DOCUMENT RESUME

ED 080 121	LI 004 426
AUTHOR	Kilgour, Frederick G., Comp.; Davis, Hillis D., Comp.
TITLE	The Development of a Computerized Regional Library System. Appendix 25. Final Report.
INSTITUTION	Ohio Coll. Library Center, Columbus.
SPONS AGENCY	Office of Education (DHEW), Washington, D.C.
BUREAU NO	BR-9-0554
PUB DATE	Jun 73
CONTRACT	OEC-0-70-2289(506)
NOTE	148p.; (0 References)
EDRS PRICE	MF-$0.65 HC-$6.58
DESCRIPTORS	Bibliographic Citations; *Cataloging; College Libraries; Computer Programs; Expenditure Per Student; Library Automation; *Library Expenditures; *Library Networks; *On Line Systems; Regional Programs; Union Catalogs; *University Libraries
IDENTIFIERS	OCLC; *Ohio College Library Center

ABSTRACT

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-five, The Ohio College Library Center Program/Subroutine Documentation; Convert Call Number (CNVT). CNVT is the first step in the formatting and production of catalog cards. The primary function is to format the call number for each catalog card request according to the predetermined specifications. To accomplish the individual format, CNVT uses a massive tree structure of information accessed by profiles, one profile per member holding library. Secondary functions of CNVT include formatting of some user data and selective deletion of unnecessary data from the member profile. (Other appendices are LI 004 423 through LI 004 425, LI 004 427 and 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 25

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)

X. The Technical Processing System, May 1972. (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)

XV. Large On-Line Files of Bibliographic Data: An Efficient Design and a Mathematical Predictor of Retrieval Behavior. P.L. Long, K.B.L. Rastogi, J.E. Rush and J.A. Wyckoff. (Not Available EDRS)

XVI. OCLC Systems: Technical Aspects. Phillip Long. and Ohio State University Libraries Systems. Gerry D. Guthrie. (Not Available EDRS)

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 (GENPDTS). (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)

XXV

THE OHIO COLLEGE LIBRARY CENTER
PROGRAM/SUBROUTINE DOCUMENTATION

Convert Call Number
(CNVT)

CONTENTS

		Section/Page
I.	OVERVIEW	I
II.	DATA FLOW	II
III.	SUMMARY OF INPUT AND OUTPUT	III
	Bibliographic Record	III.2
	CNVT Output Record	III.14
	CNVTPDTS File	III.17
IV.	FUNCTIONS	IV
V.	GENERAL INFORMATION FLOW	V
VI.	SOFTWARE INTERFACE	VI
VII.	DESCRIPTION OF SPECIAL STORAGE AREAS, SWITCHES, AND TABLES	VII
	A. Special Storage	
	Format Control Word (FCW)	VII.1
	Profile Definition Table (PDT)	VII.2
	B. Tables	VII.3
VIII.	APPENDIX	VIII
	A. Operating Requirements	VIII.2
	B. Operating Characteristics	VIII.3
	C. Detailed Description of Internal Subroutines	VIII.13
	D. Additional Subroutine Documentation	VIII.40
	E. Additional Procedure Documentation	VIII.83
	F. Examples	VIII.88

OCLC PROGRAM DOCUMENTATION

I. Overview

CNVT is the first step in the formatting and production of catalog cards to be sent to Members. For the online system, CNVT has as input the System Log tape from a day's on-line operation. There are five possible types of records on the log tape. CNVT selects only Type 1 (produce) records for its processing; the remaining records serve as archive information. CNVT may also be run off-line to produce catalog cards as requested by card input from Members. The input for the off-line CNVT is the disk data base. The functions of CNVT in both the on-line and off-line modes are the same.

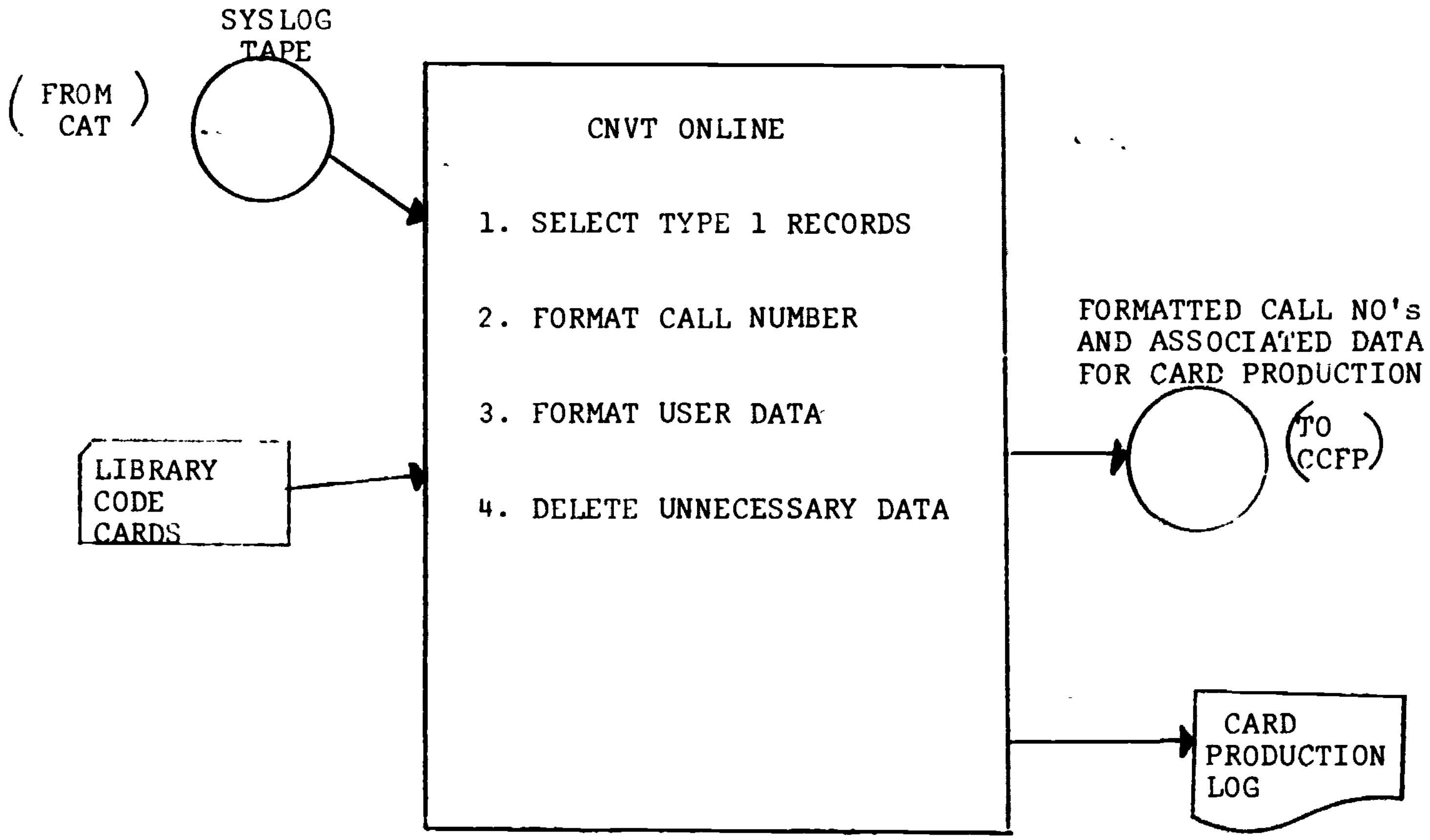
The primary function of CNVT is to format the call number for each catalog card request according to the requesting Member's pre-determined specifications. To accomplish the individual format, CNVT uses a massive tree structure of information accessed by profiles, one profile per member holding library. Each tree structure has "leaves" which indicate the routines within CNVT necessary to process the call number and associated data for the Member.

Secondary functions of CNVT include formatting of some user data and selective deletion of unnecessary data from the Type 1 record depending on the member profile.

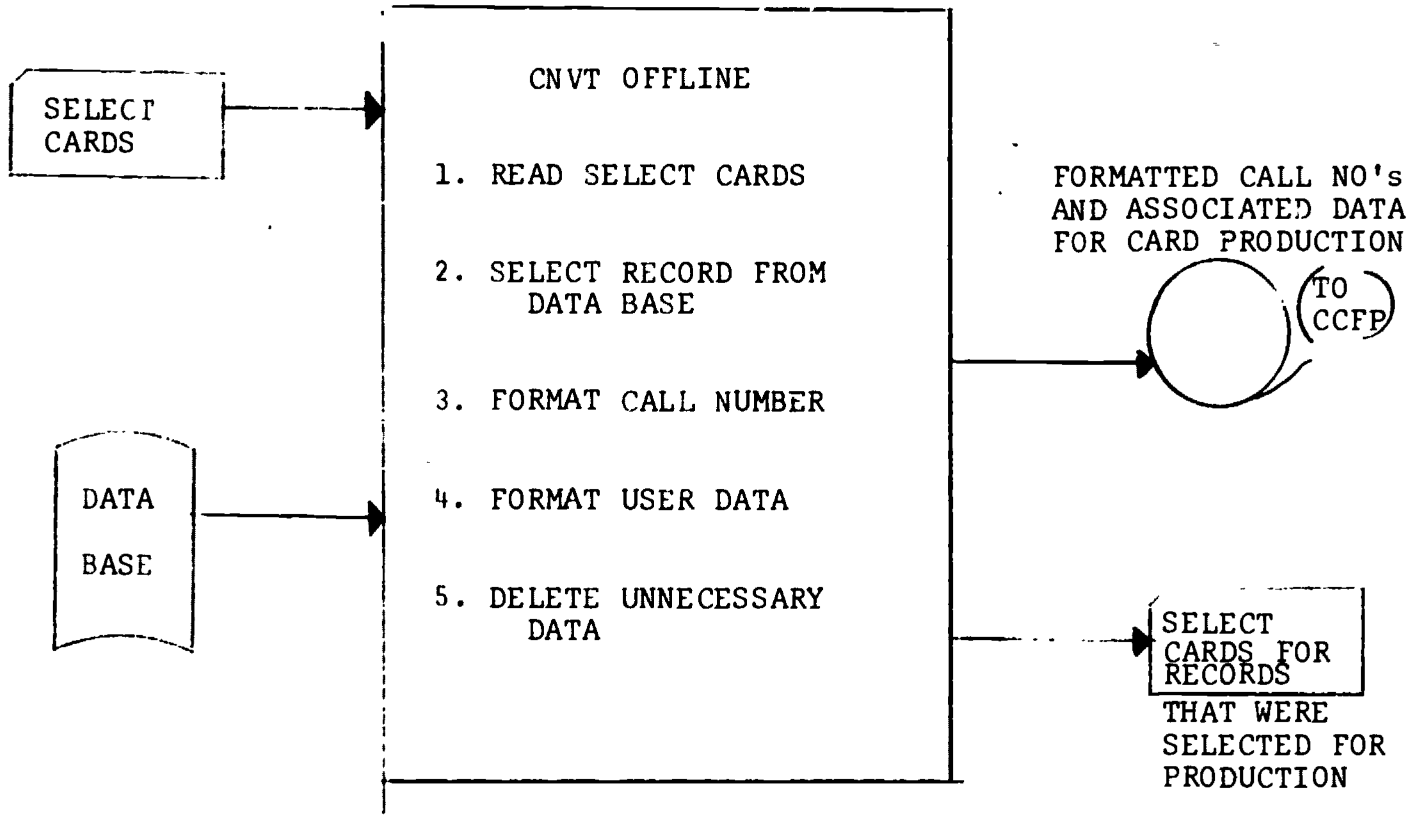
The output of CNVT is records on tape which contain the formatted call numbers, user data, and the additional data necessary to format the catalog cards.

II. DATA FLOW

DATA FLOW CHART A



DATA FLOW CHART B


SELECT CARDS
DATA BASE
CNVT OFFLINE
1. READ SELECT CARDS
2. SELECT RECORD FROM DATA BASE
3. FORMAT CALL NUMBER
4. FORMAT USER DATA
5. DELETE UNNECESSARY DATA
FORMATTED CALL NO's AND ASSOCIATED DATA FOR CARD PRODUCTION
TO CCFP
SELECT CARDS FOR RECORDS THAT WERE SELECTED FOR PRODUCTION

III. SUMMARY OF INPUT AND OUTPUT

the ohio college library center
1314 kinnear rd. - columbus ohio - 43212

Record Layout

File Name BIBLIOGRAHIC DATA FILE

Record Name ARCHIVE TAPE RECORD LEADER

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

File Organization IBM VARIABLE BLK FORMAT Record Size 4130 Block Size 4130

General Description Standard leader on all archive tape records

FIELD NAME AND DESCRIPTION	POSITION	FIELD LENGTH	FORMAT
VARIABLE BLOCK CONTROL WORD			
Logical Record Length	0-1	2	Binary
Zeros	2-3	2	Binary
RECORD TYPE CODE	4	1	Hexadecimal
X'01' - PRODUCE			
X'02' - UPDATE			
X'03' - CA UPDATE			
X'10' - MBD ADD			
X'11' - MDB REPLACE			
X'12' - MDB DELETE			
DATE OF TRANSACTION	5-7	3	Unsigned packed decimal, YYMMDD
INSTITUTION CODE	8-11	4	EBCDIC right justified, 0 filled
CATALOGER	12-15	4	EBCDIC left justified blank filled

Record Layout (Cont)

File Name BIBLIOGRAPHIC DATA FILE

Record Name ARCHIVE TAPE RECORD LEADER

FIELD NAME AND DESCRIPTION	FIELD		
	POSITION	LENGTH	FORMAT
CLASSIFICATION: 0 or 1 = LC, 2 = DC	16-17	2	Binary
TUBE NUMBER Logical tube number +1	18-19	2	Binary
"FORCE UNIT CARD" FLAG	20(0)	1 bit	Boolean
"ADDED COPY" FLAG	20(1)	1 bit	Boolean
RESERVED FOR OTHER CCFP FLAGS	20(2-7)	6 bits	Zero
NUMBER OF EXTRA CARDS	21	1	Binary
RESERVED	22-31	10	Zero

IBM DIAGRAMMING AND CHARTING WORKSHEET

REVISION OF 12/29/71

Application BIBLIOGRAPHIC DATA FILE Date 3/27/72 Page 1 of 1

~~Procedure~~ ARCHIVE TAPE RECORD LEADER Drawn By H. F. CERINE

BYTE

0	2	3
LOGICAL RECORD LENGTH	ZERO	ZERO

0	1
RECORD TYPE CODE*	DATE OF TRANSACTION (UNSIGNED PACKED DECIMAL) YYMMDD

0
INSTITUTION CODE (EBCDIC, RIGHT JUSTIFIED, ZERO FILL)

0
CATALOGER (EBCDIC, LEFT JUSTIFIED, BLANK FILL)

0	2
TYPE OF INST (1 = LC 2 = DEWEY)	TUBE NUMBER + 1

0
RESERVED

0
RESERVED

0
RESERVED

Fold under at dotted line.

Fold under at dotted line.

*RECORD TYPE CODES:

X'Ø1' – PRODUCE

X'Ø2' – UPDATE

X'Ø3' – CA UPDATE

X'1Ø' – MDB ADD

X'11' – MDB REPLACE

X'12' - MDB DELETE

the ohio college library center
1314 kinnear rd. - columbus ohio - 43212

Record Layout

File Name BIBLIOGRAPHIC DATA FILE

Record Name BIBLIOGRAPHIC RECORD

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

File Organization IBM Variable Blk Format Record Size 45-6144 Block Size 6144

General Description OCLC internal processing format of the MARC II Bibliographic Record. Access is either sequential or random.

FIELD NAME AND DESCRIPTION	POSITION	FIELD LENGTH	FORMAT
RECORD LEADER			
Logical Record Length	0-1	2	Binary
Record Status Character (MARC Manual - Page 26).	2	1	EBCDIC
Encoding Level (MARC Manual - Page 27)	3	1	EBCDIC
Leader Length - byte size of leader including terminator (X 'FD')	4	1	Binary
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.	5	Upper 4 bits	Binary

Record Layout (Cont)

File Name BIBLIOGRAPHIC DATA FILE

Record Name BIBLIOGRAPHIC RECORD

FIELD NAME AND DESCRIPTION	FIELD		
	POSITION	LENGTH	FORMAT
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.	5	Lower 4 bits	Binary
Reserved	6-7	10 Bits	Binary
Variable Control Field Length - Word length of field between supplement number and suffix character in LC card number.	7	Lower 6 bits	Binary
OCLC Number	8-11	4	Binary
Date Entered			
Year	12	1	Packed*
Month	13	1	Packed*
Day	14	1	Packed*
Type of Publication Date - Description of contents of Publication Date fields (See MARC Manual pp. 32-34).	15	1	EBCDIC
Publications Dates			
Date #1	16-17	2	Packed*
Date #2	18-19	2	Packed*
Country of Publication - First two characters of MARC field (See MARC Manual pp. 35, 290-318).	20-21	2	EBCDIC
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.	22-23	2	Binary

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

Record Layout (Cont)

File Name BIBLIOGRAPHIC DATA FILE

Record Name BIBLIOGRAPHIC RECORD

FIELD NAME AND DESCRIPTION	FIELD		
	POSITION	LENGTH	FORMAT
Form of Content Code Indexes - four 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.	24-25	2	Binary
Intellectual Level Index - index 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.	26	Upper 4 bits	Binary
Format Reproduction Code Index - Index 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.	26	Lower 4 bits	Binary
Indicators 10 thru 15 - bit switches to indicate the MARC indicators described in the MARC Reference Manual (pp. 37-38, par. 10-15). Bit values are Bit 0 - REserved 1&2 - Government Pub. Ind. 3 - Conference Pub. Ind. 4 - Festschrift Ind. 5 - Index Ind. 6 - Main Entry Ind. 7 - Function Ind.	27	1	Binary
Biography Code Index - index into 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.	28	1	Binary

Record Layout (Cont)

File Name BIBLIOGRAPHIC DATA FILE

Record Name BIBLIOGRAPHIC RECORD

FIELD NAME AND DESCRIPTION	FIELD		
	POSITION	LENGTH	FORMAT
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).	29	Upper 4 bits	Binary
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.	29	Lower 4 bits	Binary
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 "Cataloging on a Cathode Ray Tube Terminal" pp. 46-52.	30-31	2	Binary
LC-Card Number			
Prefix	32-34	3	EBCDIC
Year Part	35	1	Packed*
Number Part	36-38	3	Packed*
Supplement number	39	1	EBCDIC
Length of 1st Author Substring The number of bytes to use for the first author substring	40-41	2	Binary
Displacement of 1st Author Substring Byte displacement to the 1st author substring from end of leader	42-43	2	Binary

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

Record Layout (Cont)

File Name BIBLIOGRAPHIC DATA FILE

Record Name BIBLIOGRAPHIC RECORD

FIELD NAME AND DESCRIPTION	FIELD		
	POSITION	LENGTH	FORMAT
Length of 2nd Author Substring The number of bytes to use for the second author substring.	44-45	2	Binary
Displacement of 2nd Author Substring Byte displacement to the 2nd author substring from end of leader	46-47	2	Binary
Length of Title Substring The number of bytes to use for the title substring.	48-49	2	Binary
Displacement to Title Substring The byte displacement to the title substring from the end of the leader.	50-51	2	Binary
Holdings File Pointer Word Pointer to holdings list.	52-55	4	Binary
Institutional Holdings Bits Bit switches indicating holdings for an institution. A one indicates holdings, a zero indicates no holdings.	56-71	16	Binary
LC Suffix A variable length character string which may be absent. 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.	Variable	Variable	EBCDIC
Leader Terminator X "FD" that follows the suffix to indicate the end of the leader.	Variable	1	Binary

Record Layout (Cont)

File Name BIBLIOGRAPHIC DATA FILE

Record Name BIBLIOGRAPHIC RECORD

FIELD NAME AND DESCRIPTION	FIELD		
	POSITION	LENGTH	FORMAT
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	0-1 **	2	Binary
Element Length - length of element including tag.	2-3 **	2	Binary
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.	4-n **		EBCDIC
** These value 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

Application On-Line Bibliographic Record Date 11/1/71 Page 1 of 3

Procedure Drawn By J. Gilchrist

0 LOGICAL RECORD LENGTH | 2 RECORD STATUS | 3 ENCODING LEVEL

4 TOTAL LENGTH OF LEADER, INCL. XCF? | 5 TYPE INDEX | BIBLIOGRAPHIC LEVEL IND. | 6 RESERVED (10 BITS) | 7 WORD LENGTH OF FIELDS DE- [illegible] FIX & SUFF #

8 OCLC RECORD NUMBER

12 DATE ENTERED ON FILE (PACKED): YEAR | MONTH | DAY | 15 TYPE OF PUBLICATION DATE

16 DATE 1 (PACKED) | 18 DATE 2 (PACKED)

20 COUNTRY OF PUBLICATION (TWO CHARACTERS) | 22 ILLUSTRATION CODES INDEX (4): #1 | #2 | #3 | #4

24 FORM OF CONTENTS CODE INDEX (FOUR): #1 | #2 | #3 | #4 | 26 INTELLECTUAL LEVEL INDEX | FORM OF REPRODUCTION CODE INDEX | 27 INDICATORS 10-15 OF 008 FIELD (PACKED BITS)

28 BIBLIOGRAPHY CODE INDEX | 29 MODIFIED RECORD CODE INDEX | CATALOG SOURCE INDEX | 30 LANGUAGE CODE INDEX

32 L.C. CARD NUMBER ALPHA PREFIX | 35 L.C. CARD NUMBER 'YEAR' PART (PACKED)

36 L.C. CARD NUMBER "NUMBER" PART (PACKED) | 39 L.C. CARD NUMBER SUPPLEMENT NUMBER

40 LENGTH OF FIRST AUTHOR SUBSTRING IN BYTES | 42 BYTE DISPLACEMENT TO FIRST AUTHOR SUBSTRING (FROM END OF LDR)

44 LENGTH OF SECOND AUTHOR SUBSTRING IN BYTES | 46 BYTE DISPLACEMENT TO SECOND AUTHOR SUBSTRING (FROM END OF LDR)

Fold under at dotted line.

Fold under at dotted line.

IBM DIAGRAMMING AND CHARTING WORKSHEET

Application CA-LIAS Cumulative Bibliographic Record Date 11/?/71 Page 2 of 3

Procedure ______ Drawn By [illegible]

48 LENGTH OF TITLE SUBSTRING	50 BYTE DISPLACEMENT TO TITLE FROM END OF LEADER

52 HOLDINGS FILE POINTER WORD
56 WORD # 1 OF INSTITUTIONAL HOLDING SWITCHES
60 WORD #2
64 WORD #3
68 WORD #4 *

SUFFIX ** | LEADER TERMINATOR X'FD'

* 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 #7 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.

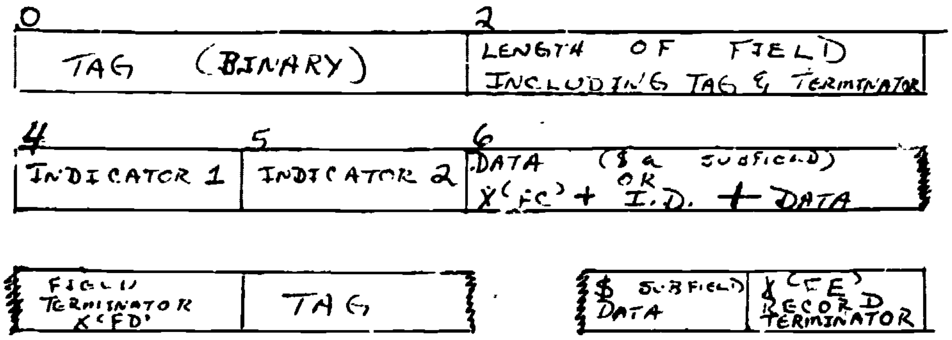
IBM DIAGRAMMING AND CHARTING WORKSHEET

Application On Line Canonical Bibliographic Record Date 11/12/71 Page 3 of 3

Procedure _____ Drawn By _____

FOLLOWING THE LEADER ARE THE TAG FIELDS. THESE FIELDS ARE VARIABLE IN LENGTH AND IMMEDIATELY FOLLOW THE PREVIOUS FIELD (OR LEADER FOR THE FIRST ELEMENT). EACH IS TERMINATED BY A DELIMITER (X'FD' OR X'FE' FOR THE LAST ELEMENT). THEY FOLLOW THE BASIC MARC-II FORMAT WITH THE FOLLOWING FOUR EXCEPTIONS

1. THE TAG ITSELF IS BINARY
2. THE FIELD LENGTH IS IN BINARY
3. THE FIRST $a IS NOT INCLUDED IN THE RECORD IF PRESENT. IF THE '$' (X'FC') does NOT IMMEDIATELY FOLLOW THE INDICATORS, THE CHARACTER IS DATA and the SUB-FIELD IS THE $a SubFIELD. OTHERWISE, The DATA IS STD.
4. THE CHARACTER X'FC' REPLACES THE '$'.



