The purpose of the Ohio College Library Center (OCLC) computerized regional library system is to provide an on-line system that makes available to faculty and students in individual colleges and universities the library resources throughout a region, while at the same time decelerating the rate of rise of per-student library costs. The research and development culminated in the successful implementation of an on-line union catalog and shared cataloging system. The final report of the project is LI 004 422. This document contains appendix twenty-four, The Ohio College Library Center Program/Subroutine Documentation; Master Data Base Update (MDBUPD). The program MDBUPD (Master Data Base Update) is used to add the records from the weekly MARC tapes to the existing data base. This program takes a record that is in MARC format, checks the entries for validity, and converts it into the current bibliographic data file format. The program assigns an OCLC number to new records and adds them to the data base. MDBUPD replaces corrected or revised records and deletes unwanted records. It also returns various statistics. (Other appendices are LI 004 423, LI 004 424 and LI 004 426 through LI 004 428.) (Author/SJ)
Final Report

Project No. 9-0554
Contract No. OEC-0-70-2289 (506)

June 1973

THE DEVELOPMENT OF A COMPUTERIZED REGIONAL LIBRARY SYSTEM

APPENDIX 24

Frederick G. Kilgour
The Ohio College Library Center
1550 West Henderson Road
Columbus, Ohio 43220

Hillis D. Davis
Cooperative College Library Center
159 Forrest Avenue N.E.
Atlanta, Georgia 30303
APPENDICES

I. Instruction Manual for Catalog Production. (LI 004 423)
II. Manual for OCLC Catalog Card Production; Revised and Enlarged. Judith Hopkins. (LI 004 423)
III. Creation of Machine Readable Catalog Entries; An Adaptation of the "Data Preparation Manual: MARC Editors." (LI 004 423)
IV. Cataloging on a Cathode Ray Tube Terminal. (LI 004 423)
V. Brief Description of the Serials Control System: A Preliminary Report. (LI 004 424)
VI. A Preliminary Description of the OCLC Serials Control System. (LI 004 424)
VII. Manual for Checking-In, Binding, and Claiming of Serials on a CRT Terminal - Draft of Preliminary Procedures. (LI 004 424)
VIII. Suggested Minimum Requirements for Serials Cataloging. (LI 004 424)
IX. OCLC Technical Processing System - A Preliminary Outline. (LI 004 424)
XI. Recommended Standards for the Cataloging of Serials. (LI 004 424)
XII. Standards for Input Cataloging. (LI 004 424)
XIII. The Technical Processing System, August 1972. (LI 004 424)
XIV. Ohio College Library Center Annual Report, 1971/1972. (LI 004 424)
XVII. Name-Title Entry Retrieval from a MARC File. Philip L. Long and Frederick G. Kilgour. (Not Available EDRS)
XVIII. A Truncated Search Key Title Index. Philip L. Long and Frederick G. Kilgour. (Not Available EDRS)
XIX. Title-Only Entries Retrieved by Use of Truncated Search Keys. Frederick G. Kilgour, Philip L. Long, Eugene B. Leiderman and Alan L. Landgraf. (Not Available EDRS)
XX. Ohio College Library Center Systems. Frederick Kilgour. (Not Available EDRS)
XXI. Evolving, Computerizing, Personalizing. Frederick Kilgour. (Not Available EDRS)
XXII. The Shared Cataloging System of the Ohio College Library Center. Frederick G. Kilgour, Philip L. Long, Alan L. Landgraf, and John W. Wyckoff. (Not Available EDRS)

XXIII. Cataloging with a Computer - OCLC Comes to Pennsylvania. Robert C. Stewart. (Not Available EDRS)

XXIV. The Ohio College Library Center Program/Subroutine Documentation; Master Data Base Update (MDBUPD). (LI 004 425)

XXV. The Ohio College Library Center Program/Subroutine Documentation; Convert Call Number (CNVT). (LI 004 426)

XXVI. The Ohio College Library Center Program/Subroutine Documentation; Generate Pack Definition Tables (GENPDT). (LI 004 427)

XXVII. The Ohio College Library Center Program/Subroutine Documentation; Catalog Card Format Program (CCFP). (LI 004 427)

XXVIII. The Ohio College Library Center Program/Subroutine Documentation; Build Print Tape (BPT). (LI 004 428)
Master Data Base Update
(MDBUPD)
## CONTENTS

<table>
<thead>
<tr>
<th>Section/Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. OVERVIEW</td>
<td>I</td>
</tr>
<tr>
<td>II. DATA FLOW</td>
<td>II</td>
</tr>
<tr>
<td>III. SUMMARY OF INPUT AND OUTPUT</td>
<td>III</td>
</tr>
<tr>
<td>MARC Record Format</td>
<td>III.1</td>
</tr>
<tr>
<td>Bibliographic Record</td>
<td>III.6</td>
</tr>
<tr>
<td>IV. FUNCTIONS</td>
<td>IV</td>
</tr>
<tr>
<td>V. GENERAL INFORMATION FLOW</td>
<td>V</td>
</tr>
<tr>
<td>VI. SOFTWARE INTERFACE</td>
<td>VI</td>
</tr>
<tr>
<td>VII. DESCRIPTION OF SPECIAL STORAGE AREAS, SWITCHES, AND TABLES</td>
<td>VII</td>
</tr>
<tr>
<td>A. Special Tables</td>
<td>VIII.1</td>
</tr>
<tr>
<td>VIII. APPENDIX</td>
<td>VIII</td>
</tr>
<tr>
<td>A. Operating Requirements</td>
<td>VIII.2</td>
</tr>
<tr>
<td>B. Operating Characteristics</td>
<td>VIII.3</td>
</tr>
<tr>
<td>C. Detailed Description of Internal subroutines</td>
<td>VIII.12</td>
</tr>
<tr>
<td>D. Additional Subroutine Documentation</td>
<td>VIII.14</td>
</tr>
<tr>
<td>E. Additional Procedure Documentation</td>
<td>VIII.58</td>
</tr>
<tr>
<td>F. Validation Processes Performed by MDBUPD</td>
<td>VIII.61</td>
</tr>
</tbody>
</table>
I. Overview

The program MDBUPD (Master Data Base Update) is used to add the records from the weekly Marc tapes to the existing data base. This program takes a record that is in MARC format, checks the entries for validity, and converts it into the current bibliographic data file format. The program assigns an OCLC number to new records and adds them to the data base. MDBUPD replaces corrected or revised records and deletes unwanted records. It also returns statistics as to the number of records added, replaced, deleted, rejected, and so on. It also returns 3 copies of the error messages on paper and 1 copy on an output tape. The records that could not be processed for one reason or another are written on the deferred tape.

CAUTION: MDBUPD is not to be run during online processing.
II. DATA FLOW

PROGRAM: MDBUPD

II.1

MARC tape → MDBUPD → Deferred Tape

Results Tape

New Data Base Records

3 copies of results

Statistics on all replaced records
III. SUMMARY OF INPUT & OUTPUT

A. MARC TAPE (INPUT)

<table>
<thead>
<tr>
<th>Volume Header Label</th>
<th>File Header Label</th>
<th>T</th>
<th>File of Records for Monographs</th>
<th>T</th>
<th>End of File Label</th>
<th>T</th>
<th>M</th>
</tr>
</thead>
</table>

TM= Tape Mark (MARC Manual¹- p. 3)

2. File Header Label -- MARC Manual Page 2
3. File of Records for Monographs:

<table>
<thead>
<tr>
<th>0</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leader</td>
<td>Record Directory</td>
</tr>
</tbody>
</table>

a. Leader -- Page III.3
b. Record Directory:

<table>
<thead>
<tr>
<th>0</th>
<th>3</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>Length</td>
<td>Starting Character Position</td>
</tr>
</tbody>
</table>

1. MARC MANUAL Vol. 1
III.2

Program: MDBUPD

<table>
<thead>
<tr>
<th>Name of Record</th>
<th>Directory Data Element</th>
<th>No. of Char's</th>
<th>Char. Pos.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tag</td>
<td>3</td>
<td>0 - 2</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>4</td>
<td>3 - 6</td>
</tr>
<tr>
<td></td>
<td>Starting Char. Pos.</td>
<td>5</td>
<td>7 - 11</td>
</tr>
</tbody>
</table>

c. Control Fields

![Control Fields Diagram]

MARC Manual - Page 30 - 39

d. Variable Fields

![Variable Fields Diagram]

MARC Manual - Page 40 - 70

### Record Layout

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>MARC RECORD LEADER FORMAT (INPUT)</td>
</tr>
<tr>
<td>Record Name</td>
<td>____________</td>
</tr>
<tr>
<td>Record Type</td>
<td>( ) Card (x) Tape ( ) Disk ( ) Other ____________</td>
</tr>
<tr>
<td>File Organization</td>
<td>SEQUENTIAL</td>
</tr>
<tr>
<td>Record Size</td>
<td>____________</td>
</tr>
<tr>
<td>Block Size</td>
<td>____________</td>
</tr>
<tr>
<td>General Description</td>
<td>____________</td>
</tr>
</tbody>
</table>

### FIELD NAME AND DESCRIPTION

<table>
<thead>
<tr>
<th>FIELD NAME AND DESCRIPTION</th>
<th>POSITION</th>
<th>LENGTH</th>
<th>FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECORD LEADER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logical Record Length</td>
<td>0 - 4</td>
<td>5</td>
<td>ASCII</td>
</tr>
<tr>
<td>Record Status (MARC Manual</td>
<td>5</td>
<td>1</td>
<td>ASCII</td>
</tr>
<tr>
<td>Page 26)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type of Record (MARC Manual</td>
<td>6</td>
<td>1</td>
<td>ASCII</td>
</tr>
<tr>
<td>Page 26)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bibliographic Level (MARC</td>
<td>7</td>
<td>1</td>
<td>ASCII</td>
</tr>
<tr>
<td>Manual Page 26)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blanks</td>
<td>8 - 9</td>
<td>2</td>
<td>ASCII</td>
</tr>
<tr>
<td>Indicator Count (MARC Manual</td>
<td>10</td>
<td>1</td>
<td>ASCII</td>
</tr>
<tr>
<td>Page 27)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### III.4

Record Layout (Cont)

File Name  **MARC RECORD FORMAT**

Record Name

<table>
<thead>
<tr>
<th>FIELD NAME AND DESCRIPTION</th>
<th>POSITION</th>
<th>LENGTH</th>
<th>FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subfield Code Count (MARC Manual Page 27)</td>
<td>11</td>
<td>1</td>
<td>ASCII</td>
</tr>
<tr>
<td>Base Address of Data (MARC Manual Page 27)</td>
<td>12 - 16</td>
<td>5</td>
<td>ASCII</td>
</tr>
<tr>
<td>Encoding Level (MARC Manual Page 27)</td>
<td>17</td>
<td>1</td>
<td>ASCII</td>
</tr>
<tr>
<td>Blanks</td>
<td>18 - 19</td>
<td>2</td>
<td>ASCII</td>
</tr>
<tr>
<td>Entry Map (proposed - MARC Manual Page 27)</td>
<td>20 - 23</td>
<td>4</td>
<td>ASCII</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Logical Record Length</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Record Status</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Type of Record</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Logical Record Level</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Blanks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Indicator Count</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Subfield Code Count</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Base Address of Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Encoding Level</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Blanks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Entry Map</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Record Layout

File Name  BIBLIOGRAPHIC DATA FILE  

Record Name  BIBLIOGRAPHIC RECORD  

Record Type  ( ) Card ( ) Tape (x) Disk ( ) Other  

IBM Variable  

File Organization  Blk Format  Record Size  5-6144  Block Size  6144  

General Description  OCLC internal processing format of the MARC II  

Bibliographic Record. Access is either sequential or random.  

<table>
<thead>
<tr>
<th>FIELD NAME AND DESCRIPTION</th>
<th>FIELD</th>
<th>POSITION</th>
<th>LENGTH</th>
<th>FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECORD LEADER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logical Record Length</td>
<td></td>
<td>0-1</td>
<td>2</td>
<td>Binary</td>
</tr>
<tr>
<td>Record Status Character (MARC Manual - Page 26)</td>
<td></td>
<td>2</td>
<td>1</td>
<td>EBCDIC</td>
</tr>
<tr>
<td>Encoding Level (MARC Manual - Page 27)</td>
<td></td>
<td>3</td>
<td>1</td>
<td>EBCDIC</td>
</tr>
<tr>
<td>Leader Length - byte size of leader including terminator (X 'RD')</td>
<td></td>
<td>4</td>
<td>1</td>
<td>Binary</td>
</tr>
<tr>
<td>Type Index - index into a table of Material Type Indicator Codes (See Cataloging on a CRT Terminal - Page 32). Note that the zero entry is used.</td>
<td></td>
<td>5</td>
<td>Upper 4 bits</td>
<td>Binary</td>
</tr>
</tbody>
</table>
### Bibliographic Level Index - index
into a table of level codes
(See Cataloging on a CRT
Terminal - Page 32). Note that
the zero entry is used.

