AD-783 406

MACHINE INDEPENDENT DATA MANAGEMENT SYSTEM (MIDMS) SYSTEM SPECIFICATIONS. CHAPTER 1 - FILE STRUCTURING CHAPTER 2 - LIBRARIAN

Defense Intelligence Agency
Washington, D. C.

1 July 1974

AD783406

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
Machine Independent Data Management System (MIDMS) System Specifications
CHAPTER 1 - File Structuring
CHAPTER 2 - Librarian

5. TYPE OF REPORT & PERIOD COVERED
User Documentation

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Defense Intelligence Agency (DIA)

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Defense Intelligence Agency
ATTN: Information Processing Division (DS-5)
Washington, D.C. 20301

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Same as 9

12. REPORT DATE
1 July 1974

13. NUMBER OF PAGES
114

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

15. SECURITY CLASS. (of this report)
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; unlimited distribution

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

MIDMS	Subsystem	Fixed Set
File Structuring	Subroutine	Variable Set
Librarian	Retrieval	Module
File Maintenance	Output	
Language Processor	Periodic Set	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
This document describes the MIDMS system specifications. Each module of the system is described in detail to include its subsystem structure, subprogram identification and description, flow charts, and error messages. Differences between the IBM and Honeywell versions of MIDMS are documented.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

MACHINE INDEPENDENT DATA MANAGEMENT SYSTEM

(MIDMS)

SYSTEM SPECIFICATIONS

FOREWORD

This manual has been developed by the Defense Intelligence Agency (DIA) for technical purposes. It does not reflect either explicitly or implicitly official DIA policy or intelligence matters. Its purpose is to provide a detailed description of the Machine Independent Data Management System (MIDMS) programs. DIA assumes no system installation, program maintenance, or system operation responsibility, nor is DIA responsible for the data upon which the system operates.

TABLE OF CONTENTS

IV - Blank

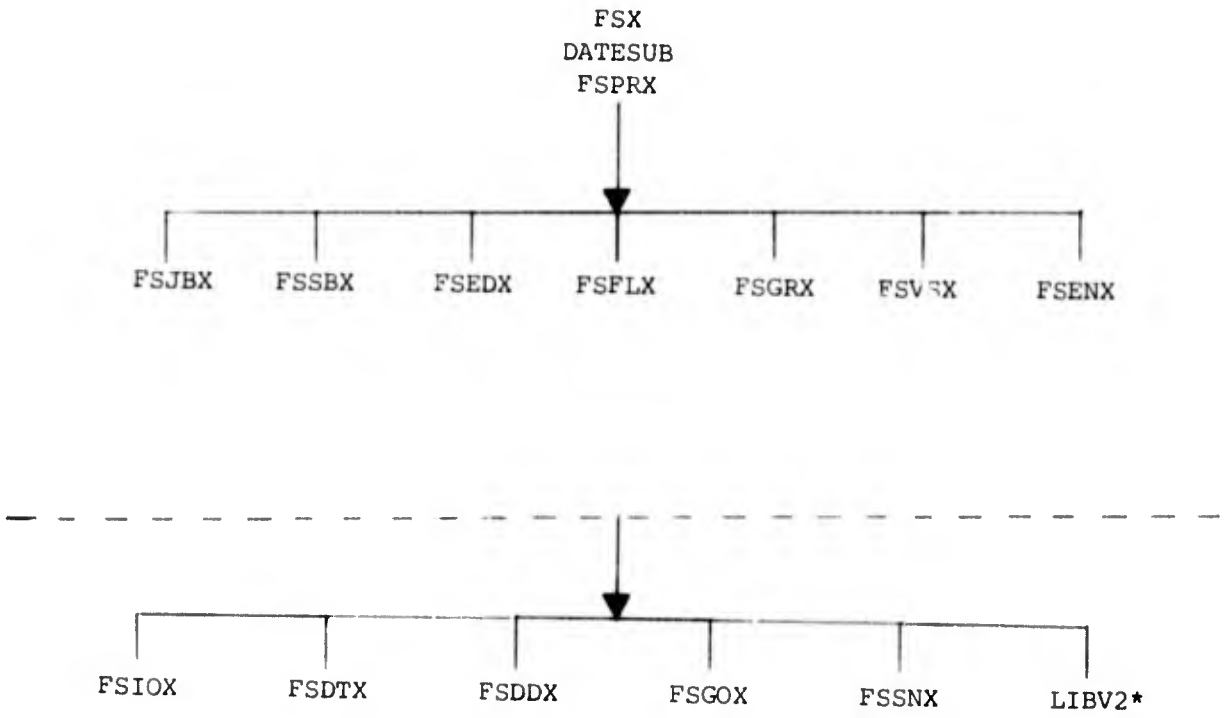
PAGE

Enclosures A, B, C & D – AD-783409

CHAPTER 1

File Structuring

1. SUBSYSTEM STRUCTURE. The File Structuring module of the Machine Independent Data Management System (MIDMS) consists of 14 subroutines that are executed during processing of this module. Through the use of the overlay structure, File Structuring requires only 84K bytes of core. All of the subroutines use the File Structuring Data Division (FS-DD) with the exception of the calls to the date subroutine (DATESUB). The parameter used in the calling sequence to DATESUB is HDR-DATE, which is defined in FS-DD. The date is taken from the computer system and passed to MIDMS FS for its use during the execution of a job. Parameters used in the calling sequence to the MIDMS table librarian are CALLING-SEQ and the particular logical record work area that is being written on the library.



* Library subroutine LIBV2 includes several object modules.

FIGURE 1-1. IBM OVERLAY STRUCTURE

2. COBOL DATA DIVISION ENTRIES. The following description of the COBOL data division is divided into three basic sections. The first section contains a description of the eleven logical records which File Structuring (FS) generates. The second section contains a description of the tables and work areas that are used by FS to generate the logical records. The last section describes how the generated COBOL data division entries are produced from the hierarchial field structure provided in logical record 10.

a. File Format Table (FFT). The MIDMS FFT is a collection of eleven tables which describes a data file to all MIDMS modules. These tables are called logical records and are numbered LR1 to LR11.

(1) FFT Logical Record 1.

(a) Data Statements:

```
01  FS-DD.
    02  FFT.
      03  FFT-LR1.
        04  FILE-MNEMONIC-1      PICTURE IS X(5).
        04  FILE-ID-1            PICTURE 999.
        04  HIST-IND             PICTURE IS X VALUE IS SPACE.
        04  XCHG-IND             PICTURE IS X VALUE IS SPACE.
        04  FIND-CNT             PICTURE IS 99.
        04  CTL-LEN              PICTURE IS 99.
      03  PSC-LR OCCURS 11 TIMES.
        04  CNT-LR               PICTURE IS 9999.
```

(b) Description: FFT-LR1 is the identification record for the FFT pertaining to the particular file.

FILE-MNEMONIC. Contains the five-character name assigned to this file. This name must be unique within the system and must consist of alphabetic or numeric characters, starting with a letter and ending with "A."

FILE-ID. Is a unique three-digit code that takes the place of the file mnemonic for the purposes of internal processing.

HIST-IND. Indicates whether or not a History File is to be maintained by FM.

Ø = No History File
F = History File consisting of fixed fields only
A = History File consisting of all fields

XCHG-IND. Tells whether or not Data Exchange is needed.

Ø = No Data Exchange
Y = Exchange desired for this file.

FIND-CNT. Contains the number of file indexes (FINDS) associated with the file. FIND-CNT may not exceed 25 (future use).

CTL-LEN. Contains the number of characters in the record control group. CTL-LEN may not exceed 30.

PSC-LR. Contains the number of entries in each of the eleven logical records of the FFT. PSC-LR(1) always has the value 1.

(2) FFT Logical Record 2.

(a) Data Statements:

```
01  FS-DD.
    02  FFT-LR2.
      03  LR2-DATA OCCURS 299 TIMES.
        04  RET-SUBRT.
          05  FILLER     PICTURE IS XXXX.
          05  DATA-TYPE  PICTURE IS X.
        04  FIELD-SIZE-2 PICTURE IS 9999.
        04  REL-HOP      PICTURE IS 9999.
        04  SET-ID-2     PICTURE IS S99.
        04  FIELD-TYPE   PICTURE IS 9.
        04  MNEMONICS-2
          05  MNM-3      PICTURE IS XXX.
          05  MNM-2      PICTURE IS XX.
```

(b) Description: FFT-LR2 provides detailed format regarding the fields and groups within the records.

RET-SUBRT contains the name of the conversion routine for a Retrieval Search Parameter. The last character DATA-TYPE specifies the nature of the data field contents.