0 TAG (BINARY)

2 LENGTH OF FIELD INCLUDING TAG & TERMINATOR

4 INDICATOR 1

5 INDICATOR 2

6 DATA ($a SUBFIELD) OR X'FC' + I.D. + DATA

FIELD TERMINATOR X'FD'

TAG

$ SUBFIELD DATA

X'FE' RECORD TERMINATOR

the ohio college library center
1314 kinnear rd. - columbus ohio - 43212

Record Layout

File Name CNVT OUTPUT RECORD

Record Name

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

File Organization SEQUENTIAL Record Size ∠4096 Block Size UNBLOCKED

General Description

FIELD NAME AND DESCRIPTION	FIELD		
	POSITION	LENGTH	FORMAT
RECORD LEADER			
PDT Number	0-1	2	Binary
Unit Card Indicator	2 (bit 0)	1 bit	Binary
Added Copy Indicator	2(bit 1)	1 bit	Binary
Reserved	2(bits 2-7)	6 bits	
Number of Extra Cards	3	1	Binary
OCLC Number	4-7	4	Binary
LC Card Number	8-19	12	EBCDIC
Reserved	20-27	8	Binary
Language Code Index	28-29	2	Binary

[illegible] OUTPUT

bit 0 ——unit card indicator
bit 1 --added copy indicator

0		2			3
PDT Number		bit 0	bit 1	bits 2-7 reserved	# of extra cards

4
OCLC Record Number

8
LC Card Number

12
LC Card Number

16
LC Card Number

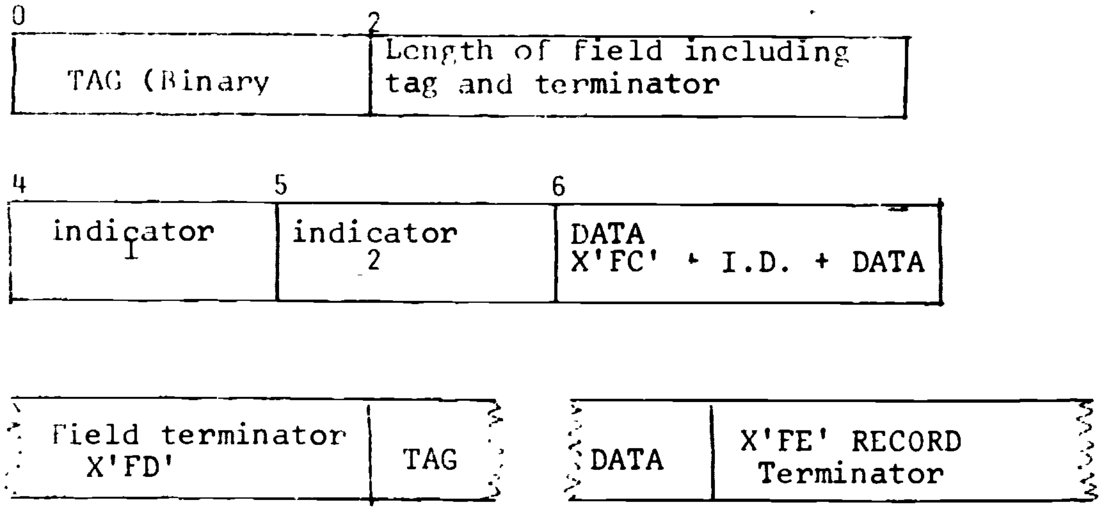
20
Reserved

24
Reserved

28
Language Code Index

Following the leader are the tag fields. These fields are variable in length and immediately follow the previous field (or leader for the first element). Each is terminated by a delimiter (S'FD' or X'FE' for the last element). They follow the basic MARC II format with the following exceptions:

1. The tag itself is Binary
2. The field length is Binary
3. ‡P (X'PC97'), ‡B (X'FC82'), or ‡C (X'FC83') within the call number field are used to indicate that a stamp is to be placed in that position relative to the call number.



the ohio college library center
1314 kinnear rd. - columbus ohio - 43212

Record Layout

File Name CNVTPDTS

Record Name TBLA, BLOCK #1

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

File Organization KEYED Record Size 256 WORDS Block Size 256 WORDS

General Description Block #1 of TBLA is prefixed by the key to TBLB and the number of entries in TBLA.

FIELD NAME AND DESCRIPTION	POSITION	FIELD LENGTH	FORMAT
KEY TO TBLB	1-2	2	BINARY
NO. ENTRIES IN TBLA	3-4	2	BINARY
TBLA, ENTRY #1	5-8	4	EBCDIC
.	.	.	
.	.	.	
.	.	.	
TBLA, ENTRY #255	1021-1024	4	EBCDIC

the ohio college library center
1314 kinnear rd. - columbus ohio - 43212

Record Layout

File Name CNVTPDTS

Record Name TBLA, BLOCK #N

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

File Organization KEYED Record Size 256 WORDS Block Size 256 WORDS

General Description A normal block of TBLA consists strictly of 256 library code entries.

FIELD NAME AND DESCRIPTION	POSITION	FIELD LENGTH	FORMAT
TBLA, ENTRY #256* KEY#	1-4	4	EBCDIC
.	.	.	
.	.	.	
.	.	.	
TBLA, ENTRY #(511* KEY#) + KEY #-1	1021-1024	4	EBCDIC

the ohio college library center

1314 kinnear rd. - columbus ohio - 43212

Record Layout

File Name CNVTPDTS

Record Name TBLB, BLOCK #1

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

File Organization KEYED Record Size 256 WORDS Block Size 256 WORDS

General Description Block #1 of TBLB is prefixed by the key to PDTTBL. The halfword TBLB entries follow the key.

FIELD NAME AND DESCRIPTION		FIELD	
	POSITION	LENGTH	FORMAT
KEY TO PDTTBL	1-2	2	BINARY
TBLB, ENTRY, #1	3-4	2	BINARY
⋮	⋮	⋮	⋮
TBLB, ENTRY, #511	1023-1024	2	BINARY

the ohio college library center

1314 kinnear rd. - columbus ohio - 43212

Record Layout

File Name CNVTPDTS

Record Name TBLB, BLOCK #N

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

File Organization KEYED Record Size 256 WORDS Block Size 256 WORDS

General Description A normal block of TBLB consists strictly of 512 halfword index entries to PDTTBL

FIELD NAME AND DESCRIPTION	POSITION	FIELD LENGTH	FORMAT
TBLB, ENTRY #512* KEY#	1-2	2	BINARY
.	.	.	.
.	.	.	.
.	.	.	.
TBLB, ENTRY #(1023* KEY#) + KEY#-1	1023-1024	2	BINARY

Record Layout

File Name CNVTPDTS

Record Name PDTTBL, BLOCK #U

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

File Organization KEYED Record Size 256 WORDS Block Size 256 WORDS

General Description A block of PDTTBL contains a variable number of PDT subtables.

FIELD NAME AND DESCRIPTION	POSITION	FIELD LENGTH	FORMAT
LIB #1 SUBTABLE			
Cycle	1-2	2	Binary
Brown	3-4	2	Binary
Blue	5-6	2	Binary
Yellow	7-8	2	Binary
Red	9-10	2	Binary
Green	11-12	2	Binary
Lit	13-14	2	Binary
PDT#	15-16	2	Binary
Oversize#	17-18	2	Binary
Tag Handler #1	21-24	4	Binary
.			
.			
.			
Tag Handler #K	K-K+4	4	Binary
End of Table Indicator	K+5-K+6	2	-1
No. of Parameters (L)	K+7-K+8	2	Binary
Parameters	K+9-K+9+2L-1	2L	Binary

IV. FUNCTIONS

CNVT reads any librarv code cards that are input and builds the table LIBSIN to control production of catalog cards for only these Members. If no library code cards are input, LIBSIN is initialized with its number of entries equal to zero. CNVT then begins normal processing. It reads a record from the daily system log tape and interrogates the record type in the archive record leader. If the record type is 01, the total number of selects is incremented by one and card production begins. If the record is a type other than 01, the next record is read.

The Library of Congress card number is picked up from the selected record and is stored in a location called 'LASTLCCN'. The institution code is stored in a location called 'LIB'.

A Link Directory is built by branching to the program 'LINK'. In this directory there are several tables. 'LNKTAG' is a table of tags; 'LNKBA' is a corresponding table of byte addresses and lengths of each of these fields. Therefore, in order to find the byte address and length of any field, a search is performed on the table 'LNKTAG' until the tag is found. This search will produce an index into the table 'LNKBA', where the byte address and the length of that field are to be found.

After the Link Directory has been built, the 049 field is found. Anything within brackets in this field indicates a stamp and is moved to the location 'STAMP'. The first three unbracketed characters indicate the cataloging library and are moved to the location 'LIB'. If there is no 049 field, the insititution code which was previously stored at 'LIB' is used as the cataloging source. If this library code is not in the list of libraries to be processed, in table 'LIBSIN', the record is counted as rejected and the next record is read. If there are zero entries in LIBSIN, all libraries may be processed.

A table, called 'TBLA', of default tag processors is built. Each tag with its corresponding processor is pulled from a stack and stored in the table. Then, the options in the program 'READPDT's' are read. If there are any special tag options indicated, the processor is picked up from the table 'TBLAOPTS'. 'TBLA' is modified based on this information from 'READPDT's' and the two processors are exchanged. Each field is then processed according to these 'TBLA' options. If the field is to be deleted, its entrv in the Link Directory is deleted.

The root number is retrieved from READPDTS and this is used as an index into 'NODETBL' to find the appropriate tree. The first element in the argument field of the 'NODE' instruction indicates the number of entries in the tree and the second element indicates where to start getting the entries. These entries are stored in the stack 'CSTK'. If there is a 'NODE' in the tree, it is expanded and these entries replace the 'NODE' in the stack. If there is a 'TEST' in the tree, the loop switch is tested. For each 'TEST', there are two alternative LEAF's or NODE's. If the switch has been set, the first alternative is pushed into the stack. If the switch is not set, the second alternative is pushed into the stack.

Each 'LEAF' entry in the stack is processed in sequence utnil a 'NODE' or 'TEST' is encountered, at which time the appropriate replacement routines are pushed into the stack.

The first number in the argument field of the 'LEAF' is an index into the table of routines called 'EXUTBL', where the address of the routine is loaded into R7. For the routines 3:U001 - 3:U087, the address of the Format Control Word (FCW) is also loaded into R6 at this time.

The second number in the argument field of the 'LEAF' is used as an indicator within the routine and is always passed to the routine in R3.

A branch is performed through R7 to each routine in the stack in sequence. Upon return from the routine, all registers are cleared and the address of the next routine is retrieved from the table. When the stack is empty, the next record is read and processed.

In order to save the contents of a register when going from one routine to another, its contents are stored in a core image location called 'REGx + 16' where 'x' is the register number. Through most of the processing R1, R2, and R3 are saved in this manner. R1 is used as an index to the temporary call number field; R2 has the complement of the call number width; and R3 has the byte address of the source of the unformatted record.

The call number formating routines basically take each element, one at a time, format it and move it to an area called '090T' which is a temporary call number field. From here, it is moved in its completed form, along with stamps and oversize symbols, if any, to an area called 'FIELD2'. The elements of the call number and their corresponding numbers are:

1) LC alpha prefix	AA
2) Classification-numeric portion	NNN
3) Classification-decimal portion	.NN
4) First date	NNNNA
5) First cutter	ANN
6) Second date	NNNNA
7) Second cutter	ANN

The sequence of the routines in the tree follows a pattern. The first routine is always a set-up routine. If any element or elements are to be suppressed, the routine to do this must precede the routine to process the suppressed element. If any stamp or symbol is to go in the left margin, the call number width must be decremented before any element is processed.

If no elements are to be suppressed and nothing is to go in the left margin, the next seven routines, after the initial set-up, will process each element, one at a time. For a Dewey call number, the routine to process the first element is absent.

The next routine will usually be 3:U008 which will process any other elements.

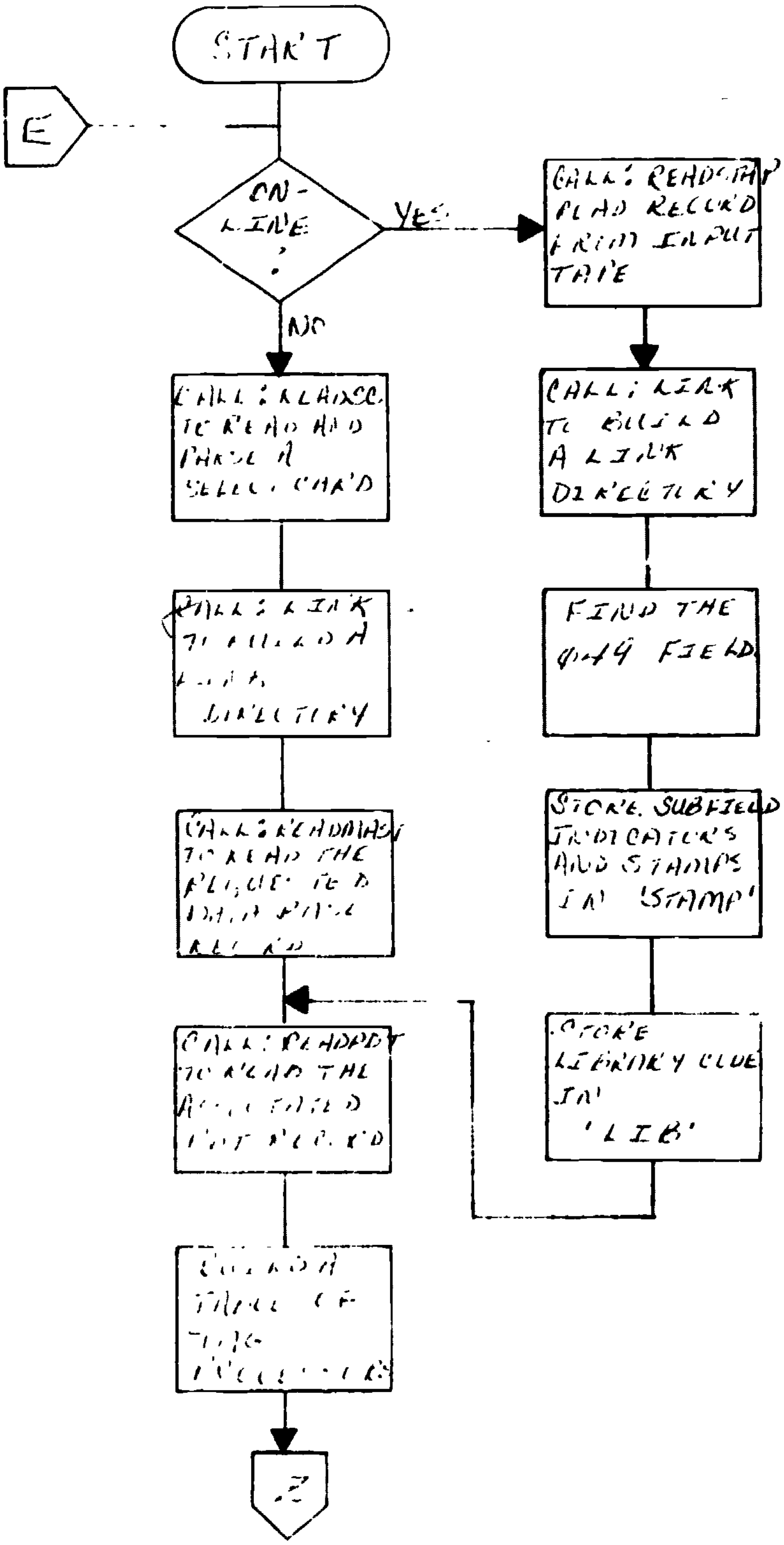
The next routines after the elements have been formatted, are the routines to set up and move the user data to 'FIELD2' and link FIELD2 to the call number field. The routines to create extra cards are also found here. The next set of routines determine the arrangement of the three stamps and the oversize symbol in relation to the call number. These routines move the stamps, the symbols, and the call number to 'FIELD2' in the order in which they will appear on the card.

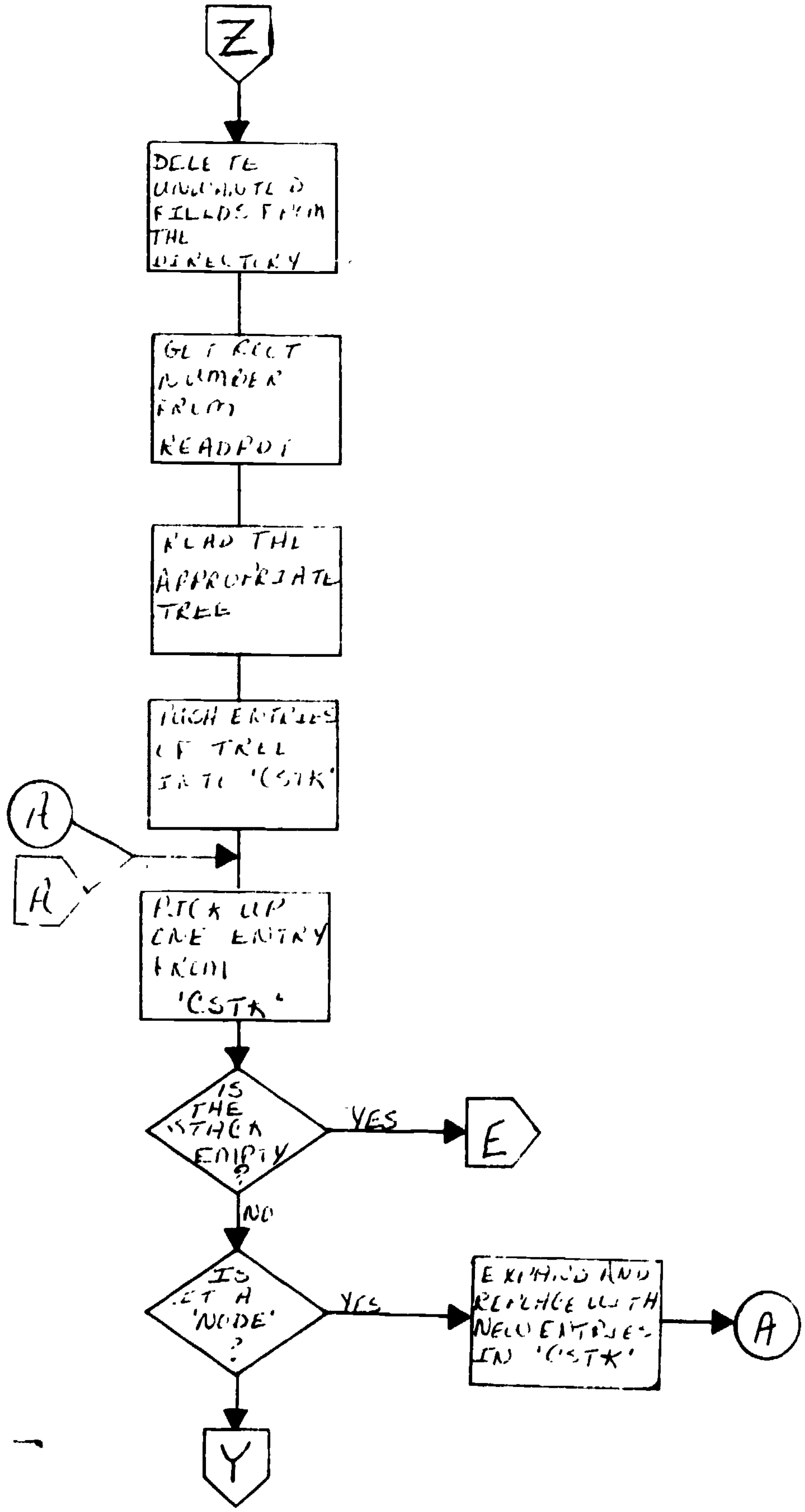
The last four routines are the same for every set-up. The first routine, 5:U999, provides for holdings. The last three will link 'FIELD2' to the rest of the record (4:U999); provide for the card to be produced (X:U000); and log it as having been selected (X;U001).

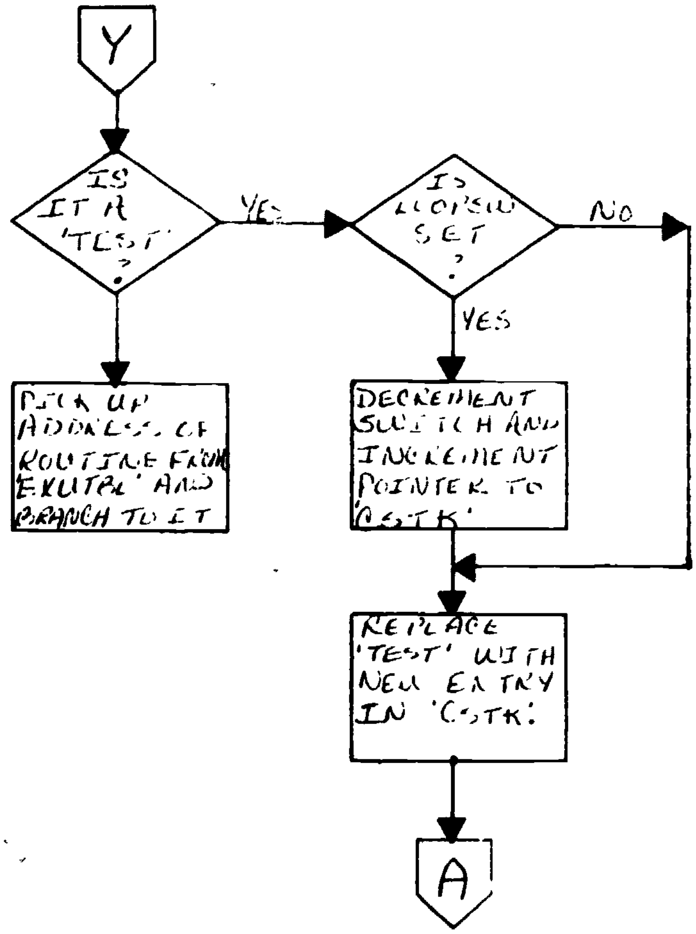
For the off-line CNVT, the functions are basically the same. The table LIBSIN is built from any library code cards input. CNVT then reads a member select card and accesses the disk data base for the records necessary for card production. The member select cards are color keyed depending on their function. A description of the color codes of select card input is given in Appendix B, Operating Characteristics.

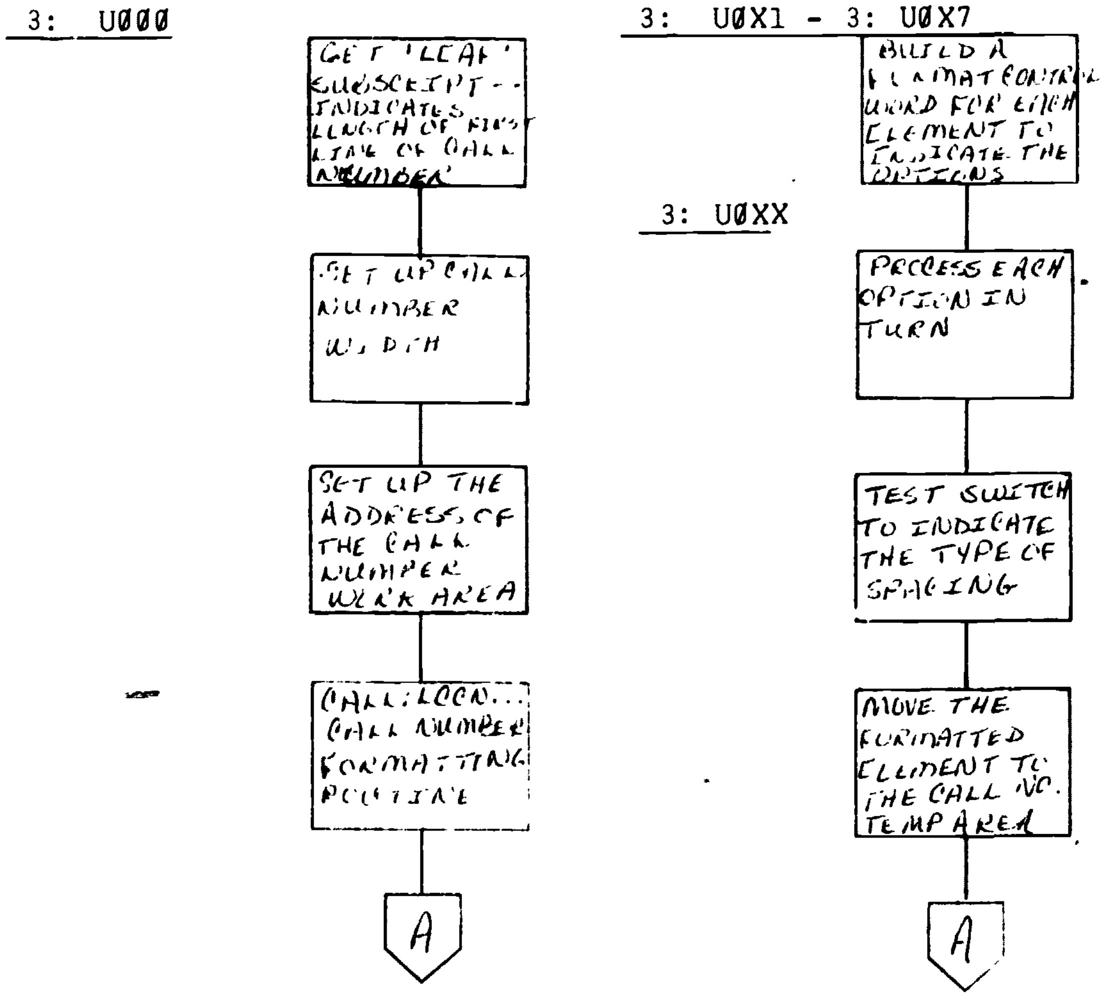
After the select card is read and the records for card production are obtained from the data base, card production proceeds the same as for the online system. As each set of cards is produced by the offline system, a card is punched which contains the unpacked Library of Congress card number and the library code. These cards are used to eliminate from the input cards, those cards which were selected for production.

