets (AUDPL), 5-4, 5-8
directive, 5-12
PURGE, 1-7, 3-20
Purged identifiers, 1-16
PURGEDK, 1-14, 3-20
Quick mode, 1-12, 2-2, 2-6, 2-14, 2-20
Random PL organization, C-3
READ, 1-8, 1-21
Read from alternate datasets example, 4-9
Reconstructed AUDPL modification sets, 5-17
Record formats
regular length, 1-13
variable length, 1-13, 2-4, 2-6
Regular deck, 1-5
Regular length records, 1-13
Resequencing a PL example, 4-14
RESTORE, 1-7, 3-21
Restrictions, 1-9
AUDPL, 5-1
Revision level, 1-15
REWIND, 3-22
Rewind, 2-9
AUDPL, 5-5
none, 2-15
Run
creation, 1-2
modification, 1-2
Run option directives, 1-6, 3-3
/COMMENT, 3-7
COMPILE, 3-9

Index-4
Run option directives (continued)
COPY, 3-10
DECLARE, 3-12
DEFINE, 3-12
LIST, 3-18
MASTER, 3-19
NOLIST, 3-18
READ, 3-21
REWRITE, 3-22
SKIPF, 3-24

Sample listing (AUDPL), 5-20
SEGLDR maps with UPIC, 7-4
with UPIC, 7-1
SEQ, 3-22
Sequence
information, 2-5, 2-19
suppress line, 2-8
number, 1-8, 3-5
generation, example, 3-5
old scheme, 2-8, 2-15, 2-22
specification, NUPDATE, 2-16
Sequential PL organization, C-1
SKPL, 3-23
SKIPF, 3-24
SKOL with UPIC, 7-1
Source
code, 1-1
dataset, 1-5, 1-6, 1-7, 2-3, 2-5
contents, 1-12
name, 2-3, 2-11
dec, 1-4, 1-5, 1-6, 1-7
output update, 2-8
Standard error, 2-16
Standard out, 2-16, 2-21
Summary
edited line, 2-7, 2-21
identifier, 2-7, 2-21
conditional text, 2-7, 2-21
Symbolic reference map, list with UPIC, 7-4

Text
editor, 1-1
summary, 1-16
line options (AUDPL), 5-5, 5-8
Time, 1-15
UNICOS command line, 2-16
conventions, 1-3
execution, 2-16
program library sequence, 1-8
UNICOS command line, 2-16
compile file name, 2-17
data width value, 2-18
input file name, 2-16
IF directive names, 2-17
message level, 2-20
mode run, 2-20
new PL file name, 2-18
UNICOS command line (continued)
output options, 2-21
user common deck file name, 2-17
UNYANK, 1-5, 3-26
UPDATE, 2-1
data flow, figure 1-3
differences with NUPDATE, 1-1
identifiers and UPIC, 7-3
invoking, 2-1
under COS, 2-1
under UNICOS, 2-16
listings with UPIC, 7-2
messages, 8-1
modes, 1-12, 2-3, 2-6, 2-20
program library
formats, C-1
generate
from NUPDATE, 2-15
from UNICOS, 2-22
run, example, 4-16
UPDATE directives, 3-1 through 3-26
begin conditional text, 3-17
call common deck, 3-8
change input master character, 3-19
change line width in compile dataset, 3-25
close file, 3-15
comment, 3-7
conditionally skip a block of
directives, 3-23
conditionally write end-of-file, 3-11
copy text, 3-10
declare deck for modifications, 3-12
define names, 3-12
delete (restore) decks and modification sets, 3-21
delete lines, 3-13
delit decks, 3-13
delete lines, 3-13
derective texts, 3-15
delete line, 3-15
identify modification set, 3-16
insert after a line, 3-18
insert before a line, 3-7
introduce a common deck, 3-8
introduce a deck, 3-12
move a deck, 3-19
reactivate lines, 3-21
read alternative input, 3-21
remove deck, 3-20
remove modification set, 3-20
resume (stop) listing, 3-18
reverse condition, 3-14
skip dataset files, 3-24
specify compile or source datasets, 3-9
start (or stop) sequence number
writing, 3-22
summary of, E-1 through E-3
test condition, 3-14
write end-of-file, 3-24
UPIC
COS listing extraction program, 7-1
Log file messages, B-11
User
common deck file name, 2-17
field, 2-1
Variable length records, 1-13, 2-4, 2-6,
C-6
Verb, 1-3

WEOF, 1-7, 3-24
Width, data, value, 2-12
WIDTH, 1-7, 3-25
example, 4-16

Yank flag, 5-19, C-5
YANK, 1-7, 3-26
Yanked identifiers, 1-16
Reader's Comment Form

UPDATE Reference Manual

Your reactions to this manual will help us provide you with better documentation. Please take a moment to complete the following items, and use the blank space for additional comments.

List the operating systems and programming languages you have used and the years of experience with each.

Your experience with Cray Research computer systems: ______0-1 year ______1-5 year ______5+ years

How did you use this manual: ______in a class ______as a tutorial or introduction ______as a procedural ______as a reference ______for troubleshooting ______other

Please rate this manual on the following criteria:

<table>
<thead>
<tr>
<th>Category</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Appropriateness (correct technical level)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accessibility (ease of finding information)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical qualities (binding, printing, illustrations)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terminology (correct, consistent, and clear)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of examples</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quality of examples</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Index</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Please use the space below for your comments about this manual. Please include general comments about the usefulness of this manual. If you have discovered inaccuracies or omissions, please specify the number of the page on which the problem occurred.

Name __________________________ Address __________________________
Title __________________________ City __________________________
Company _________________________ State/Country __________________
Telephone _______________________ Zip code ______________________
Today's date ____________________ Electronic mail address ___________

Your reactions to this manual will help us provide you with better documentation. Please take a moment to complete the following items, and use the blank space for additional comments.

List the operating systems and programming languages you have used and the years of experience with each.

Your experience with Cray Research computer systems:    0-1 year  1-5 year  5+ years

How did you use this manual:  in a class  as a tutorial or introduction  as a procedural
                                  as a reference  for troubleshooting  other

Please rate this manual on the following criteria:

<table>
<thead>
<tr>
<th></th>
<th>Excellent</th>
<th>3</th>
<th>2</th>
<th>Poor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Appropriateness</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(correct technical level)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accessibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ease of finding information)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical qualities</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(binding, printing, illustrations)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terminology</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(correct, consistent, and clear)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of examples</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quality of examples</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Index</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Please use the space below for your comments about this manual. Please include general comments about the usefulness of this manual. If you have discovered inaccuracies or omissions, please specify the number of the page on which the problem occurred.