0 = Variable field (no subroutine)
1 = Alpha field (no subroutine)
2 = Numeric field (no subroutine)
3 = Signed numeric field (no subroutine)
S = Right-most character of the conversion subroutine (alpha field assumed)
9 = System generated field

FIELD-SIZE-2. Contains the character size of the field group. Since variable sets are variable in length, their field lengths will be described as 9999.

REL-HOP. For fixed fields or groups, this element locates their position relative to the first character in the data record. For periodic fields or groups, REL-HOP locates their position relative to the first character of the periodic subset in which they are included. For subset IDs fields (PSSnn) and variable sets, REL-HOP will always be ØØØØ.

SET-ID-2. Contains the number and type of the set in which this field is included. Zero (actually +ØØ) identifies fixed fields. Negative numbers indicate periodic sets. Positive numbers indicate variable sets. For example:

+ØØ = Fixed Set
-Ø1 = Periodic Set 1
-Ø2 = Periodic Set 2
+Ø1 = Variable Set 1
+Ø2 = Variable Set 2

FIELD-TYPE. Is a one-character code, indicating the kind of field within a data record.

1 = Control field or group
2 = Fixed field or group
3 = Periodic field or group
4 = Variable Set
5 = Periodic subset ID
6 = Periodic Set control field
7 = Variable Set control field

MNEMONIC-2. Contains a five-character field or group name which frequently is used to find applicable entries in the table.

(3) FFT Logical Record 3.

(a) Data Statements:

```
01  FS-DD.
    02  FFT-LR3.
      03  LR3-DATA OCCURS 50 TIMES.
        04  SETCTL-HOP  PICTURE IS 9999.
        04  SET-LENGTH  PICTURE IS 9999.
        04  SET-ID-3    PICTURE IS S99.
```

(b) Description: FFT-LR3 contains information pertinent to the sets defined for a MIDMS file. The sequence of entries will be the fixed set, followed by all periodic sets in sequence, followed by all variable sets in sequence.

SETCTL-HOP. Contains the relative high order position within the data record of the set control field (PSCnn or VSCnn) which pertains to the particular set. For the fixed set, SETCTL-HOP will contain ØØØØ.

SET-LENGTH. Contains the size of the fixed set or periodic subset in the file. For a variable set, the value of SET-LENGTH is ØØØØ.

SET-ID-3. Contains the number and type of the set being described. Zero (actually +ØØ) identifies the fixed set. Negative numbers indicate periodic sets. Positive numbers indicate variable sets. See the description of SET-ID-2 in FFT LR2 for an example.

(4) FFT Logical Records 4 and 5.

(a) Data Statements: (FFT-LR4 and FFT-LR5 for future use)

```
01  FS-DD.
    02  FFT-LR4.
      03  LR4-DATA OCCURS 25 TIMES.
        04  FIELD-SIZE-4   PICTURE IS 9999.
        04  FIELD-HOP      PICTURE IS 9999.
        04  SUBRT-NAME     PICTURE IS X(5).
        04  SET-ID-4       PICTURE IS S99.
        04  FIND-HOP       PICTURE IS 999.
        04  FIND-LENGTH    PICTURE IS 9999.
01  FS-DD.
    02  FFT-LR5.
      03  LR5-DATA OCCURS 75 TIMES.
        04  TYPE-ID        PICTURE IS X.
        04  LR4-ENTRY      PICTURE IS 99.
        04  FIND-ID        PICTURE IS 99.
        04  MNEMONIC-5     PICTURE IS X(5).
```

(b) Description: FFT-LR4 and FFT-LR5 contains information required to support the file index (FIND) capability of MIDMS. They describe the fields/groups in the file that affect FINDs. These logical records will be referenced by the RETRIEVAL program to process query terms against the FINDs and by File Maintenance which will build the FINDs. (They are not currently used.)

(5) FFT Logical Record 6.

(a) Data Statements: (FFT-LR6 not used at present)

```
01  FS-DD.
    02  FFT-LR6.
      03  LR6-DATA OCCURS 99 TIMES.
        04  OPR-ID    PICTURE IS XXXX.
        04  MNEMONIC-6 PICTURE IS X(5).
```

(6) FFT Logical Record 7.

(a) Data Statements:

```
01  FS-DD.
    02  FFT-LR7.
      03  LR-DATA OCCURS 299 TIMES.
        04  OP-SUB.
          05  EDIT-LOC      PICTURE IS 9999.
          05  OP-ID         PICTURE IS X.
        04  REL-SIZE        PICTURE IS S999.
        04  OUTPUT-SIZE     PICTURE IS 9999.
        04  LAB-LOC         PICTURE IS 9999.
```

(b) Description: FFT-LR7 contains output information for each field/group in the data file.

OP-SUB. Contains three kinds of information. When OP-ID is blank, all of OP-SUB consists of spaces and indicates that no conversion is needed for Output. When OP-ID has the value "S," OP-SUB contains the name of a conversion subroutine to be used for outputting the field. When OP-ID has the value "E," Output editing is specified and EDIT-LOC contains the subscript of the LR9 character starting the edit information for this field. Since EDIT-LOC has a numeric specification, it can only be referenced when OP-ID is "E," OP-SUB must be used whenever OP-ID is "Ø" or "S" to avoid the problem of alphanumeric characters in a numeric field.

REL-SIZE. Specifies the size of the Output extract label relative to the output size of the field/group which it references.

OUTPUT-SIZE. Specifies the actual output size of the field/group. When subroutine conversion is specified, the output size of the subroutine is used. For editing, the size of the edit literal is used. For all other fields, OUTPUT-SIZE is the same as FIELD-SIZE-2 in LR2.

LR2-ENTRY. Contains a pointer to the FIELD/GROUP definition within LR2 in order to obtain the field's mnemonic.

LAB-LOC. Is the subscript of the LR8 character at the high order position of the OUTPUT extract label for this field. For fields with no extract labels, LAB-LOC has a zero value. Checks for LR8 overflow.

(7) FFT Logical Record 8.

(a) Data Statements:

```
01  FS-DD.
    02  FFT-LR8.
      03  LR8-DATA OCCURS 3000 TIMES.
        04  LR-8 CHAR       PICTURE IS X.
```

(b) Description: FFT-LR8 contains all Output extract labels for the fields in the file. It is defined as 3000 individual characters because COBOL cannot directly handle variable length items. The LAB-LOC field of LR7 indicates the beginning of the label for a particular

data field/group. The length of the label is computed as the sum of REL-SIZE and OUTPUT-SIZE, also in LR7.

(8) FFT Logical Record 9.

(a) Data Statements:

```
01  FS-DD.
    02  FFT-LR9.
      03  LR9-DATA OCCURS 3000 TIMES.
        04  LR9-CHAR          PICTURE IS X.
    02  SAMPLE-LR9
      03  EDIT-SIZE           PICTURE IS 999.
      03  DATA-SIZE           PICTURE IS 999.
      03  EDIT-WORD           PICTURE IS X(20).
      03  FILLER              PICTURE IS X(2974).
```

(b) Description: FFT-LR9 contains all edit information for the fields in this file. It is defined as 3000 individual characters because COBOL cannot directly handle variable length items. The EDIT-LOC field of LR7 indicates the beginning of the edit information (the first character of EDIT-SIZE) for the particular field whenever OP-ID contains an "E." The edit data characters are associated in the manner described in SAMPLE-LR9. The names EDIT-SIZE, DATA-SIZE, and EDIT-WORD cannot be used by any program. The definitions are used only as documentation of the format of an item in LR9.

EDIT-SIZE. Contains the number of characters in the edit literal. It may not exceed 132 characters.

DATA-SIZE. Contains the maximum number of characters in a data field to be edited by the edit literal. Its value is derived from the number of blanks and zeros in EDIT-WORD.

EDIT-WORD. Contains the actual literal. Its length is indicated by EDIT-SIZE. The PICTURE length of 20 is for an example only and is not necessarily the length of any particular EDIT-WORD.

(9) FFT Logical Record 10.

(a) Data Statements:

```
01  FS-DD.
    02  FFT-LR10.
      03  LR10-DATA OCCURS 500 TIMES.
        04  LEVEL-NO      PICTURE 99.
        04  ITEM-TYPE     PICTURE X.
        04  PURPOSE       PICTURE X.
        04  LR2-ENTRY     PICTURE 999.
```

(b) Description: FFT-LR10 contains information necessary to build COBOL Data Division statements describing the data file.

LEVEL-NO. Is a two-digit number containing the Data Division level number associated with the item.