V. GENERAL INFORMATION FLOW

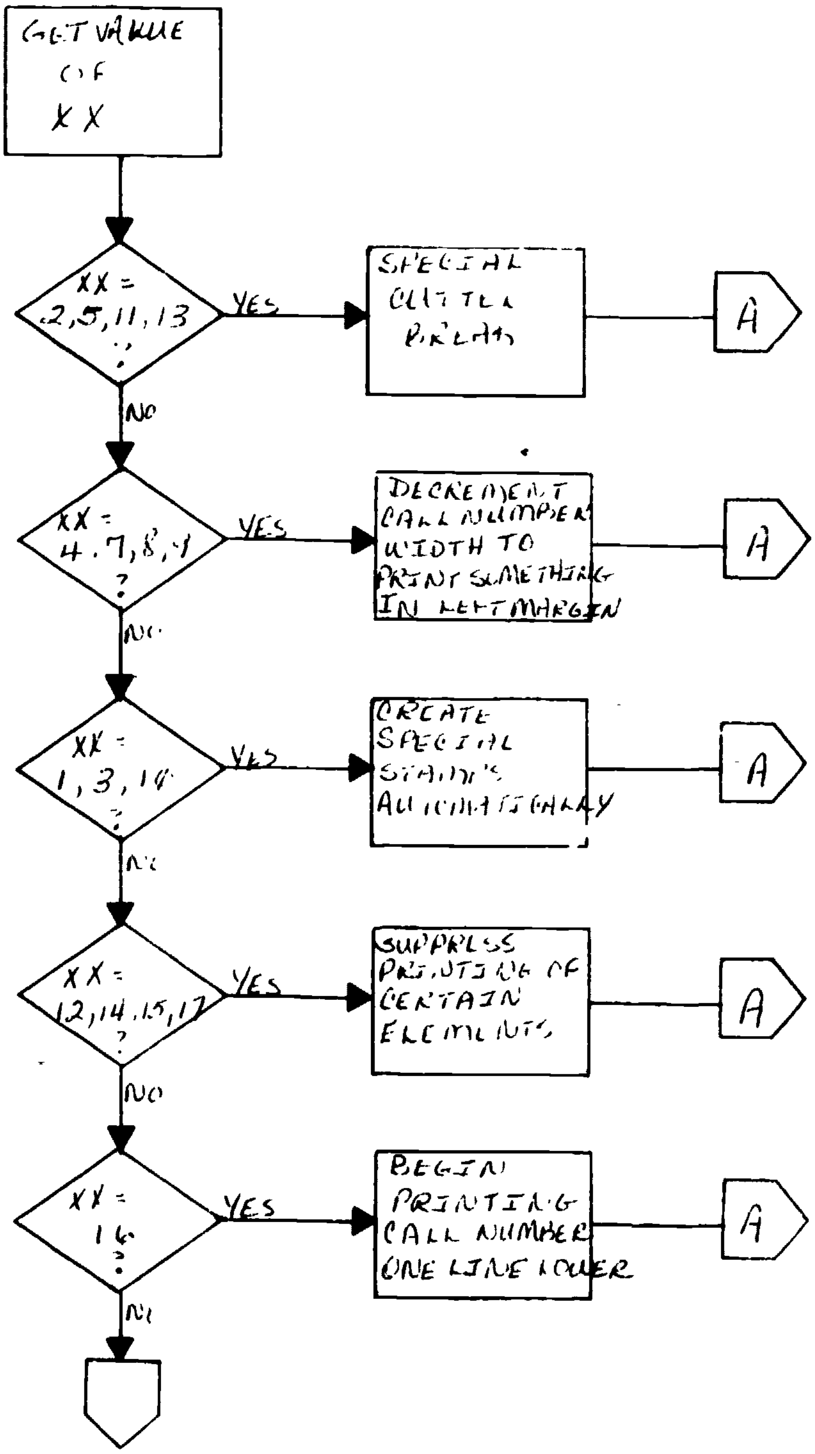

START
E
ON-LINE?
YES
NO
CALL: READTAP READ RECORD FROM INPUT TAPE
CALL: READSC TO READ AND PARSE A SELECT CARD
CALL: LINK TO BUILD A LINK DIRECTORY
CALL: LINK TO BUILD A LINK DIRECTORY
FIND THE 049 FIELD
CALL: READMAS TO READ THE REQUESTED DATA BASE RECORD
STORE SUBFIELD INDICATORS AND STAMPS IN 'STAMP'
CALL: READPDT TO READ THE ASSOCIATED PDT RECORD
STORE LIBRARY CODE IN 'LIB'
[illegible] A TABLE OF THE [illegible]
Z


Z
DELETE UNWANTED FIELDS FROM THE DIRECTORY
GET RECT NUMBER FROM READPOT
READ THE APPROPRIATE TREE
PUSH ENTRIES OF TREE INTO 'CSTK'
A
A
PICK UP ONE ENTRY FROM 'CSTK'
IS THE STACK EMPTY ?
YES
E
NO
IS IT A 'NODE' ?
YES
EXPAND AND REPLACE WITH NEW ENTRIES IN 'CSTK'
A
Y

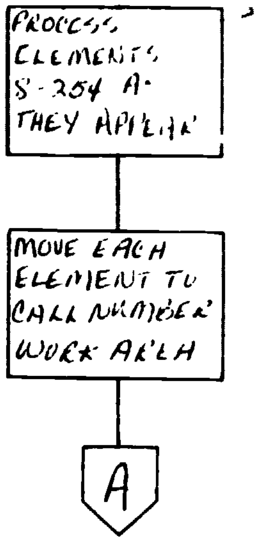

Y
IS IT A 'TEST' ?
YES
IS LOOPSW SET ?
NO
YES
PICK UP ADDRESS OF ROUTINE FROM 'EXUTBL' AND BRANCH TO IT
DECREMENT SWITCH AND INCREMENT POINTER TO 'CSTK'
REPLACE 'TEST' WITH NEW ENTRY IN 'CSTK'.
A


3: UØØØ
GET 'LEAF' SUBSCRIPT -- INDICATES LENGTH OF FIRST LINE OF CALL NUMBER'
SET UP CALL NUMBER WIDTH
SET UP THE ADDRESS OF THE CALL NUMBER WORK AREA
CALL: LOCN... CALL NUMBER FORMATTING ROUTINE
A
3: UØX1 - 3: UØX7
BUILD A FORMAT CONTROL WORD FOR EACH ELEMENT TO INDICATE THE OPTIONS
3: UØXX
PROCESS EACH OPTION IN TURN
TEST SWITCH TO INDICATE THE TYPE OF SPACING
MOVE THE FORMATTED ELEMENT TO THE CALL NO. TEMP AREA
A

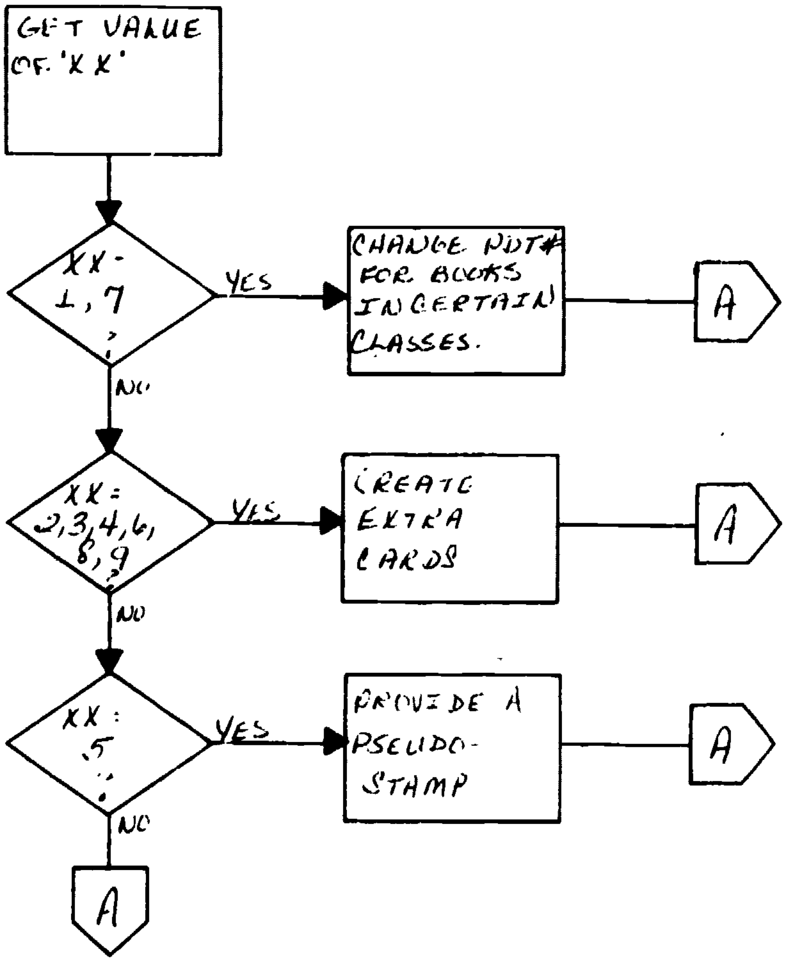
3:XØXX


GET VALUE OF XX
XX = 2,5,11,13 ?
YES
SPECIAL CUTTER BREAKS
A
NO
XX = 4,7,8,9 ?
YES
DECREMENT CALL NUMBER WIDTH TO PRINT SOMETHING IN LEFT MARGIN
A
NO
XX = 1,3,10 ?
YES
CREATE SPECIAL STAMPS AUTOMATICALLY
A
NO
XX = 12,14,15,17 ?
YES
SUPPRESS PRINTING OF CERTAIN ELEMENTS
A
NO
XX = 16 ?
YES
BEGIN PRINTING CALL NUMBER ONE LINE LOWER
A
NO

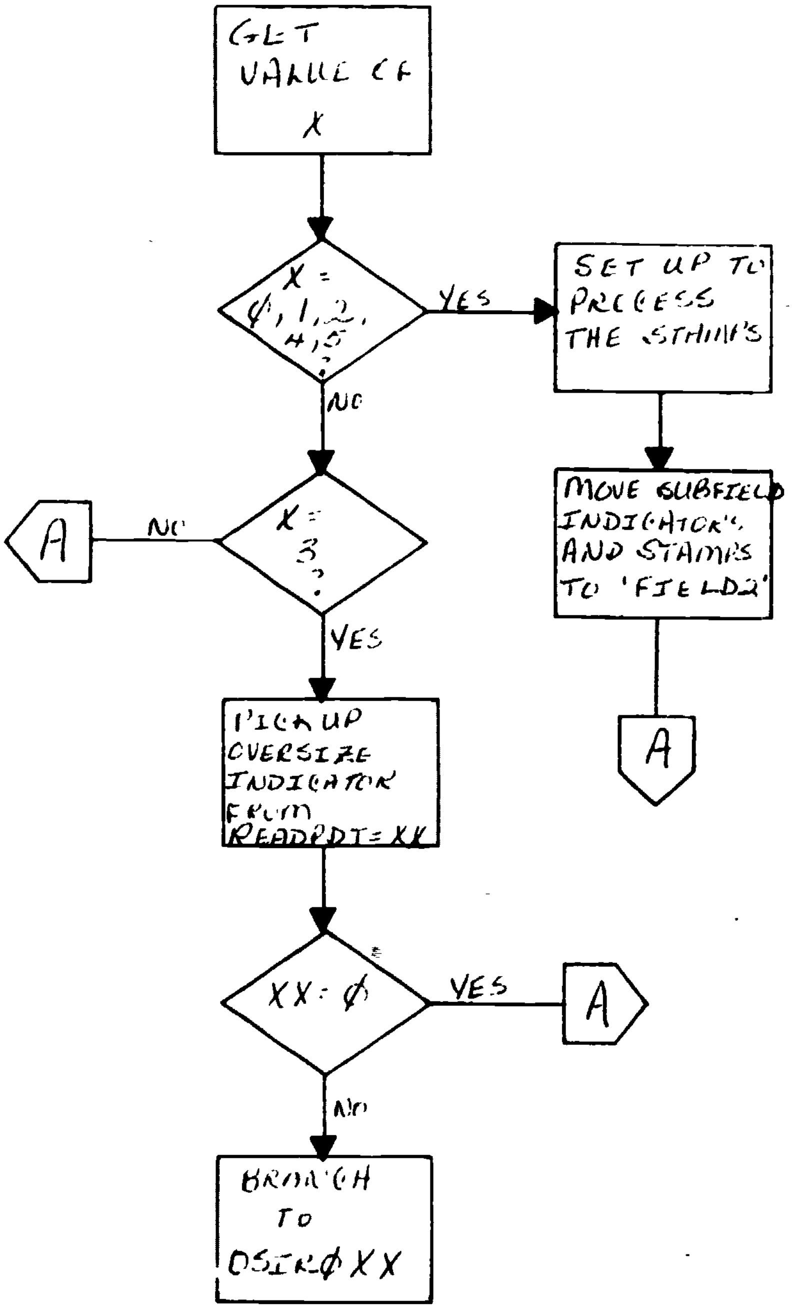
3: UØØ8


PROCESS ELEMENTS S-254 AS THEY APPEAR
MOVE EACH ELEMENT TO CALL NUMBER WORK AREA
A

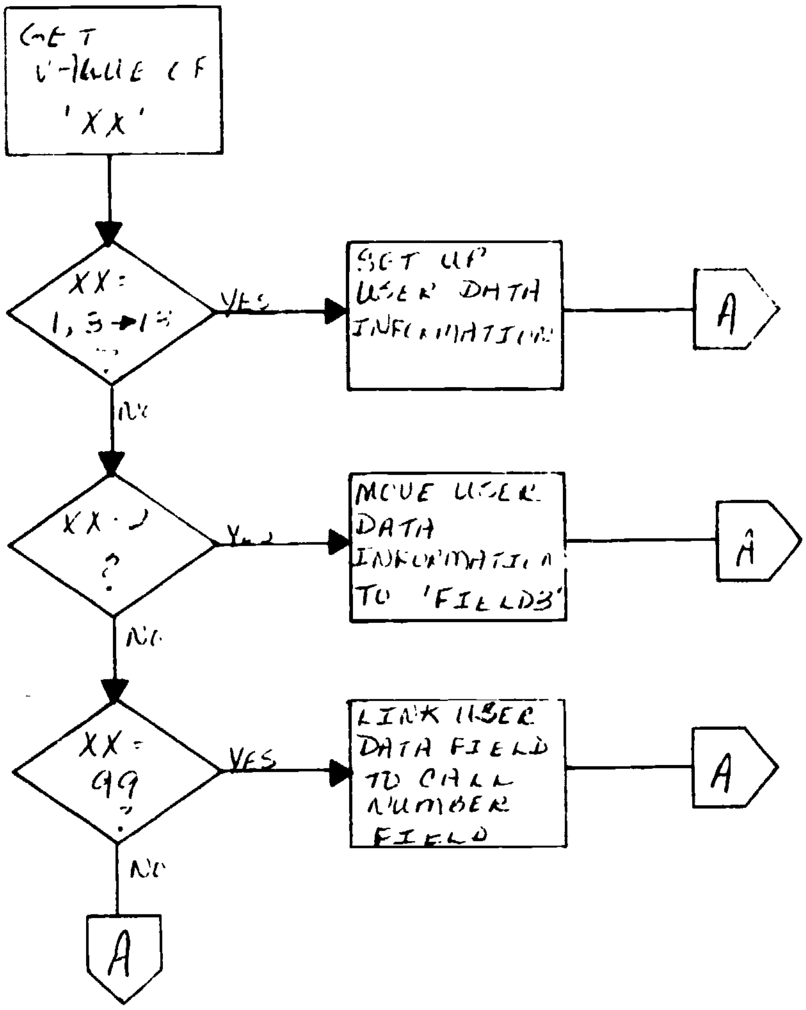
4: XØXX


GET VALUE OF 'XX'
XX = 1,7 ?
YES
CHANGE NDT# FOR BOOKS IN CERTAIN CLASSES.
A
NO
XX = 2,3,4,6, 8,9
YES
CREATE EXTRA CARDS
A
NO
XX = 5 ?
YES
PROVIDE A PSEUDO-STAMP
A
NO
A

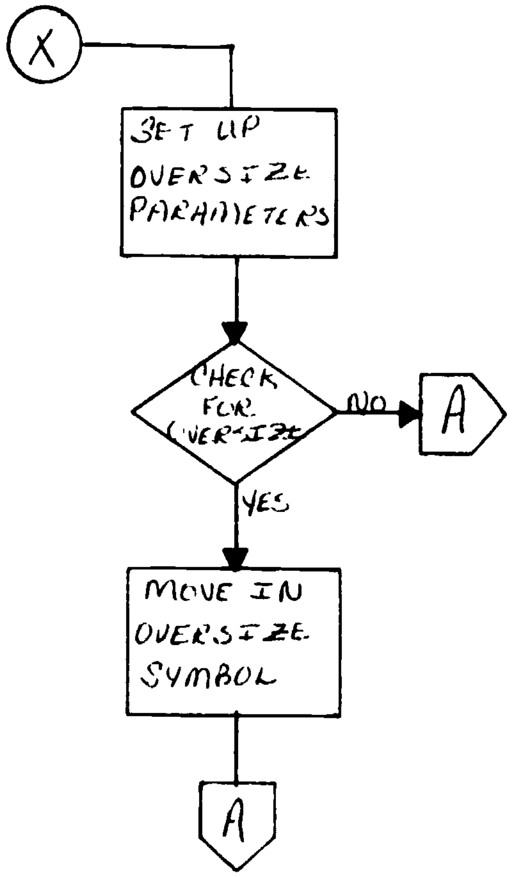
4: UØXX


GET VALUE OF X
X = Ø, 1, 2, 4, 5 ?
YES
SET UP TO PROCESS THE STAMPS
MOVE SUBFIELD INDICATORS AND STAMPS TO 'FIELD2'
A
NO
A
NO
X = 3 ?
YES
PICK UP OVERSIZE INDICATOR FROM READPDT=XX
XX = Ø
YES
A
NO
BRANCH TO OSIRØXX

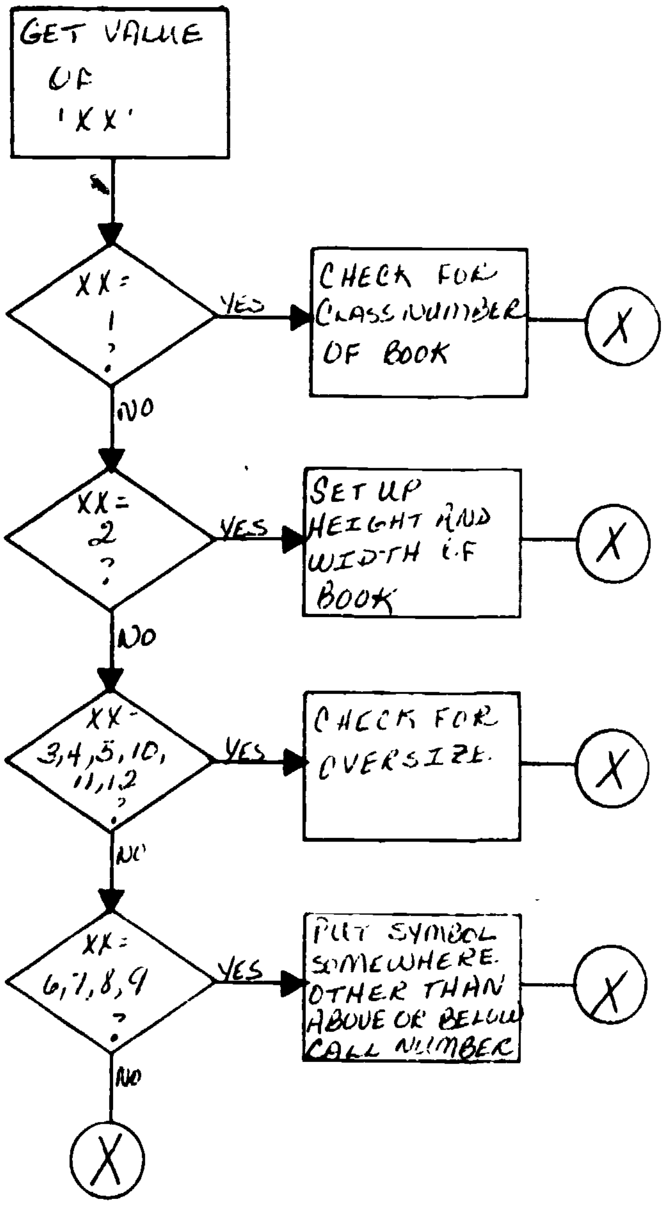
5: UØXX


GET VALUE OF 'XX'
XX = 1, 3 → 18 ?
YES
SET UP USER DATA INFORMATION
A
NO
XX = 2 ?
YES
MOVE USER DATA INFORMATION TO 'FIELD3'
A
NO
XX = 99 ?
YES
LINK USER DATA FIELD TO CALL NUMBER FIELD
A
NO
A

OSIRØXX



OSIUØXX


GET VALUE OF 'XX'
XX = 1 ?
YES
CHECK FOR CLASS NUMBER OF BOOK
X
NO
XX = 2 ?
YES
SET UP HEIGHT AND WIDTH OF BOOK
X
NO
XX = 3,4,5,10, 11,12 ?
YES
CHECK FOR OVERSIZE
X
NO
XX = 6,7,8,9 ?
YES
PUT SYMBOL SOMEWHERE OTHER THAN ABOVE OR BELOW CALL NUMBER
X
NO
X

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 -

READPDT (Alternate EP, READPDT1)

NODETBL

CBIEB

READSC

PUNCHSC (Alternate EP, CLOSESC)

LOGMSG

READMAST

TAPEIO

FMTREC

TAPEIO

LINK (Alternate EP's, LINKDLT, LINKINST)

LCCN000 (Alternate EP's, LCCN000B, LCCN000D)

F. OCLC Procedures Referenced

WRTMSG

WRTSELD

WRTMISS

WRTEJECT

PUNCHSLD

NOTE

ATBL

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

A. Special Storage

FORMAT CONTROL WORD (FCW) - a word defined for each of the different ways an element of a Library of Congress call number may be formatted on a catalog card. Provides a mask to direct the formatting of the LC call number. A sample FCW follows:

```
FCW025  FCW  XUT00:2,XUTM1:2,2,5,3,0,2,0
```

with each field comprised of the following number of bits:

8,8,2,6,2,2,2,2

The first 8 bits are an index into UT00TBL to determine which 'UT00' move routine to use.

The second 8 bits serve as an index into the same table 'UT00TBL' to determin‹ which 'UTM1' move routine to use. The next 2 bits indicate whether or not this element must be present.

=2 - - - need not be present
=3 - - - must be present

The next 6 bits give the element number of the call number.
The next 8 bits (4 bytes) determine the following options:

Bits 0-1	00	NOOP
	01	Start a new line if previous element was 4 or 6
	10	NOOP
	11	Start a new line
Bits 2-3	00	NOOP
	01	Supply a blank
	10	Supply a decimal
	11	Supply a new line and a blank if the element will not fit on previous line
Bits 4-5	00	use UT00 Routine (Move)
	01	if blank first, use UTM1 Routine (delete)
	10	if decimal first, use UTM1 Routine (delete)
	11	if blank or decimal first, use UTM1 routine

```
Bits 6-7  00 NOOP
          01 NOOP
          10 NOOP
          11 Force next element to new line
```

These 8 control Bits are processed in the following sections of the routine 3:UOXX respectively; PH1, PH2, PH3, and PH4.

Therefore, for the FCW in the example, the element of the call number would be processed as follows: It would use the second UT00 move routine and the second UTM1 move routine. This is element number 5 (first cutter) and it need not be present. This element will start a new line, and if it begins with a decimal, the decimal will be deleted.

PROFILE DEFINITION TABLE (PDT) - a table defined for each member holding library which describes that library's specifications for formatting the call number and formatting or deleting user data. The items of information in a PDT include

1. A three digit holding library code.
2. A PDT number which is used by the following format program in the processing sequence (CCFP).
3. The cycling period, which is the number of weeks the data base will be searched for a Library of Congress Card Number before the request is returned to the user (offline system only).
4. A table of indexes into the roots of the table NODETBL, one halfword for each color-coded card (see CNVT - Appendix B, OPERATING CHARACTERISTICS, Parameter Cards Required). In the offline mode, the index in the color branch table indicated by the color code on the input request cards is used for processing. For online processing, the index for a blue card is always used.
5. A table of tag numbers and the number of a special processor for each tag.
6. The call number width
7. The number of the oversize routine for this library.
8. Any parameters needed for processing this holding library including oversize parameters and parameters for specific internal subroutines.