<table>
<thead>
<tr>
<th>FIELD NAME AND DESCRIPTION</th>
<th>POSITION</th>
<th>LENGTH</th>
<th>FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bibliographic Level Index - index</td>
<td>5</td>
<td>Lower, 4 bits</td>
<td>Binary</td>
</tr>
</tbody>
</table>

Reserved

<table>
<thead>
<tr>
<th>FIELD NAME AND DESCRIPTION</th>
<th>POSITION</th>
<th>LENGTH</th>
<th>FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>6-7</td>
<td>10 Bits</td>
<td>Binary</td>
</tr>
</tbody>
</table>

Variable Control Field Length -
Word length of field between
supplement number and suffix
character in LC card number.

<table>
<thead>
<tr>
<th>FIELD NAME AND DESCRIPTION</th>
<th>POSITION</th>
<th>LENGTH</th>
<th>FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable Control Field Length</td>
<td>7</td>
<td>Lower, 6 bits</td>
<td>Binary</td>
</tr>
</tbody>
</table>

OCLC Number

<table>
<thead>
<tr>
<th>FIELD NAME AND DESCRIPTION</th>
<th>POSITION</th>
<th>LENGTH</th>
<th>FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCLC Number</td>
<td>8-11</td>
<td>4</td>
<td>Binary</td>
</tr>
</tbody>
</table>

Date Entered

<table>
<thead>
<tr>
<th>FIELD NAME AND DESCRIPTION</th>
<th>POSITION</th>
<th>LENGTH</th>
<th>FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date Entered Year</td>
<td>12</td>
<td>1</td>
<td>Packed*</td>
</tr>
<tr>
<td>Date Entered Month</td>
<td>13</td>
<td>1</td>
<td>Packed*</td>
</tr>
<tr>
<td>Date Entered Day</td>
<td>14</td>
<td>1</td>
<td>Packed*</td>
</tr>
</tbody>
</table>

Type of Publication Date -
Description of contents of
Publication Date fields (See

<table>
<thead>
<tr>
<th>FIELD NAME AND DESCRIPTION</th>
<th>POSITION</th>
<th>LENGTH</th>
<th>FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of Publication Date</td>
<td>15</td>
<td>1</td>
<td>EBCDIC</td>
</tr>
</tbody>
</table>

Publications Dates

<table>
<thead>
<tr>
<th>FIELD NAME AND DESCRIPTION</th>
<th>POSITION</th>
<th>LENGTH</th>
<th>FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publications Dates Date #1</td>
<td>16-17</td>
<td>2</td>
<td>Packed*</td>
</tr>
<tr>
<td>Publications Dates Date #2</td>
<td>18-19</td>
<td>2</td>
<td>Packed*</td>
</tr>
</tbody>
</table>

Country of Publication - First two
characters of MARC field (See

<table>
<thead>
<tr>
<th>FIELD NAME AND DESCRIPTION</th>
<th>POSITION</th>
<th>LENGTH</th>
<th>FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Country of Publication</td>
<td>20-21</td>
<td>2</td>
<td>EBCDIC</td>
</tr>
</tbody>
</table>

Illustration Code Indexes - Four
4-bit indexes into the table of
Illustration codes (See MARC
Manual pp. 35). Note that the
zero entry is used to indicate
an invalid code was received
and that entry contains a
blinking blank.

<table>
<thead>
<tr>
<th>FIELD NAME AND DESCRIPTION</th>
<th>POSITION</th>
<th>LENGTH</th>
<th>FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Illustration Code Indexes</td>
<td>22-23</td>
<td>2</td>
<td>Binary</td>
</tr>
</tbody>
</table>

* Packed data is numeric data
which has had the upper four
bits of each numeral removed and
has been packed two digits per byte.
### Field Name and Description

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form of Content Code Indexes - four</td>
<td>4-bit indexes into a table of codes describing the form of work (See MARC Manual pp. 36-37). Note that the zero entry contains a blinking blank to indicate an invalid code was received.</td>
</tr>
<tr>
<td>Intellectual Level Index - index</td>
<td>into a table of intellectual level codes (See MARC Manual pp. 36). Note the zero entry is used to indicate that the input code was invalid and contains a blinking blank.</td>
</tr>
<tr>
<td>Format Reproduction Code Index - Index</td>
<td>into a table of codes describing the type reproduction, if any. Note the zero entry is used to indicate that the input code was invalid and contains a blinking blank.</td>
</tr>
<tr>
<td>Indicators 10 thru 15 - bit</td>
<td>switches to indicate the MARC indicators described in the MARC Reference Manual (pp. 37-38, par. 10-15). Bit values are</td>
</tr>
<tr>
<td>Biography Code Index - index into</td>
<td>a table of biography codes (See MARC Reference Manual pp. 33). Note that the zero entry contains a blinking blank to indicate an invalid code was received.</td>
</tr>
</tbody>
</table>
Record Layout (Cont)

File Name BIBLIOGRAPHIC DATA FILE
Record Name BIBLIOGRAPHIC RECORD

<table>
<thead>
<tr>
<th>FIELD NAME AND DESCRIPTION</th>
<th>POSITION</th>
<th>LENGTH</th>
<th>FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified Record Indicator Index - Index into a table of codes describing the type of change. Note that the zero entry contains the blinking blank character to indicate a code was received in error. (See MARC Manual pp. 38-39).</td>
<td>29</td>
<td>Upper 4 bits</td>
<td>Binary</td>
</tr>
<tr>
<td>Catalog Source Index - index into a table of codes to describe other sources of catalog records. (See MARC Manual - page 39.) Note that the zero entry contains a blinking blank to indicate an error code was received.</td>
<td>29</td>
<td>Lower 4 bits</td>
<td>Binary</td>
</tr>
<tr>
<td>Language Index - index into a table of language codes to describe the text of the data. Although the codes are not arranged exactly as shown, see the manual &quot;Cataloging on a Cathode Ray Tube Terminal&quot; pp. 46-52.</td>
<td>30-31</td>
<td>2</td>
<td>Binary</td>
</tr>
<tr>
<td>LC-Card Number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prefix</td>
<td>32-34</td>
<td>3</td>
<td>EBCDIC</td>
</tr>
<tr>
<td>Year Part</td>
<td>35</td>
<td>1</td>
<td>Packed*</td>
</tr>
<tr>
<td>Number Part</td>
<td>36-38</td>
<td>3</td>
<td>Packed*</td>
</tr>
<tr>
<td>Supplement number</td>
<td>39</td>
<td>1</td>
<td>EBCDIC</td>
</tr>
<tr>
<td>Length of 1st Author Substring</td>
<td>40-41</td>
<td>2</td>
<td>Binary</td>
</tr>
<tr>
<td>Displacement of 1st Author Substring</td>
<td>42-43</td>
<td>2</td>
<td>Binary</td>
</tr>
</tbody>
</table>

* Packed data is numeric data which has had the upper four bits of each numeral removed and has been packed two digits per byte.
## Field Name and Description

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Position</th>
<th>Length</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length of 2nd Author Substring</td>
<td>44-45</td>
<td>2</td>
<td>Binary</td>
</tr>
<tr>
<td>The number of bytes to use for the second author substring.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Displacement of 2nd Author Substring</td>
<td>46-47</td>
<td>2</td>
<td>Binary</td>
</tr>
<tr>
<td>Byte displacement to the 2nd author substring from end of leader.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Length of Title Substring</td>
<td>48-49</td>
<td>2</td>
<td>Binary</td>
</tr>
<tr>
<td>The number of bytes to use for the title substring.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Displacement to Title Substring</td>
<td>50-51</td>
<td>2</td>
<td>Binary</td>
</tr>
<tr>
<td>The byte displacement to the title substring from the end of the leader.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Holdings File Pointer Word</td>
<td>52-55</td>
<td>4</td>
<td>Binary</td>
</tr>
<tr>
<td>Pointer to holdings list.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Institutional Holdings Bits</td>
<td>56-71</td>
<td>16</td>
<td>Binary</td>
</tr>
<tr>
<td>Bit switches indicating holdings for an institution. A one indicates holdings, a zero indicates no holdings.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LC Suffix</td>
<td>Variable</td>
<td>Variable</td>
<td>EBCL</td>
</tr>
<tr>
<td>A variable length character string which may be absent.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Displacement to suffix equal to 40 + (4^n) where (n) equals the binary value of bits 2-7 of byte #7 of leader. Length of suffix is equal to the leader length, byte #4, minus the displacement to the suffix minus one.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Leader Terminator</td>
<td>Variable</td>
<td>1</td>
<td>Binary</td>
</tr>
<tr>
<td>X &quot;FD&quot; that follows the suffix to indicate the end of the leader.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### VARIABLE FIELDS

The following fields of the record are repeated for as many times as there are bibliographic elements. The fields are variable in the data that they contain and the length of each data item. The elements have the following format:

- **Tag** - element field descriptor number
- **Element Length** - length of element including tag
- **Subfields and Indicators** - the remainder of the element fields are identical to the MARC format with the exception that the '$a' subfield code is deleted if this field is present and the data begins immediately following the indicators. The code is a 'X'FD' for end of subfield and 'X'FE' for end of record.

**These values are the relative positions within the variable fields.**

**Packed data is numeric data which has had the upper four bits of each numeral removed and has been packed two digits per byte.**
### IBM DIAGRAMMING AND CHARTING WORKSHEET

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application Code</td>
<td>Date</td>
<td>Page 1 of 3</td>
</tr>
<tr>
<td>Procedure</td>
<td>Drawn by</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>LOGICAL RECORD LENGTH</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>RECORD STATUS</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>ENCODING LEVEL</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL LENGTH OF LEADING INDEX</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>TYPE IDENTIFICATION</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>RESERVED</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>WOODLAND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INDEXES</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCLC RECORD NUMBER</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE ENTERED ON FILE</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE (PACKED)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>YEAR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MONTH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAY</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TYPE OF PUBLICATION</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE (PACKED)</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>COUNTRY OF PUBLICATION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(TWO CHARACTERS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ILLUSTRATION CODES INDEX (2)</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#1</td>
<td></td>
<td>#2</td>
</tr>
<tr>
<td>#3</td>
<td></td>
<td>#4</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FORM OF COUNTRY CODE INDEX (TWO)</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#1</td>
<td></td>
<td>#2</td>
</tr>
<tr>
<td>33</td>
<td></td>
<td>34</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIBLIOGRAPHY CODE INDEX</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>PUBLISHED CATALOG RECORD SOURCE CODE INDEX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LANGUAGE CODE INDEX</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L.C. CARD NUMBER ALPHA PREFIX</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L.C. CARD NUMBER &quot;NUMBER&quot; PART</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E.C. CARD NUMBER SUPPLEMENT NUMBER</td>
<td>49</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LENGTH OF FIRST AUTHOR'S NAME STRING IN BYTES</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td>BYTE DISPLACEMENT TO FIRST AUTHOR'S NAME STRING FROM END OF CARD</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LENGTH OF SECOND AUTHOR'S NAME STRING IN BYTES</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>BYTE DISPLACEMENT TO SECOND AUTHOR'S NAME STRING FROM END OF CARD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Procedure</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>LENGTH OF TITLE STRINGING</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>POS DISPLACEMENT TO TITLE FROM END OF LEADER</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>HOLDINGS FILE POINTER WORD</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>WORD # 1 OF INSTITUTIONAL HOLDING SWITCHES</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>WORD # 2</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>WORD # 3</td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>WORD # 4</td>
<td></td>
</tr>
</tbody>
</table>

**Suffix** *LEADER TERMINATOR X'FF'*

* VARIABLE DATA WILL BE INSERTED HERE TO ASSIST IN INTERNAL PROCESSING AS THE NEED ARISES. TOTAL NUMBER OF WORDS BETWEEN L.C. SUFFIX AND SUPPLEMENT IS REFLECTED IN BITS 2-7 OF BYTE # 2 OF LEADER.  

** THE SUFFIX IS THE LAST FIELD IN THE LEADER SINCE IT IS VARIABLE IN LENGTH ANY MAY EVEN BE ABSENT; THE LENGTH IS EQUAL TO THE LEADER LENGTH (BYTE # 4) MINUS 40 MINUS FOUR TIMES THE VALUE OF BITS 2-7 OF BYTE # 7 MINUS ONE. THE ADDRESS OF THE SUFFIX IS EQUAL TO THE LENGTH OF THE FIXED LEADER (40) PLUS THE BINARY VALUE OF BITS 2-7 OF BYTE # 7 TIMES FOUR. **
Following the header are the tag fields. These fields are variable in length and immediately follow the previous field (or header for the first element). Each is terminated by a delimiter (X'FD' or X'FE') for the last element. They follow the basic make-up format with the following four exceptions:

1. The tag itself is binary.
2. The field length is in binary.
3. The first X'0A' is not included in the record if present. If the $ (X'FC') does not immediately follow the indicators, the character is data and the sub-field is the $X'0A'. Otherwise, the data is STD.
4. The character X'FC' replaces the $.