ITEM-TYPE. Is a one-character code, indicating the kind of Data Division statement to be generated. Possible values for ITEM-TYPE are:

F - field
G - group
P - PSCnn or VSCnn
C - control
S - set (fixed or periodic)
V - vset

PURPOSE. Is a one-character code, specifying the variations in the Data Division statement to be generated. Possible values for PURPOSE are:

D - define
N - define of a numeric group
S - special define (used for overlapping groups)
R - redefine (used for overlapping groups)
F - filler

LR2-ENTRY. Is a three-digit field, containing the subscript of the LR2 entry associated with this item. The entry in LR2 is used to obtain the field/group mnemonic and PICTURE information. LR2-ENTRY is zero for "C" or "S" item types because they refer to group definitions with standard MIDMS-generated names.

(10) FFT Logical Record 11.

(a) Data Statements:

```
01  FS-DD.
    02  FFT-LR11.
      03  LR11-DATA  OCCURS 303 TIMES.
        04  MNEMONIC-11   PICTURE X(5).
        04  LR2-LR7-ENTRY PICTURE 999.
```

(b) Description: FFT-LR11 contains the sorted field mnemonics of LR2 and a pointer to the LR2 and LR7 item, which contains information about the field so that FM, RT and OP can locate field mnemonics by a modified binary search technique and obtain the FIELD's associated LR2 or LR7 entry.

MNEMONIC-11. Contains a field name. For the first two entries in LR11, MNEMONIC-11 has the value LOW-VALUE. MNEMONIC-11 has the value HIGH-VALUE for the last two entries in LR11. The modified binary search requires these special values to operate properly.

LR2-LR7-ENTRY. Contains the subscript of the entry in LR2 and LR7, where MNEMONIC-2 or MNEMONIC-7 matches MNEMONIC-11. (LR2 and LR7 always have the same number of elements and corresponding entries have the same mnemonic.) When MNEMONIC-11 has the value LOW-VALUE or HIGH-VALUE, LR2-LR7-ENTRY contains zero.

b. FS Tables and Work Areas.

(1) ALL-INFO AREA.

(a) Data Statements

```
01  FS-DD.
    02  ALL-INFO.
      03  WORD-COUNT          PICTURE IS 99 USAGE COMPUTATIONAL.
      03  DATA-WORDS.
        04  DATA-ITEM OCCURS 50 TIMES.
          05  CHAR-COUNT      PICTURE IS 9 USAGE COMPUTATIONAL.
          05  DATA-WORD.
            06  DATA-CHAR OCCURS 9 TIMES PICTURE IS X.
      03  LITCOUNT   PICTURE IS S999 USAGE COMPUTATIONAL.
      03  LIT-DATA.
        04  LIT-CHAR OCCURS 132 TIMES  PICTURE IS X.
```

(b) Description: ALL-INFO is an area composed of two tables made from the data in the STORAGE-WORD Group in STORAGE-AREA. The FSSNX SUBROUTINE builds this area each time a statement is read, in order to aid in the analysis of the statement. There is one entry of DATA-ITEM for each valid word (a series of characters terminated by a comma).

WORD-COUNT. Contains the number of entries in DATA-WORDS.

CHAR-COUNT. Contains the length of the word.

DATA-WORD. Contains the characters of a word, as defined above, left-justified, in the 9-character area. For words longer than 9 characters only the leading characters are present, since the maximum valid word size is 9.

LITCOUNT. Contains the length of the literal in LIT-DATA. If no literal is present, the value of LITCOUNT is set at -999. If there is only one at-sign, the value is set at -1. Valid literals have lengths between 0 and 132 characters.

LIT-DATA. Contains the literal from an FS card, left-justified, in the 132-character area. Only the first 132 characters of a literal exceeding 132 characters are present.

(2) STORAGE-AREA.

(a) Data Statements:

```
01  FS-DD.
    02  STORAGE-AREA.
      03  THIRTEEN-FIRST.
        04  FILLER              PICTURE IS X.
        04  CARD-ID             PICTURE IS X(5).
        04  FILLER              PICTURE IS X.
        04  FS-CARD-TYPE        PICTURE IS X(5).
        04  FILLER              PICTURE IS X.
      03  STORAGE-WORD.
        04  STORAGE-CHAR OCCURS 236 TIMES PICTURE IS X.
      03  STORAGE-INFO  REDEFINES STORAGE-WORD.
        04  STORAGE-DATA OCCURS 4 TIMES PICTURE IS X(59).
```

(b) Description: STORAGE-AREA is an area composed of two parts which allow for the building of a single statement when continuation cards are used. The first part, THIRTEEN-FIRST, is used to identify the card. The second part, STORAGE-INFO, holds the continued card image.

CARD-ID. Contains the name field of the card.
FS-CARD-TYPE. Contains the statement identifier.
STORAGE-WORD. A 236-character area into which column 14 thru 72 of up to four cards (the original and three continuations) comprising an FS statement are placed.

(3) ERROR-INFO.

(a) Data Statements:

```
01  FS-DD.
    02  ERROR-INFO.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'FFT IN LIB.  IT WILL BE CHANGED          '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'NOT IN LIBRARY                           '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'EXTRA PARAMETERS ON STATEMENT IGNORED    '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'NEXT JOB CARD OUT OF ORDER               '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'HAS SAME ID AS SUBROUTINE                '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'SUBROUTINE TABLE OVERFLOW                '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'LR10 OVERFLOW                            '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'LR9 OVERFLOW                             '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'LR8 OVERFLOW                             '.
      03  FILLER                    PICTURE IX X(40)   VALUE IS
        'LR6 OVERFLOW                             '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'LR5 OVERFLOW                             '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'LR4 OVERFLOW                             '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'LR3 OVERFLOW                             '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'LR2 OVERFLOW                             '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'OUTPUT SIZE IN ERROR                     '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'THE 4TH CHAR OF SUBNAME MUST BE N OR T   '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'ILLEGAL SUB NAME                         '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'DOUBLY DEFINED SUBROUTINE ID             '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'DOUBLY DEFINED MNEMONIC                  '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'FIELD OUT OF SEQUENCE                    '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'UNDEFINED FIELD ID                       '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'MAX NUMBER OF FIELDS PER GROUP EXCEEDED  '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'EXCEEDS CAPACITY OF EDIT FIELD           '.
      03  FILLER                    PICTURE IS X(40)   VALUE IS
        'UNDEFINED INPUT FUNCTION                 '.
```

```
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'O/P OF I/P FUNCT. NOT EQUAL TO FLD SIZE   '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'FUNCTION REQUIRES SUBSCRIPT               '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'ILLEGAL SUBSCRIPT                         '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'MISSING ASTERISK                          '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'MORE THAN THREE CONTINUATION CARDS        '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'ILLEGAL FILE NAME                         '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'JOB NOT COMPLETED BECAUSE OF ERRORS       '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'ILLEGAL MNEMONIC                          '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'CARD MUST PRECEED THE FIELD-GROUP CARDS   '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'UNDEFINED SUBRT SPECIFIED FOR BASE        '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'FIELDS HAVE DIFFERENT SET IDS             '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'FIELDS HAVE DIFFERENT TYPE IDS            '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'ILLEGAL SUB BASE                          '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'SUBROUTINE OR TABLE DEFINED BUT NOT USED  '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'MISSING QUOTE ON EDIT LITERAL             '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'UNDEFINED OUTPUT FUNCTION                 '.
003  FILLER                    PICTURE IS X(40)   VALUE IS
  'UNDEFINED FIELD/GROUP ID ON OPR CARD      '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'FIND ID OUT OF SEQUENCE                   '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'COLUMN 4 OF CARD MUST CONTAIN B OR F      '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
   FUNCTION LENGTH CANNOT EXCEED 50          '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'ELEMENT FOR FIND BASE EXCEEDS 20 CHARS    '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'DOUBLY DEFINED EDIT FIELD ID              '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'SET ID OUT OF SEQUENCE                    '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'CNTL FLD MUST BE THE FIRST FLD DEFINED    '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'ILLEGAL CARD TYPE                         '.
03  FILLER                    PICTURE IS X(40)   VALUE IS
  'CARD ILLEGAL HERE                         '.
```

```
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'CREATE, CHANGE, OR TESTIT MODE REQUIRED   '.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'FILE ID NOT AVAILABLE IN FICON TABLE      '.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'INVALID OPTION                            '.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'INVALID LITERAL                           '.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'ENDFS CARD MISSING                        '.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'FIELD SIZE INVALID                        '.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'ILLEGAL SET ID                            '.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'SUBROUTINE USED FOR BOTH I/P AND O/P      '.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'CONTROL FIELD EXCEEDS 30 CHARACTERS       '.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'RECCT CANNOT BE GROUPED                   '.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'FIELD NOT IN LR10  - PROGRAM ERROR        '.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'INCORRECT GROUP TYPE  -  PROGRAM ERROR    '.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'LAST FIELD IS NOT LAST FIELD DEFINED      '.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      SPACES.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'THIS CARD FORMAT CANNOT BE FIRST IN SET   '.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'ASSUMED CONTROL FIELD                     '.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'ASSUMED FIXED FIELD                       '.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'ASSUMED PERIODIC SET                      '.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'ASSUMED ALPHA INPUT FUNCTION              '.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'ASSUMED BLANK OUTPUT FUNCTION             '.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'ASSUMED VSET SIZE, 100 CHARACTERS         '.
    03  FILLER                     PICTURE IS X(40)   VALUE IS
      'EXCEEDS MAXIMUM RECORD SIZE               '.
  02  ERR-DATA REDEFINES ERROR-INFO.
    03  ERR-MSG OCCURS 72 TIMES PICTURE IS X(40).
```