The PDT's are defined in three tables on a direct access device. The tables are generated by the program, CNVTPDTS, and accessed in CNVT by the subroutine READPDT.

B. Special Switches
None

C. Tables

OSIT - An indexable table of oversize routines. The index into the table is REG.5 which is the number of the routine picked up from the program 'READPDT's'. A branch is taken from OSIT to the appropriate oversize routine.

EXUTBL - An indexable table of routines. The index into the table is R2 which is the first entry in the argument field of the 'LEAF' instruction. For the routines 3:U0X1 - 3:U0X7, the byte address of the format control word (FCW) is loaded into R6 and the address of the routine, 3:U0XX, which will format each element is loaded into R7. The instruction: B 0, R7 will effectively cause the processing program to continue at the appropriate routine.

OSIS - A table of oversize symbols. Each symbol is identified by number. The first byte of each of the fields is the length in bytes of the symbol +5 to account for 2 bytes for the subfield code (FC81), 1 byte for the end of field delimiter (FD), and 2 bytes for the sort skip characters on either end of the symbol [0 (zero)].

TBLA - A table of default tag processors. The default processors are pushed into a stack called 'TBLASTK'. 'ITBL' is used to pick up the tag number and the processor in 'TBLA'. After the PDT's have been read, any tag that is to have a special processor is replaced in 'TBLASTK' and all the processors are put back into TBLA. TBLASTK1 and TBLASTK2 are used as stack pointer doublewords. #1 is used to fill up the stack; #2 is used to empty the stack. For each entry in TBLA, the argument field contains the tag number followed by the address of the routine required to process that field.

TBLAOPTS - A table of special tag handlers. The tag number and the index into this table are picked up from 'READPDTS'. These are the processors which will replace the entries in TBLA.

STAMPS - A table of stamps, each of which is set up in the same manner as entries for the table 'OSIS'.

STOPLIST - An alphabetical list of words and abbreviations. It is used to reduce the cataloging source field to 22 characters by abbreviating commonly used words and phrases.

CALL NUMBER BREAKDOWN CONTROL WORD (BRKCW) a word defined

for each way a call number may be set up for formatting. BRKCW provides parameters for the call number set-up routine (3:UØØØ). A sample BRKCW is:

BRKCW 1,45,1

where each field is composed of the following number of bits:

8,8,16

The first 8 bits in the example devote the type of call number

Ø - L.C. call number
1 - Dewey call number
2 - Medical library call number

The next 8 bits are the call number length of first line.

Ø - default to CNWIDTH
7 - Dewey, short first line

45 - long first line

The final 16 bits are an index to a brance table to entry points in the call number parse subroutine LCCNØØØ

0 - LCCN000
1 - LCCN000D
2 - LCCN000B

In the example a set-up for a Dewey call number with a long first line is desired. The entry point in LCCN000 is to be LCCN000D.

VIII. APPENDIX

APPENDIX A

OPERATING REQUIREMENTS

1. Computer - Xerox Sigma 5
2. I/O Devices - Two 800-BPI tape drives, line printer, card reader/punch
3. Operating System - RBM/OBM
4. Execution Time - average 10 minutes clock
5. Run schedule - daily
6. Job Control Language

   a. ONLINE

```
! JOB     OCLC,CP
! RUN     BP,IMG002A E1
! PAU - - MOUNT PRINT TAPE ON 'o81', RING IN
! RUN     BP,IMG002A F1
! RUN     BP,ONCNVT
      }  Library Codes (optional)
! FIN
```

   b. OFFLINE

```
! JOB     OCLC,CP
! PAU    --EAC
! PAU    --SYC
! ATT
! RADEIT
! ALLOT (FILE,D3,SCARD), (FSIZE,1000),
    (FORMAT,C)
! ASS  (F:SIN,CR)
! ASS  (F:SOUT,D3,SCARD)
! POOL1
! SORT  F,80,,,A,12,3,A,W1,W2,A,9,3,
    A,19,1,A,15,4,A,21,1,A,20,1
      }     SELECT CARDS
! PAU     READY CNVT OUTPUT TAPE ON T1
! RUN       BP,CNVT
```

APPENDIX B

OPERATING CHARACTERISTICS

1. CONSOLE MESSAGES
   !!PAU -- MOUNT PRINT TAPE ON '081', RING IN
   RESPONSE: Mount tape as directed

2. PROGRAMMED ABNORMAL COMPLETION - CNVT will terminate abnormally via a

```
     CAL1,9  3
```

instruction under the following conditions. The message printed out to signal the abort is included in each case.

MESSAGE	REASON
'***CNVT CONTROL CARD ERROR-- CHECK DECK'	Three possible reasons 1. Table size is negative 2. Control card read error 3. Exceeded table length
'E3 DATA BASE READ ERROR' '*****PROGRAM ABORTED***'	Unable to read data base record
'E8 UNABLE TO WRITE FORMATTED RECORD'	Unable to write on output tape.
'E18 NODE TABLE ERROR' '*****PROGRAM ABORTED***'	Either less than two branches or more than thirteen branches were found for a single node.
'E19 CONTROL STACK OVERFLOW' '*****PROGRAM ABORTED***'	Control stack has overflow condition.

3. DIAGNOSTICS

MESSAGE	ACTION
E2 SELECT CARD READ ERROR	Card is counted as rejected.
E3 DATA BASE READ ERROR	Program is aborted.
E4 PDT READ ERROR	Select card is rejected
E5 UNABLE TO LINK FIELDS	Select card is rejected
E6 UNLISTED FIELD TAG	Select card is rejected

MESSAGE	ACTION
E7 UNABLE TO BUILD FORMATTED RECORD	Select record rejected
E8 UNABLE TO WRITE FORMATTED RECORD	Program has aborted
W9 EOT ON FMT OUTPUT TAPE	Go to end of file routine
E10 ILLEGAL TAG IN DATA BASE RECORD	Select record rejected
*****PROGRAM ABORTED***	Program has been aborted
W12 OUTPUT FMT TAPE WRITE PROT	A retry is initiated
W14 **CAN'T DO**	Select record missing; counted as rejected
W15 ILLEGAL COLOR CODE	Select record rejected
W16 INVALID 050 INDICATOR	Select record rejected
W17 IMPOSSIBLE TO FORMAT 050 FIELD	-------
VT STAT SUMMARY	
VT NORMAL END	
E18 NODE/TABLE ERROR	Program has aborted
E19 CONTROL STACK OVERFLOW	Program has aborted
** CNVT CONTROL CARD ERROR--CHECK DECK	Check input cards for possible errors

4. PARAMETER CARDS REQUIRED

VIII.6

PARAMETER CARD DESCRIPTION
(ONELINE OR OFFLINE)

the ohio college library center
1314 kinnear rd. - columbus ohio - 43212

Record Layout

File Name CNVT CONTROL CARD INPUT

Record Name CNVT CONTROL CARD

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

File Organization Sequential Record Size 80 Block Size UNBLOCKED

General Description Control cards contain two-digit library codes to control selection of records for card production. Card input to CNVT is optional.

FIELD NAME AND DESCRIPTION	POSITION	FIELD LENGTH	FORMAT
LIBRARY CODE 1	1-2	2	EBCDIC
LIBRARY CODE 2	3-4	2	EBCDIC
.			
.			
.			
LIBRARY CODE n	n-n+1	2	EBCDIC

VIII.7
PARAMETER CARD DESCRIPTION

(OFFLINE)

the ohio college library center
1314 kinnear rd. - columbus ohio - 43212

Record Layout

File Name SELECT CARD INPUT

Record Name SELECT CARD

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

File Organization Sequential Record Size 80 Block Size UNBLOCKED

General Description Select cards are input to CNVT OFFLINE to initiate production of sets of catalog cards by Library of Congress card numbers.

FIELD NAME AND DESCRIPTION	FIELD		
	POSITION	LENGTH	FORMAT
	1-3		BLANK
L.C. CARD NUMBER	4-11	8	EBCDIC
HOLDING LIBRARY CODE	12-14	3	EBCDIC
	15-18		BLANK
COLOR CODE*	'9-19	1	EBCDIC
	20-21		BLANK
WEEK CODE	22-22	1	EBCDIC
	23-25		BLANK
0-8-2 PUNCH DENOTES FIRST VARIABLE LENGTH FIELD	26-26		
+ DENOTES EXTRA CARD COPIES DESIRED			
'b̸' BLANK DENOTES END OF CARD			

SELECT CARD (cont.)

* Each user input card is color keyed as to its function.

Color	Code	Function
Brown	1	Input modifications to LC descriptive cataloging
Orange	2	Input holding statements
Blue	3	Request cards with LC call numbers
Yellow	4	Request cards with local call numbers
Red	5	Request unit card with Dewey class number
Green	6	Request unit card with LC call number
Green Lit	7	Request unit card with alternate LC call number or class number

Beginning in column 27 of the select card are a series of variable length fields. Each field is preceded by a special character which indicates what type of data follows. The fields are delimited by a 0-8-2 punch. The last field is followed by a vertical bar (|) indicating the end of the select card. If more than one card is required, the data may be continued in column 1 of succeeding cards.

0	1	n
FIELD INDICATOR	DATA	0-8-2 PUNCH or | FOR END OF CARD

INDICATOR	TYPE OF FIELD
+	NON-PDT STAMP
11-PUNCH	COPIES
=	USER DATA
*	CALL NUMBER *

* The call number appears on the select card formatted as the user requires it on the finished catalog card. The punctuation desired must be included. A new line indicator (¬) must also be included before the first character of each new line of the call number.

APPENDIX B

5. EXAMPLE OF OUTPUT

VIII.10
OFFLINE CNVT OUTPUT

the ohio college library center
1314 kinnear rd. - columbus ohio - 43212

Record Layout

File Name CNVT PUNCHED OUTPUT

Record Name CARD SELECTED FOR PRODUCTION

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

File Organization SEQUENTIAL Record Size 80 Block Size UNBLOCKED

General Description A card entry of sixteen bytes is punched for each input select card for which catalog cards are produced.

FIELD NAME AND DESCRIPTION	POSITION	FIELD LENGTH	FORMAT
L.C. CARD NUMBER #1	1-11	11	ALPHA-NUMERIC
HOLDING LIBRARY CODE #1	12-14	3	ALPHA
	15-16	2	BLANK
	.		
	.		
	.		
L.C. CARD NUMBER #5	65-75	11	
HOLDING LIBRARY CODE #5	76-78	3	
	79-80	2	BLANK

```
20:04 OCT 21,'71     CCVT RUNTIME LOG

9CLC..155948 BLU CIN SLD    9CLC..162546 BLU CIN SLD    9CLC..159975 BLU CIN SLD    8CLL..127960 BLU BUN SLD
9CLC..154927 BLU CIN SLD    9CLC..156474 BLU CIN SLD    9CLC..159283 BLU CIN SLD    9CLL..158539 BLU CIN SLD
9CLC...99106 BLU CIE SLD    9CLC..155836 BLU CIN SLD    9CLC..155653 BLU CIN SLD    8CLL..159062 BLU BSA SLD
9CLC..106308 BLU 9RE SLD    9CLC...82156 BLU 6RE SLD    9CLC..161858 BLU 9SJ SLD    9CLL..162446 BLU BSA SLD
9CLC..145909 BLU 9U. SLD    9CLC...33578 BLU 9RE SLD    9CLC...59131 BLU 9SE SLD    8CLL..127179 BLU WSU SLD
9CLC..164941 BLU 9UN SLD    9CLC..105473 BLU 9SE SLD    9CLC..161872 BLU 9JN SLD    8CLL...39273 BLU BBE SLD
9CLC..157122 BLU 9SP SLD    9CLC...73507 BLU 9RE SLD    9CLC...74570 BLU BJN SLD    8CLL...66530 BLU BUN SLD
9CLC..123247 BLU 9UN SLD    9CLC..117145 BLU 9SE SLD    9CLC..132976 BLU AKR SLD    8CLL..165910 BLU BUN SLD
9CLC..145911 BLU 9U  SLD    9CLC..151667 BLU 9UN SLD    9CLC...12403 BLU BSE SLD    8CLL..161420 BLU BSE SLD
9CLC..141864 BLU AKS SLD    9CLC..135878 BLU AKR SLD    9CLC..137407 BLU AKR SLD    9CLL..152909 BLU BUN SLD
9CLC..115523 BLU 9TE SLD    9CLC..121142 BLU 9S. SLD    9CLC..154359 BLU CIN SLD    8CLL..139826 BLU AKR SLD
9CLC..140076 BLU CIL SLD    9CLC..159907 BLU 9UN SLD    9CLC..165912 BLU 9SA SLD    8CLL..141634 BLU AKS SLD
9CLC..147536 BLU KSU SLD    9CLC..133520 BLU AKR SLD    9CLC..156141 BLU CIN SLD    8CLL..156693 BLU LIN SLD
9CLC..159924 BLU CIN SLD    9CLC..132190 BLU 9S. SLD    9CLC..156439 BLU CIN SLD    9CLL..156447 BLU LIN SLD
9CLC...74134 BLU CIN SLD    9CLC..127754 BLU KSU SLD    9CLC..145377 BLU AKR SLD    9CLL..159512 BLU LIN SLD
9CLC..107791 BLU 9UN SLD    9CLC..125574 BLU AKS SLD    9CLC...59832 BLU BBE SLD    9CLL...17833 BLU KSU SLD
9CLC...99397 BLU AKR SLD    9CLC..140548 BLU AKR SLD    9CLC..165913 BLU BUN SLD    9CLL...77371 BLU LIU SLD
9CLC...96724 BLU 9 E SLD    9CLC..107559 BLU KSU SLD    9CLC..154651 BLU BJN SLD    8CLL...96944 BLU BSE SLD
9CLC..107215 BLU 9UN SLD    9CLC..165914 BLU AKR SLD    9CLC..145947 BLU BSJ SLD    9CLL..162591 BLU BSA SLD
9CLC..163015 BLU KSU SLD    9CLC..104479 BLU SSS SLD    9CLC..129268 BLU 9UN SLD    9CLL..138449 BLU BSA SLD
9CLC..111362 BLU KSU SLD    9CLC..145174 BLU 9SU SLD    9CLC..112259 BLU ASL SLD    8CLL...96732 BLU KSL SLD
9CLC...99060 BLU 9SU SLD    9CLC..154916 BLU AKR SLD    9CLC..136243 BLU BSJ SLD    8CLL..136739 BLU BUN SLD
9CLC..145917 BLU AKR SLD    9CLC..114675 BLU 9SU SLD    9CLC..154710 BLU BSU SLD    8CLL..112963 BLU KSU SLD
9CLC..145916 BLU AKR SLD    9CLC..157894 BLU 9SJ SLD    9CLC..165919 BLU BJN SLD    9CLL...97467 BLU BSA SLD
9CLC..154076 BLU 9UN SLD    9CLC...95491 BLU 992 SLD    9CLC..165920 BLU AKR SLD    8CLL..148580 BLU BUN SLD
9CLC..109135 BLU 9S  SLD    9CLC..165921 BLU AKR SLD    9CLC..165922 BLU AKR SLD    9CLL..149873 BLU BSE SLD
9CLC..155371 BLU 9UN SLD    9CLC..104130 BLU KSU SLD    9CLC..155821 BLU CSU SLD    9CLL..165923 BLU MIA SLD
9CLC..145320 BLU 9SK SLD    9CLC..165924 BLU AKR SLD    9CLC..165925 BLU AKR SLD    9CLL...51956 BLU BSJ SLD
9CLC..123765 BLU 9U. SLD    9CLC..187239 BLU KSU SLD    9CLC..133205 BLU BJN SLD    8CLL..165926 BLU AKR SLD
9CLC..127735 BLU 9S. SLD    9CLC....1941 BLU KSU SLD    9CLC..151962 BLU 9SA SLD    8CLL..165927 BLU AKR SLD
9CLC..137651 BLU 9SU SLD    9CLC..144592 BLU 9SA SLD    9CLC..152375 BLU 9S8 SLD    9CLL...86507 BLU BSE SLD
9CLC..106723 BLU KSU SLD    9CLC...73678 BLU DEF SLD    8CLC..165928 BLU AKR SLD    8CLL...73990 BLU YNG SLD
9CLC..165929 BLU AKR SLD    9CLC..147745 BLU HS8 SLD    9CLC..147316 BLU 9SA SLD    9CLL..165930 BLU AKR SLD
9CLC..140269 BLU 9UN SLD    9CLC...71232 BLU PCJ SLD    9CLC..165931 BLU AKR SLD    9CLL..162302 BLU KSU SLD
9CLC..145932 BLU AKR SLD    9CLC..155100 BLU CSJ SLD    9CLC...39896 BLU PCJ SLD    8CLL..147167 BLU BS9 SLD
9CLC..165933 BLU AKR SLD    9CLC..120677 BLU 9SA SLD    9CLC...95992 BLU PCJ SLD    9CLL....9186 BLU BSD SLD
9CLC..136328 BLU 9 N SLD    9CLC...30844 BLU 9UN SLD    9CLC..157499 BLU TJL SLD    9CLL..165934 BLU AKR SLD
9CLC...55640 BLU PCJ SLD    9CLC..134821 BLU 9SL SLD    9CLC..138415 BLU 9SA SLD    8CLL...65135 BLU PCJ SLD
9CLC..130753 BLN AIT SLD    9CLC..106963 BLU ASU SLD    9CLC..129438 BLU 9S8 SLD    8CLL..142205 BLU BUN SLD
9CLC...43658 BLU PCJ SLD    9CLC..153971 BLU CSJ SLD    9CLC..152314 BLU 9SU SLD    9CLL...12532 BLU PCJ SLD
9CLC..115405 BLU 9AE SLD    9CLC..156627 BLU 9SX SLD    9CLC..104613 BLU KSU SLD    9CLL...92165 BLU BS! SLD
9CLC..103128 BLU PCJ SLD    9CLC..111524 BLU PCJ SLD    9CLC..152631 BLU KSL SLD    9CLL..117938 BLU KSU SLD
9CLC..114469 BLU 9UN SLD    9CLC..104554 BLU PCJ SLD    9CLC..134564 BLU ASC SLD    8CLL..159466 BLU TOL SLD
9CLC..147397 BLU 9UN SLD    9CLC..159055 BLU 9S8 SLD    9CLC..143120 BLU 9SU SLD    9CLL...93525 BLU PCJ SLD
9CLC..141194 BLU ASC SLD    9CLC..151073 BLU 9SU SLD    9CLC..130120 BLU AKR SLD    8CLL..159089 BLU BUN SLD
9CLC..140355 BLU KSU SLD    9CLC..140236 BLU ASC SLD    9CLC..141946 BLU BSE SLD    9CLL...60324 BLU PCJ SLD
9CLC...54057 BLU ASC SLD    9CLC...93064 BLU 9S/ SLD    9CLC..148596 BLU BJN SLD    8CLL..142548 BLU PCJ SLD
9CLC..165935 BLU 9TC SLD    9CLC..142141 BLU 9S/ SLD    9CLC..134142 BLU AKR SLD    8CLL...99102 BLU BSU SLD
9CLC..161473 BLU 9UN SLD    9CLC....6651 BLU CIN SLD    9CLC..154346 BLU CSU SLD    8CLL..111208 BLU ASC SLD
9CLC..105774 BLU AKR SLD    9CLC..140257 BLU 9SJ SLD    9CLC...92100 BLU ASL SLD    9CLL..140086 BLU AKR SLD
9CLC..152504 BLU 9SJ SLD    9CLC...94397 BLU PCJ SLD    9CLC..132512 BLU BUN SLD    9CLL..116045 BLU PCJ SLD
9CLC..135727 BLU 9S8 SLD    9CLC..118205 BLU AKR SLD    9CLC..157439 BLU KSU SLD    8CLL..141375 BLU BUN SLD
9CLC..127331 BLU AKR SLD    9CLC..135970 BLU 9SE SLD    9CLC...67961 BLU PCJ SLD    9CLL..135890 BLU AKR SLD
6CLC..113924 BLU ASC SLD    9CLC..150468 BLU 9SE SLD    9CLC...52482 BLU 9S8 SLD    8CLC..135336 BLU AKR SLD
```

```
20:04 OCT 21,'71      CNVT STAT SUMMARY
    1110 SELECT CARDS READ
    1110 SELECTED
       0 REJECTED
```

APPENDIX C

DETAILED DESCRIPTION OF INTERNAL SUBROUTINES

1:U001 This routine prints the series field but does not trace it. R3 is set and is stored in the Link Directory - Tags. A branch is taken to 'AEND' which will keep track of the last entry and point to the next entry in the directory.

1:U002 This routine is a special processor of 'TBLA' options for the 600 fields. The byte address and the length of the field are picked up in R4 and R5 from 'LNKBA'. R4 is used to pick up the second indicator of the field. If the indicator is 0 (zero) a branch is taken to 'AEND' to keep the field. If the 2nd indicator is 1 (one), a bit is set in 'LNKTAG' and a branch is taken to 'AEND' to keep the field but enclose it in brackets. If the indicator is not 0 or 1, a branch is taken to 'ADEL' to delete the field from the directory.

1:U003 This routine is a special processor of 'TBLA' options for the 600 fields. The byte address and the length of the field are picked up in R4 and R5 from 'LNKBA'. R4 is used to pick up the second indicator of the field. If the indicator is 0 (zero) or 2 (two) a branch is taken to 'AEND' to keep the field. Otherwise a brance is taken to 'ADEL' to delete the field from the directory.

1:U004 This routine is a special processor of 'TBLA' options for the 600 fields. The byte address and the length of the field are picked up in R4 and R5 from 'LNKBA'. R4 is used to pick up the second indicator of the field. If the indicator is 2 (two) a brance is taken to 'AEND' to save the field. If the indicator is not 2, a branch is taken to 'ADEL' to delete the field from the directory.

3:U000 This is the routine that does the housekeeping prior to formatting the call number. At entry, R3 is the sub-script from the 'LEAF' instruction which indicates which type of formatting to use:

=0	LC, short first line
=1	Dewey, long first line
=2	LC, long first line
=3	LC, special, short first line
=4	Dewey, shrot first line
=5	Medical, short first line

Upon entry to the routine 3:U000, the type of library is checked in the Break-down Control Word (BRKCW -- see page VII.4) If it is 0 (L.C. call number), the 090 field is checked. If present it is used as the call number. If not present, the 050 is checked. If present, it is used as the call number; if not, a unit card is forced. The first line length of the call number is set up. If the length in the BRKCW is zero (0), then CNWIDTH (from the PDT) is used. Then control is transferred to the appropriate parse.

If the library is Dewey (type 1), the 092 field is checked. If present, it is used as the call number. If not present, the 090 is and then the 050 fields are checked. If none of these fields are present, a unit card is produced.

If the library is a medical library (type 2), the 096 field is checked first. If present it is used as the call number. If not present, the 060 is checked. If present, the 060 is used as the call number. If neither the 096 or 060 is present, the 090 and then the 050 are checked. If none of these fields are present, a unit card is produced. For an off-line request card, only the 050 field is used.

After the appropriate call number field has been found, R2 and R3 are loaded with the byte address and the elngth respectively from'LNKBA'. If the 050 field is used and the book is not LC and there is no subfield 'b', a unit card is produced.

R8 is loaded with the address of a two-word parm list called LCCN000P. The first word is the byte address of the beginning of the call number field. The second word is the byte address of a work area in which the parsed call number will be returned. R7 is then used as the linking register to the appropriate entry point in the call number parse routine, LCCN000.

REG1, REG2, and REG3 are then set up to begin formatting the elements of the call number. REG1 is set to 0 to indicate the index into the call number field. REG2 is set with the negative call number width. REG3 is set with the byte address of the call number field (LCCN WRKA)

The first byte of the call number field will be the element number of the first element. If this is not 0 (zero), a brance is taken to RETURN. Otherwise REG4 is loaded with the element number (0) and R7 is used to store a new line character (X'5F') at the beginning of the temporary formatted call number field (090T). R7 is used as the linking register to LCCNUT00 to move element 0 to the 090T field. LOOPSW 1 is set with a 1 (one) if element 0 was present.

3:U009 This is a special 'end and link' routine to be used when the routine 3:U008 (Format Remaining Elements) is not used. It is used specifically when the call number consists solely of an element 0 and all other elements are to be suppressed. This routine closes the 090T (call number field) with an end of field delimiter (X'FD'). The routine also increments the record length (090T-1) by 1 to account for the delimiter. A branch is taken to 4:U000B to store a '4' into the second half-word of FIELD2 (formatted call number field), since the call number is now complete and ready to be moved to FIELD2.

3:U0X1 - 3:U0X7 These routines set up the format control word needed to process each of the first seven elements of the call number. A branch is taken to routine 3:U0XX after the 'FCW' has been set up.

3:U0XX This routine is used to format elements 1 through 7. Processing gets to this routine from the 'EXUTBL' with a "LD,R6" (six) instruction, which will load R6 with the byte address of the 'FCW' (Format Control Word) and will load R7 with the address of this routine. R6 is used to pick up the bits from the 'FCW'. The 8 Control Bits of the FCW are processed in the following sections of this routine, respectively; PH1, PH2, PH3, and PH4.