<table>
<thead>
<tr>
<th>TAG (BINARY)</th>
<th>LENGTH OF FIELD INCLUDING TAG &amp; TERMINATOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>INDICATOR 1</td>
</tr>
<tr>
<td></td>
<td>SIGN INDICATOR X'FD'</td>
</tr>
</tbody>
</table>

$ SUBFIELD RECORD TERMINATOR
Record Layout

File Name _BIBLIOGRAPHIC DATA FILE_

Record Name _AUTHOR TITLE & TITLE ONLY INDEX (AT & TO)_

Record Type - ( ) Card ( ) Tape ( ) Disk ( ) Other

File OrganizationRandom Record Size 12-bytes Block Size 1020

General Description This file is used to access the required Bibliographic Record(s) from the input Author-Title Key (3,3) which is hashed to a disk address within the file. If the input key and the title key within the record agree, the index points to one of the possible matching BIB records.

<table>
<thead>
<tr>
<th>FIELD NAME AND DESCRIPTION</th>
<th>POSITION</th>
<th>LENGTH</th>
<th>FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chaining Flag - Bit flag indicating end of chain. 0 = Not last in entry chain 1 = End of entry chain</td>
<td>0</td>
<td>Bit 0</td>
<td>Binary</td>
</tr>
<tr>
<td>Master BIB Record Present Flag - Bit flag indicating whether BIB record is present or absent (deleted) 0 = Master record deleted 1 = Master record present</td>
<td>0</td>
<td>Bit 1</td>
<td>Binary</td>
</tr>
<tr>
<td>Bibliographic Record Address - 30 bit disk address of Bibliographic record.</td>
<td>0-3</td>
<td>30 Bits</td>
<td>Binary</td>
</tr>
<tr>
<td>Author-Title &amp; Title Only Key - String of six EBCDIC characters which make up the 3,3 key of the bibliographic record pointed to by the index.</td>
<td>4-9</td>
<td>6</td>
<td>EBCDIC</td>
</tr>
<tr>
<td>Reserved</td>
<td>10-11</td>
<td>2</td>
<td>---</td>
</tr>
</tbody>
</table>
Record Layout

File Name **BIBLIOGRAPHIC DATA FILE**

Record Name **OCLC INDEX (OCLC)**

Record Type ( ) Card ( ) Tape ( ) Disk ( ) Other

File Organization Random Record Size 8-bytes Block Size

General Description This file is used to access the required Bibliographic Record(s) from the input OCLC # Key which is hashed to a disk address within the file.

<table>
<thead>
<tr>
<th>FIELD NAME AND DESCRIPTION</th>
<th>POSITION</th>
<th>LENGTH</th>
<th>FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chaining Flag - Bit flag indicating end of chain. 0 = Not last in entry chain 1 = End of entry chain</td>
<td>0</td>
<td>Bit 0</td>
<td>Binary</td>
</tr>
<tr>
<td>Master BIB Record Present Flag - Bit flag indicating whether BIB record is present or absent (deleted) 0 = Master record deleted 1 = Master record present</td>
<td>0</td>
<td>Bit 1</td>
<td>Binary</td>
</tr>
<tr>
<td>Bibliographic Record Address - 30 bit disk address of Bibliographic record.</td>
<td>0-3</td>
<td>30 Bits</td>
<td>Binary</td>
</tr>
<tr>
<td>Reserved</td>
<td>4-7</td>
<td>4</td>
<td>----</td>
</tr>
</tbody>
</table>
Record Layout

**File Name**  BIBLIOGRAPHIC DATA FILE

**Record Name**  LIBRARY OF CONGRESS CARD NO. INDEX (LCCN)

**Record Type**  - ( ) Card ( ) Tape (x) Disk ( ) Other

**File Organization**  Random  **Record Size**  12-bytes  **Block Size**  1020

**General Description**  This file is used to access the required Bibliographic Record(s) from the input LCCN key which is hashed to a disk address within the file. If the input key and the title key within the record agree, the index points to one of the possible matching BIB records.

<table>
<thead>
<tr>
<th>FIELD NAME AND DESCRIPTION</th>
<th>POSITION</th>
<th>LENGTH</th>
<th>FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chaining Flag - Bit flag indicating end of chain. 0 = Not last in entry chain 1 = End of entry chain</td>
<td>0</td>
<td>Bit 0</td>
<td>Binary</td>
</tr>
<tr>
<td>Master BIB Record Present Flag - Bit flag indicating whether BIB record is present or absent (deleted) 0 = Master record deleted 1 = Master record present</td>
<td>0</td>
<td>Bit 1</td>
<td>Binary</td>
</tr>
<tr>
<td>Bibliographic Record Address - 30 bit disk address of Bibliographic record.</td>
<td>0-3</td>
<td>30 Bits</td>
<td>Binary</td>
</tr>
<tr>
<td>LCCN Key - Library of Congress Card Number Prefix</td>
<td>4-6</td>
<td>3</td>
<td>EBCDIC</td>
</tr>
<tr>
<td>L.C. Card Number Part</td>
<td>7-10</td>
<td>4</td>
<td>Packed</td>
</tr>
<tr>
<td>L.C. Supplement</td>
<td>11</td>
<td>1</td>
<td>EBCDIC</td>
</tr>
</tbody>
</table>
IV. FUNCTIONS

A request is made on the OC device to mount the MARC tape on 9TA80 and the output tape for deferred records on 9TA81. If Sense Switch's are '0' the deferred tape is positioned past the last tape marks. If Sense Switches are not 0, a reply is requested from the operator. A reply of 'x' will abort the job. A reply of 'NEW' indicates that the deferred tape is new and to start processing. A reply of anything other than 'x' or 'NEW' will cause the request for a reply from the operator to occur again. An area called 'RESULTS' is assigned on the RAD for error and warning messages.

After initialization, the program 'READMARC' reads a record from the MARC tape. If this record is the volume header label or the end of file label, it is converted to F'1CDIC and written on the deferred tape.

The routine 'CASBI' is used to convert fields to binary, the result being returned in R1. The routine 'CASEB' is used to convert fields to EBCDIC, the address of the result being returned in the third word of the parm list.

The base address of the data and the record length are picked up from the MARC leader and converted to binary. Positions 6-11 of the leader are verified as being 'am 22'. Positions 18-23 are verified as being all blanks.

The record directory is taken from the MARC tape and a corresponding binary directory is built for the bibliographic record at location 'DIR', where each field has a 2-word entry:

Word #0 -- tag (1 byte), BA(field)
Word #1 -- length of field

The record status is now checked. If it is 'D' the record is to be deleted. This is accomplished by changing the Encoding level to 'J' and skipping the error checking and formatting of the record. Otherwise, the bibliographic record leader is built taking the data from the leader, field 001 and field 008 of the MARC record. The next available OCLC # is found and is put into the leader. Each data entry is converted from ASCII to either EBCDIC or PACKED, is checked for validity using the tables in MDBUPD, and is
moved to the appropriate position in the leader. There are 2 instructions indicated in the 'BLDLEADR' section of the program listing which must be changed if the last 8 control fields are to be added to the leader.

The length of each field and the indicators are checked for validity against the tables. Any invalid characters are reported as errors and replaced with blinking blanks (X'77'). If the field begins with a 'a' subfield code, the subfield code is deleted, and the rest of the field is converted to the proper format (either EBCDIC, binary or packed), checked for valid characters, and moved to the output buffer. Then the subfield codes are checked for validity using the table 'SFLD'. Each field is moved in this manner until an end of record indicator is encountered (X'FE').

If there were any errors in the record, the error message and the record are written on the RAD and printed in the format as shown in APPENDIX B. All Warning messages are also written on the RAD. The subroutine 'INDEXER' is then the author-title key, and the LC card number key. The subroutine 'SEARCHER' performs several checks on these keys. This MARC record is checked against the other existing MARC records. If the LC card numbers match, the new MARC is logged as a duplicate and is entered on the deferred tape. The MARC record is checked against the records input by the member libraries (those with encoding level 'I'). If all 3 keys are equal, the holdings from the data base record are added to the new MARC record and the data base record is deleted but its OCLC# is kept. If a weekly record that has a 'D' (delete) status has the same LCCN as an old MARC record, the old MARC record is deleted as long as it has no holdings. If it does have holdings, its encoding level is changed to 'I' and it remains on the data base. The new MARC record is checked against the old MARC records which have an encoding level of 'J' (deleted). If the LC card numbers are equal, the old data base record is replaced with the new MARC record. Then the new MARC record is checked against the user data base records and the MARC data base records which have an encoding level of 'J'.

If the card numbers are not present or are not the same, but the title key and the author-title key match, the record is written on the differed tape and awaits human intervention. If the new MARC record has been proven not to be a duplicate, the record is either written at the end of the Bibliographic file, or is written in an available slot where another record had been deleted.
Before reading the next MARC record, a check is made to determine if the previous record was a replacement record. If it was, the LC card number and the OCLC number of the Marc record and the OCLC number of the Data Base record are punched on the card punch. If the previous record was the last record on this weekly tape, the Summary statistics are written on the RAD file.
V. GENERAL INFORMATION FLOW

1. START
   - MOUNT WEEKLY & DEFERRED TAPES
     - SENSE SWITCHES
       - YES
         - POSITION DEFERRED TAPE
         - DEFERRED TAPE AFTER LAST TAPE MARK
       - NO
         - OPERATOR RESPOND

2. 'X'? (OPERATOR RESPOND)
   - YES
     - ABORT
   - NO
     - 'NEW'? (OPERATOR RESPOND)
       - NO
         - 2A
       - YES
         - GET NEXT WEEKLY RECORD

3. 1A

4. B

5. A
PROGRAM: MDBUPD

V.3

3A

ANY INVALID CHAR'S. YES

REPLACE WITH BLINKING BLANK X'77'

NO

MOVE FIELD OUTPUT BUFFER

REPORT ERROR

2B

ANY INVALID SUBFIELD CODES YES

REPLACE WITH BLINKING BLANK X'77'

NO

REACHED END OF RECORD

REPORT ERROR

YES

SET UP ACCESS KEYS
LCCN, TITLE
6 AT/TO

3B

Is there a record with same LCCN

NO

5B

YES

DB RECORD HAVE HOLDINGS

NO

YES

ADD HOLDINGS TO WEEKLY RECORD FROM DB RECORD

4A
PROGRAM: MDBUPD

1A

4A

STATUS = 'd'

NO

STATUS = 'n'

YES

DB record =Marc

NO

NO

DELETE DB RECORD & SAVE OCLC#

YES

3B

DUPLICATE WARNING MESSAGE

3B

IS DB RECORD NO MARC

YES

DOES DB RECORD HAVE HOLDINGS

YES

CHANGE DB ENCODING LEVEL TO 'J'

DELETE DB RECORD

REWRITE MODIFIED DB RECORD ON BIB FILE

1A
PROGRAM: MDBUPD

V.5

5A

- IS THERE A DB RECORD WITH SAME 3,3 or 3,1,1,1
  - YES
    - DB RECORD HAVE HOLDINGS
      - YES
        - ADD HOLDINGS TO WEEKLY RECORD
      - NO
        - DELETE DB RECORD & SAVE OCLC#
  - NO

C

- IS THERE AN AVAILABLE OCLC
  - NO
    - Len. of weekly < len. of DB
      - YES
        - ADD WEEKLY RECORD IN THIS SLOT
      - NO
        - GET A NEW OCLC#
  - YES
    - ADD RECORD AT END

1A
VI. SOFTWARE INTERFACE

A. Linkage - background linkage with OBM

B. Parameter List Description - none

C. Return Codes - none

D. Other Entry Points - none

E. OCLC Subroutines Referenced

CASEB
CASBI
MDBUPDLG
OPENMARC
READMARC
CLOSEMARC
PACK
UNPK
$MBS
$MBSS
NDX123
ADDMAST
RADDMAST
DELTMAST
LCCN0000 (Alternate EP,LCCN0000T)
INDEXER
VII. DESCRIPTION OF SPECIAL STORAGE AREAS, SWITCHES, AND TABLES

A. Special Storage
None

B. Special Switches
None

C. Tables

ECV - This is a table of sub-tables which will set up the table of parameters required for the program 'CBIEB' which will convert binary data to EBCDIC. An example of an 'ECV' table would be:

<table>
<thead>
<tr>
<th>Table</th>
<th>Data 0</th>
<th>Data 1</th>
<th>Data 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECV1</td>
<td>0</td>
<td>5,'C',+','C'-'</td>
<td>0</td>
</tr>
</tbody>
</table>

where the first word is the number to be converted; the second word contains the length in bytes to be converted, a fill character (usually a blank), a plus sign and a minus sign; the third word has the byte address of the area where the converted number is to go.

PACK_ _ _ _ The data statements PACK_ _ _ _ are used to generate the parameter list needed for the OCLC subroutine 'PACK' which is used to convert ASCII fields to packed format. A sample statement would be:

PACK LCC# DATA BA(UNPLCCN) +3,8, BA(PLCCN) +3,0

The first word gives the address of the field to be packed. The second word gives the number of bytes to be converted. The third word gives the byte address of where to put the packed field. The fourth word is where the condition of the conversion is passed.