(b) Description: ERROR-INFO is a table containing all of the FS error diagnostics. Each FS subroutine references this table upon encountering an error. Reference to the ERROR-INFO Table is achieved by supplying ERR-CODE with the subscript number of the desired diagnostic. The subscript value is determined by simply counting the number of entries in ERROR-INFO. An indication of the ERR-MSG is provided by a NOTE statement whenever it is referenced. MAX-CODE holds the number of error diagnostics contained in the table, so that, if a new diagnostic is added to ERROR-INFO, the only modification to the existing program will be:

1. Changing the value of MAX-CODE.

2. Adding a filler to ERROR-INFO.

3. Changing the "OCCURS" clauses of ERR-MSG.

(4) JOB TYPE TABLE.

(a) Data Statements:

```
01  FS-DD.
    02  FSJOB-TYPE.
      03  FILLER              PICTURE IS X(5)     VALUE IS 'FSJOB'.
      03  FILLER              PICTURE IS X(5)     VALUE IS 'EDIT '.
      03  FILLER              PICTURE IS X(5)     VALUE IS 'SUB  '.
      03  FILLER              PICTURE IS X(5)     VALUE IS 'TAB  '.
      03  FILLER              PICTURE IS X(5)     VALUE IS 'FIELD'.
      03  FILLER              PICTURE IS X(5)     VALUE IS 'GROUP'.
      03  FILLER              PICTURE IS X(5)     VALUE IS 'VSET '.
      03  FILLER              PICTURE IS X(5)     VALUE IS 'OPR  '.
      03  FILLER              PICTURE IS X(5)     VALUE IS 'FIND '.
      03  FILLER              PICTURE IS X(5)     VALUE IS 'ENDFS'.
    02  JOB-TYPE-TBL REDEFINES FSJOB-TYPE.
      03  TYPE-CARD OCCURS 10 TIMES  PICTURE IS X(5).
```

(b) Description: FSJOB-TYPE is a table of all the control cards acceptable to File Structure. The FS supervisor identifies each card read in by scanning the FSJOB-TYPE Table. TYPE-CNT holds the number of card types contained in the Table, so that, if a new card type is later added to FS, the only modification to the existing program will be:

1. Changing the value of TYPE-CNT.

2. Adding a filler to FSJOB-TYPE.

3. Changing the "OCCURS" clause of TYPE-CARD.

4. Adding another "Go To" label to the TYPE-ROUTINE paragraph in the supervisor.

(5) FS-JOB-OPTIONS TABLE.

(a) Data Statements:

```
01  FS-DD.
  02  FS--JOB-OPTIONS.
    03  FILLER                    PICTURE IS XXXX   VALUE IS 'DATE'.
    03  FILLER                    PICTURE IS XXXX   VALUE IS 'HFIX'.
    03  FILLER                    PICTURE IS XXXX   VALUE IS 'HALL'.
    03  FILLER                    PICTURE IS XXXX   VALUE IS 'XCHG'.
    03  FILLER                    PICTURE IS XXXX   VALUE IS 'LST1'.
    03  FILLER                    PICTURE IS XXXX   VALUE IS 'LST2'.
    03  FILLER                    PICTURE IS XXXX   VALUE IS 'LST3'.
    03  FILLER                    PICTURE IS XXXX   VALUE IS 'LST4'.
    03  FILLER                    PICTURE IS XXXX   VALUE IS 'LST5'.
    03  FILLER                    PICTURE IS XXXX   VALUE IS 'LST6'.
    03  FILLER                    PICTURE IS XXXX   VALUE IS 'LST7'.
    03  FILLER                    PICTURE IS XXXX   VALUE IS 'LST8'.
    03  FILLER                    PICTURE IS XXXX   VALUE IS 'LST9'.
    03  FILLER                    PICTURE IS XXXX   VALUE IS 'NODK'.
  02  JOB-OPTIONS REDEFINES FS-JOB-OPTIONS.
    03  JOB-OPTION OCCURS 15 TIMES  PICTURE IS XXXX.
```

(b) Description: FS-JOB-OPTION is a table of all the options which may be specified on a JOB card. OPTION-CNT holds the number of options contained in the table, so that, if a new option is later added to the JOB card, the only modification to the existing program will be:

1. Changing the value of OPTION-CNT.

2. Adding a filler to FS-JOB-OPTIONS.

3. Changing the "OCCURS" clause of the JOB-OPTIONS.

4. Adding another "Go To" label to the OPTION-LU paragraph in FSJOB subroutine.

The existing FS-JOB-OPTION Table contains the following options:

DATE. Allows for the management date fields, CREDT and CHGDT, to be included in the FFT. The analyst cannot specify them on the source card.

HFIX. Causes a History Tape containing fixed fields of obsolete records to be maintained by FM.

HALL. Causes a History Tape containing entire obsolete data file records to be maintained.

XCHG. An indicator which is placed in the FFT if a Data Exchange Tape is to be written, whenever the file defined by the FFT is updated.

LSTx. Specifies the number of extra listings to be produced by FS for a valid FFT. The last digit specifies the number of listings desired, in addition to the one that is always produced.
NODK. Has no current purpose, other than compatibility with Mark III. It is acceptable, but otherwise ignored.

(6) EDIT-TABLE.

(a) Data Statements:

```
01  FS-DD.
    02  EDTBL.
      03  EDIT-INFO OCCURS 100 TIMES.
        04  EDIT-NAME              PICTURE IS X(5).
        04  EDIT-USE               PICTURE IS X.
        04  EDIT-LEN               PICTURE IS 999.
        04  DATA-LEN               PICTURE IS 999.
        04  EDIT-POINTER           PICTURE IS 9999.
    02  EDIT-WORDS.
      03  EDIT-CHAR OCCURS 3000 TIMES  PICTURE IS X.
```

(b) Description: EDTBL and EDIT-WORDS describe EDIT fields so that they can be referenced on FIELD or GROUP cards. There is one entry of EDIT-INFO for each valid EDIT card.

EDIT-CNT. Contains the number of entries in EDTBL.
EDIT-NAME. Contains the name that is specified on the EDIT-CARD.
EDIT-USE. Indicates whether or not an EDIT has been referenced on a FIELD/GROUP card. (A value of space means not used, a "U" means used.)
EDIT-LEN. Contains the length of the EDIT literal.
DATA-LEN. Contains the maximum size data field which can be edited by the EDIT literal. It is determined by counting the number of blanks and zeroes.
EDIT-POINTER. Contains the subscript of the HOP within FFT-LR9 when EDIT-USE has the value of "U."
EDIT-CHAR. Contains the number of characters which have been used in the EDIT-WORDS area.

(7) SUBROUTINE AND TABLE TABLE.

(a) Data Statements:

```
01  FS-DD.
    02  SUB-TAB.
      03  SUB-ITEM OCCURS 100 TIMES.
        04  SUB-NAME            PICTURE IS X(5).
        04  I-O-SUB             PICTURE IS X.
        04  LEN-CNT             PICTURE IS 9.
        04  OUT-POINT           PICTURE IS 9999.
    02  OUT-TAB.
      03  OUT-ITEM OCCURS 300 TIMES  PICTURE IS 999.
```

(b) Description: SUB-TAB and OUT-TAB describe subroutines or tables, so that they can be referenced on FIELD/GROUP cards. There is one entry of SUB-ITEM for each valid SUB-TAB card. There is an OUT-ITEM entry for each OP size of every subroutine or table defined in the FFT.