R8 is set up in this routine. R8 acts as an indicator to determine which type of spacing to use in the UT00 move routine.

3:U008 This routine is used to format all remaining elements, numbers 8-254. If R3, the sub-script of the 'LEAF' instruction, is set = 1, every element is preceded by a blank character in the left margin. R12 is used as a working register to save the value of the sub-script. R2 and R3 are set up with the negative call number width and the byte address of the source, respectively. R7 is used as the linking register to branch to 2 sub-routines within this routine. The sub-routines are 'NLT' and 'NL' which check to see if the element will fit on the previous line, and provide a new line character, respectively. They both use R14 as a working register. R15 is used as a working register to pick up characters and check element numbers. R13 is used to make sure that no element is tried more than 3 times. R7 is used as the linking register to branch to a call number move routine.

3:XQ01 This routine will create an extra non-PDT stamp 'Campus' for two holding libraries for Robert Morris College. The stamp is provided if there is a parameter of '1' provided in READPDT's. If the parameter is zero, the stamp is not provided. R1 is used as a working register to test the parameter which has been previously set up in 'PDTPARMS'. R1, R2, and R3 are loaded with the size, the subfield code, and the stamp. These are then stored at the location 'STAMP'.

3:X002 This routine is a special processor for class 'PZ' books. This routine is used in the tree as the processor for the 7th element (2nd cutter). R15 is set up with the letters 'PZ' and R7 is used as the linking register to OSIU01, the class check routine. R3 (subscript indicator) is set to 1 for class 'PZ'. It will format as 1*A1NNN. R3 is otherwise set to 0 to format as 1*ANN.

3:X003 This routine is used to provide the stamp 'J' before the numeric portion of a Dewey call number. If the characters 'Fic' are in the $a subfield of the 090 field, they are replaced with 'J'. If there is anything else in the $a subfield, it is preceded by 'J'. If nothing is entered to precede the numerics of the call number, a 'J' is placed there. R1 is used as a working register to test the contents of the $a subfield. R4 is used to store the 'J' in the 090T area. R1, R2, and R4 are used to shift the contents over two places to the right if the $a subfield does not contain the characters 'Fic'.

3·X004 This routine is used to decrement the call number width by 1 for Dewey call numbers so that a stamp will go into the left margin. R1 is used as a working register.

3:X005 This routine provides for special cutter breaks. Both cutters are put on one line if they fit; otherwise, the first cutter ends with a '-' (for sub-script = 0) or '.' (for sub-script = 1) and the 2nd cutter begins with a blank on the next line. R4, R6, and R7 are used as working registers.

3:X006 This routine will test the loopswitch indicated by R2. Prior to entering this routine, R2 is set with the first indicator from a 'TEST' instruction. If the loopswitch is not set, a branch is taken to 'RETURN'. If the loopswitch is set, it is incremented by 1.

3:X007 This routine is used to decrement the call number width when there is only one set of oversize parameters, and the symbol is to go in the left margin. R10 is used as a working register to save and restore PDTPARMS. The routine OSIU02 is used to set up R14 and R15 (height and width); R5 is also used as a working register.

3:X008 This routine decrements CNWIDTH (Call number width) by one for special stamps which go in the left margin. Prior to entering this routine, R3 is set with the sub-script indicator from the 'LEAF' instruction. R2 is used to pick up the stamp and R3 (sub-script indicator) is used as an index into a table of stamps. R4 is used as a working register.

3:X009 This routine decrements CNWIDTH (Call number width) by 2 for books larger than 42 cm. so that the oversize symbol can go into the left margin. If the book is not larger than 42 cm., a branch is taken to the routine 3:X007 to decrement the call number width by one for books less than 42 cm. but larger than 26 cm. The routine OSIU02 is used to set up R14 and R15 (height and width). R5 is used as a working register. R10 is used to save PDTPARMS. A branch is taken to 3:X007 to restore PDTPARMS.

3:X010 This routine supplies a special pseudo-'stamp' for Dewey Juvenile books. R3 is set up with the sub-script indicator from the 'LEAF' instruction and is used as an index into a table of routines to determine which stamp to use and where it is to be placed.

R5 is set up with the address of the stamp and R7 is set with the length of the stamp before branching to the move routine 4:UCM1. When the stamp is to go in the left margin, R9 is set up with the address of the stamp and a branch is taken to the move routine OSIU09.

3:X011 This routine will put both cutters on 1 line unless the book is class 'PZ', then the cutters go on separate lines. R15 is set with 'PZ' and R3 is set equal to 0 before branching to the class check routine OSIU01A. R6 and R7 are loaded with the address of the FCW (format control word), and the address of the routine to process the second cutter.

3:X012 This routine will suppress printing of elements 2 and 3 and supply a blank line instead for all books in class 'K' except class 'KF'. R4 is set as a switch before branching to 3:X014 which will suppress printing of elements 2 and 3 for class 'K' books. R2 is used as a working register to move in a blank line (5F) and end of field delimiter (FD).

3:X013 This routine will put the second cutter on a new line, preceded by a decimal point unless the book is class 'PZ', then the second cutter is put on a new line with no decimal point. R15 is set with 'PZ' and R3 is set equal to zero before branching to the class check routine, OSIU01A. R6 and R7 are loaded with the address of the FCW (format control word), and the address of the routine to process the second cutter.

3:X014 This routine suppresses printing of elements 2 and 3 for all books in class K if the first position of element 2 is zero (0). R15 is set with the letter 'K' and R3 is set to zero before branching via R7 to 3:X100 to initialize 090T-1 (size), then to 0$IU01 to check the class. If the class is not 'K' a branch is made to RETURN to continue processing the call number. If the class is 'K', R8 is used as a working register to find the first position of element 2. If the first character is zero, the fourth element is located and REG3 is set up from R3 as the byte address of the source (i.e. elements 2 and 3 are suppressed). If the first position of element 2 is not zero, a branch is taken to RETURN without suppressing elements 2 and 3. R14 is used as a linking register within the routine.

3:X015 This routine is used to suppress printing of the $a subfield of the 092 field when the subfield contains one of two different three-letter entries. The entries to be suppressed must be entered consecutively in the table 3:X017T. The index to the first entry is passed to this routine in R3 where R3 is set up with the subscript from the LEAF instruction. R3 is then incremented by 1 to obtain the index for the second entry. A branch and link on R7 is then taken to the routine 3:X017 to do the actual checking and suppression of the entry.

3:X100 This routine is used to initialize 090T-1 (size) before branching to the class check routine, OSIU01. The check must go through this routine first if the check occurs before the first 7 elements have been formatted. R8 is used as a working register to set up 090T-1. 3:X101 is used to set up a branch to OSIU01A which will check 2 class letters. 3:X100 is used to set up a branch to OSIU01 which will check 1 class letter. R15 and R3 must be set up prior to coming through this routine. R15 contains the 1 or 2 class letters or numbers to be checked and 3 is set equal to 0.

3:X016 This routine will cause the printing of the call number to begin one line lower. This is accomplished by moving in the 4 characters -- $a, sort skip, and a blank in front of the call number. R14 and R15 are used as working registers to pick up the byte address and the length of FIELD2, the formatted call number field. R2, R3, and R4 are used as working registers to move the 4 characters to FIELD2. R2 is used as an index into FIELD2, and R2 also has the length of the call number which is stored into the second half-word of FIELD2 after the characters have been moved. R3 is used to store an end of field delimiter (FD) at the end of the call number field.

3:X017 This routine is used for a Superintendent of Documents collection. The characters 'Doc' are entered in the $a subfield of the 090 field. The printing of this subfield is suppressed with this routine. R4 is used to test for the characters 'Doc'. R1 is used to change the index into the 090T area to zero. This will effectively cause the remainder of the call number to be printed over the characters 'Doc'.

3:X018 This routine will suppress elements two and three for all books. R3 is set from REC3 and contains the byte address of the source. R14 links to 3:X014B and returns with the byte address of the fourth element in R3.

3:X019 This routine will create an extra stamp '(LC)' for all books if the sub-script =1 (R3 = subscript from the 'LEAF' instruction), or for all books except those in classes 'Q', 'D', or 'P' (not including 'PA, 'PB, etc.) if the subscript = 0. R15 is loaded with the class letter to be checked, and then a branch and link on R7 is taken to OSIU01 which does the class check. R14 is a switch for Yale to be set if R3=0 and a class 'P' is being checked. If the stamp is to be provided, R3 is set to act as an index into the table 3:X0019T. R4 is loaded with the address of the stamp using R3 as an index. The stamp is then loaded into R1, R2, and R3 and then stored into the field 'STAMP2' to be treated as a Non-PDT stamp #3.

4:U000 This routine is used to set up for formatting off-line yellow request cards and any on-line record that uses the free-text call number field (099).

For the on-line version, R4 and R5 pick up the byte address and the elngth of the 099 field from LNKBA. R4 is then incremented to point to the first byte of the call number and R7 is loaded with the length of the call number.

For the off-line version, the call number has already been moved to the location TEXT. If TEXT is longer than 160 bytes an end of field delimiter (X'FD') is moved to the 090T field (temporary formatted call number field). R4 is loaded with the byte address of the first byte of the call number and R7 is loaded with the length of the call number.

For both versions, R5 is loaded with the byte address of the 090T field and the length of the call number is stored at 090T-I. The call number is then moved to the 090T field and an End of Field delimiter (X'FD') is stored at the end of the field. a '4' is moved into the second half-word of FIELD2 (formatted call number field) so that the call number will be moved to FIELD2 beginning at the fourth byte.

4:U001 This routine is used to process the PDT or holding library stamp. R2 is loaded with the length of the call number as it is in FIELD2 at this point. R2 is then used as an index into FIELD2 to move in a subfield code (X'FC97' -- $P). An end of field delimiter (X'FD') is then moved in after the subfield code. The adjusted length of the call number (R2+3) is then stored in the second half-word of FIELD2.

When the program CCFP processes this record, it will pick up the subfield code and replace it with the holding library stamp as requested by the institution.

4:U002 This routine is used to process the non-PDT stamp #1 ($b). R3 is set prior to entering this routine with the sub-script of the 'LEAF' construction. The current possible values of R3 are as follows:

1 Supply a blank line before the stamp
2 Put the stamp in the left margin
3 If the stamp is 'j' move it into the left margin
4 If the stamp is 'j', move it in front of the first cutter.
5 If the stamp is 'j', move it in front of the element 2
6 Supply a blank line after the stamp

The stamp is picked up from the 049 field and moved to the temporary location STAMP. R7 is used in this routine to test for the presence of the stamp. R7 is then set with the length of the stamp and R5 is set with the byte address of the stamp. If no special handling is to be performed (R3 = 0), a branch is taken to the move routine, 4:UCM1.

If R3 = 1, R13 is loaded with a X'82' (lower case 'b') and R12 is used as a linking register to 4:U0012. 4:U0012 will move ~~a subfield delimiter~~ (X'FC') and the lower case 'b' followed by an end of field delimiter (X'FD') into FIELD2 (formatted call number field). This will provide a blank line. Control is then passed to the routine 4:UCM1 to move the stamp into FIELD2.

If R3 = 2, R9 is loaded with the stamp (assuming the stamp is only one character in length)and R7 is used to link to the routine OSIU09 where the contents of R9 will be moved into the left margin.

If R3 is not 0, 1, 2, or 6 and the stamp is not 'j', a direct brance is taken to 4:UCM1 to move the stamp into FIELD2, and Control is returned to this routine. A direct brance is then taken to 3:X016 where a subfield code (X'FC81') and an end of field delimiter (X'FD') are moved to FIELD2. This action effectively produces a blank line after the stamp.

If the stamp is 'j' and R3 = 4 or 5, R15 is set with the element number after which the stamp is to appear. R14 is set with a 'F' to indicate to the move routine that this is a stamp and not an oversize symbol. A branch is then made to OSIU06 to move the stamp.

4:U003 This routine sets up the oversize routine. R5 is used to pick up the index to the oversize routine from PDT set-up.

4:U004 This routine sets up R5 and R7 before branching to the move routine to move in the formatted call number. R7 has the length of the 090T area and R5 has the byte address of 090T.

4:U005 This routine is used to process the non-PDT stamp #3 ($c). R7 is used to test for the presence of the stamp. Prior to entering this routine, R3 is set with the sub-script of the 'LEAF' instruction. If it is not a 1 (one) a branch is taken to the move routine (4:UCM1). If the sub-script is a 1, R13 is loaded with the subfield code (c) and a branch is taken on R12 to 4:U001 which will effectively provide a blank line before the stamp.

4:U999 This routine is used to end and link FIELD2 (formatted call number field) to the rest of the record. R4 and R5 are loaded with the byte address and length of FIELD2, respectively. An end of field delimiter is moved at the end of FIELD2. The byte address and the adjusted length of FIELD2 are then re-set in the first word of FIELD2. The adjusted length is also stored into the second word of the parameter list FLDPARM2. R2 is then loaded with the address of FLDPARM2 and R3 is loaded with the address of the field that is to immediately precede this new field. In the case of the call number, this field is the last field before the 100 field and the address of this field is found at location XLF100. Control is then passed to the routine X:U999 to do the actual linking.

4:UCM1 This routine is used to move the formatted call number into FIELD2. An alternate entry point to this routine is 4:UCM1E which will return control back to the calling routine after the move is performed. The calling sequence is: BAL,R3 4:UCM1E. Prior to entering this routine, R5 and R7 are set up with the byte address and the length of the field to be moved, respectively. The second half-word of FIELD2 has the length of the call number so far. This value is loaded into R4 which is then used as an index register to FIELD2. If the field does not begin with a subfield code, a subfield "a' (X'FC81') is stored at the beginning of the field. For every new line indicator in the field (X'5F' or X'FB'), it is replaced with a subfield "a" (X'FC81'). The field is then moved to FIELD2 using R1, R4, R5, and R7 until an end of field delimiter (X'FD') is found. The byte address and the length of FIELD2 are adjusted in the first and second half-words respectively.

4:X001 This routine will change the PDT# from that of 'CIN' to that of 'CII' if the book is in class 'M'. R14 is used as a working register to pick up the PDT# from FMTDATA and check for 'CIN'. R15 is set with class letter 'M' before branching on R7 to the class check routine, OSIU01. R14 is used as a working register to change the PDT# and store it back into FMTDATA.

4:X002 This routine checks for class 'QA' and creates one extra card for those books. R15 is set with two class letters 'QA' before branching on R7 to the class check routine, OSIU01A. 'FMTDATA' is incremented by one to force an extra card, if the book is in class 'QA'.

4:X003 This routine causes two extra cards to be created for books in class 'QD'. R15 is set with the two class letters 'QD' before branching on R7 to the class check routine, OSIU01A. 'FMTDATA' is incremented by two to force two extra cards.

4:X004 This routine causes one extra card to be created for books in class 'K' or class 'JX'. R15 is first set with class letter 'K' and then 'JX' before branching on R7 to the class check routines OSIU01 and OSIU01A. 'FMTDATA' is incremented by one to force an extra card.

4:X005 This routine provides a pseudo-stamp 'J' above the call number. R5 has the byte address of the stamp 'J' and R7 has the length of the stamp (one). The routine then branches to the move routine 4:UCM1.

4:X006 This routine creates extra cards for MIAMI. Two extra cards are created for books with stamp 'REF.H' or stamp 'B'. No extra cards for books with any other stamp. Two extra cards are created for books in class 'A', 'B', 'N', 'M', 'P', or 'Z', or Dewey class 'F' or '800'. R7 is used as the working register to check for the presence of $b stamp. R3 (sub-script on 'LEAF' instruction) is set up prior to coming to this routine. R3 = 1 for Dewey books and R3 = 0 for LC books. R2 is used as a working register to check for 'REF H' or 'B' stamp. A table is set up with the list of class letters to be checked, and R5 acts as an incrementing index through the table. R15 is set with the class letter from the table before branching on R7 to the class check routine OSIU01. 'FMTDATA' is incremented by 2 to force extra cards when needed.

4:X007 This routine is a special processor for Pittsburgh. If the book is in the main holding library (PIT) and it is in class 'N' the PDT# is changed to '396' -- holding library (PIR). If it is in the main holding library and class 'M' the PDT# is changed to '383' -- Music holding library (PIK). The PDT stamp is deleted in both cases. R14 is used as a working register to check for main holding library; to change the PDT# in 'FMTDATA'; to set Loopswitch 2 to delete processing of PDT stamp ($p); and to cancel out the last library processed code since the library code has been changed. R15 is set first with 'M' then with 'N' before branching to the class check routine, OSIU01.

4:X008 This routine will automatically cause two extra cards to be created for any book with a non-PDT stamp. R7 is used as a working register to test for the presence of either stamp. If there is a stamp present, 'FMTDATA' is incremented to create two extra cards.

4:X009 This routine will automatically cause one extra card to be created for all books in class 'L' with a date of publication of 1972 or later. R15 is loaded with the class letter 'L' and R7 is used as the linking register to OSIU01, the class check routine. If it is in class 'L', the first half-word of the 5th word of the record is checked to see if the date of publication is 1972 or later. R4 and R15 are used as working registers to do this. The first entry in the link directory is the byte address of the record. This is used to get the date of publication.

5:U000 This routine will set the unit card flag. R2 and R3 are used to set the flag and to zero the number of extra cards. Since the call number is printed in the user data field for unit cards, the X910 field (user data field) is deleted from the directory. R7 and R8 are used to delete the entry by branching to the program 'LINKDLT'. R2 is used to zero the address to the X910.

4:X010

This routine will set the second loopswitch which will effectively cause elements 1-7 of the call number to be suppressed whenever the $a subfield of the 092 field is 'Fiction'. The first two words of the call number are loaded into R4 and R5 and compared with 'Fiction'. If they are equal, LOOPSW +2 is turned on.

5:U001 This routine sets up the registers to put the Dewey class number into the user data area if it is present; otherwise put the LC number there instead. R1 is used to check for the presence of the Dewey class number. R1 is also used to store an apostorophe character in REG15. R1 is used as an index into the link directory table to pick up the byte address and the length of the 082 field into R2 and R3 respectively. R2 is used to point past the indicators of the 082 field. REG5 then has the byte address of the 1st byte past the indicators of the 082 field.

5:U002 This routine will move the user data into 'FIELD3' which is the formatted user data field. Prior to branching to this routine, R5 and R15 are set up with the byte address of the user data and the character which is to replace all slash characters, respectively. R4 is used as an index to FIELD3 beginning at the second word. R4 will also have the length of the user data (in bytes) +4. This value is stored in the second half-word of 'FIELD3'. R1 is used as a working register to pick up and test for a slash character, in which case, the slash is replaced by the contents of R15. R7 is set with the size of the 910 field and is used as a decrementing register to fill up FIELD3. The characters are moved to FIELD3 in a 'load-byte; store-byte' fashion with R5 as the index to the data and R4 as the index to FIELD3.

5:U003 This routine sets up the registers to put the LC card number in the user data area. The X050 field (LC call number) is loaded into R1. R2 and R3 are used as working registers to pick up the byte address and the length of the 050 field from the link directory. REG5 is loaded with the byte address of the 1st byte of the 050 field past the indicator. R1 is used as a working register to store a slash character in REG15. If there is no LC number present (X050 = 0), R1 is used to move '40FE' --"Blank, end of record' -- into REG5.

5:U004 This routine sets up the registers to put the alternate LC card number in the user data area for "green lit" cards. R1 is used as a working register to check the presence of an 050 field. R1 is also used as an index to the link directory to pick off the bytc address and the length of the 050 into R2 and R3. R1 and R2 are then used to test the second indicator which tells whether or not it is a green lit card. R3 and R4 are loaded with a subfield delimiter character and a lower case 'a', respectively. R2 is used as the index to the 050. R2, R3, and R4 are used to find the $a subfield. REG5 is loaded with the byte address of the alternate LC card number. R1 is used to store a slash character in REG15. The character in REG15 is used to replace all slashes in the call number field.

5:U005 This routine sets up the registers to put the Dewey card number in the user data area. R1 is used to check for the presence of the 082 (Dewey class number) field. If the field is not present, a branch is taken (5:U003C) to store the byte address of the Dewey card number in REG5, and to store a slash character in REG15.

5:U010 This routine sets up the registers to provide the printing date in the user data area. R1 is used as a working register to store a slash character in REG15. REG15 contains the character which will replace all slashes in the card number. R1, R2, and R3 are used to pick off the printing data from MSG0. R1 is used to store an end of field indicator after the date in the 910T area. R1 is used to store the byte address of the printing date in REG5.

5:U011 This routine provides the cataloging source in the user data area. R1 is used to pick up the address of the cataloging source. For the on-line system, the source is in 'RECDBUF', the 8-word record leader. For the off-line system, the source is found from 'DBBA', the byte address of the record. If the source is one of the three standard: Library of Congress, National Agriculture Library, or National Library of Medicine, its abbreviated format is moved immediately into FIELD3 (user data field). If it is not one of the standard forms, it must be reduced to 22 characters.

5:U011 (continued)
R1 - R5 are used to pick up the source and move it to 090T (user data temp area), deleting all decimals and commas. If the field is less than 22 characters, it is moved to FIELD3. If not, each word is then checked against a table called 'STOPLIST'. This is an alphabetical list of words and abbreviations. If the word is in this list, it is replaced with its substitute. The word is then checked for a vowel group. In which case, all vowels in each group except for the first vowel in the group are suppressed. If the cataloging source is still longer than 22 characters, any word with more than 6 characters and with more than 1 vowel group is truncated until the data is less than 22 characters. At which point the contents of 910T are moved to FIELD3. R1 - R9 are used in this substitution and truncation process.

5:U012 This routine sets up the register to put the year of the print run, plus the cataloging source, if it is LC; otherwise put nothing into the user data area. R1 is used as a working register to store a slash character in REG15, and R1 is used to store the byte address of the 910T area (temporary user data field) into REG5. R1 is used to pick up the byte address of the record, add cataloging source displacement (29) and pick off the first byte in R2. R2 is used to test to see if the cataloging source is LC. R1 - R5 are used to pick up the abbreviation for 'Library of Congress' and move it to the 910T area. R2 and R3 are used to pick up the year of the print run from MSG0 and move it to the 910T area. R1 is used to store an end of field indicator (X'FD') in the 910T area.

5:U013 This routine will provide the date of the print run plus the catalogers indentification in the user data area for the on-line system only. The routine 5:U010 is used to provide the date of the print run. R4 is used to pick up the catalogers initials from 'RECDBUF', the 8-word record leader, and store it in the 910T (user data field) after the printing date. R1 is used to store an end of field indicator in the 910T area.

5:U014 This routine is used to provide the date of the print run and the OCLC # in the user data area. The OCLC# can only be retrived if CNVT is run on-line. In order for this routine to work, the routine 5:U010 must be in the tree before this routine. The address of the OCLC# is picked up from the record leader and stored into the first word of the parm list labeled OCLC#FPT. The address of the parm list is loaded into R8. A branch and link on R7 is taken to the external subroutine CBIEB to convert the OCLC# from Binary to EBCDIC format, and to move the converted number to the user data temporary field (910t). R1 and R2 are used to store an end of field delimiter (X'FD') at the end of the 910T field.

5:U015 This routine will provide in the user data area the date of the print run plus a two-word entry from the table 5:U015T. The routine 5:U010 (which must precede this routine in the TREE) provides the date. R3 has the subscript from the LEAF instruction which serves as an index into 5:U015T. R4 and R5 are used to move the data to the temporary user data field (910T). R1 is used to store an end of field indicator (X'FD') at the end of the 910T area.

6:U001 This routine will provide a bracketed blank line between the main entry and the title field whenever there is no 240 field. The address of the 240 field is retrived from X240. If the field is present, a branch is taken to RETURN. XLT245 has the address of the last entry in the directory before the 245-field. This is loaded in R3. R2 is loaded with the address of FLDPARM4. This is a three-word parm list. The first word is the byte address of FIELD4 which is a field of 40 blanks preceeded by a '1'. The second word is the byte length of the field. The third word is the tag (240) in hexadecimal format. A branch is then taken to X:U999 to input the blank field and link it to the other fields. The LEAF for this routine is entered in the tree after 4:U999 which links the 090 field.

5:U999 This routine will end and link the 910 field (user data). "FIELD3" is set up with the user data prior to coming to this routine. The second half-word of FIELD3 has the length of the data field +4; the rest of FIELD3 has the user data. The second half-word of R5 is set up with the length. R1 is used as a working register to store X'FD' at the end of the field. R5 is incremented to account for the :EOF and the value is stored in FLDPARM3 +1 and is also stored into the first word of FIELD3. R2 is set with the byte address of FIELD3 and R3 is set with the last entry in the directory which has been processed. A branch is taken to X:U999 to end and link the 090 field.

X:U000 This is the routine which causes the formatted cards to be produced. For the on-line system, R8 is loaded with the parameter list to build the formatted record and a branch is taken to the program 'FMTREC'. Upon return from formatting the record, R1 is loaded with the address of 'WORKAREA' and R8 is loaded with the address of the parameters for writing the record onto another Tape. A branch is taken to 'TAPEIO' to read the records and write them onto another tape. The completion status of the tape is checked after it has been created. For the off-line system, the completion status of the type is checked first. R8 is loaded with the parameter list to build the formatted record and a branch is taken to 'FMTREC'. Upon return from formatting the record, R1 and R8 are set up as for the on-line system. A branch is taken to 'TAPEIO' to read the tape and write out the records.