CASBI3, CASBI4, CA2BI5 -- These are data statements which are used to set up the parameter list for the OCLC subroutine 'CASBI' which converts a field in ASCII format to binary. A sample data statement would be:

CASBI3 DATA 3,0,0

The first word gives the number of bytes to convert. The second word is used for the byte address of the field to be converted. The third word is used to return the condition of the conversion.
DELTPARM, ADDParm, RADDParm, RESTParm.–These are data statements which set up the parameter lists for the OCLC subroutines DELTMAST, ADDMAST, and RADDMAST. A sample data statement would be:

DELTPARM DATA 0, DBBUF, MASTWORK, ATKEY+2, DBADDR

The first half-word is where the status is returned. The second word is the word address of the bibliographic record. The third word is the word address of the work area. The fourth word is the word address of the returned OCLC entry. The fifth word is the word address of the index (for RADDMAST and DELTMAST only).

MSG TABLE – This is a table of partial text messages from which a message will be built. A sample entry would be:

MSG8 TEXTC 'FIELD TERMINATOR MISSING IN POSITION'

PRCS - This is a branch table of routines that are used to check the tag fields for required data and to check that the subfields are correct and in the correct sequence. The index to this table is retrieved from the table 'PRC'. The calling sequence for this table is:

LB, R1
PRC, R6
B
PRCS, R1

where R6 has the field index obtained from the table 'TAGS' for this field.

ERL - and ABL - These are tables of procedures that set up the FPT's and the parameters for printing the error messages. These tables are described more thoroughly in Section VIII, Part E with the description of the procedures 'MSG', 'DEC', 'HEX', and 'MEND'.

SUL – This is a table of sub-tables which set up the parameters for printing the messages concerning the weekly tape totals. An example of a sub-table would be:

SUL1 DEC ECV6
MSG MSG42
MEND

The procedure 'DEC' sets up the parameters for the conversion to EBCDIC. The procedure 'MSG' causes the message to be copied. The procedure 'MEND' indicates the end of the parameter list. For a more thorough description of the procedures, refer to Section VIII, Part E.
TAGS - This is a table of valid tag indicators. The procedure 'DTAG' will cause these entries to expand to a table with 841 byte entries. The procedure 'DTAG' is described more thoroughly in Section VIII, Part E. Each tag from 001 to 840 has a corresponding byte entry which can be found by using the tag number as an index. The byte entry in the table will be used as an index for that field into all the other tables. This byte will be called the field index. If the corresponding byte is 0 when the table is indexed by a tag number, that tag is invalid. A partial expansion of the table would be:

<table>
<thead>
<tr>
<th>T001</th>
<th>DATA,1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DATA,1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DATA,1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DATA,1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DATA,1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DATA,1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DATA,1</td>
<td>0</td>
</tr>
<tr>
<td>T008</td>
<td>DATA,1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>DATA,1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DATA,1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DATA,1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DATA,1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DATA,1</td>
<td>0</td>
</tr>
<tr>
<td>T015</td>
<td>DATA,1</td>
<td>3</td>
</tr>
</tbody>
</table>

In other words, the first entry is a '1' (field index for the 001 field); the eighth entry is a '2' (field index for the 008 field); the fifteenth entry is a '3' (field index for the 015 field), and so on. All other entries are '0' to indicate an invalid tag. This byte (the field index) is then stored at the beginning of the field's entry in the directory.

BTAGS - This is a table which contains the half-word binary value of the tag value. This table is indexed by the field index which was found from the table 'TAGS'.

MINL - This is a byte table whose entries indicate the minimum byte length acceptable for each tagged field. This table is indexed by the field index for this field. The 001 and 008 fields are handled separately.

IND1 - This is an indexable byte table of byte displacements into the 'NL' table. This table and the 'NL' table are used to verify and edit the first indicator for each individual field. The byte displacement into the 'NL' table for a particular field is picked up from this table using R6, the field index. If the byte
The displacement code for a field is 'FBL', the first indicator should be a blank and a blank is loaded into the record in the first indicator position. If the byte displacement code for a field is 'PBL' and the first indicator is a blank, it is replaced with a blanking blank. The indicator for fields 001 and 008 are handled separately.

**IND2**
This is an indexable byte table of byte displacements in the 'NL' table. This table and the 'NL' table are used to verify and edit the second indicator of each individual field. This table is set up the same as the table 'IND1'.

**SFLD**
This is an indexable byte table of byte displacements into the table 'AL'. This table and the table 'AL' are used to verify the subfield codes for each individual field. The byte displacement into the table 'AL' for a particular field is picked up from this table using R6 the field index. The subfield codes for the 001 and 008 fields are handled separately.

**PRC**
This is an indexable byte table of word indexes into the branch table 'PRCS'. This table and the table 'PRCS' are used to verify and edit the required data for each individual field. The word index into the table 'PRCS' for a particular field is picked up from this table using R6, the field index. The data for the 001 and 008 fields are handled separately.

**MINCNT**
An indexable byte table whose entries indicate the minimum allowable times a tagged field may appear within each MARC record. This table is indexed by the field index obtained from the table 'TAGS'.

**MAXCNT**
This is an indexable byte table whose entries indicate the maximum allowable times a tagged field may appear within each MARC record. This table is indexed by the field index obtained from the table 'TAGS'.

**AL**
This is an indexable byte table of valid subfield codes that may be used for a particular field. This table contains 30 subtables, one of which is associated with each valid field. The first entry of each sub-
PROGRAM: MDBUPD

VII.5

table is: the number of byte entries for this subtable. The remaining entries are the valid subfield codes. The correct subtable is located by first retrieving the byte displacement from the table 'SFLD' for a particular field. This is then used as an index to point to the correct subtable in the table 'AL'. The program 'NDX123' uses this subtable to verify the subfield indicators. The table 'AL' is used with the following sequence of instructions:

| LB,R3     | SFLD,R6  | pick up the byte displacement using tag as an index |
| AI,R3     | AL       | point to the appropriate subtable of subfield codes |
| BAL,R7    | NDX123   | go check for a valid subfield code |

where R6 equals the field index obtained from the table 'TAGS' for this field.

NL This is an indexable byte table of valid indicators that may be used for a particular field. This table contains 15 sub-tables, one of which is associated with each valid field. The first entry of each sub-table is the number of byte entries for this sub-table. The remaining entries are the valid indicators. The correct sub-table is located by first retrieving the byte displacement from either the table 'IND1' or 'IND2' for a particular field. This is then used as an index to point to the correct sub-table within the table 'NL'. The program 'NDX123' uses this sub-table to verify the indicators. The table 'NL' requires the following sequence of instructions:

| LB,R3     | IND1,R6  | pick up byte displacement using tag as index |
| AI,R3     | NL       | point to the appropriate sub-table of indicators |
| BAL,R7    | NDX123   | go check for a valid indicator |

where R6 equals the field index obtained from the table 'TAGS' for this field.

LNGLIST This is a word table of valid language codes. The language code from the MARC record is checked sequentially against the entries in this table until a match is found. The index that was used to verify
the MARC language code is then used in the OCLC bibliographic record instead of the characters. The table is ordered so that the most frequently used language is found first.

**CPLIST** This is a half-word table of valid country of publication codes. If the work was published in the United States, Canada, Russia, or the United Kingdom, this table contains the mnemonic code for the state, province, or other subdivision. The country of publication code from the MARC record is checked sequentially against the entries in this table until a match is found. The index used to find the match is then used in the OCLC bibliographic record instead of the characters.
VIII.1

PROGRAM: MDBUPD

VIII APPENDIX
Operating Requirements

1. Computer - Xerox Sigma 5
2. I/O Devices - 2 9 track tape drives, line printer, card reader/punch
3. Operating System - OBM
4. Execution Time - depends upon number of records to be added; approximately 45 minutes for 1500 records.
5. Run Schedule - weekly
6. Job Control Language

```
!JOB ONLINE, MDBUPD
!IMG002A E1
!RAEDIT
: CLEAR D2
: ALLOT (FILE, D2, RESULTS), (FSI, 8950), (FOR, C)
: ASSIGN (M: SO, CP), (TRIES, 100)
!MDBUPD
!IMG002A M1
!STD (LL, 0)
!RAEDIT
: COPY (FILE, D2, RESULTS), (OUT, LO), VFC
: COPY (FILE, D2, RESULTS), (OUT, LO), VEC
!STD (LL, LP)
!STD (BI, TO), (BO, 0)
!COPY BI, 1000
SFI TO, BACK
!RAEDIT
: COPY (FILE, D2, RESULTS), (OUT, TO)
!WEC TO, 2
!UNL TO
!JOB IDLE,
!IMG002A CCR
!FIN
```
1. CONSOLE MESSAGES

! MES -- IF A MARC WEEKLY TAPE IS NOT TO BE ADDED
!! PAU -- TO THE DATA BASE, THEN ABORT THIS JOB NOW.

RESPONSE: ACTION:

'X' Abort the job
'C' Continue to add the MARC tape

!! PAU -- READY CARD PUNCH *
RESPONSE: 'C' - New Line

MOUNT OUTPUT ON 081
RESPONSE: 1) Mount tape as directed -- ring in
2) 'C' - New Line

MOUNT WEEKLY ON 080
RESPONSE: 1) Mount tape as directed -- no ring
2) 'C' - New Line

If a new deferred tape is to be started, set all Sense Switches to 'l' and the following message will appear on the console:

RESET SENSE SWITCHES
BEGIN WAIT

RESPONSE: ACTION:

'X' Abort the job
'NEW' Start a new deferred tape

DISMOUNT WEEKLY FROM 080
RESPONSE: 1) Dismount tape as directed
2) 'C' - New Line

! MES -- MOUNT RESULTS TAPE ON TO
RESPONSE: 1) Mount tape as directed -- ring in
2) 'C' - New Line

! PAU -- VERIFY RESULTS TAPE MOUNTED ON TO, RING IN *
RESPONSE: 'C' - New Line

! PAU -- DISMOUNT ALL TAPES *
RESPONSE: 1) Dismount tapes as directed -- remove the rings
2) 'C' - New Line
PROGRAM: MDBUPD

VIII.4

2. PROGRAMMED ABNORMAL COMPLETION
   - MDBUPD will terminate abnormally via a CALL,9 3
   instruction under the following conditions. The message
   printed out to signal the abort is included in each case.

   MESSAGE         REASON
   'ERROR READING MASTER   four possible reasons:
       DATA BASE'           1. Invalid logical file or
                              block no.
                              2. Permanent I/O error
                              3. Invalid Index
                              4. End-of-file delimiter read

   'ERROR REPLACING MASTER   three possible reasons:
       DB RECORD'           1. OCLC index entry found
                              occupied
                              2. Could not replace Bib. record
                              3. Could not replace an index

   'ERROR DELETING MASTER   three possible reasons:
       DB RECORD'           1. Invalid index
                              2. Permanent I/O error
                              3. End-of-file delimiter read

   'ERROR ADDING MASTER     three possible reasons:
       DB RECORD'           1. OCLC index entry found occupied
                              2. Could not add Bib. record
                              3. Could not add an index

3. DIAGNOSTICS

   MESSAGE         REASON
   NON-NUMERIC X ' ' in    There was an error in converting
       POSITION X ' '          from ASCII to Binary

   LRECL ( ) DOES NOT      Record length does not equal
       EQUAL PRECL ( )         length of actual data read

   ILLEGAL CHARACTER X ' '  Three possible reasons:
       IN POSITION _           1. Illegal encoding level
                               2. Bytes 18-23 of the MARC
                               leader are not all blank
                               or of the form '4500'.
                               3. Bytes 6-11 of the MARC leader
                               are not of the form 'AM 22'

   BADDR INCORRECT        MODULO 12 of the Base address
       LENGTH                does not equal 1

   FIELD TERMINATOR       Field terminator is missing at
       MISSING IN POSITION _   the end of the directory
MESSAGE

RECORD TERMINATOR MISSING
IN POSITION _

ILLEGAL TAG IN POSITION _

TOO MANY FIELDS

MASTER RECORD MISSING --
CANNOT DELETE

MASTER RECORD PRESENT --
CANNOT ADD

MASTER RECORD MISSING --
STATUS CHANGED TO NEW

001 FIELD IS NOT FIRST

001 FIELD INCORRECT LENGTH

NON-MARC CHARACTER
X ' ' IN POSITION
( FIELD)

NON-NUMERIC IN LC CARD
NUMBER

008 FIELD IS NOT SECOND

008 FIELD INCORRECT LENGTH

REASON

Record terminator is missing
at the end of the record

Two possible reasons:
1. Tag is too big
2. Tag is not in table of
   legal tags

There are more than 50 fields
in the record

Record status equals 'D'
but there is not master
record to delete

This record is already on
the master data base file,
log it as a duplicate

If the status of the record
is not 'N' and no existing
record was found, the status
is changed to 'N'.

The 001 field must be the
first field in the record

The 001 field must be at
least 13 bytes and not
more than 30 bytes

An Invalid character was
returned when converting
from ASCII to EBCDIC

The LC card number must
be all numeric

The 008 field must be the
second field in the record

The 008 field must be 41
bytes including the field
terminator
VIII.6

MESSAGE

INVALID DATA IN ELEMENT ' ' OF 008 FIELD

RECORD HAS FEWER THAN 3 FIELDS

FIELD IS TOO SHORT

FIELD IS SHORTER THAN EXPECTED

INVALID FIRST INDICATOR X ' ' IN POSITION ' ' ( FIELD)

INVALID SECOND INDICATOR X ' ' IN POSITION ' ' ( FIELD)

SUBFIELD DELIMITER IS NOT FIRST IN POSITION ( FIELD)

FIELD TERMINATOR MISSING IN POSITION ( FIELD)

ILLEGAL SUBFIELD CODE X ' ' IN POSITION ( FIELD)

FIELD OR REC'D TERMINATOR END'D IN POSITION BEFORE END OF FIELD

OBSOLETE TAG CHANGED TO ___ ___

REASON

The 008 field of the MARC record contains invalid data

Every record must have at least 3 fields

Every field must be at least 5 bytes long

The field is shorter than minimum length required for that field

The first indicator for this field is not one of the valid entries in the table 'IND1'.