OUT-CNT. Contains the number of elements in OUT-TAB.
SUB-CNT. Contains the number of entries in SUB-TAB.
SUB-NAME. Contains the name that is specified on the SUB-TAB card.
I-O-SUB. Indicates whether or not a subroutine has been referenced and the use of the subroutine. (Blank means not used, "I" means Input Conversion Subroutine and "F" means Find Base Generation Subroutine.)
LEN-CNT. Contains the number of OP size defined for the subroutine.
OUT-POINT. Contains the subscript of the OUT-ITEM for the first or only OP size of the subroutine. Any other OP sizes for the subroutine are contained in subsequent OUT-TAB entries in the order specified on the SUB/TAB card.

c. COBOL Generated Data Division. The File Structuring (FS) subsystem of MIDMS produces a series of COBOL Data Division statements describing the hierarchy of fields and groups defined for a data file. These data descriptions are eventually placed in MIDMS-generated COBOL programs so that user-defined field and groups names can be referenced. The method which FS uses to produce the COBOL data descriptions from the FS statements consists of two separate algorithms which together permit the proper statements to be generated. One assigns level numbers, the other determines the sequencing of the lines in the Data Division. The various rules are used each time a new FS card is processed.

(1) Basic Rules for Assigning Level Numbers. FS uses the following basic rules to assign level numbers:

(a) Originally, each field gets a level number of 49 to allow the most possible levels in the field/group hierarchy (fields within groups within groups).

(b) Each group is given a level one less (numerically) than the lowest level of any field or group within the new group.

(c) All fields/groups within a group are assigned the level number equal to the lowest level of any of those fields/groups if they have not previously been included in a group.

For example, suppose the following statements are input to FS:

```
1  AA FIELD
2  BB FIELD
3  CC GROUP AA,BB
4  DD FIELD
5  EE GROUP CC,DD
```

As cards (1) and (2) are processed, fields AA and BB are given level 49. Group CC gets level 48 because the minimum level number of its component fields/groups is 49. The level numbers for field AA and BB remain at 49 since both already have the "lowest level" mentioned in paragraph 1.c.(1)(c). In accordance with paragraph 1.c.(1)(a), field DD is initially given level 49. Since group EE contains a field and a group at different levels, the level of DD is changed to 48 (the lowest level number) and EE is assigned 47.

(2) Basic Rules for Determining Sequence. As cards are processed by FS, one entry is generally added to the end of a table containing level numbers and an indication of the kind of Data Division statements to be produced. In addition, a special sequence number field is attached to each table entry so that the entries can be resequenced at the completion of the data description. The table entries are not physically reordered as processing takes place; only the sequence number field is actually changed until after the last field or group card is interpreted.

(a) The following are the basic rules used by FS to determine the value of the sequence number field and eventually the order of the Data Division statements:

1. A field is initially given the next available sequence number.

2. A group is given the lowest sequence number found in the table after the entry for the first field/group in the specification list.

3. The sequence number of all table entries with numbers greater than or equal to the sequence of the group is increased by 1.

(b) Continuing with the example given for assigning level numbers, figure 1-1 entries 1 - 5 indicate changes in sequencing for the fields and groups. On the left is a list of the field/group specifications. The top of the table indicates when the sequence number in the table is changed.

(c) Note that during the processing of group CC, the sequence number of CC comes from the sequence of AA and the sequence of AA and BB are each adjusted by 1. Since CC is the first field/group for EE, EE gets the sequence of CC and all other sequence numbers are increased.

(3) Additional Rules. The basic rules taken into account the vast majority of formats specified in MIDMS, i.e. fields within groups within larger groups. The basic rules do not cover situations where overlapping groups are defined. Overlapping groups are two groups containing the same fields without one of the groups containing the other. Specifically, groups will be considered overlapping whenever the same field/group appears in the field/group list of two or more groups. Since FS statements are not saved after they are processed, FS determines that overlap is present when either:

(a) The sequence number for any table entry after the first field or group in the list is lower than the sequence number for the first field or group, or

(b) An overlapping group entry is located in the series of table entries after the entry for the first field/group in the new group.

(c) The extended rules required as a result of allowing overlapping groups are:

1. When the new group overlaps a previous group, the table entry for the group is considered a "special group" item The level number and sequence number is the same as described in paragraphs 1.c.(1)(b) and 1.c.(2)(a)2.

2. Another table entry is built for a "group redefine" item. Its level number is the same as described in paragraphs 1.c.(1)(b) and 1.c.(2)(a)2.

3. For each field in the table with a sequence number greater than the sequence of the new group and smaller than the sequence of the first field/group on the group card, a new table entry is produced for "filler field" to represent the space in the special group before the definition of the group. Each of these entries is given level 49 and the next available sequence number.

4. A table entry for the group is built after the first field/group is encountered by paragraph 1.c.(3)(c)3. The entry specifies a "field define" item instead of the usual "group define". The level number of the entry is 49 and the next available sequence number is used.

(d) Figure 1-2 entry (6) shows field FF and group GG being defined. Since both EE and GG include DD in their field/group lists, a special group item (indicated by -S) is placed in the table followed by a group redefine (-R). Then filler fields for AA and BB (-F suffix) are generated as is a field define for group GG.

(e) As a result of examining column (7) of the table in figure 1-2, COBOL Data Division statements are produced in the sequence shown in figure 1-3.

				(1)	(2)	(3)	(4)	(5)	(6)	(7)
	(1)	AA FIELD	AA	1		2		3		4
	(2)	BB FIELD	BB		2	3		4		5
BASIC	(3)	CC GROUP AA,BB	CC			1		2		3
	(4)	DD FIELD	DD				4	5		6
	(5)	EE GROUP CC,DD	EE					1		2
	(6)	FF FIELD	FF						6	7
	(7)	GG GROUP DD,FF	GG-S							1
EXTENDED			GG-R							8
			AA-F							9
			BB-F							10
			GG							11

FIGURE 1-2. FIELD/GROUP SEQUENCING

```
46 GG-S.
  47 EE.
    48 CC.
      49 AA  _______.
      49 BB  _______.
    48 DD    _____.
  47 FF      _______.
46 GG-R REDEFINES GG-S.
      49 FILLER _____.
      49 FILLER _____.
      49 GG     _____.
```

FIGURE 1-3. GENERATED DATA DEFINITION

3. FS SUBPROGRAM IDENTIFICATION AND DESCRIPTION.

a. FSX (SUPERVISOR).

(1) Summary. FSX is the supervisor subroutine for the File Structuring module; it initializes various switches and work areas. FSX calls the DATESUB subroutines in order to obtain the system date for MIDMS use in generated reports. RSX also reads all cards and their continuations into a work area. The card scan subroutine (FSSNX) is called to perform preliminary processing on each card read. Control is returned to the supervisor which then determines the card type. Once the card type is determined, the appropriate subroutine is called to perform detail analysis of the card in question. After a certain amount of processing is completed by one of the subroutines, control is then given back to FSX. FSX then determines whether the PRINT subroutine (FSPRX) should be called to print a message and whether it should return to the processing subroutine or whether the next card should be read.

(a) Function. Controls the execution of all FS subroutines.

(b) Calling Sequence.

```
CALL  'DATESUB'  USING  HDR-DATE
CALL  'FSSN'     USING  FS-DD
CALL  'FSDD'     USING  FS-DD
CALL  'FSPR'     USING  FS-DD
CALL  'FSJB'     USING  FS-DD
CALL  'FSED'     USING  FS-DD
CALL  'FSSB'     USING  FS-DD
CALL  'FSFL'     USING  FS-DD
```

```
CALL  'FSGR'     USING  FS-DD
CALL  'FSEN'     USING  FS-DD
CALL  'FSML'     USING  FS-DD
CALL  'FSER'     USING  FS-DD
```

(2) Description.

PROCEDURE DIVISION. Initializes switches and storage area, makes I/O areas available for use and obtains date from Operating System.

READ-AFTER-JOB. Reinitializes work areas and switches for new input card.

BUILD-AREA. Reads input card into temporary hold area.

COME-BACK. Determines if card is a COMMENT or ENDFS card.

NEW-CARD. The first thirteen columns of the card image just read in and which is now located in a temporary hold area is compared to the first thirteen columns of the preceding card which is held in a storage area called ALL-INFO to determine if the card is to be continued. If continuation is required, a multi-card statement is built in ALL-INFO area. If there is no continuation, control is passed to FSSN to generate the word tables.

CALL-FSSN THRU CALL2-FSSN. Calls subroutine which examines the storage area for the presence of words. A word is indicated by a string of characters separated by a comma.

ERROR-PRINT. Prints error message which states that there are too many continuations.