X:U001 This routine will log the record as being selected. The procedure "WRTSELD' will print the message that this record has been selected. The procedure 'PUNCHSLD' will punch out the select card information. This procedure being for the off-line system only. In either case the total number of select cards read is incremented by one.

VIII.26A

X:U999 This routine is used to link another field (Generally the 090) to the rest of the record. Upon entering this routine, R3 must have the address of the field that is to immediately precede the new unlinked field. R2 must have the byte address of a 3-word parm list. The first word is the byte address cf the field to be linked. The second word is the length of the field. The third word is the tag in hexadecimal format. R8 is loaded with the address of the parm list for the link and a branch and link on R7 is taken to the external subroutine LINKINST.

MAKEGRN- This routine is used when it has been determined that a unit card is to be produced for a record. A unit card is a single card with no call number. The usual cause of a unit card is an illegal call number. This routine changes the color code for the record to '6' (green) which causes CNVT to produce a unit card. The call number field is printed in the user data area for a unit card. This routine finds the call number field for this record and changes it to an 050 field if it is not already an 050. This change is performed because the user data routine expects the call number to be in the 050 field.

LCCNUA00,-M1 This routine is a special routine for Oberlin to move the call number to the 090T (temporary call number field). It moves a maximum of 7 characters per line with a decimal point after every 3 characters, and lines are broken only at a decimal point. LCCNUAM1 increments R3 (the byte address of the source), and R5 is set to the element length -2 to skip past the element number and the blank. LCCNUA00 is used if the element is to be preceded by a blank. LCCNUA00 does not increment R3 and sets R5 with the element length -1 to skip past the element number but include the blank. R7 is used as the link register to this routine. R4 is used to pick off characters from the source. R6 is loaded with a '4' (= 3 characters plus 1 decimal point to move per group). R1 is the index to the 090T and also has the number of characters that have already been moved. R2 has the negative call number width. R9 is used as a working register to move in 'new line' 'decimal' (5F4B).

LCCNUG00,-M1 This routine is a special routine for Hebrew Union to move the call number to the 090T area (temporary call number field). It moves 6 digits per line, with a space after 'I' and 'O' if they begin a cutter, and a decimal after the first numeric in the cutter. LCCNUGM1 increments the index to the source (R3) and loads R4 with the length of the element -2 to skip past the element number and the blank. LCCNUG00 is used if the element is to be preceded by a blank. LCCNUG00 loads R4 with the length of the element -1 to skip past the element number and include the blank. R2 is set with the negative call number width which will effectively indicate how many more characters will fit on one line. R6 is used as a linking register to pick off and store the characters. R5 is used as the index to a temporary storage area. R8 and R9 are used as working registers. Two routines within the routine are 'IOT' and 'NL' which test for 'I' or 'O' and 'new line'.

. LCCNUT00,-M1 This routine moves one element of the call number into the 090T area (call number temporary field). LCCNUT00 is used if the element is to be preceded by a blank. R5 is loaded with the (element length -1) because the first byte of the element is the element number. All other elements use the routine LCCNUTM1. The address of the source (R3) is incremented by 1 to point past the element number and the blank. The element length is decremented by 2 to account for these 2 characters and the resulting length is loaded into R5. R8 is set up in 3:U0XX as a switch to indicate which type of spacing to use. If R8 =

1	new line after alpha
2	space after 'I' if not followed by a numeric
24	space before 'I' and 'O'
32	space after alpha
63	supply decimal if element is in $a
192	supply decimal if element is in $a, newline if it is in $b
30	space before 'I' and 'O' and space after 'I' and 'O' if followed by a numeric
6	space after 'I' and 'O'

R3 is set equal to the byte address of the source, and R4 is used as a working register to pick off characters from the source. R6 (six) is used as a linking register within the routine. R2 is set equal to the complement of the call number width. R9 is used as a working register. R1 is used as an index into the 090T area and is also used to indicate the number of characters already moved. R7 is the linking register to this routine. If an element over-flows the call number width and the next character is not a comma (indicator to start a new line), a return is made back to 3:U00X ('X' being the element number) to try moving the element again.

OSIU01 This routine will check one (OSIU01) or two (OSIU01A) characters of the class number. R15 is set with the 1 or 2 class letters prior to entering this routine. R6 is set with the number of characters to be checked for (1 or 2). R1 is set with the size of the 090T (temporary call number area). R7 is the linking register to this routine. R1 is used as an index to the 090T to pick off the class number into R2. The first byte of the class number is found, skipping over subfield codes and new line indicators. R2 is compared to R15 and a return to the calling routine is taken one (not equal) or zero (equal) instructions past R7.

OSIU02 This routine sets up R14 and R15 (height and width) used in checking for oversize. R7 is the linking register to this routine. R5 is used as a linking register within the routine. R4 is used as a working register to check for the presence of the 300 field (identification block). R4 and R5 are used to pick up the byte address and the length of the 300 field from the Link Directory. R11, R12, and R13 are used as working registers and are set with X'FD', X'FC', and X'83' respectively. R4 is used to pick up the bytes of the 300 field and find the $c subfield (height and width). R11 is used to convert the height and width from EDCDIC to binary.

OSIU03 This routine will check for oversize for two sets of parameters. R14 and R15 are already set up with the height and width of the book. R5 is used as an index into the table of oversize parameters for this institution. R9 has the address of the symbol used for the larger oversize books. If the book does not meet this criterion, R9 is loaded with the address in R8 which is the symbol for the smaller oversize books. A branch is then taken to OSIU04 to either move in the symbol or to check the next set of parameters.

OSIU04 This routine will check one set of oversize parameters. R14 and R15 are already set up with the height and width of the book. R9 is already set up with the byte address of the oversize symbol. R5 is used as an index into the table of oversize parameters for this institution. If the book is oversize, the size of the symbol is picked up from R9 and loaded into R7. R5 points to the first byte of the symbol. A branch is taken to 4:UCM1 to move the symbol into the call number field.

OSIU05 This routine is used to check one set of oversize parameters when the symbol is to go somewhere other than above or below the call number. R14 and R15 are already set up with the height and width of the book. R5 is used as an index into the parameter table used by this institution. If the book is oversize, a branch is made back to the oversize routine. R7 is used as the linking register.

OSIU06 This routine will put the oversize symbol in front of the element indicated by R15. R9 is already set up with the byte address of the oversize symbol. R1 is loaded with the length of the 090T field (formatted call number field). R2 is loaded with the byte address of the call number field. R3 is used to pick up characters from the call number and check for alphas and numerics until the proper element is found (indicated by R15). R7 is used as a linking register to two routines -- OSIU07 and OSIU08. OSIU07 will adjust R2 to point past new line and subfield indicators and past blanks or decimals. OSIU08 will adjust R2 to point to the first new line, subfield, or end of field indicator it encounters. Once the element has been found, R3 is loaded with the byte address of a temporary storage area called 'TEMP'. R4 is used as a working register to move the remainder of the call number into TEMP. R4 is used to store an end of field character at the end of TEMP. The oversize symbol is taken from R9 and moved into the call number field. Then the remainder of the call number is taken from TEMP and stored back into the call number field after the symbol, using R1 - R4. R4 is used to store an end of field character at the end of the call number. R2 is used to store the revised length (including the symbol) into 090T-1.

OSIU07 This routine will adjust R2 to point to the first character past all new line and subfield indicators, and past blanks or decimals. R2 is already set prior to entering this routine with a byte address. R3 is used to pick up bytes from R2 and to check for new line and subfield indicators and blanks or decimals. R1 is already set prior to entering this routine with the remaining length of the field. The linking register to this routine is R7.

OSIU08 This routine will adjust R2 to point to the first new line or subfield indicator or the first end of field character it encounters. R2 is already set up with a byte address from which the search is to begin and R1 is already set up with the remaining length of the field. R3 is used to pick up the bytes one-by-one from R2 and test for the characters. R7 is the linking register to this routine.

OSIU09 This routine is used to move the oversize symbol into the left margin and shift every other element of the call number over 1 space. R7 is used as the linking register to this routine but the address in R7 is immediately loaded into R6 so that the return to the calling routine is made through R6. R7 is used as the linking register within the routine. Prior to entering this routine, R9 contains the byte address of the oversize symbol.

If the symbol is to go in the left margin in front of the first cutter, R14 and R15 must be set up prior to entering the routine. R14 will have a '1' if the first cutter is to identified as the first element that begins with a decimal followed by at least one alpha. R14 will have a '2' if the first cutter is to be identified as the second call number element that is one alpha followed by at least one numeric. In either case, R15 will contain the symbol (in hexadecimal format) that is to go in front of the first cutter.

One subroutine is used outside this routine. OSIU08 is used to position R2 (byte address of the call number field) at the first new line, subfield, or end of field indicator.

R1 is loaded with the length of the 090T area (temporary formatted call number field.) R2 is loaded with the byte address of the 090T area. R3 is used to pick up the characters pointed to by R2. R9 is used to put "sort skip" characters (X'70') around the oversize symbol if they are not already there. The 'sort skip' character is used so that a call number sort will not include the oversize symbol. R4, R5, and R8 are used as working registers to shift the entire call number over 1 byte to the right. R9 is then loaded with a blank character (X'40') to be moved into the left margin in front of every other element. The length of the call number (090T-1) is incremented to account for the extra blanks and the stamp.

If R14=2, the second loopswitch is used to determine when the second alpha-numeric element is encountered. The switch is set after the first element is found.

OSIU10 This routine is used to check three sets of oversize parameters. R14 and R15 are already set up with the heignt and width of the book. R5 is us d as an index into the parame-er table used by this institution. R9 has the addres. of the symbol used for the largest oversize books. If the book does not fit this criterion, R9 is loaded with the address in R6, the symbol used for the oversize books meeting the middle criterion. A branch is taken to OSIU03 to check the next smaller set of parameters. If the book is oversize, a branch is taken to OSIU04 to move the symbol into the call number field.

OSIU11 This routine is used to check two sets of oversize parameters when the symbol is to go somewhere other than above or below the call number. R14 and R15 are already set up with the height and width of the book. R5 is used as an index into the table of oversize parameters used by this institution. If the book is oversize, a branch is made back to the oversize routine. R9 has the address of the symbol. If the book is not oversize, R9 is loaded with the address in R8, the symbol used for the smaller oversize books. A branch is taken to OSIU05 to check the smaller set of parameters.

OSIU12 This routine is used to check three sets of oversize parameters when the symbol is to go somewhere other than above or below the call number. R14 and R15 are already set up with the height and width of the book. R5 is used as an index into the table of parameters used by this institution. The address of the oversize symbol is in R9. If the book is not larger than the largest set of parameters, a branch is taken to OSIU11 to check the next smaller set of parameters. If the book does meet this criterion, a branch and link is taken to OSIU09 to move the symbol into the left margin. Upon return, R9 is loaded with a lower case 'F' and another branch and link is taken to OSIU09 to move this also into the left margin.

OSIR001 This routine will provide oversize symbols for all oversize books except for those in class 'Z'. 'fo' is placed above the call number for books larger than the largest parameter, and 'f' is used for books larger than the smaller parameter. R15 is loaded with class letter 'Z' and a branch is taken on R7 to OSIU01, the class check routine. R8 and R9 are loaded with the byte address of 'f' and 'fo' respectively and a branch is taken to OSIU03 to see if the book is oversize.

OSIR002 This routine will place 'f' in the left margin for all oversize books. A branch is taken on R7 to OSIU02 to set up the height and width of the book. R9 is loaded with the byte address of the stamp and a branch is taken to OSIU04 to see if the book is oversize.

OSIR003 This routine will provide 'f' in front of the first cutter for all oversize books. R7 is used as the linking register to OSIU02 and OSIU05 to set up the height and width of the book; and to check if the book is oversize. If the book is oversize, R9 is loaded with the byte address of the stamp. R15 is loaded with a '4' to place the stamp after the fourth element. A branch is taken to OSIU06 to move in the symbol.

OSIR004 This routine will provide the symbol 'folio' for the larger books and 'quar' for books larger than the smaller set of oversize parameters. R7 is used as the linking register to branch to OSIU02 to set up the height and width of the book. R8 and R9 are loaded with the byte address of 'quar' and 'folio', respectively. A branch is taken to OSIU03 to check for oversize.

OSIR005 This routine will provide the symbol 'FOLIO' for the larger books and 'QUAR' for the books larger than the smallest parameter. R7 is used as the linking register to OSIU02 to set up the height and width of the book. R8 and R9 are loaded with the byte address of the 'QUAR' stamp and the 'FOLIO' stamp, respectively. A branch is taken to OSIU03 to check for oversize.

OSIR006 This routine will provide the stamp 'f' for the larger books and 'q' for books larger than the smallest oversize parameter. R7 is used as the linking register to OSIU02 to set up the height and width of the books. R8 and R9 are loaded with the byte address of the stamp 'q' and the stamp 'f', respectively. A branch is taken to OSIU03 to check for oversize.

OSIR007 This routine will provide the symbol 'f' for all oversize books except for those in class 'M'. R15 is loaded with the class letter 'M' and a branch is taken on R7 to OSIU01, the class check routine. R7 is used as the linking register to OSIU02 to set up the height and width of the book. R9 is loaded with the byte address of the symbol 'f', and a branch is taken to OSIU04 to check for oversize.

OSIR008 This routine will provide the symbol 'g' for the larger books and 'f' for books larger than the smallest oversize parameter. R7 is used as the linking register to OSIU02 to set up the height and width of the book. R8 and R9 are loaded with the byte address of the symbol 'f' and the symbol 'g', respectively. A branch is taken to OSIU03 to check for oversize.

OSIR009 This routine will provide the symbol 'Folio' for all oversize books. Books in class 'N' have a different set of oversize parameters. R15 is set with the class letter 'N' and a branch is taken on R7 to OSIU01, the class check routine. R7 is used as a linking register to OSIU02 to set up the height and width of the book. R9 is loaded with the byte address of the symbol 'Folio' and a branch is taken to OSIU04 to check for oversize. R5 is used as a working register to change the oversize parameters for books in class 'N'.

OSIR010 This routine will provide the symbol 'Q' for all oversize books except for those books which have any non-PDT stamp #1. R7 is used as a working register to test for the presence of the stamp. R7 is used as a linking register to OSIU02 to set up the height and width of the book. R9 is loaded with the byte address of the symbol 'Q' and a branch is taken to OSIU04 to check for oversize.

OSIR011 This routine will provide the symbol 'XX' for all oversize books except for those books which have the non-PDT stamp 'ATLAS' or 'DISC'. R7 is used as a working register to test for the presence of a $b stamp. R2 is used as a working register to test for the stamp 'ATLAS' or 'DISC'. R7 is used as the linking register to OSIU02 to set up the height and width of the book. R9 is loaded with the byte address of the symbol 'XX' and a branch is taken to OSIU04 to check for oversize.

OSIR012 This routine will provide the symbol 'F' for the larger books and 'Q' for books larger than the smallest oversize parameter unless the book has the non-PDT stamp 'REF'. R7 is used as a working register to test for the presence of a $b stamp. R2 is used as a working register to test for the stamp 'REF'. R7 is used as a linking register to OSIU02 to set up the height and width of the book. R8 and R9 are loaded with the byte address of the symbol 'Q' and of the symbol 'F', respectively and a branch is taken to OSIU03 to check for oversize.

OSIR013 This routine will provide the symbol 'Folio' for all oversize books with a non-PDT stamp #1. R7 is used as a working register to check for the presence of a $b stamp. R7 is used as a linking register to OSIU02 to set up the height and width of the book. R9 is loaded with the byte address of the symbol 'Folio' and a branch is taken to OSIU04 to check for oversize.

OSIR014 This routine will provide the symbol '*' at the end of the numeric portion of the call number for all oversize books. R7 is used as a linking register to OSIU02 and OSIU05 to set up the height and width of the book and to check for oversize. If the book is oversize, R9 ~~is loaded~~ with the byte address of the symbol. R15 is loaded with a '3' to indicate placement of the symbol after the third element. A branch is taken to OSIU06 to move the symbol.

OSIR015 This routine will provide the symbol 'QUARTO' for all oversize books except those with a non-PDT stamp #1 other than 'CHEM'. R7 is used as a working register to check for the presence of the stamp. R2 is used as a working register to test for the stamp 'CHEM'. R7 is used as a linking register to OSIU02 to set up the height and width of the book. R9 is loaded with the byte address of the symbol 'QUARTO'. A branch is taken to OSIU04 to check for oversize.

OSIR016 This routine will provide the symbol 'F' for the larger books and 'Q' for books larger than the smallest oversize parameter for all oversize books which do not have a non-PDT stamp #1. R7 is used as a working register to check for the presence of the stamp. R7 is used as a linking register to OSIU02 to set up the height and width of the book. R8 and R9 are loaded with the symbol 'Q' and the symbol 'F', respectively. A branch is taken to OSIU03 to check for oversize.

OSIR017 This routine will provide the symbol 'folio' for all oversize books. R7 is used as a linking register to OSIU02 to set up the height and width of the book. R9 is loaded with the byte address of the symbol 'folio'. A branch is taken to OSIU04 to check for oversize.

OSIR018 This routine will provide the symbol 'f' in the left margin for all oversize books. R7 is used as a linking register to OSIU02 and OSIU05 to set up the height and width of the book and to check for oversize. If the book is oversize, R9 is loaded with the symbol 'f'. R7 is used as a linking register to OSIU09 to move the symbol into the left margin.

OSIR019 This routine will provide the symbol 'f' for all oversize books. R7 is used as a linking register to OSIU02 to set up the height and width of the book. R9 is loaded with the byte address of the symbol 'f'. A branch is taken to OSIU04 to check for oversize.

OSIR020 This routine will provide the symbol 'OVERSIZE' for all oversize books. R7 is used as a linking register to OSIU02 to set up the height and width of the book. R9 is loaded with the byte address of the symbol 'OVERSIZE'. A branch is taken to OSIU04 to check for oversize.

OSIR021 This routine will provide the symbol 'F' for all oversize books. R7 is used as a linking register to OSIU02 to set up the height and width of the book. R9 is loaded with the byte address of the symbol 'F'. A branch is taken to OSIU04 to check for oversize.

OSIR022 This routine will provide the symbol '+' for all oversize books. R7 is used as a linking register to OSIU02 to set up the height and width of the book. R9 is loaded with the byte address of the symbol '+'. A branch is taken to OSIU04 to check for oversize.

OSIR023 This routine will provide the symbol 'q' in the left margin for all oversize books. R7 is used as a linking register to OSIU02 and OSIU05 to set up the height and width of the book and to check for oversize. R9 is loaded with the symbol 'q'. R7 is used as a linking register to OSIU09 to move the symbol into the left margin.

OSIR024 This routine will provide the symbol 'Folio' for all oversize books and will also create one extra card for all oversize books. R7 is used as a linking register to OSIU02 and OSIU05 to set up the height and width of the book and to check for oversize. If the book is oversize, 'FMTDATA' is incremented by 1 to create an extra card and R9 is loaded with the byte address of the symbol 'Folio'. A branch is taken to OSIU04 to set up the registers and move the symbol.

OSIR025 This routine will provide the symbol 'O-SIZE' for all oversize books. R7 is used as a linking register to OSIU02 to set up the height and width of the book. R9 is loaded with the byte address of the symbol 'O-SIZE'. A branch is taken to OSIU04-to check for oversize.

OSIR026 This ~~routine~~ will provide the symbol 'F' for the smallest oversize books, 'FF' for the larger oversize books, and 'FFF' for the largest oversize books. R7 is used as a linking register to OSIU02 to set up the height and ~~width of the book~~. R8, R6, and R9 are loaded with the byte address ~~of~~ the symbol 'F', of the symbol 'FF', and of the symbol 'FFF', respectively. A branch is taken to OSIU10 to check for oversize.

OSIR027 This routine will provide the symbol 'f' for the larger oversize books and 'q' for books larger than the smallest oversize parameters, both of which will go into the left margin. R7 is used as a linking register to OSIU02 to set up the height and width of the book. R8 and R9 are loaded with the symbols, 'q' and 'f', respectively. R7 is used as a linking register to OSIU11 and OSIU09 to check for oversize and to move the symbol into the left margin.

OSIR028 This routine will provide the symbol 'q' for the smaller oversize books, 'f' for the larger oversize books, and 'ff' for the largest books; all of which will go into the left margin. R7 is used as the linking register to OSIU02 to set up the height and width of the book. R8 and R9 are loaded with the symbol 'q' and the symbol 'f', respectively. R7 is used as a linking register to OSIU12 and OSIU09 to check for oversize and to move the symbol into the left margin.

OSIR029 This routine will provide the symbol 'q' in front of the first cutter in the left margin for all oversize books. R7 is used as a linking register to OSIU02 and OSIU05 to set up the height and width of the book and to check for oversize. If the book is oversize, R14 is loaded with a '1' (one) to serve as a switch in the move routine. R7 is used as a linking register to OSIU09 to move a 'q' in the left margin in front of the first cutter.

OSIR030 This routine will provide the symbol 'Folio' for all oversize books that do not have either of the non-PDT stamps. R7 is used as a working register to test for the presence of the $b or $c stamp. If no stamp is present, R7 is used as a linking register to OSIU02 to set up the height and width of the book. R9 is loaded with the byte address of the symbol 'Folio'. A branch is taken to OSIU04 to check for oversize.

OSIR031 This routine will provide the symbol 'Folio' for all oversize books except for those in class 'ML' or class 'MT'. R15 is loaded with the class letter 'ML' then 'MT'. R7 is used as a linking register to OSIU01, the class check routine. R7 is used as a linking register to OSIU02 to set up the height and width of the book. R9 is loaded with the byte address of the stamp 'Folio'. A branch is taken to OSIU04 to check for oversize.

OSIR032 This routine will provide the symbol 'F' for larger oversize books or 'Q' for the smaller oversize books. The symbol will appear in front of the LC class alpha. R7 is used as a linking register to OSIU02 to set up the height and width of the book. R8 and R9 are loaded with the byte address of the symbols 'Q' and 'F', respectively. R7 is again used as a linking register to OSIU11 to check for oversize. A branch is then taken to OSIU06 to move the symbol in front of the first element of the call number.

OSIR033 This routine will provide a 'q' for the smallest oversize books, an 'f' for the larger books, and an 'ff' for the largest oversize books. R7 links to OSIU02 to find the height and width. R6, R8, and R9 are loaded with the byte address of the symboles 'f', 'q', and 'ff', respectively. A branch to OSIU10 checkes for oversize for three sets of parms.

OSIR034 This routine provides the symbole 'Oversize' for all oversoze books in classes 'M' and 'N'. R15 is loaded with the class ('M' or 'N') and R7 links to OSIU01, the class check routine. If either class is found, a BAL on R7 to OSIU02 sets up the height and width. R9 is loaded with the byte address of the symbol 'Oversize' and a branch is taken to OSIU04 to check for oversize.

OSIR035 This routine will provide the symbol 'q' for all oversize books except those with a non-PDT stamp 'Ref':R7 is the working register for determining if the stamp is 'Ref' or not. If it is not 'Ref', R7 links to OSIU02 to set up the height and width. R9 is then loaded with the byte address of the symbol 'q' and a branch is taken to OSIU04 to check for oversize.

OSIR036 This routine will provide the symbol 'F' for all oversize books that do not have the stamp 'Ref' or are not clann 'N'. R15 is loaded with the class and R7 links to OSIU01, the class check routine. R7 is then used as the working register to check for a stamp; R2 is the working register to check for 'Ref'. If there is no stamp or the book is not in class 'N', R7 will link

OSIR036 (cont'd) to OSIU02 to set up the height and width, while R9 is loaded with the symbol 'F'. A branch is taken to OSIU04 to check for oversize.

OSIR037 This routine will provide the symbol 'Q' for all oversize books. R7 links to OSIU02 to get the height and width, R9 is loaded with the byte address of the stamp; a branch is taken to OSIU04 to check for oversize.

OSIR038 This routine provides the symbol 'Folio' above the call number for the larger oversize books and a 't' in front of the first element of the call number for the smaller oversize books. R7 links to OSIU02 and gets the height and width of the book. R9 and R8 are loaded with byte address of 'Folio' and 't', repectively. A BAL on R7 to OSIU11 checks for oversize. Upon return, R9 contains the oversize symbol. If it is 'Folio' a branch is taken to OSIU04A; if 't', a branch is taken to OSIU06.

OSIR039 This routine will provide the symbol 'oversize' for all oversize books. R7 links to OSIU02 to set up the ehight and width; R9 is loaded with the byte address of the symbol, and there's a branch to OSIU04 to check for oversize.

OSIR040 This routine will provide the symbol 'oversize' for all books in class 'N'. R15 is loaded with the class letter and R7 links to OSIU01 to check the class. A BAL on R7 to OSIU02 sets up the height and width. R9 is loaded with byte address of 'oversize', and the routine OSIU04 checks for oversize.