The second indicator for this field is not one of the valid entries in the table 'IND2'.

Each data element in a field must begin with a subfield delimiter (X'FC')

If the field terminator (X 'FD') is missing, provide one.

The subfield code is not one of the valid entries for this field in the table 'SFLD'

The field terminator (X 'FD') or the record terminator (X 'FC') are encountered before the actual end of field

The 652 field is obsolete and will be changed to a 651. The 750 field is obsolete and will be changed to a 710.
PROGRAM: MDBUPD

VIII.7

MESSAGE

\$a SUBFIELD IS NOT FIRST IN ... FIELD

CONTENTS OF 050 FIELD CAN'T BE PARSED

ILLEGAL STATUS CHARACTER X ' ' CHANGED TO 'N'

ENCODING LEVEL CHANGED TO 'E'

ENCODING LEVEL CHANGED TO 'W'

FIELD OCCURS ... TIMES

REASON

The first subfield for every field must be a \$a subfield.

The contents of the 050 field cannot be parsed.

The status of the MARC record is not one of the valid codes (D,N,C,P) so it is changed to 'N'.

The encoding level changed to 'E' on every record that has an error and is dumped to the 'RESULTS' area.

The encoding level is changed to 'W' on every record that has a warning message associated with it.

The field occurs less than the minimum or more than the maximum number of allowable times as indicated in the tables 'MINCNT' and 'MAXCNT'

4. Parameter Cards Required - none

5. Example of Output - next page
<table>
<thead>
<tr>
<th>Ti</th>
<th>LDR</th>
<th>S</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>2.0</td>
<td>1C3</td>
<td>721</td>
<td>974</td>
</tr>
<tr>
<td>2.0</td>
<td>010</td>
<td>1C3</td>
<td>721</td>
<td>974</td>
</tr>
<tr>
<td>2.5</td>
<td>010</td>
<td>1C3</td>
<td>721</td>
<td>974</td>
</tr>
<tr>
<td>3.0</td>
<td>010</td>
<td>1C3</td>
<td>721</td>
<td>974</td>
</tr>
<tr>
<td>3.5</td>
<td>010</td>
<td>1C3</td>
<td>721</td>
<td>974</td>
</tr>
<tr>
<td>4.0</td>
<td>010</td>
<td>1C3</td>
<td>721</td>
<td>974</td>
</tr>
<tr>
<td>4.5</td>
<td>010</td>
<td>1C3</td>
<td>721</td>
<td>974</td>
</tr>
<tr>
<td>5.0</td>
<td>010</td>
<td>1C3</td>
<td>721</td>
<td>974</td>
</tr>
<tr>
<td>5.5</td>
<td>010</td>
<td>1C3</td>
<td>721</td>
<td>974</td>
</tr>
<tr>
<td>6.0</td>
<td>010</td>
<td>1C3</td>
<td>721</td>
<td>974</td>
</tr>
<tr>
<td>6.5</td>
<td>010</td>
<td>1C3</td>
<td>721</td>
<td>974</td>
</tr>
<tr>
<td>7.0</td>
<td>010</td>
<td>1C3</td>
<td>721</td>
<td>974</td>
</tr>
<tr>
<td>7.5</td>
<td>010</td>
<td>1C3</td>
<td>721</td>
<td>974</td>
</tr>
<tr>
<td>8.0</td>
<td>010</td>
<td>1C3</td>
<td>721</td>
<td>974</td>
</tr>
<tr>
<td>8.5</td>
<td>010</td>
<td>1C3</td>
<td>721</td>
<td>974</td>
</tr>
<tr>
<td>9.0</td>
<td>010</td>
<td>1C3</td>
<td>721</td>
<td>974</td>
</tr>
<tr>
<td>9.5</td>
<td>010</td>
<td>1C3</td>
<td>721</td>
<td>974</td>
</tr>
<tr>
<td>10.0</td>
<td>010</td>
<td>1C3</td>
<td>721</td>
<td>974</td>
</tr>
</tbody>
</table>
APPENDIX C

DETAILED DESCRIPTION OF INTERNAL
SUBROUTINES
This routine is used to compare bytes for equal. Upon entry to this routine, R1 has the number of bytes to be checked, R2 has the byte address of the record, R3 has the byte address of the field to be checked. R7 is the link register. At exit, R2 is pointing to the next byte of the record. If the two fields are equal, the condition is set to zero.

DUMPLN

This routine is used to translate a field to EBCDIC and hexadecimal and move it to output buffers. The output buffer for the EBCDIC translation is called 'TXTFLD'; for the hexadecimal translation, it is called 'HEXFLD'. The characters are converted and moved 32 bytes at a time. The link register is R9. Upon entry, R2 has the byte address of the field to be translated, and R3 has the length of the field.

SEARCHER

This routine is used to check the access keys for a particular MARC record against the existing access keys for the other bibliographic records. Before branching to this routine, an FPT is set up in the following manner:

```
SCHFPT DATA 0,0,0, SCHWORK,OCL NUMBER
```

Where the first word is for the completion status and function code; the second word is the logical file number; the third word is either the title key or the author-title key; the fourth word is the word address of the work area; the fifth word is the OCLC number.

The link register to this routine is R5. A description of the processes of this routine are given in Section IV -- FUNCTIONS.

DEFER

This routine will cause a troublesome record to be copied onto the deferred tape. The routine picks up the beginning address of the record from the location 'RECORG' and picks up the record length from the location 'RECLN'. The routine then sets up a 'copy' FPT and issues the copy command.

FLDERR

This routine causes the error message to be logged. R9 is the link register to this routine. R8 has the address of the error message to be used. This routine sets up the required parameters and links to the external sub-routine, 'LOGERR'.

FLDWAR

This routine causes the warning messages to be logged. R9 is the link register to this routine. R8 is set with the address of the warning message to be used. This routine sets up the required parameters and links to the external sub-routine, 'LOGWAR'.

ADDITIONAL SUBROUTINE DOCUMENTATION
I. FUNCTIONS

The external subroutine 'CASEB' is used to convert ASCII data to EBCDIC. The address of the parameters needed for this subroutine is passed in R8.
II. SOFTWARE INTERFACE

A. Linkage

```
LI,R8 CASEBF
BAL,R7 CASEB
```

B. Parameter List Description

```
CASEBF RES 1 number of bytes to be converted
RES 1 beginning address of data
RES 1 beginning address of storage area
RES 1 completion status
```

C. Return Codes

The status of the convert operation is returned in the fourth word of the parameter list.

```
STATUS = 80000000' - Normal completion; all characters were legal
= C0010000' - Abnormal completion
```

D. Other Entry Points - none

E. OCLC Subroutines Referenced - none

F. OCLC Procedures Referenced - none

III. DESCRIPTION OF SPECIAL STORAGE AREAS, SWITCHES, AND TABLES

ASEBTBL

The conversion from ASCII to EBCDIC is accomplished with this one large table. Using the ASCII value of the character as an index, the corresponding EBCDIC value is retrieved from this table. If the corresponding table entry is a '0', the character is considered illegal.
I. FUNCTIONS

The external subroutine CASBI is used to convert numeric data which is in ASCII format to Binary format. The ASCII character is first checked to be numeric (between the limits of X'30' and X'39'), and is then converted to binary.
II. SOFTWARE INTERFACE

A. Linkage

LI,R8 CASBI
BAL,R7 CASBI

B. Parameter List Description

The address of the three word parameter list is passed to this subroutine in R8.

RES 1 number of bytes to be converted
RES 1 beginning address of field to be converted
RES 1 completion status

C. Return Codes

The status of the convert operation is returned in the third word of the parameter list.

STATUS = X'80000000' Normal completion
         = X'C0010000' Abnormal completion; character to be converted is not numeric

D. Other Entry Points

CEBBI - This entry point converts numeric data in EBCDIC format to Binary format.

E. OCLC Subroutines Referenced - none

F. OCLC Procedures Referenced - none
I. FUNCTIONS

The external subroutine 'MDBUPDLG' is used as the logging routine for the program MDBUPD only. MDBUPDLG is a program of subroutines as follows:

<table>
<thead>
<tr>
<th>Name of subroutine</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGMSG</td>
<td>type and log a text message.</td>
</tr>
<tr>
<td>LOGTEXT</td>
<td>log a text message</td>
</tr>
<tr>
<td>LOGERR</td>
<td>log a message with the preface 'ERROR'</td>
</tr>
<tr>
<td>LOGWAR</td>
<td>log a message with the preface 'WARNING'</td>
</tr>
<tr>
<td>LOGADD</td>
<td>log the LC card number and the word 'ADDED'</td>
</tr>
<tr>
<td>LOGREP</td>
<td>log the LC card number and the word 'REPLACED'</td>
</tr>
<tr>
<td>LOGDEL</td>
<td>log the LC card number and the word 'DELETED'</td>
</tr>
<tr>
<td>LOGREJ</td>
<td>log the LC card number and the word 'REJECTED'</td>
</tr>
<tr>
<td>LOGIGN</td>
<td>log the LC card number and the word 'IGNORED'</td>
</tr>
<tr>
<td>LOGPAGF</td>
<td>write the previous line and supply the format control character to eject the page</td>
</tr>
<tr>
<td>LOGDEF</td>
<td>log the LC card number and the word 'DEFERRED'</td>
</tr>
<tr>
<td>LOGDUMP</td>
<td>log the message to be listed on the DO device (teletype)</td>
</tr>
<tr>
<td>LOGPUNCH</td>
<td>log the message to be listed on the SO device (card punch)</td>
</tr>
</tbody>
</table>

MDBUPDLG begins with an indexable branch table which branches to one of these routines. When a message is 'hogged', it is written on the RAD area assigned by MDBUPD. The RAD file will later be used to print copies of the messages on the appropriate devices.
II. SOFTWARE INTERFACE

A. Linkage

LOGTEXT EQU MDBUPDLG + 1

LI,R8 MSGNPARN
BAL,R7 LOGTEXT

B. Parameter List Description

R8 is set up with the address of the parameter list required to format the message. Each parameter list must be at least two words long. Every word except the last word is of the format:

- **GEN, 8, 24**
- **FC, BA** (message)

<table>
<thead>
<tr>
<th>FC</th>
<th>Function Code</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>single message is to be logged</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>message is to be converted to hexadecimal before being logged</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>message is to be converted to EBCDIC before being logged</td>
<td></td>
</tr>
</tbody>
</table>

The last word of the parameter list must be a '0' to indicate to MDBUPDLG that the end of the parameter list has been reached. For additional documentation and examples, see the APPENDIX, Section E, under the procedure description for 'MSG'.

C. Return Codes

No completion codes are set. The return is made back to the calling program through R7.

D. Other Entry Points

None

E. OCLC Subroutines Referenced

- $MBS -- Move Byte String
  (described on page VIII.28)
- CBIEB -- Convert Binary to EBCDIC
  (described on page VIII.49)

F. OCLC Procedures Referenced

None
III. DESCRIPTION OF SPECIAL STORAGE AREAS, SWITCHES, AND TABLES

Upon entry to MDBUPDLG, there is an indexable branch table of routines. This table effectively produces the same result as having separate entry points. A branch is taken from this table to the appropriate routine.

IV. INTERNAL SUBROUTINE DESCRIPTION

CBIHE12 - - This routine converts one byte in binary format to 2 digits in hexadecimal format. The link register to this routine is R7.

PURGLINE - This routine is used to test the status of the buffer, print the previous line if there is one, clear out the line image and re-set the buffer status. A buffer status of '-1' indicates that the buffer is ready to go. A status of '0' indicates that the buffer is empty. A buffer status of anything else will cause the previous line to be written. The link register to this routine is R5.
1. FUNCTIONS

The external subroutines OPENMARC, CLOSE MARC, and READMARC are used to perform the MARC tape input operation.

The external subroutine OPEN MARC sets up the parameters needed for the subroutine TAPEIO (described on page VIII.53). TAPEIO performs the operation of reading the MARC tape input.

OPENMARC sets up the two function parameter tables (FPT's) needed to perform this read operation. The first FPT is for the read operation itself; the second FPT is used to sense the device status. OPENMARC also sets up R1 with the address of a work area and R8 with the address of the parameters needed for TAPEIO.

At its alternate entry point, READMARC, OPENMARC actually reads the record by means of the subroutine TAPEIO. READMARC exits with the length of the record in R1 and the byte address of the record in R2. The return to the calling program is made either 0,1,2, or 3 instructions past the calling instruction depending upon whether the record read was a header label, a MARC record, a trailer label, or the end of a particular file, respectively.