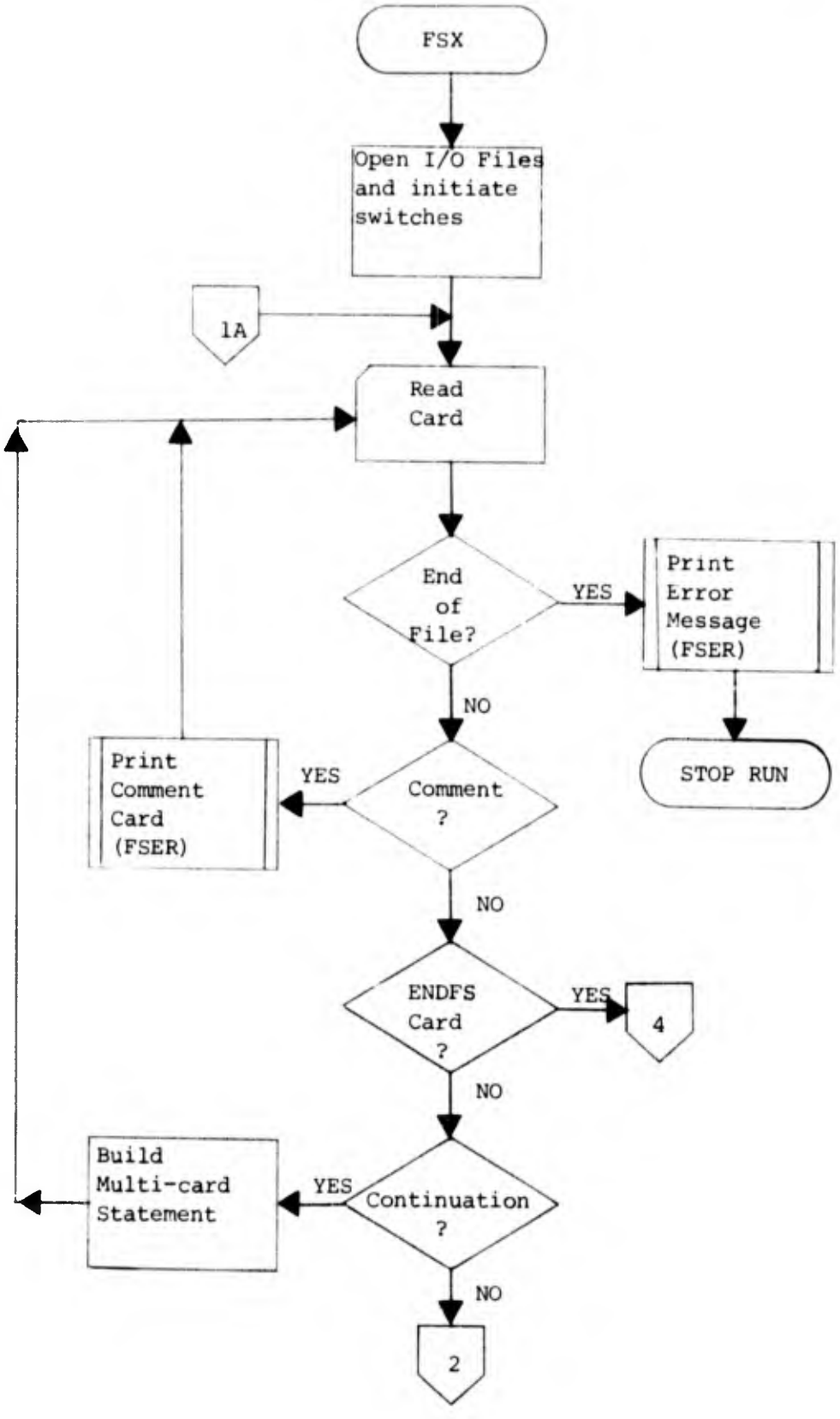
END-OF-CARDS. Prints error message which states that the ENDFS card is missing.

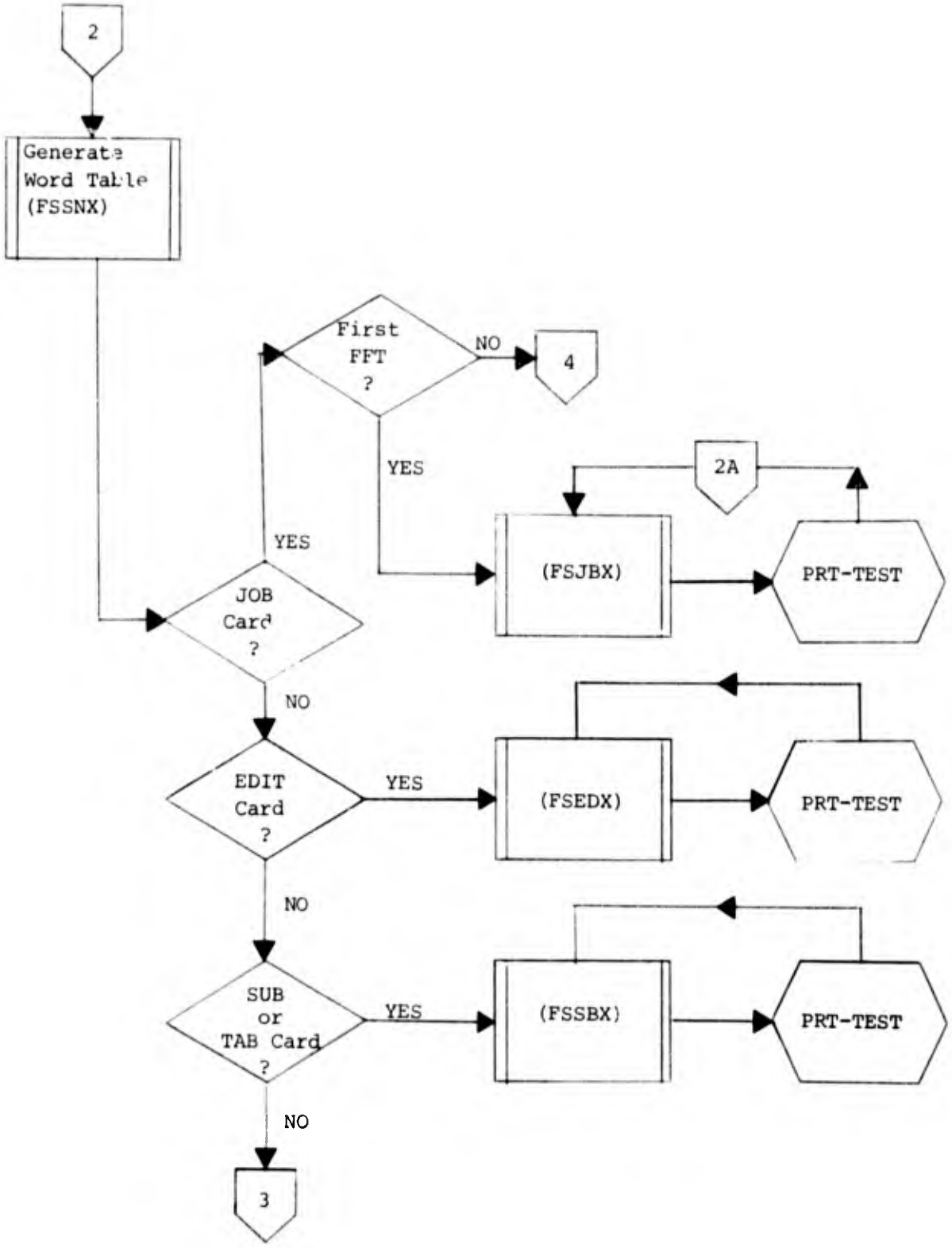
JOB-ERROR. Prints error message which states that the job did not run to completion because of diagnostic errors.