OSIR043 This routine will provide the symbol 'tt' for larger oversize books and 't' for the smaller oversize books. R7 links to OSIU02 to set up the height and width. R8 and R9 are loaded with the byte address of the symbols 't' and 'tt', respectively. A branch taken to OSIU03 checks for oversize.

OSIR044 This routine will provide the symbol 'folio' for the larger oversize books and 'oversize' for the smaller books. R7 links to OSIU02 to get the height and widrh. The byte address of the symbols 'oversize' and 'folio' are loaded into R8 and R9 respectively. The branch to OSIU03 checks for oversize.

OSIR045 This routine provides the symbol 'FF' for the largest oversize books, an 'F' for the smaller books, and a 'Q' for the smallest oversize books. A BAL on R7 tc OSIU02 sets up the height and width, while R6 is loaded with the byte address of the symbol 'F', R8 with the byte address of 'Q', and R9 with the byte address of "ff". A branch is taken to OSIU010 to check for oversize for a set of three parms.

OSIR046 This routine provides the symbol 'ff' for the larger oversize books and an 'F' for the smaller books. R7 links to OSIU02 to set up the height and width. R8 and R9 are loaded with the byte address of 'F' and 'FF', respectively. A branch to OSIU03 checks for an oversize book.

OSIR047 This routine will provide the symbol 'f' in the left margin in front of the first cutter for all oversize books. R7 links to OSIU02 to set up the height and width of the book and then R7 links to OSIU05 to check for oversize. If the book is oversize, R14 is loaded with a '1' to indicate that the symbol is to go in front of the first cutter where the first cutter is the first element that begins with a decimal followed by at least one alpha. R15 is loaded with an 'f' and R7 links to OSIU09 to move the symbol inco the left margin.

OSIR048 This routine provides the symbol 'Oversize' for all oversize books. R7 links to OSIU02 to set up the height and width; R9 is loaded with the byte address of 'Oversize'; OSIU04 checks for an oversize book.

OSIR049 This routine will provide the symbol 'f' for the larger oversize books and 'q' for the smaller ones. The symbol will appear in the left margin in front of the first cutter. R7 links to OSIU02 to set up the height and width of the book. R8 and R9 are loaded with symbols 'q' and 'f' respectively. R7 links to OSIU011 to check for oversize. If the book is oversize, R14 is loaded with a '2' to indicate to the move routine that the symbol is to go in ıront of the first cutter where the first cutter is identified as the second element that is one alpha followed by at least one numeric. R7 then links to OSIU09 to move the symbol into the left margin.

OSIR050 This routine will provide the symbol 'q' for all oversize books. It will appear in the left margin in front of the first cutter where the first cutter is identified as the second element that has one alpha followed by at least one numeric. R7 links to OSIU02 to set up the height and width of the book. R7 then links to OSIU05 to check for oversize. If the book is oversize, R14 is loaded with a '2' to indicate to the move routine to place the symbol in front of the first cutter and how to identify the first cutter. R15 is loaded with the 'q' and R7 links to OSIU09 to move the symbol.

FUNCTIONS

READMAST searches the disk data base using the Library of Congress card number and reads a bibliography file record. READMAST constructs the function parameter tables (FPT's) for and issues a CAL3,2 to search the indexes, lock, and unlock the OCLC number index for the LC card number. After the index entry is found, a CAL3,4 is issued to read a bibliography record. The record is read into a user-supplied buffer, and a pointer to the index is returned to the user along with status information about the completion of the read operation.

Upon entry to READMAST, the user parameter list contains a pointer to a double-word aligned workarea. After the FPT's for the CAL3's are constructed, they are stored in the workarea. The Library of Congress card number index is then searched using the user-supplied packed LCCN search key. The LCCN index entry allows access to the OCLC number index for this LCCN. If more than one LCCN entry is found for the same number, a code is returned at completion. The first entry for the LCCN is used to search for the OCLC number.

Before the OCLC number entry is read, a LOCK is requested on the entr' chain to prevent its use by other tasks until after READMAST is finished with it. Then the OCLC number entry is read. This provides a pointer to the bibliography record for the LCCN in question. If the CCLC number read was successful, a CAL3,4 is issued to read the bibliography record. Then the OCLC number entry chain is unlocked, and control is returned to the calling program.

If an error is detected for any of the reads, a completion code is set and control is returned. An error is also declared if the index entry for a number cannot be found.

SOFTWARE INTERFACE

A. LINKAGE

```
LI,R8    READPARM
BAL,R7   READMAST
```

B. PARAMETER LIST DESCRIPTION

```
RES   1           STATUS
PZE   BUFFER      WA(BUFFER)
PZE   WRKAREA2    WA(WORKAREA)
PZE   4096        MAX BUFF. SIZE
PZE   INDEX       WA(INDEX TO RECORD)
PZE   PCKDLCCN    WA(PACKED LCCN SEARCH KEY)
```

Where WORKAREA is an area of 302 words aligned on a doubleword boundary; BUFFER is a 1024 word area; and INDEX is one word.

C. RETURN CODES -

The status of the read operation is returned in the first halfword of the user parameter list.

```
STATUS = X'8000' - NORMAL COMPLETION, NO DUPLICATE KEYS
                     IN THE LCCN INDEX FILE
         X'8001' - NORMAL COMPLETION - DUPLICATE LCCN KEYS
                     HAVE BEEN FOUND AND FIRST BIBLIO. RECORD
                     READ
         X'C000' - LCCN KEY NOT YET ENTERED INTO THE INDEX
                     FILE
         X'C001' - BIBLIO RECORD READ ERROR - ONE OR MORE KEYS
                     DOES EXIST IN THE INDEX FILE
         X'C002' - READ ERROR OCCURED WHILE SEARCHING FOR
                     THE INDEX
         X'C003' - READ ERROR OCCURED WHILE READING THE
                     OCLC# CONTROL FILE
```

D. OTHER ENTRY POINTS - none

E. OCLC SUBROUTINES REFERENCED - none

F. OCLC PROCEDURES REFERENCED - none

FUNCTIONS

FMTREC performs final housekeeping on the data to be output by CNVT and builds the output records for each catalog card production request.

After setting up parameters upon entry, FMTREC builds the output record leader. Then the variable length data fields are moved to the output buffer. The following cleanups are made on the data fields:

1) field indicators are added if missing.
2) indicators are unpacked if existant.
3) a '‡a' subfield delimiter and code are added if the text portion of a field begins with no'‡' delimiter.
4) the 240 tag is changed to a 130 tag in the absence of a 1XX tag. Indicators are not changed.
5) the 690 tag is changed to a 650 tag. Indicators are not changed.

Field lengths are adjusted to account for all modifications. Each variable length field is processed and inserted in the output buffer in the order in which it appears in the forward link chain, LNK1, in CNVT. Processing is stopped when the link points back to the input record leader.

At the end of processing a record terminator is inserted at the end of the record. Control is then returned to CNVT. The following parameters are passed back to CNVT.

1) Output record length
2) Status bits indicating whether or not an error was encountered while formatting the record.

SOFTWARE INTERFACE

A. LINKAGE

Control is passed to FMTREC from CNVT using the following sequence:

```
LI,R8     PARMLIST
BAL,R7    FMTREC
```

B. PARAMETER LIST DESCRIPTION

The following list of parameters is passed to FMTREC.

```
PARMLIST   GEN,4,12,16    0,0,0   STATUS/.../...
           DATA     LNKDV      LINK DIRECTORY's
                                DUPE VECTOR
           DATA     WA(UNPACKED CARD NO)
           DATA     FMTDATA    WA(PDTNO, INDICAT,
                                   RESERVED BYTES)
           DATA     BA(OUTPUT BUFFER)
           DATA     :BUFSZ     LENGTH OF OUTPUT BUFFER
           DATA     FMTLEN     RETURN RECORD LENGTH
```

Where STATUS bits 0-3 have the following meaning.

```
CC1 - 4 = 0  NORMAL COMPLETION
CC1 -   = 1  PROCESSING ERROR
CC2 -   = 1  BUFFER OVERFLOW
CC3 -   = 1  ...
CC4 -   = 1  ...
```

C. RETURN CODES - see status bits in parameter list above.

D. OTHER ENTRY POINTS - none

E. OCLC SUBROUTINES REFERENCED - none

F. OCLC PROCEDURES REFERENCED - none

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'
BYTE0 = X'C0'  ERROR
BYTE1 = X'01'  NO ROOM IN FIELD FOR SIGN
BYTE0 = X'C0'  ERROR
BYTE1 = X'02'  FIELD OVERFLOW
```

D. OTHER ENTRY POINTS - none

E. OCLC SUBROUTINES REFERENCED - none

F. OCLC PROCEDRURES REFERENCED - none

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 LOBMSG.

```
GEN,8,24      FUN,BA(MESSAGE)
DATA          WA(UNPACKED LC CARD NUMBER)
DATA          WA(COLOR CODE)
DATA          WA(LIBRARY CODE)
```

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

```
FUN = 0    Print message only, do no logging.
    = 1    Log as selected before printing a message.
    = 2    Log as missing before printing a message.
    = 3    Log as rejected before printing a message.
    = 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

FUNCTIONS

LINK initially builds a link directory of addresses, lengths, and tags for the variable data fields of the input record for CNVT. Then at its alternate entry points, LINKINST and LINKDELT it respectively inserts or deletes an entry in its directory.

LINK is entered after a record has been read and its leader processed by CNVT. Included in the parameters passed are the byte address of the record and the word addresses of the areas in CNVT where the directories are to be built. LINK systematically scans the variable length data fields, one field at a time and stores the field's byte address and length and its tag in the tables LNKBA(byte address and length) and LNKTAG(tags). Two tables of index values are kept as a directory to the tables, LNKBA and LNKTAG. The directory provides forward and backward links among the fields and tags. A duplicate set of index tables (a duplicate directory) is built as the working directory is built. The duplicate directory is retained as a map of the original record during processing. When the end of the record is encountered, the directories are complete. The total number of entries in the directory (the largest index entry in the forward link table) is stored at the beginning of the backward link tables and in LNKTAG. An error condition arises when no delimiter is found for a field. In this case a condition code of X'80' is returned in the status byte of the parameter list. Otherwise, for normal returns, the status byte is set to zero and control is returned to the calling program.

When LINKINST is entered the parameter list contains the word addresses of the directory and tables; but instead of the byte address of the input record, an index into the forward link table is present to indicate where the new field is to be inserted. Also present is a pointer to the byte address of the field to be inserted, its length and its tag. Upon entry the total number of directory entries is incremented by one. This value will be the index value for the new field. The index value in the parameter list is used to get a forward and backward link to the field immediately following where the new field is to be inserted. The index for the existing field is moved to the end of the directory, and the new index is inserted in its place. Then the byte address and length of the new field are stored as the last entry in LNKBA, and the tag becomes the last entry in LNKTAG. The total number of directory entries is brought up to date. An error occurs if the index in the parameter list is greater than the number of entries in the directory. In this case the status byte in the parameter list is set to X'80' and control is returned. Upon normal completion the status byte is set to zero, and control is returned.

When LINKDLT is entered, the parameter list still contains the word addresses for the directory and tables in CNVT. An index is also present to the forward link directory indicating the field to be deleted. To delete the field, the index entries in the directory for the field are nullified. This is accomplished by setting the index value for the field in the forward links equal to that immediately following it in the directory. The index value for the field to be deleted in the backward links is set equal to the index value immediately preceding it. The lengths of the directories are not changed. An error is declared if the index value in the parameter list is greater than the total number of entries in the directory. If this condition is encountered, the status byte in the parameter list is set to X'80' and control is returned. For normal completion, the status byte is zeroed and control is returned to the calling program.

SOFTWARE INTERFACE (LINK)

A. LINKAGE

- Control is transferred to LINK by the following sequence of instructions.

```
LI,R8      LNKPARMS
BAL,R7     LINK
```

B. PARAMETER LIST DESCRIPTION

```
LNKPARMS      GEN,4,12,16    0,0,0      STATUS/ERROR NO./...
              DATA     BA(INPUT RECORD)
              GEN,8,24    0,LNKDV       LINK DIRECTORY
                                         DUPE VECTOR
```

Where LNKDV is the following list

```
LNKDV    EQU  $    LINK DIRECTORY DUPE VECTOR
         DATA     WA(INITIAL SIZE OF LINKED DIRECTORY)
         DATA     WA(LINK TAG TABLE)
         DATA     WA(LINK BYTE ADDRESS AND LENGTH TABLE)
         DATA     WA(FORWARD LINKS 1)
         DATA     WA(BACKWARD LINKS 1)
         DATA     WA(FORWARD LINKS 2)
         DATA     WA(BACKWARD LINKS 2)
```

C. RETURN CODES

The status byte in LNKPARMS reflects the completion of LINK.

```
STATUS = X'00'    Normal completion
       = X'80'    Invalid or missing field
                   delimiter encountered
```

D. OTHER ENTRY POINTS

LINKINST
LINKDLT

E. OCLC SUBROUTINES REFERENCED - none

F. OCLC PROCEDURES REFERENCED - none

SOFTWARE INTERFACE (LINKINST)

A. LINKAGE

Calling Sequence is

```
LI,R8   LNKPARMI
BALMR7  LINKINST
```

B. PARAMETER LIST DESCRIPTION

```
LNKPARMI    DATA   WA(FIELD BYTE ADDRESS, LENGTH, TAG)
            DATA   INDEX INTO DIRECTORY FOR FIELD
                    PRECEDING FIELD TC BE INSERTED.
            DATA   LNKDV
```

Where LNKDV is the same as for LINK.

C. RETURN CODES

The first byte of LNKPARMI is used as the status byte on return from LINKINST.

```
STATUS = X'00'   Normal completion
       = X'80'   Index for entry to be inserted is
                  greater than total no. of entries
                  in the directory.
```

D. OTHER ENTRY POINTS - none

E. OCLC SUBROUTINES REFERENCED - none

F. OCLC PROCEDURES REFERENCED - none

SOFTWARE INTERFACE (LINKDLT)

A. LINKAGE

Calling sequence is

```
LI,R8          LNKPARMD
BAL,R7         LINKDLT
```

B. PARAMETER LIST DESCRIPTION

```
LNKPARMD        GEN,4,12,16   0,0,0    STATUS/ERROR NO./ . . .
                DATA          Index into directory for field
                              to be deleted.
                DATA          LNKDV
```

Where LNKDV is the same as for LINK.

C. RETURN CODES

The status byte in LNKPARMD reflects the completion of LINKDLT.

STATUS = X'00'	Normal completion
=X'80'	Index for entry to be deleted is greater than total no. of entries in directory

D. OTHER ENTRY POINTS - none

E. OCLC SUBROUTINES REFERENCED - none

F. OCLC PROCEDURES REFERENCED - none

FUNCTIONS

TAPEIO is a general purpose input/output subroutine which performs the following functions depending on a function code passed from the calling program.

FUNCTION CODE	FUNCTION
X'00'	READ
01	WRITE
02	READ REVERSE
03	WEOF
04	SKIP ONE RECORD FORWARD
05	SKIP ONE RECORD BACKWARD
06	SKIP ONE FILE FORWARD
07	SKIP ONE FILE BACKWARD
08	REWIND (ONLINE)
09	UNLOAD

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:

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

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 normally, the condition code returned is the result of the last retry.

If the operation was a READ but the error was declared noncorrectable (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'C0') 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

WORD 0	FUNCTION	DEVICE ADDRESS
WORD 1	BA (BUFFER)	
WORD 2	BYTE COUNT	
WORD 3	EVENT STATUS	

For functions 3-9

WORD 0	FUNCTION	DEVICE ADDRESS
WORD 1	EVENT STATUS	

C. RETURN CODES

NORMAL COMPLETION: EVENTWORD BYTE 0 = X'80'
BYTE 1 = X'00'

ABNORMAL COMPLETION: EVENT WORD BYTE 0 = X'C0'
BYTE 1 = XX - CODE INDICATING NATURE OF ABNORMAL COMPLETION.

Possible event words for abnormal completion and their meanings are listed below:

EVENT WORD	TDV STATUS	MEANING
C001	0200	NORMAL TERMINATION BEYOND END OF TAPE MARKER
C002	0400	NORMAL TERMINATION AT BEGINNING OF TAPE
C00A	000E	IOP ERROR
C009	0010	MEMORY ADDRESS ERROR
C008	2000	WRITE PROTECT VIOLATION
C003	1000	END OF FILE
C007	8000	DATA OVERRUN
C004	0800	NON-CORRECTABLE READ ERROR
C005	0040	TRANSMISSION DATA ERROR
C006	0020	TRANSMISSION MEMORY ERROR
C000	----	UNIT UNRECOGNIZED
C00B	----	SOFTWARE ERROR

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

FUNCTIONS

READSC reads and interprets select cards input to CNVT in the offline mode. The Library of Congress card number and the library code are stored in areas provided by the user. The color code and function code are converted to binary and stored in user fields. Then the remainder of the card is scanned for a X'E0' which denotes the beginning of each subfield. When the field delimiter is encountered, the following character is interrogated to determine what type of field is present. A X'4E' denotes the stamp field, X'60' denotes copies, X'7E' denotes user data, and X'5C' denotes text. When a field type is recognized, the tag for that field is stored in the user area and the data length, data, and a subfield delimiter are moved in. This procedure is repeated for each valid field type until a X'4F' is encountered, signaling the end of the fields. Error conditions are as follows:

1. Read error
2. Invalid character found in card column 26
3. Invalid character found in card column 20
4. The card subfield has overflowed the user storage area

For all error conditions the status byte in the user parameter list is set to X'80' before returning. For normal completion the status returned is X'00'.

SOFTWARE INTERFACE

A. LINKAGE

Control is transferred to READSC via the following instructions.

```
LI,R8      RDSCPARM
BAL,R7     READSC
```

B. PARAMETER LIST DESCRIPTION

```
RDSCPARM   EQU    $
           DATA   0         STATUS
           DATA   WA(USER AREA FOR OCLC NO.)
           DATA   WA(AREA FOR LIBRARY CODE)
           DATA   WA(AREA FOR COLOR CODE)
           DATA   WA(AREA FOR FUNCTION CODE)
           DATA   WA(AREA FOR TAG 1)
           DATA   WA(AREA FOR TAG 2)
           DATA   WA(AREA FOR NO. OF EXTRA CARDS)
           DATA   WA(AREA FOR STAMP SUBFIELD)
           DATA   WA(AREA FOR EXTRA COPY SUBFIELD)
           DATA   WA(AREA FOR TEXT SUBFIELD)
           DATA   WA(AREA FOR USER DATA SUBFIELD)
```

C. RETURN CODES

```
STATUS = X'00'     NORMAL COMPLETION
       = X'80'     ONE OF THE FOLLOWING ERRORS HAS OCCURRED

                     1.  READ ERROR
                     2.  INVALID CHARACTER IN CARD COLUMN 20
                     3.  INVALID CHARACTER IN CARD COLUMN 26
                     4.  CARD SUBFIELD HAS OVERFLOWED USER
                          STORAGE AREA
```

D. OTHER ENTRY POINTS - none

E. OCLC SUBROUTINES REFERENCED - none

F. OCLC PROCEDURES REFERENCED - none

FUNCTIONS

PUNCHSC formats cards for CNVT which will be used to select out the input cards for which catalog cards were produced. Upon entry to PUNCHSC, the first two words of the Library of Congress card number are stored in the next available position in the output buffer. The final word of the L.C. card number and the holding library code are formatted to insure the proper position of the library code; then the two fields are stored in the output buffer. The index to the output buffer is advanced, and control is returned to CNVT. When the output buffer is full, it is written to the output device.

At its alternate entry point, CLOSESC, the output buffer is padded to its maximum. Then the last buffer is written, end of file housekeeping is performed, and control is returned.

SOFTWARE INTERFACE (PUNCHSC)

A. LINKAGE

Linkage to PUNCHSC is obtained via a

```
BAL,R7    PUNCHSC
```

where the parameter list described below immediately follows the BAL instruction.

B. PARAMETER LIST DESCRIPTION

```
DATA    WA(L.C. CARD NO.)
DATA    WA(LIBRARY CODE)
```

C. RETURN CODE - none

D. OTHER ENTRY POINTS - CLOSESC

E. OCLC SUBROUTINES REFERENCED - none

F. OCLC PROCEDURES REFERENCED - none

SOFTWARE INTERFACE (CLOSESC)

A. LINKAGE

BAL,R7 CLOSESC

B. PARAMETER LIST DESCRIPTION - none

C. RETURN CODES - none

D. OTHER ENTRY POINTS - none

E. OCLC SUBROUTINES REFERENCED - none

F. OCLC PROCEDURES REFERENCED - none

FUNCTIONS

LCCN000 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.

LCCN000 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 090 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 ⎬     Different name values for the
BLANK ⎪      same procedure
TERMN ⎪
COMMA ⎭
BREAK
```

PROGRAM: CNVT
SUBROUTINE: LCCN000

PROCEDURE: NEXT

PROCEDURE DESCRIPTION

PURPOSE: NEXT generates a BAL,R7 :NXT where: NXT is an internal subroutine of LCCN000

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
```

PROGRAM: CNVT
SUBROUTINE: LCCN000
PROCEDURE: SPAN

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 AF(3) and AF(5). If AF(3) =1 and AF(5) =0 a B $+2 is generated. If AF(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,YES,T4
+ 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.

PROGRAM: CNVT
SUBROUTINE: LCCN000
PROCEDURE: SAVE

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
```

PROGRAM: CNVT
SUBROUTINE: LCCN000
PROCEDURE: ALPHA

PROCEDURE DESCRIPTION

PURPOSE: ALPHA compares the character value in R15 to its name value shifted left one position (1**NAME). 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

PROGRAM: CNVT
SUBROUTINE: LCCN000
PROCEDURE: BREAK

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 CHARVAL

+               LI,14    2
+               BAL,R7   :BRK
```

FUNCTIONS

Trees (entry point, NODETBL) is a large table of index values to processing routines within CNVT. The trees consist of nodes, leaves, and tests which are generated by the procedures, NODE, LEAF, and TEST. The initial index into NODETBL is obtained from the profile definition table for a member holding library.

Each NODE has as the first argument field the number of leaves in its particular tree. The minimum is two and the maximum is thirteen leaves. The second argument field is the address of the first leaf in the tree. The leaves are picked up sequentially beginning with the leaf indicated by the second argument field. If any nodes are encountered in the list, they are expanded in place.

A TEST causes a check to be made on a switch in CNVT and a choice of entries to be made based on the value of the switch. When a TEST is encountered, the value of argument field (1) tells which loop switch in CNVT is to be tested. Argument field (2) gives the address of the first of the two alternate entries. If the switch tested is set, the first alternative is selected; if the switch is not set, the second alternative is selected. The alternatives may be nodes or leaves.

The first argument field of a LEAF is an index into a table of routines called EXUTBL. The second argument field is used as an indicator within the routine set up by EXUTBL.

PROGRAM: CNVT
SUBROUTINE: TREES
(NODETBL)

SOFTWARE INTERFACE

A. LINKAGE - not applicable

B. Parameter List Description - none

C. Return Codes - none

D. Other Entry Points - none

E. OCLC Subroutines Referenced

F. OCLC Procedures Referenced

NODE
LEAF
TEST

PROGRAM: CNVT
SUBROUTINE: TREES
(NODETBL)
PROCEDURE: NODE

PROCEDURE DESCRIPTION

PURPOSE:

The purpose of the procedure NODE is to generate a word in NODETBL which indicates the number of leaves to be picked up from NODETBL and where in the table to find the leaves.

There are two alternate names for this procedure, TEST and LEAF. The CNAME value assigned to each name indicates to the processing program which type of entry in NODETBL it is using.

NODE has a CNAME value of 8. TEST has a CNAME value of 4. TEST is used to generate a word which indicates a switch to be tested. It also includes the location in NODETBL for the two alternative branches to be taken depending on the value of the switch.

LEAF has a CNAME value of Ø. It generates a word which contains an index value into the table 'EXUTBL' in CNVT. Also included in the word is an indicator to be passed to the routine pointed to in EXUTBL.

FORMAT:

```
     NODE  AF(1), AF(2)
```

EXAMPLE 1:

```
     NODE 1Ø, #1ØØ.
+    GEN,4,12,16   8,1Ø,#1ØØ
```

CNVT will pick up ten entries in NODETBL beginning with #1ØØ.

EXAMPLE 2:

```
     TEST    2,#1ØØØ
+    GEN,4,12.16  4,2,#1ØØØ
```

CNVT will test loop switch 2. If the switch is set, the first entry at #1ØØØ is selected. If the switch is not set, the second entry at #1ØØØ is selected.

EXAMPLE 3:

```
     LEAF 65,1
+    GEN,4,12,16  0,65,1
```

CNVT will execute the load instruction at 'EXUTBL'+65. The value 1 will be passed to the processing routine.

APPENDIX E

ADDITIONAL PROCEDURE DOCUMENTATION

PROCEDURE DESCRIPTION

PURPOSE:

WRTMSG establishes parameters and links to the external subroutine LOGMSG. The CNAME value of WRTMSG is 0 and indicates to LOGMSG that a single message is to be printed. There are four alternate names for WRTMSG.

WRTSELD - has a CNAME value equal to 1 and indicates that a request is to be logged as selected.