At its alternate entry point, CLOSEMARC, OPENMARC sets up the tape for the next record. If there was an end of file, the tape is unloaded, otherwise CLOSEMARC causes TAPEIO to back space over the last record read.
II. SOFTWARE INTERFACE

A. Linkage

The calling sequence for OPENMARC is as follows:
LI,R8 X'80' device address needed for the FPT's
BAL,R7 OPENMARC

B. Parameter List Description

None

C. Return Codes

None

D. Other Entry Points

READMARC
CLOSEMARC

E. OCLC Subroutines Referenced

TAPEIO (Page VIII.53)

F. OCLC Procedures Referenced

None
II. SOFTWARE INTERFACE

A. Linkage
The calling sequence for READMARC is as follows:

```
BAL,R7 READMARC
B File 1 header label
B File 2 data
B File 3 trailer label
Bal,R7 CLOSEMARC done with that EOF
```

B. Parameter List Description
None

C. Return Codes
None

D. Other Entry Points
None

E. OCLC Subroutines Referenced

LOGMSG (page VIII.19)
LOGWAR (page VIII.19)

F. OCLC Procedures Referenced
None
II. SOFTWARE INTERFACE

A. Linkage

BAL,R7 CLOSEMARC

B. Parameter List Description

None

C. Return Codes

None

D. Other Entry Points

None

E. OCLC Subroutines Referenced

TAPEIO (page VIII.53)

F. OCLC Procedures Referenced

None
I. FUNCTIONS

These external subroutines are used to Pack and Unpack numeric data. The characters are first checked to be numeric and are then either packed or unpacked depending on the entry point.
II. SOFTWARE INTERFACE

A. Linkage

LI,R8 PACKPARM
BAL,R7 PACK

B. Parameter List Description

The address of the 4-word parameter list is passed to these routines in R8.

RES 1 byte address of field to be packed
RES 1 number of bytes to convert
RES 1 byte address of storage area
RES 1 completion status

C. Return Codes

The status of the conversion is returned in the fourth word of the parameter list

STATUS = X'80000000' - Normal completion
        = X'C0010000' - Abnormal completion; character to be converted is non-numeric

D. Other Entry Points

UNPK - - This entry point is used to unpack data. The parameter list is the same as for PACK. The return code for a normal completion is the same. There is no abnormal completion.

E. OCLC Subroutines Referenced

NDX123 - This routine is used to check for valid numeric characters for the PACK routine. NDX123 is described more thoroughly on page

F. OCLC Procedures Referenced - none

III. DESCRIPTION OF SPECIAL STORAGE AREAS, SWITCHES, AND TABLES

PTBL - This is a table of valid numeric characters for the PACK routine.
I. FUNCTIONS

This external subroutine is used to move byte strings. If the address of the field to be copied and the address of the location for the output are not both word aligned, they are adjusted to compensate. $MBSS is used to move less than 15 bytes.
II. SOFTWARE INTERFACE

A. Linkage

BAL,R7 $MBS

B. Parameter List Description

Enter with the number of bytes to be moved in R1. The beginning byte address of the field must be R2. The byte address of the location where the field to be moved must be in R3.

C. Return Codes

The return is made back to the calling program through R7 with no completion status posted.

D. Other Entry Points

$MBSS - This entry point is used when 14 bytes or less are to be moved. There is no check for the word alignment of the addresses.

E. OCLC Subroutines Referenced - none

F. OCLC Procedures Referenced - none
I. FUNCTIONS

This external subroutine is used to search for a particular character (byte) in a list. The search is performed from the end of the list and the index used to find the character is returned to the calling program.
II. SOFTWARE INTERFACE

A. Linkage

BAL,R7   NDX123

B. Parameter List Description

Enter with the byte to be searched for in R2 and the byte address of a list in R3. The first entry of the list must be the number of entries in the list.

C. Return Code

There are no completion codes returned to the program. Return to the calling program is made through R7. The index used to find the character in the list is returned in R1. R1 is '0' if the character was not found. The search is performed from right to left.

D. Other Entry Points - none

E. OCLC Subroutines Referenced - none

F. OCLC Procedures Referenced - none
The external subroutine 'ADDMAST' is used to add and delete records from the master database. If the OCLC file entry is not occupied, the search keys are extracted for this record and the indexes are added to the index file. The record is also added to the database.

If the OCLC file entry is occupied, a check is made to determine if the subroutine 'DELTMAST' had been called. 'DELTMAST' is another entry point to the subroutine 'ADDMAST'. If the OCLC file entry is occupied and 'DELTMAST' is not called, this constitutes an error and the OCLC file entry is returned and nothing is added.

Another entry point to 'ADDMAST' is 'RADDMAST' which will re-add a bibliographic record and its indexes.
II. SOFTWARE INTERFACE

A. Linkage

LI,R8     ADDPARM
BAL,R7     ADDMAST

B. Parameter List Description

RES 1 completion status
RES 1 word address of the bibliographic record
RES 1 WA (WORKAREA)
RES 1 word address of OCLC file entry and indexes
RES 1 WA(index)

Where WORKAREA is a 312 word double word aligned workarea. The workarea is 313 words if it is not double word aligned. The fifth word is used only for DELTMAST and RADDMAST.

C. Return Codes

The status of the addition operation is returned in the first half-word of the parameter list.

STATUS = X'8000' - Normal completion
         = X'C000' - Abnormal completion; OCLC index found occupied, nothing has been added to the data base
         = X'C001' - Abnormal completion; could not add the Bib. record, no index were added to the data base
         = X'C001' + INDEX FILE NO.-- Abnormal completion; could not add an index, the Bib. record and/or other indexes were added

D. Other Entry Points

DELTMAST - This entry point is used to delete a record and its indexes.

RADDMAST - This entry point is used to re-add a record and its indexes.

E. OCLC Subroutines Referenced

INDEXER - This external subroutine (described more fully elsewhere in this section) is used to set up the access keys for a particular record.

F. OCLC Procedures Referenced - none
FUNCTIONS

LCN000 breaks a call number into components to aid in the formatting of the call number. A code set at the entry point determines the type of call number which has been input. At LCCN000, the code is set to zero; at LCCN000B, the code is set to four; at LCCN000D, the code is set to two; and at LCCN000T, the code is set to one. Upon entry to either LCCN000, LCCN000B, LCCN000D, or LCCN000T, R8 points to a parameter list which contains the byte address of the input call number. The second word of the parameter list is the byte address of a work area.

LCN000 scans and interrogates the call number using a set of internal procedures. The components to be broken down by LCCN000 are as follows:

0 - A string of alphas, followed by a blank, which precedes the rest of the call number.
1 - Alpha portion of the Library of Congress class number.
2 - Numeric portion of the Library of Congress class number.
3 - Decimal portion of the Library of Congress class number.
4 - Date type element that precedes the first Cutter. In reality this is any field preceded by a blank which precedes the first Cutter.
5 - First Cutter. It must begin with a decimal followed by an alpha string and a numeric string.
6 - Date type element that precedes the second Cutter. In reality this is any field preceded by a blank which precedes the second Cutter.
7 - Second Cutter. It is preceded by a decimal if component 6 is present; otherwise it immediately follows the first Cutter. The second Cutter is a numeric string followed by an alpha string.
8-254 These components are variable in format. Bit 7 of the component number set to 1 indicates an element followed by a comma.

Component 255 always marks the end of the call number in the work area.

When an error is encountered in the format of the call number, the condition code is set and control is returned to CNVT.

As each component of the call number is found, it is stored in the workarea preceded by its component number. When the end of the call number field is encountered, if all required components are present, control is returned normally.
SOFTWARE INTERFACE

A. LINKAGE

LI,R8 LCCNPARM
BAL,R7 LCCN000 (or LCCN000B, LCCN000D, or LCCN000T)

B. PARAMETER LIST DESCRIPTION

LCCNPARM DATA BA(050 FIELD) or 050 FIELD IF PRESENT
DATA BA(WORKAREA) AREA WHERE FORMATTED CALL NO. WILL BE RETURNED

C. RETURN CODES

LCCN000, LCCN000B, LCCN000D, & LCCN000T set the condition code as follows:

CC1 = 4 = 0 NORMAL RETURN
CC3 = 1 DEFAULT TO UNIT CARD
CC4 = 1 BREAKDOWN WAS UNSUCCESSFUL

D. OTHER ENTRY POINTS

LCCN000B
LCCN000D
LCCN000T

E. OCLC SUBROUTINES REFERENCED - none

F. OCLC PROCEDURES REFERENCED -

NEXT
BACK
SPAN
POWER
ANY
SAVE
MARK
OPT
ALPHA
NUMER
POINT
BLANK
TERMN
COMMA
BREAK

Different name values for the same procedure
PROCEDURE DESCRIPTION

PURPOSE: NEXT generates a BAL,R7 :NXT where: NXT is an internal subroutine of LCCNO00

FORMAT: NEXT No. operands are required

EXAMPLE: NEXT BAL,R7 :NXT
PROCEDURE DESCRIPTION

PURPOSE: BACK sets up a parameter value and provides a link via R7 to the internal subroutine :BCK.
If the value of AF(1) is less than two, a BAL,R7:BCK-1 is generated. If AF(1) is loaded into R14 and a BAL,R7:BCK is generated.

FORMAT: BACK AF(1)

EXAMPLE 1:
Back up one character.

```
BACK  AL WHERE AL=1
+ BAL,R7 :BCK+1
```

EXAMPLE 2:
Back up four characters

```
BACK  PO where PO=4
LI,R14  4
BAL,R7 :BCK
```
PROCEDURE DESCRIPTION

PURPOSE: SPAN sets up a parameter value and links to internal subroutine :PWR-1 via R7. R14 is loaded with the argument field. Its range of values is the table CHARVAL.

FORMAT: SPAN AF(1)

EXAMPLE: Scan to the next non-numeric character.

SPAN NU where NU = char value for a numeric in CHARVAL

+ LI,14 2
+ BAL,R7 :PWR -1
PROCEDURE DESCRIPTION

PURPOSE: POWER sets up a counter in R13 from AF(2) and a value in R14 from AF(1); then links to the internal subroutine :PWR. On return from :PWR, an unconditional branch is taken. The effective address of the branch is determined by the value of A1(3) and AF(5). If AF(3) = 1 and AF(5) = 0 a B $+2 is generated. If A1(3) = 0 and AF(5) = 1, three instructions are generated:

B $+2
B $+3
BAL,R7 :RST

If AF(5) = 1 an unconditional branch to AF(4) is generated.

FORMAT: POWER AF(1),AF(2),AF(3),AF(4),AF(5)

EXAMPLE: Scan to see if there is a blank in the next 4 characters. If so, branch to T4. If not, restore R1 and branch to T4.

POWER BL,PO,NO,T4
+ LI,R13 4
+ LI,R14 8
+ BAL,R7 :PWR
+ B $+2
+ B $+3
+ BAL,R7 :RST
+ B T4

EXAMPLE 2: Scan to see if there is a blank in the next four characters. If not, skip the branch to T4 and continue with the next sequential instruction. If so, branch to T4.

POWER BL,PO,YESET4
+ LI,R13 4
+ LI,R14 8
+ BAL,R7 :PWR
+ B $+2
+ B T4
PROCEDURE DESCRIPTION

PURPOSE: ANY compares the character value which is in R15 to a table value or a combination of table values [AF(1)]. The succeeding branch instructions are generated on the basis of AF(2) which has the value 0 or 1 and the presence or absence of AF(4). The effective address of the branch instruction is AF(3).

FORMAT: ANY AF(1), AF(2), AF(3), AF(4)

EXAMPLE: Is next character a period or a blank:
ANY PO/BL,NO,ABT
+ CI,15 12
+ BAZ ABT

where PO = 4
BL = 8
NO = 1

Is the value in R15 4 or 8? If neither, branch to ABT, otherwise fall through to the next sequential instruction.
PROCEDURE DESCRIPTION

PURPOSE: SAVE generates a STW,1 :SAVE instruction to save the pointer to the current location in the TEMP area.

FORMAT: SAVE

EXAMPLE: SAVE
+ STW,R1 :SAVE
PROCEDURE DESCRIPTION

PURPOSE: MARK sets up the component number and links to the routine :MRK which will move the component to WORKAREA.

FORMAT: MARK AF(1)

EXAMPLE: MARK 1
         + LI,14 1
         + BAL,R7 :MRK

Mark component #1 and move it to the WORKAREA.
PROCEDURE DESCRIPTION

PURPOSE: OPT interrogates the next sequential character value. If it is not equal to AF(1) a branch is taken to $+2. If the character value is equal to AF(1), a BAL,R7 :NXT is taken.

FORMAT: OPT AF(1)

EXAMPLE: Is the next character a blank. If so, look at following character.

```
OPT BL
+ CI,15 8
+ BAZ  $+2
+ BAL,R7 :NXT
```
PROCEDURE DESCRIPTION

PURPOSE: ALPHA compares the character value in R15 to its name value shifted left one position (1NAME). The shifted name value equals the alpha character value from the table CHARVAL. The conditions of the succeeding branch instruction are generated depending on the value of AF(1) which may be 0 or 1 and the presence or absence of AF(3). If AF(3) is absent, the effective address of the branch is AF(2). If the branch is not taken, the next sequential instruction is executed. If AF(3) is present the effective address of the generated branch is $+3. If the branch is not taken, the next instruction is a BAL,R7 :RST followed by an unconditional branch to AF(2).