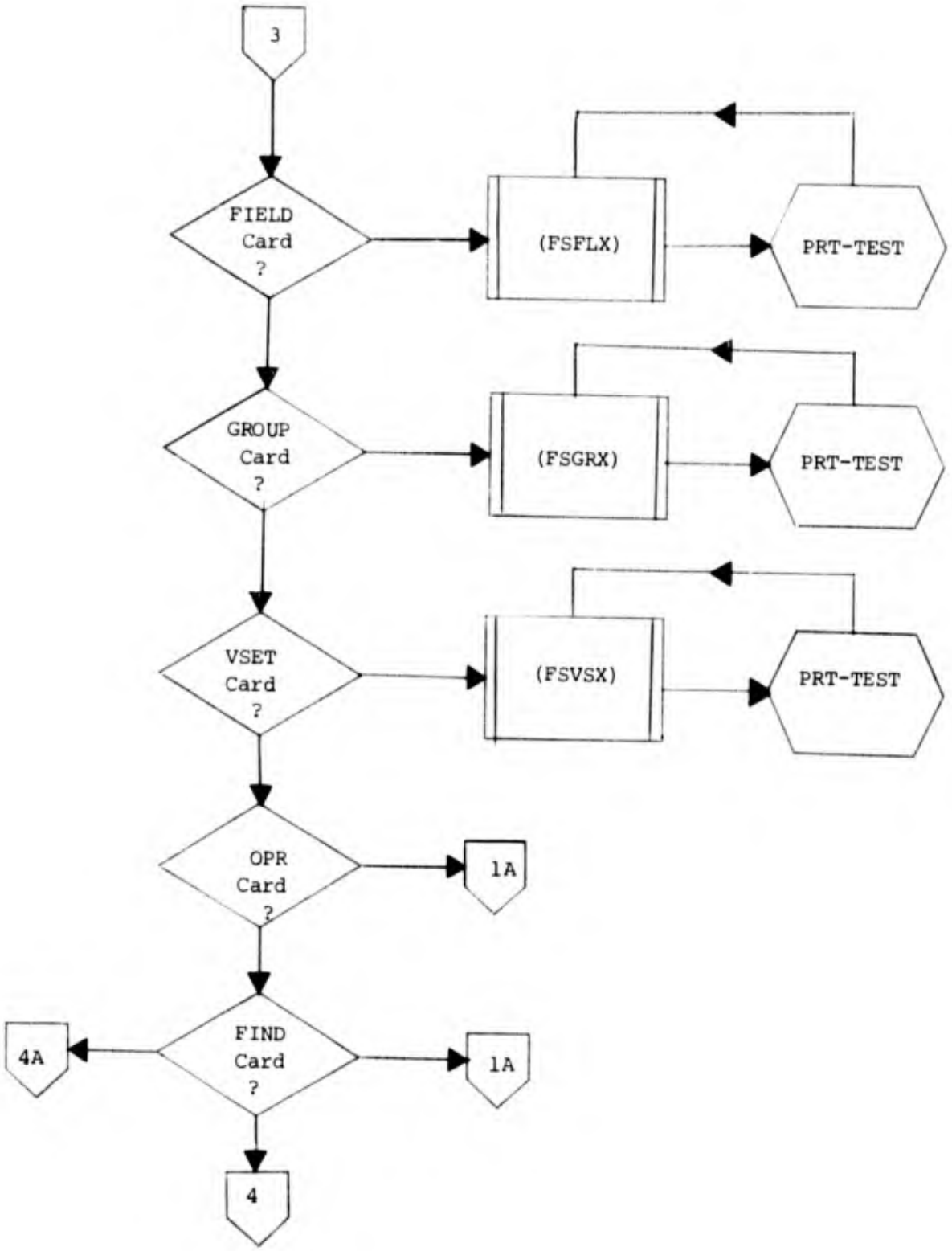
STOPIT. Stops the execution of the File Structuring program.

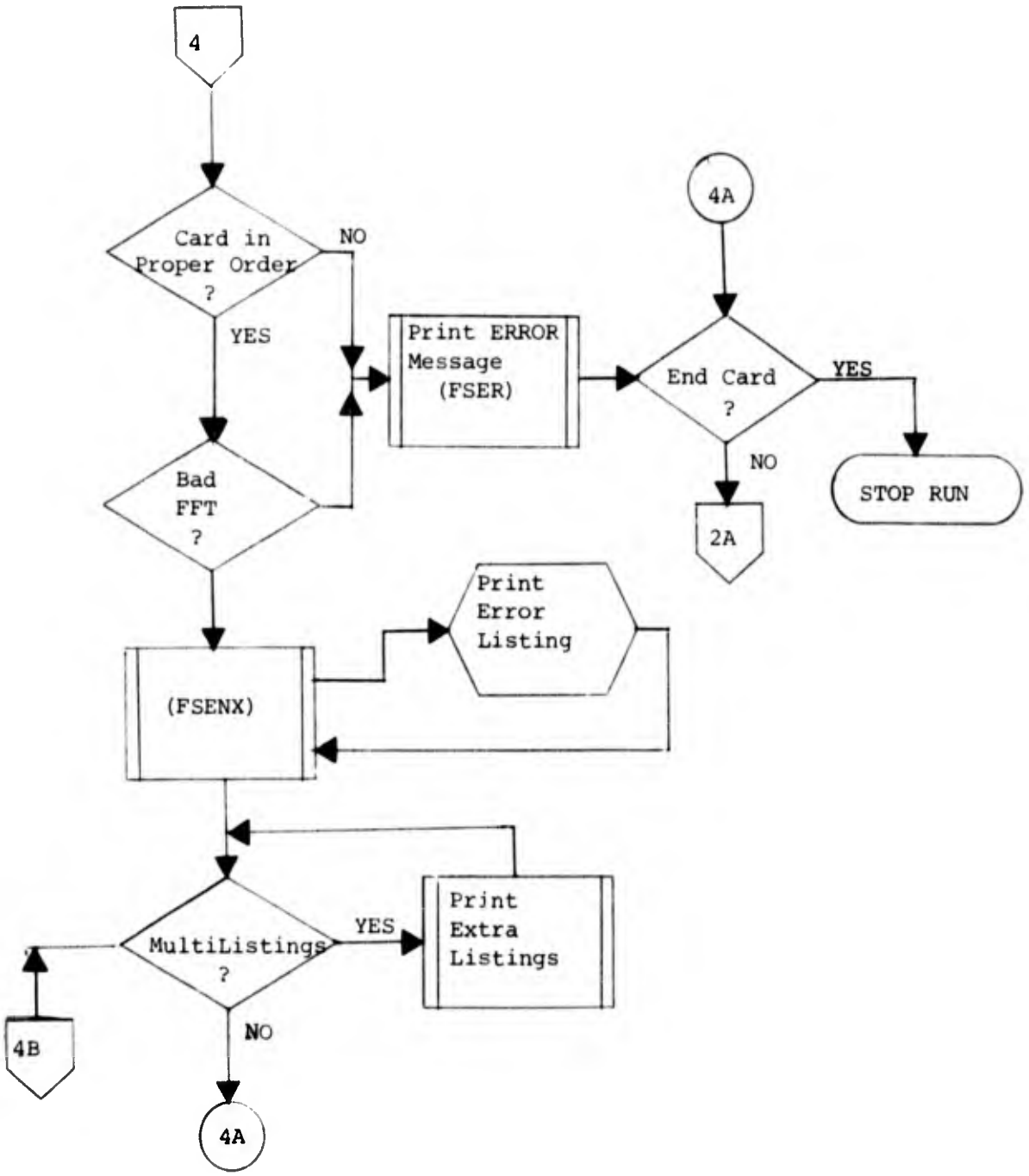
ANALYZE/WRONG-CARD. Analyzes card type and branches to the appropriate subroutine. Also prints listings, multiple copies, errors, and advisory messages. Determines if subroutines are to be re-entered to continue processing of ALL-INFO or to return and process a new card.

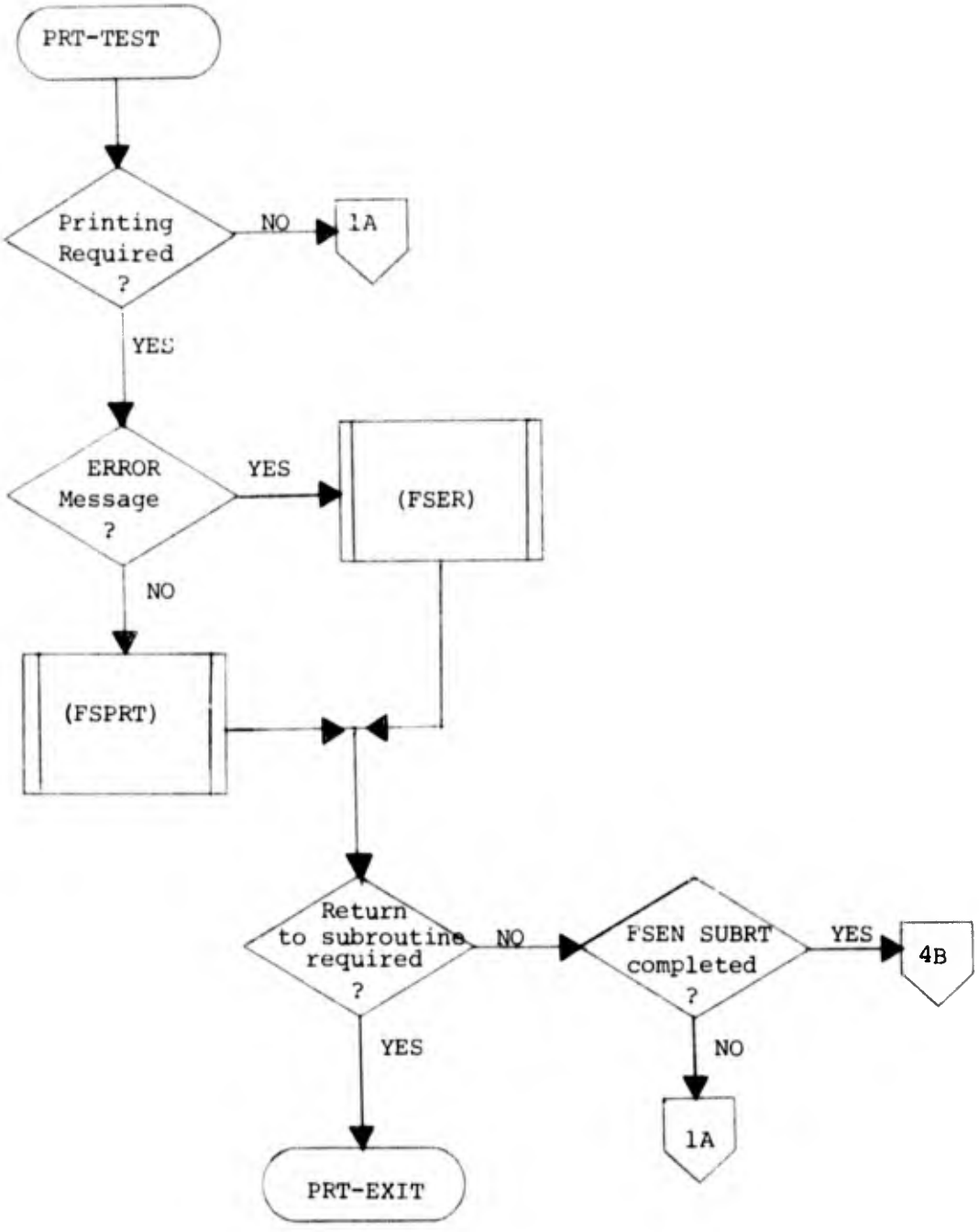
(3) FSX Flowchart.