WRTMISS - has a CNAME value of 2 and indicates that an OCLC number is to be logged as missing.

WRTRJD - has a CNAME value of 3 and indicates that a request is to be logged as rejected.

WRTEJECT - has a CNAME value of 15 and indicates that the page is to be ejected when printing the message.

FORMAT:

```
     WRTMSG  AF(1)
```

EXAMPLE 1:

```
     WRTMSG STATHEAD

+    BAL,R7  LOGMSG
+    GEN,8,24 0,BA(STATHEAD)
+    DATA    MSGPARMS
```

EXAMPLE 2:

```
     WRTSELD

+    BAL,R7  LOGMSG
+    GEN,8,24  1,0
+    DATA MSGPARMS
```

PROCEDURE DESCRIPTION

PURPOSE:

PUNCHSLD checks to see if any select cards are to be punched. If so, a BAL to PUNCHSC is taken to punch the cards for a call number. Also included in PUNCHSLD are the parameters for PUNCHSC.

FORMAT:

PUNCHSLD

EXAMPLE:

```
  PUNCHSLD

+             LW,R7  STATSW
+             BGZ    $+4
+             BAL,R7  PUNCHSC
+             DATA   UNPACKED   WA(UNPACKED LC CARD NO.)
+             DATE   LASTLIB    WA(LIBRARY CODE)
```

PROGRAM: CNVT
PROCEDURE: ATBL

PROCEDURE DESCRIPTION

PURPOSE:

ATBL buildds entries in the table TBLA. Each entry consists of a displacement and an address which will be the effective address of a branch instruction.

FORMAT:

```
     ATBL    AF(1), AF(2)
```

```
EXAMPLE 1:  ATBL          015, ALOW

+      ORG,1              BA(ITBL)+15
+      DATA,1             0
+      ORG                TRLA+0
+       B                 ALOW
```

```
EXAMPLE 2:  ATBL

+      ORG,1              BA(ITBL)+20
+      DATA,1             1
+      ORG                TBLA+1
+      B                  ALOW
```

PROGRAM: CNVT
PROCEDURE: NOTE:

PROCEDURE DESCRIPTION

PURPOSE:

NOTE: advances the location counter 1 byte.

FORMAT:

NOTE:

EXAMPLE

```
    NOTE:
+   BOUND 1
```

APPENDIX F

EXAMPLES

Ohio University example -

From 'READ PDT'S' it is found that Ohio University uses the default processors of BLUE 1 and YELLOW 1. From the control section 'TREES', it is found that a BLUE 1 takes ten instructions beginning with #100; and a YELLOW 1 takes ten instructions beginning with #11Ø. BLUE 1 yields the following instructions:

```
LEAF    1,0
LEAF    3,0
LEAF    4,0
LEAF    5,0
LEAF    6,0
LEAF    7,0
LEAF    8,0
LEAF    9,0
LEAF    2,0
NODE    9,#111                       LEAF     11,0
                                     LEAF     12,0
                                     LEAF     13,0
                                     LEAF     14,0
                                     LEAF     15,0
                                     LEAF     16,0
                                     LEAF     17,0
                                     LEAF     18,0
                                     LEAF     25,0
```

The first number in the argument field of the 'NODE' instruction tells how many instructions to take and the second number tells from where to start taking them. Those instructions at that location replace the original 'NODE' instruction.

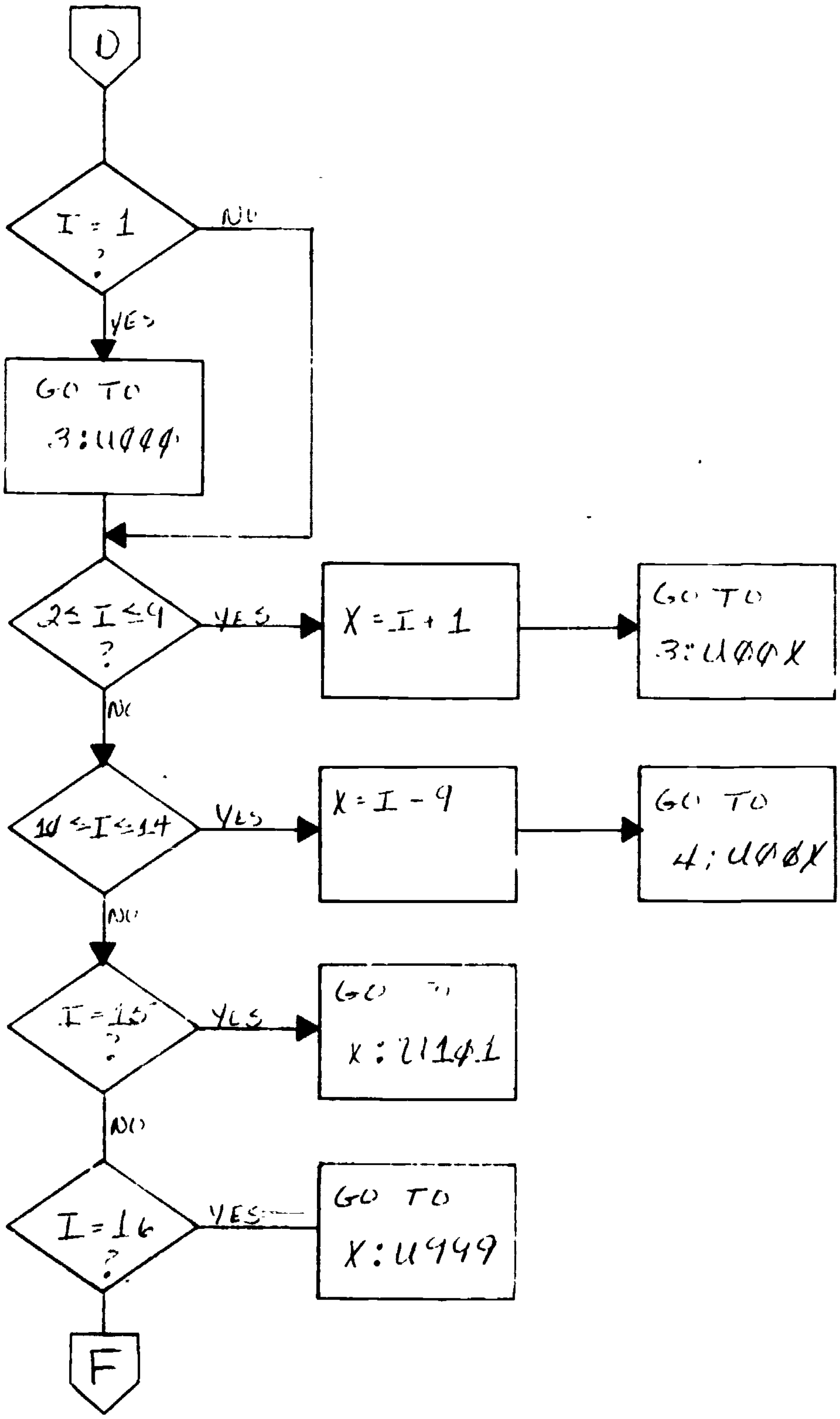
The first number in the argument field of a 'LEAF' instruction is an index into the table 'EXUTBL' from which the addresses of the formatting routines are pushed into a stack. To format the call number for Ohio University, the following routines are used in order:

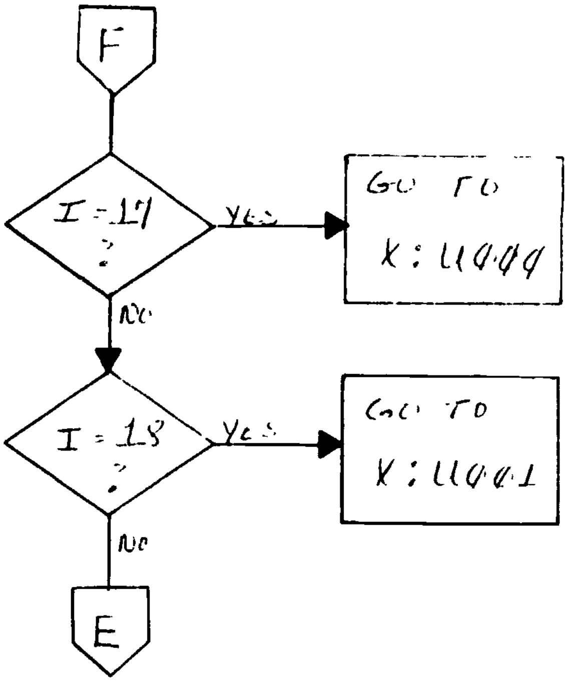
```
 1.)  3:U000
 2.)  3:U001
 3,)  3:U002
 4.)  3:U003
 5.)  3:U004
 6.)  3:U005
 7.)  3:U006
 8.)  3:U007
 9.)  3:U008
10.)  4:U001
11.)  4:U002
12).  4:U003
13.)  4:U004
14.)  4:U005
15.)  4:U101
16.)  4:U999
17.)  X:U000
```

This will process a call number in this manner:

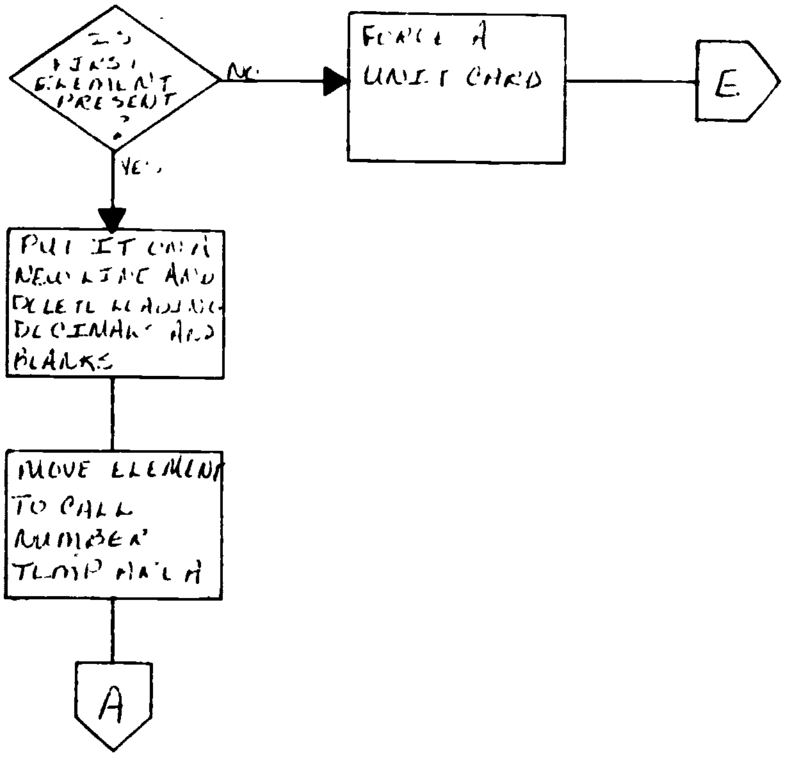
```
PDT STAMP
NON-PDT STAMP # 1
AB
123.45
1962
C78
1979
D96
NON-PDT STAMP #2
```

With no user data automatically supplied, no automatic oversize symbol, and no special tag processors.

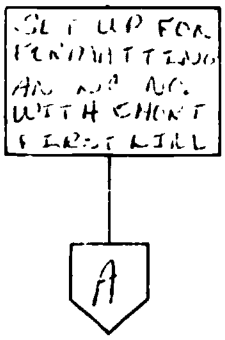

D
I = 1 ?
NO
YES
GO TO 3:U0ØØ
2≤I≤9 ?
YES
X = I + 1
GO TO 3:U0ØX
NO
10≤I≤14
YES
X = I - 9
GO TO 4:U0ØX
NO
I = 15 ?
YES
GO TO X:U1Ø1
NO
I = 16 ?
YES
GO TO X:U999
F


F
I = 17
?
Yes
GO TO
X: 11000
No
I = 18
?
Yes
GO TO
X: 11001
No
E

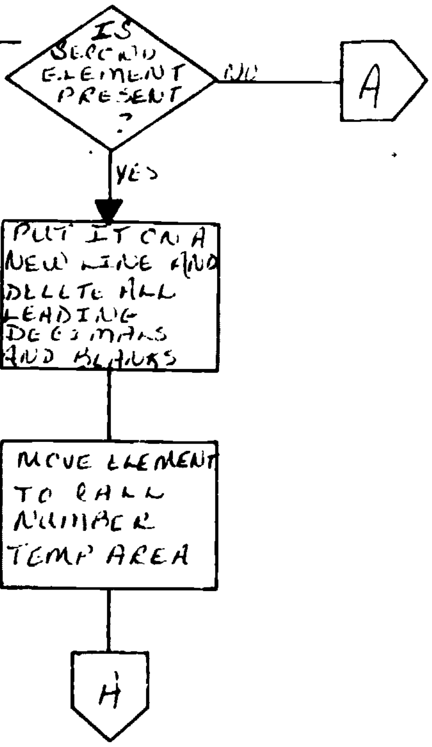
3:JØØ1


IS FIRST ELEMENT PRESENT ?
NO
FORCE A UNIT CARD
E
YES
PUT IT ON A NEW LINE AND DELETE LEADING DECIMALS AND BLANKS
MOVE ELEMENT TO CALL NUMBER TEMP AREA
A

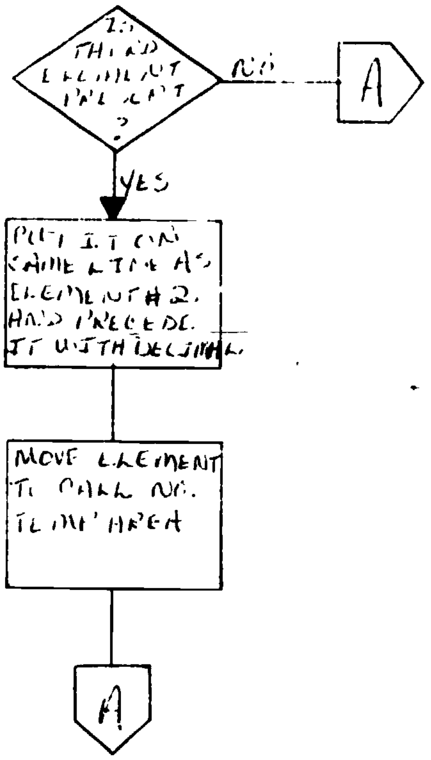
3:UØØØ


SET UP FOR FORMATTING AN NP NO. WITH SHORT FIRST LINE
A

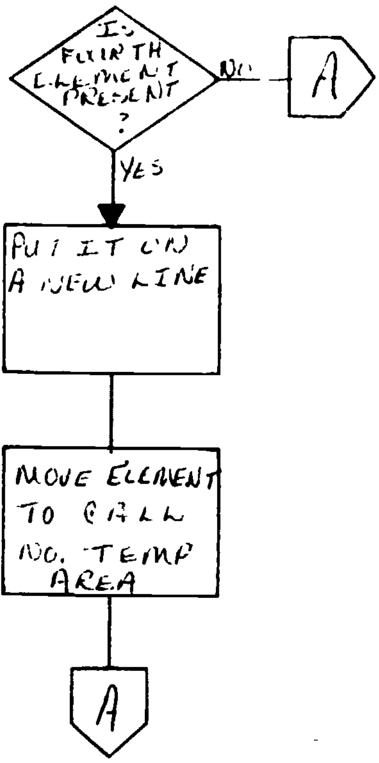
3:UØØ2


IS SECOND ELEMENT PRESENT ?
NO
A
YES
PUT IT ON A NEW LINE AND DELETE ALL LEADING DECIMALS AND BLANKS
MOVE ELEMENT TO CALL NUMBER TEMP AREA
H

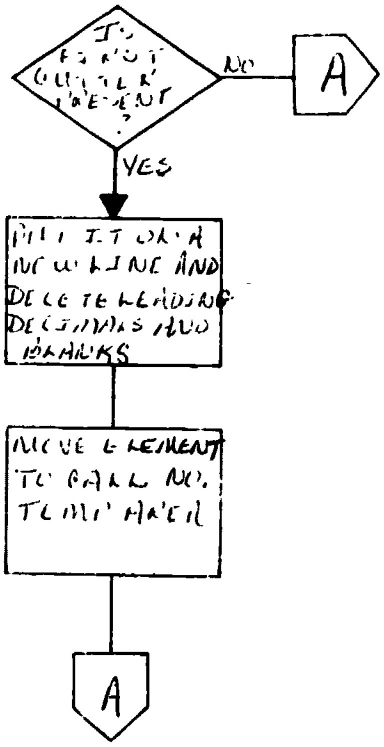
3:UØØ3


IS THIRD ELEMENT PRESENT ?
NO
A
YES
PUT IT ON SAME LINE AS ELEMENT #2. AND PRECEDE IT WITH DECIMAL
MOVE ELEMENT TO CALL NO. TEMP AREA
A

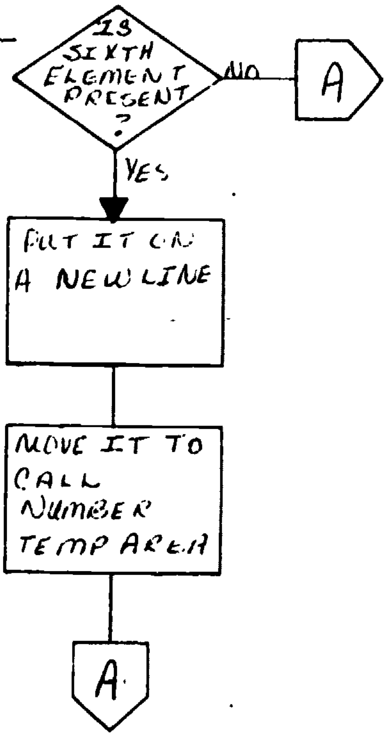
3:UØØ4


IS FOURTH ELEMENT PRESENT ?
NO
A
YES
PUT IT ON A NEW LINE
MOVE ELEMENT TO CALL NO. TEMP AREA
A

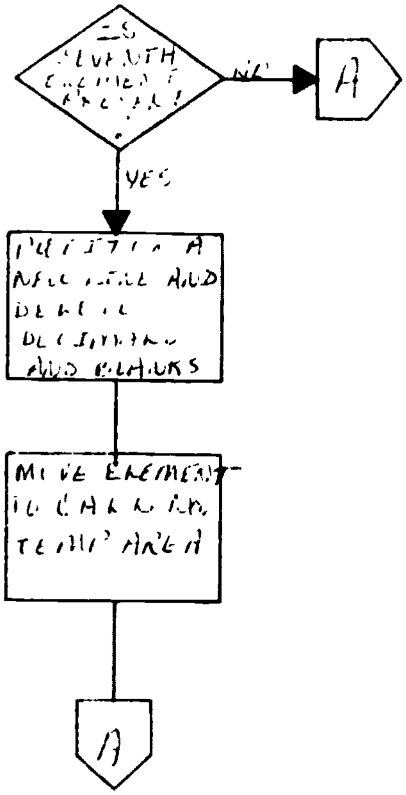
3:UØØ5


IS FIRST CUTTER PRESENT ?
NO
A
YES
PUT IT ON A NEW LINE AND DELETE LEADING DECIMALS AND BLANKS
MOVE ELEMENT TO CALL NO. TEMP AREA
A

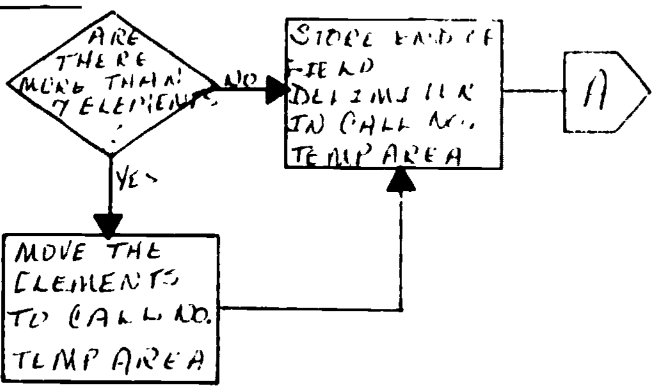
3:UØØ6



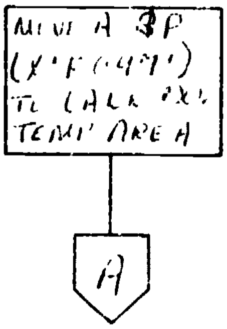
3:UØØ7


IS SEVENTH ELEMENT PRESENT ?
NO
A
YES
PUT IN A NEW FIELD AND DELETE DELIMITERS AND BLANKS
MOVE ELEMENT TO CALL NO. TEMP AREA
A

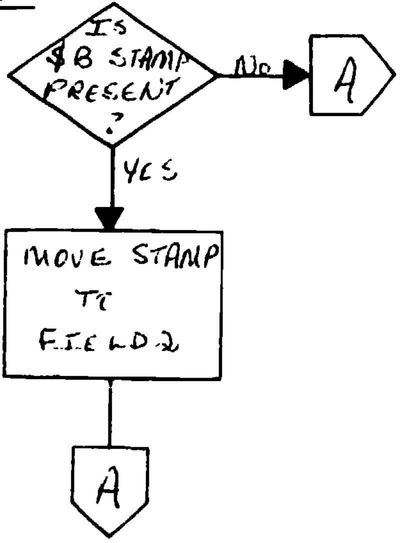
3:UØØ8


ARE THERE MORE THAN 7 ELEMENTS ?
NO
STORE END OF FIELD DELIMITER IN CALL NO. TEMP AREA
A
YES
MOVE THE ELEMENTS TO CALL NO. TEMP AREA

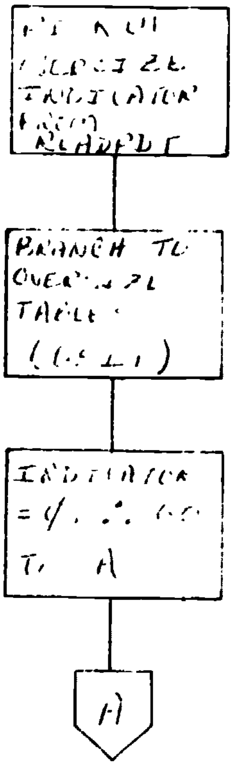
4:UØØ1



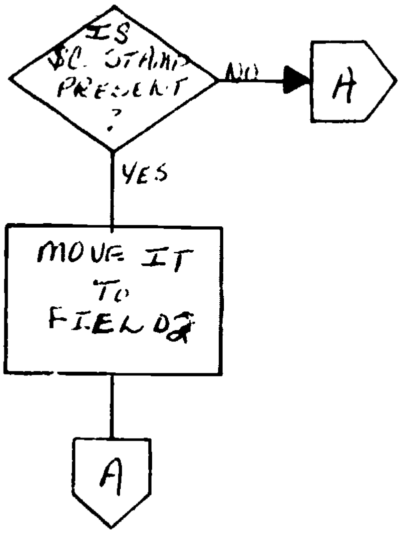
4:UØØ2



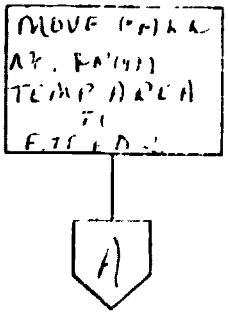
4:UØØ3



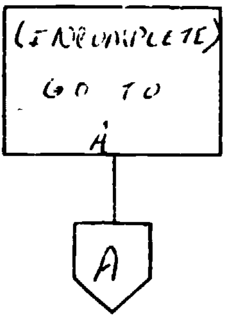
4:UØØ5

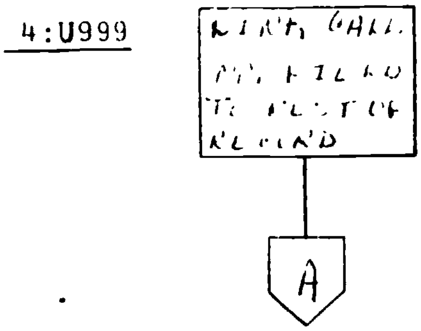


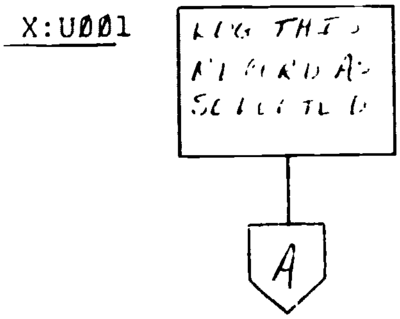
4:UØØ4

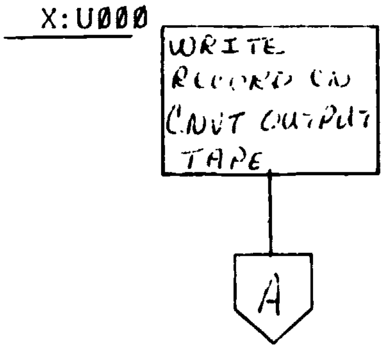


4:U1Ø1




4:U999
[illegible] [illegible]
[illegible] FILE [illegible]
TO [illegible] OF
RECORD
A


X:U001
LOG THIS
RECORD AS
SELECTED
A


X:U000
WRITE
RECORD ON
CNVT OUTPUT
TAPE
A