There are five alternate names that may be used to invoke this procedure.

NUMER - its name value equals the numeric character value

POINT - its name value equals the character value for a period

BLANK - its name value equals the character value for a blank

TERMN - its name value equals the character value for a field delimiter

COMMA - its name value equals the character value for a comma

These procedures are used to interrogate the value of a character.

FORMAT: ALPHA AF(1), AF(2), AF(3)

EXAMPLE 1: Is the character in question numeric. If not, declare an error.

        NUMER    NO,ABT
+        CI,15     2
+        BAZ      ABT

where no = 0

EXAMPLE 2: Is the character alpha. If it is skip around; if not restore R1 to previous character and branch to T8

        ALPHA    NO,T8,REST
+        CI,15     1
+        BANZ     $+3
+        BAL,7     RST
+        B        T8

where no = 0 and REST = 1
PROCEDURE DESCRIPTION

PURPOSE:

BREAK sets up a parameter value and links to the internal subroutine, :BRK. R14 is loaded with AF(1). Its range of values is equal to the range of values in the table CHARVAL.

FORMAT:

BREAK AF(1)

EXAMPLE:

Scan to find the next numeric character.

BREAK NU

where NU = CHAR. value

for a numeric is CHAFVAL

+ LI,14 2

+ BAL,R7 :BRK
I. FUNCTIONS

The external subroutine 'INDEXER' scans the given bibliographic record. It returns a 6-byte (left-justified) author-title access key in the third and fourth words of the workarea which is provided by the calling program. It also returns a 6-byte (left-justified) title-only access key in the first and second words of the workarea. It also returns the 8-byte OCLC semi-packed representation of the first 12 characters of the LC card number in the fifth and sixth words of the work area.

INDEXER also prepares and stores three pointers at word displacements 10, 11, and 12 in the OCLC bibliographic record (see Bibliographic record format - page 14). The content of these pointers is as follows:

1. First Pointer --
   Bits 0 - 15: Byte length of first part of the author string (1XX field).
   Bits 16 - 31: Byte displacement to the first author substring from the end of the leader.

2. Second Pointer --
   Bits 0 - 15: Byte length of the second part of the author string
   Bits 16 - 31: Byte displacement to the second author string from the end of the leader.

3. Third Pointer --
   Bits 0 - 15: Byte length of the title (245 field) string.
   Bits 16 - 31: Byte displacement to the first character of the title string from the end of the leader.
II. SOFTWARE INTERFACE

A. Linkage

<table>
<thead>
<tr>
<th>LI,R6</th>
<th>XRPAPMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAL,R7</td>
<td>INDEXER</td>
</tr>
</tbody>
</table>

B. Parameter List Description

<table>
<thead>
<tr>
<th>RES</th>
<th>1</th>
<th>word address of the cell containing the word address of the record</th>
</tr>
</thead>
</table>

where WORKAREA is 14 words and double word aligned. The work area is set up as follows:

<table>
<thead>
<tr>
<th>WKAREA</th>
<th>RES</th>
<th>0</th>
<th>title access key returned here</th>
</tr>
</thead>
<tbody>
<tr>
<td>3111</td>
<td>RES</td>
<td>2</td>
<td>author/title access key returned here</td>
</tr>
<tr>
<td>33KEY</td>
<td>RES</td>
<td>2</td>
<td>LC card number returned here</td>
</tr>
<tr>
<td>LCCARONO</td>
<td>RES</td>
<td>2</td>
<td>scratch area</td>
</tr>
<tr>
<td>SCRATCH</td>
<td>RES</td>
<td>8</td>
<td>scratch area</td>
</tr>
</tbody>
</table>

C. Return Codes

If the title cannot be parsed, is missing, or appears after the 260 field, the title portion of the author/title key will be 3 blanks, and the title only key will be 6 blanks. There is no returned status.

D. Other Entry Points

PULLKEYS - This entry is used when the access keys are to be set up and returned but the pointers are not to be prepared or stored.

E. OCLC Subroutines Referenced

None

F. OCLC Procedures Referenced

None
III. DESCRIPTION OF SPECIAL STORAGE AREAS, SWITCHES, AND TABLES

A. Special Switches
None

B. Special Storage Areas
None

C. Special Tables

ALLOWED -- This is a byte table of characters values that are permitted in the keys. If the byte is a ' ', it is an allowed character. The table is indexed by the character itself.
FUNCTIONS

CBIEB converts variable-length binary fields to EBCDIC. The user specifies what sign is to be given to the result and what fill character is to be used in padding the field. Error conditions are encountered when there is an overflow condition in the output field or when the output field is not large enough to contain the sign. The return code is posted in the first two bytes of the parameter list upon return.
SOFTWARE INTERFACE

A. LINKAGE

The calling sequence is

LI,R8 CBPARMS
BAL,R7 CBIEB

B. PARAMETER LIST DESCRIPTION

CBPARMS DATA BA(BINARY FIELD TO BE CONVERTED)
DATA,1 WIDTH, FILL, PLUS, MINUS
DATA BA(OUTPUT FIELD)

C. RETURN CODES

The return code is found in bytes 0 and 1 of CBPARMS.

BYTE0 = X'80' Normal completion
BYTE1 = X'00' ERROR
BYTE0 = X'CO' NO ROOM IN FIELD FOR SIGN
BYTE1 = X'01' ERROR
BYTE1 = X'02' FIELD OVERFLOW

OTHER ENTRY POINTS - none

OCLC SUBROUTINES REFERENCED - none
OCLC PROCEDURES REFERENCED - none
VIII.51

PROGRAM: MDBUPD
SUBROUTINE: LOGMSG

FUNCTIONS

LOGMSG formats and prints a log entry for each OCLC record number which is selected for catalog card production. A log entry on the CNVT Log consists of the OCLC control number, the color code, and the holding library code followed by a statistical code showing whether the record was selected (SLD) or rejected (RJD). LOGMSG also prints diagnostic messages when required by CNVT.
SOFTWARE INTERFACE

A. LINKAGE

Control is transferred from CNVT via a

BAL,R7 LOGMSG

This instruction must be immediately followed by the parameter list described below. Upon entry to LOGMSG, R7 points to the parameter list.

B. PARAMETER LIST DESCRIPTION

The following list of parameters must be passed to LOGMSG.

<table>
<thead>
<tr>
<th>GEN,8,24</th>
<th>FUN,BA(MESSAGE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>WA(UNPACKED LC CARD NUMBER)</td>
</tr>
<tr>
<td>DATA</td>
<td>WA(COLOR CODE)</td>
</tr>
<tr>
<td>DATA</td>
<td>WA(LIBRARY CODE)</td>
</tr>
</tbody>
</table>

Where the byte indicator 'FUN' may assume the following values:

- FUN = 0 Print message only, do no logging.
- FUN = 1 Log as selected before printing a message.
- FUN = 2 Log as missing before printing a message.
- FUN = 3 Log as rejected before printing a message.
- FUN = 15 Eject page when printing a message.

No message will be printed if BA(MESSAGE) is equal to zero.

C. RETURN CODES - none

D. OTHER ENTRY POINTS - none

E. OCLC SUBROUTINES REFERENCED - none

F. OCLC PROCEDURES REFERENCED - none
TAPEIO is a general purpose input/output subroutine which performs the following functions depending on a function code passed from the calling program.

<table>
<thead>
<tr>
<th>FUNCTION CODE</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'00'</td>
<td>READ</td>
</tr>
<tr>
<td>01</td>
<td>WRITE</td>
</tr>
<tr>
<td>02</td>
<td>READ REVERSE</td>
</tr>
<tr>
<td>03</td>
<td>WEOF</td>
</tr>
<tr>
<td>04</td>
<td>SKIP ONE RECORD FORWARD</td>
</tr>
<tr>
<td>05</td>
<td>SKIP ONE RECORD BACKWARD</td>
</tr>
<tr>
<td>06</td>
<td>SKIP ONE FILE FORWARD</td>
</tr>
<tr>
<td>07</td>
<td>SKIP ONE FILE BACKWARD</td>
</tr>
<tr>
<td>08</td>
<td>REWIND (ONLINE)</td>
</tr>
<tr>
<td>09</td>
<td>UNLOAD</td>
</tr>
</tbody>
</table>

TAPEIO sets up the FPT to be used in IOEX CAL2 from parameters passed by the calling program. If the function required does not involve data transfer (in the range of codes 3-9), the only parameters needed by TAPEIO are the function code, the unit address, and an event word. If data transfer is to be performed (codes 0,1,2), TAPEIO must also have the address of a buffer and the length of the data to be read or written. Upon entry to TAPEIO, general register 1 should be pointing to a user-defined work area on a double word boundary.

If the function to be performed involves data transfer or is a WEOF, two function parameter tables (FPT's) are set up. The first FPT is for the operation requested; the second is used to sense the device status in the event the requested operation does not end normally. For non-data transfer functions, only one FPT is constructed.

TAPEIO contains its own end action routine, STDEA. STDEA uses the Test Device (TDV) status returned by the IOEX CAL2 to determine the end action required. If the I/O operation terminated normally, the first byte of the event word in the first FPT is set to X'80' and control is returned. If the operation ended abnormally, the TDV status is interrogated more closely to determine the exact result of the operation.

A table of TDV status values and their meanings follows:
SUBROUTINE: TAPFIO

<table>
<thead>
<tr>
<th>TDV STATUS</th>
<th>EXPLANATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0200</td>
<td>NORMAL TERMINATION BEYOND END OF TAPE</td>
</tr>
<tr>
<td>0400</td>
<td>NORMAL TERMINATION AT BEGINNING OF TAPE</td>
</tr>
<tr>
<td>B87E</td>
<td>NORMAL TERMINATION</td>
</tr>
<tr>
<td>000E</td>
<td>IOP ERROR</td>
</tr>
<tr>
<td>0010</td>
<td>MEMORY ADDRESS ERROR</td>
</tr>
<tr>
<td>2000</td>
<td>WRITE PROTECT VIOLATION</td>
</tr>
<tr>
<td>1000</td>
<td>END OF FILE</td>
</tr>
<tr>
<td>8000</td>
<td>DATA OVERRUN</td>
</tr>
<tr>
<td>0800</td>
<td>NON-CORRECTABLE READ ERROR</td>
</tr>
<tr>
<td>0040</td>
<td>TRANSMISSION DATA ERROR</td>
</tr>
<tr>
<td>0020</td>
<td>TRANSMISSION MEMORY ERROR</td>
</tr>
</tbody>
</table>

A TDV status of 'B87E' initiates the return of a normal completion code (X'80') to the user. If the status is '1000', an end of file indication is returned. If the TDV status is '000E', '0010', or '2000', the error is not attributed to the I/O device, and no retry is attempted. If the status is one of the last four in the table, the retry count is interrogated. The retry count is arbitrarily set in TAPEIO to ten for data transfer operations (function codes 0-2) and WEOF (code 3) and is set to zero for non-data operations (codes 4-9). If the retry count for this operation is zero, an abnormal return code is posted, and control is returned to the calling program. If the retry count is greater than zero, retry procedures are initiated based on the type of I/O function that was attempted.

If the status is '0200' or '0400', a code is returned to indicate the position of the tape.

If the operation was a READ and the error is correctable (TDV status of '8000', '0040', or '0020'), the second FPT is pulled from the work area and used to sense the device. If the sense does not take, an unconditional backspace and retry are initiated; otherwise STDEA will alternately backspace, or forward space (depending on whether the READ was forward or reverse), sense, retry, and sense until either the retry count is zero or the I/O operation has been performed. If the retry count reaches zero before the operation has been terminated, the condition code returned is the result of the last retry.

If the operation was a READ but the error was declared non-correctable (TDV status '0800'), STDEA initiates an unconditional retry. It backspaces, or forward spaces if the operation was READ REVERSE, and attempts to READ again. The TDV status is interrogated after each retry of the READ. If the error status becomes correctable before the retry count is zero, STDEA will initiate sensing of the device and the correctable READ error procedure. In any case, retry continues until the operation is completed normally or the retry count reaches zero. If the retry count becomes zero before the operation has terminated normally the condition code returned is the result of the last retry.
If the operation was a WRITE or WEOF, STDEA automatically backspaces, senses, and attempts the operation again. This procedure continues until the I/O is complete or the retry count is zero. If the retry count reaches zero before the operation has been terminated normally, the condition code returned is the result of the last retry.

At its alternate entry point, TAPEWAIT, TAPEIO checks for completion of an I/O operation performed by TAPEIO. If the event is not complete TAPEWAIT issues a CAL2,9 0 to wait for completion. When the event is posted complete, the status is interrogated. If the completion is normal (X'80'), control is returned to the return address plus one. If the completion is abnormal (X'CO') control is returned at the return address. In either case BYTE0 of the event word is returned in bits 24-31 of R8.
SOFTWARE INTERFACE

A. LINKAGE

The calling sequence for TAPEIO is as follows:

LI,R1  WORKAREA
LI,R8  PARMS
BAL,R7  TAPEIO

Where WORKAREA is a 16-word storage area aligned on a doubleword boundary.

B. PARAMETER LIST DESCRIPTION

For function codes 0, 1, 2

<table>
<thead>
<tr>
<th>WORD 0</th>
<th>FUNCTION</th>
<th>DEVICE ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD 1</td>
<td>BA (BUFFER)</td>
<td></td>
</tr>
<tr>
<td>WORD 2</td>
<td>BYTE COUNT</td>
<td></td>
</tr>
<tr>
<td>WORD 3</td>
<td>EVENT STATUS</td>
<td></td>
</tr>
</tbody>
</table>

For functions 3-9

<table>
<thead>
<tr>
<th>WORD 0</th>
<th>FUNCTION</th>
<th>DEVICE ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD 1</td>
<td>EVENT STATUS</td>
<td></td>
</tr>
</tbody>
</table>

C. RETURN CODES

NORMAL COMPLETION: EVENTWORD BYTE 0 = X'30'
BYTE 1 = X'00'

ABNORMAL COMPLETION: EVENT WORD BYTE 0 = X'0C'
BYTE 1 = XX - CODE INDICATING NATURE OF ABNORMAL COMPLETION.

Possible event words for abnormal completion and their meanings are listed below:
<table>
<thead>
<tr>
<th>EVENT WORD</th>
<th>TDV STATUS</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>0200</td>
<td>NORMAL TERMINATION</td>
</tr>
<tr>
<td>002</td>
<td>0400</td>
<td>BEYOND END OF TAPE MARKER</td>
</tr>
<tr>
<td>003</td>
<td>0610</td>
<td>NORMAL TERMINATION AT BEGINNING OF TAPE</td>
</tr>
<tr>
<td>004</td>
<td>0800</td>
<td>IOP ERROR</td>
</tr>
<tr>
<td>005</td>
<td>0040</td>
<td>MEMORY ADDRESS ERROR</td>
</tr>
<tr>
<td>006</td>
<td>0020</td>
<td>WRITE PROTECT VIOLATION</td>
</tr>
<tr>
<td>007</td>
<td>8000</td>
<td>END OF FILE</td>
</tr>
<tr>
<td>008</td>
<td>1000</td>
<td>DATA OVERRUN</td>
</tr>
<tr>
<td>009</td>
<td>2000</td>
<td>NON-CORRECTABLE READ ERROR</td>
</tr>
<tr>
<td>010</td>
<td>8000</td>
<td>TRANSMISSION DATA ERROR</td>
</tr>
<tr>
<td>011</td>
<td>0040</td>
<td>TRANSMISSION MTDY ERROR</td>
</tr>
<tr>
<td>012</td>
<td>0020</td>
<td>UNIT UNRECOGNIZED SOFTWARE ERROR</td>
</tr>
</tbody>
</table>

For codes C000-C003 and C008-C00B, no retry has been attempted. For codes C004-C007, retry has been attempted only if the function was a data transfer or WEOF.

D. OTHER ENTRY POINTS
   TAPEWAIT

E. OCLC SUBROUTINES REFERENCED - none

F. OCLC PROCEDURES REFERENCED - none
SOFTWARE INTERFACE (TAPEWAIT)

A. LINKAGE

LI,R8       PARMS
BAL,R7      TAPEWAIT

B. PARAMETER LIST DESCRIPTION

same as for TAPEIO

C. RETURN CODES:

BYTE 0 of the user provided EVENT WORD is returned in
bits 24-31 of R8

D. OTHER ENTRY POINTS - none

E. OCLC SUBROUTINES REFERENCED - none

F. OCLC PROCEDURES REFERENCED - none
APPENDIX E

ADDITIONAL PROCEDURE DOCUMENTATION
PURPOSE:

DTAGS establishes the parameters for the table 'TAGS' which assigns an index to each field to be used for all the other tables.

FORMAT:

DTAG AF(1)

EXAMPLE:

| DATA,1 | 0 | index for 001 field |
| DATA,1 | 1 | index for 001 field |
| DATA,1 | 0 | index for 008 field |
| DATA,1 | 0 | index for 008 field |
| DATA,1 | 0 | index for 008 field |
| DATA,1 | 0 | index for 008 field |
| DATA,1 | 2 | index for 008 field |
PROCEDURE DESCRIPTION

PURPOSE:

MSG established the parameters for the external subroutine 'MDBUPDLG'. The CNAME value of MSG is 1 and indicates to MDBUPDLG that a single message is to be logged. There are three alternate names for MSG:

- HEX - has a CNAME equal to 2 and indicates that the message is to be converted to hexadecimal
- DEC - has a CNAME equal to 3 and indicates that the message is to be converted to BCD/DFC.
- MEND - has a CNAME equal to 0 and indicates the end of the parameter list.

FORMAT:

Example 1:

```plaintext
MSG MSG71
HEX ECH1
MSG MSG72
MEND
+ GEN,8,24 1, BA (MSG71)
+ GEN,8,24 2, BA (ECH1)
+ GEN,8,24 3, BA (MSG72)
+ DATA 0
```

Example 2:

```plaintext
DEC ECV3
MSG MSG75
DEC ECV4
MSG MSG76
MEND
+ GEN,8,24 1, BA (ECV3)
+ GEN,8,24 2, BA (MSG75)
+ GEN,8,24 3, BA (ECV4)
+ GEN,8,24 3, BA (MSG76)
+ DATA 0
```
1. All data must be correct ASCII data.
2. The actual length of the record must equal the record length in the leader.
3. The Type of Record must be 'a' (lower case ASCII).
4. The Bibliographic Level must be 'm' (lower case ASCII).
5. 'BLANKS' (position 8-9 in the leader) must be blank.
6. The Indicator Count must be 2.
7. The base address of the data must be a multiple of 12 (leader = 24, Directory = 12).
8. The Subfield Code Count must be 2.
9. There must be a field terminator (X'1E') at the end of the directory.
10. There must be a record terminator (X'1D') at the end of the record.
11. Leader positions 18-19 must be all blank.
12. The Entry Map in the Leader must be all blank or be of the form '4500'.
13. There can be no more than 50 fields in a record.
14. The value of the tag can not exceed '340'.
15. The value of the tag must be in the following list of valid tags:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>086</td>
</tr>
<tr>
<td>008</td>
<td>100</td>
</tr>
<tr>
<td>015</td>
<td>110</td>
</tr>
<tr>
<td>020</td>
<td>111</td>
</tr>
<tr>
<td>025</td>
<td>130</td>
</tr>
<tr>
<td>040</td>
<td>240</td>
</tr>
<tr>
<td>041</td>
<td>241</td>
</tr>
<tr>
<td>043</td>
<td>245</td>
</tr>
<tr>
<td>050</td>
<td>250</td>
</tr>
<tr>
<td>051</td>
<td>260</td>
</tr>
<tr>
<td>050</td>
<td>300</td>
</tr>
<tr>
<td>070</td>
<td>350</td>
</tr>
<tr>
<td>082</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
16. The record is rejected if the LC card number is not all numeric.

17. The Record Status must be one of the following: D (delete), NC (new), C (corrected), or P (full replacing CIP). If it is not one of the above, the Record Status is changed to N (new) and a warning is logged.

18. The Encoding Level must be one of the following: 1 (sublevel 1), 8 (full replacing CIP), or blank (full level).

19. The record is rejected if the 001 field is not the first field in the directory.

20. The record is rejected if the 001 field is not at least 13 characters or is more than 30 characters long.

21. The record is rejected if the 008 field is not the second entry in the directory.

22. The record is rejected if the length of the 008 is not 41 characters including the field terminator.

23. The Type of Publication Date Code must be one of the following:
   s (date is known), c (consists of two dates), n (date is unknown), r (previously published), m (multiple date), q (one or more digits are missing).

24. The Country of Publication Code must be in the list of valid codes.

25. The Illustration Codes must be one of the following:
   a (illustration), b (maps), c (portraits), d (charts),
   e (plans), f (plates), g (music), h (facsimiles),
   i (coates of arms), j (genealogical tables), k (forms),
   l (samples), m (phono disc), M = blank (no illustrations).

26. Elements 11-15 of the 008 field (Conference, Festschrift, Index, Main Entry, and Fiction Indicators) must be either 0 or 1.

27. The Government Publication Indicator must be 0.

28. The Form of Reproduction Code must be one of the following:
   a (Microfilm), b (Microfiche), c. (Micropage), d (Large-print),
   M = blank (not a reproduction).

29. The Intellectual Level Code must be either 'j' or M = blank.
30. The Form of Content Codes must be one of the following:
   b (Bibliographies), c (Catalogs),
   i (Indexes), a (Abstracts),
   d (Dictionaries), e (Encyclopedias),
   r (Directories), y (Yearbooks),
   s (Statistics), h (Handbooks),
   p (Programmed Testbooks), or
   $= blank (not a specified form).

31. The Language Code must be in the list of valid language codes.

32. The Cataloging Source Code must be one of the following:
   a (National Agricultural Library),
   b (National Library of Medicine),
   c (Cooperative Cataloging), or
   $ = blank (Library of Congress).

33. The Modified Record Indicator must be either d, s, x, or a blank.

34. The Biography Code must be one of the following:
   a (Autobiography), b (Individual),
   c (Collective), d (Contains biographical information),
   or $ = blank (not biographical).

35. The record is rejected if the directory has only 1 or 2 entries.

36. Every field must be at least five characters long.

37. The minimum byte length for the 020 field is 14.

38. The minimum byte length for the 025 field is 10.

39. The minimum byte length for the 040 and 043 fields is 12.

40. The minimum byte length for the 041 field is 11.

41. The minimum byte length for the 082 field is 6.

42. The minimum byte length for every other field (excluding 001, 008, 020, 025, 040, 041, 043, and 082 fields) is 7.

43. Check to see that fields 041, 050, 240, 241, 245, 260, 490, and 750 have a first indicator of 0 or 1.

44. Check to see that fields 110, 111, 410, 411, 505, 510, 611, 710, 711, 810, and 811 have a first indicator of 0, 1, or 2.

45. Check to see that fields 100, 400, 600, 700, and 800 have a first indicator of 0, 1, 2, or 3.
46. Check to see that the first indicator for every other field is a blank.

47. Check to see that the second indicator for the 100, 110, 111, 130, 400, 410, and 411 fields is a 0 or 1.

48. Check to see that the second indicator for the 700, 710, 711, 730, 740, and 750 fields is a 0, 1, or 2.

49. Check to see that the second indicator for the 600, 610, 611, 630, 650, 651, and 652 fields is a 0, 1, 2, 3, or 4.

50. Check to see that the second indicator for every other field is a blank.

51. Every field must begin with a subfield delimiter (X'1F').

52. Every field must end with a field terminator (X'FD').

53. Verify that the subfield codes for the 041, 050, 060, 070, and 250 fields are either 'a' or 'b'.

54. Verify that the subfield codes for the 051, 245, 260, and 300 fields are either 'a', 'b', or 'c'.

55. Verify that the subfield codes for the 100 field are either 'a'. 'b', 'c', 'd', 'e', 'k', or 't'.

56. Verify that the subfield codes for the 110 field are either a, b, e, k, or t.

57. Verify that the subfield codes for the 111 field are either a, b, c, d, e, g, k, or t.

58. Verify that the subfield codes for the 130 field are either a or t.

59. Verify that the subfield codes for the 400 and 800 fields are either a, b, c, d, e, k, t, or v.

60. Verify that the subfield codes for the 410 and 810 fields are either a, b, e, k, t, or v.

61. Verify that the subfield codes for the 411 and 811 fields are either a, b, c, d, e, g, k, t, or v.

62. Verify that the subfield codes for the 440 and 840 fields are either a or v.

63. Verify that the subfield codes for the 600 field are either a, b, c, d, e, k, t, x, y, or z.

64. Verify that the subfield codes for the 610 field are either a, b, e, k, t, x, y, or z.

65. Verify that the subfield codes for the 611 field are either a, b, c, d, e, g, k, t, x, y, or z.
66. Verify that the subfield codes for the 630 field are either a, t, x, y, or z.

67. Verify that the subfield codes for the 650 and 651 fields are either a, b, x, y, or z.

68. Verify that the subfield codes for the 652 field are either a, x, y, or z.

69. Verify that the subfield codes for the 700 field are either a, b, c, d, e, k, t, or u.

70. Verify that the subfield codes for the 710 field are either a, b, e, k, t, or u.

71. Verify that the subfield codes for the 711 field are either a, b, c, d, e, g, k, t, or u.

72. Verify that the subfield codes for the 730 field are either a, t, or u.

73. Verify that the subfield codes are 'a' for following fields:
   (015 043 350 504)
   (020 082 490 505)
   (025 086 500 520)
   (040 240 501 740)
   (241 502 750)

74. A field terminator must come directly before the record terminator.

75. Verify that the 245 and 260 fields are present in every record.

76. Verify that the 025, 255, and all the fields from 400 to 840 do not occur more than 255 times in one record.

77. Verify that there are no more than two 050 fields in each record.

78. Verify that all the other fields (all fields from 001 to 350 except 025, 050, and 255) do not occur more than once in each record.