FSX
Open I/O Files and initiate switches
1A
Read Card
End of File?
YES
Print Error Message (FSER)
NO
STOP RUN
Print Comment Card (FSER)
YES
Comment ?
NO
ENDFS Card ?
YES
4
NO
Build Multi-card Statement
YES
Continuation ?
NO
2


2
Generate
Word Table
(FSSNX)
First
FFT
?
NO
4
YES
2A
YES
(FSJBX)
PRT-TEST
JOB
Card
?
NO
EDIT
Card
?
YES
(FSEDX)
PRT-TEST
NO
SUB
or
TAB Card
?
YES
(FSSBX)
PRT-TEST
NO
3


3
FIELD Card ?
(FSFLX)
PRT-TEST
GROUP Card ?
(FSGRX)
PRT-TEST
VSET Card ?
(FSVSX)
PRT-TEST
OPR Card ?
1A
FIND Card ?
4A
1A
4


4
Card in Proper Order ?
NO
YES
Print ERROR Message (FSER)
4A
End Card ?
YES
NO
STOP RUN
2A
Bad FFT ?
Print Error Listing
(FSENX)
MultiListings ?
YES
Print Extra Listings
NO
4B
4A


PRT-TEST
Printing Required ?
NO
1A
YES
ERROR Message ?
YES
(FSER)
NO
(FSPRT)
Return to subroutine required ?
NO
FSEN SUBRT completed ?
YES
4B
YES
NO
1A
PRT-EXIT

b. FSSNX (WORD SCAN).

(1) Summary. FSSNX is the card scan subroutine. This subroutine performs initial processing of every card by scanning forward, looking for a comma. If a comma is found, the preceding string of characters is placed in a word table. The scan is continued until a period is found which indicates that the rest of the card is treated as comments or the scan is continued until an at-sign (@) is found which indicates a literal is on the card. The literal is validated and placed in a literal table. Certain switches are set and used later by certain subroutines.

(a) Function. Builds the word table.

(b) Calling Sequence.

```
ENTRY 'FSSN' USING FS-DD
```

(2) Description. Procedure Division receives control from calling sequence.

CHECKIT/REVERS-SCAN. Locates last character on card. A reverse scan is performed on the card SAVE-AREA (the original and up to three continuations are stored there, if this is the situation).

BUILDIT/LEAVE-PERFORM. A forward scan is performed on the card SAVE AREA to determine the first character of the word. The first forward scan is continued until a comma is found. This comma indicates that a word is completed. The word and its length is moved to the word table. If the word is longer than nine positions, a length of nine is entered into the word table. With each word encountered, a word count is incremented by one. This process is continued until either an at-sign is encountered, which indicates the beginning of an extract label, or a period is encountered which indicates the remaining positions will be treated as comments, or the last character is encountered, which indicates there will be no extract label or comments on the card image.

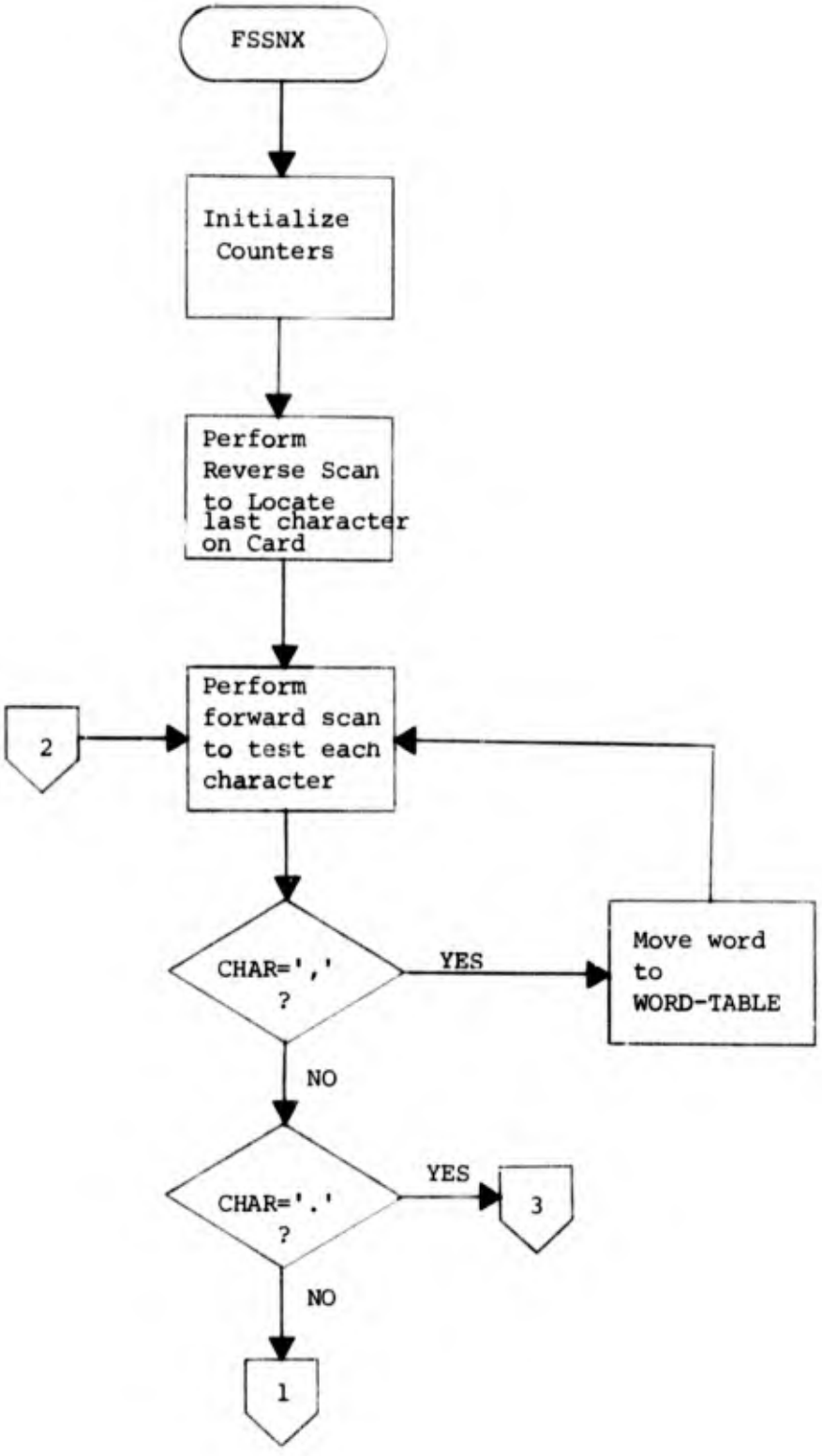
FOUND-AT-SIGN/LIT-SCAN. When an at-sign is encountered, the building of words is terminated and the building of a literal is begun. The beginning and terminating at-signs (@) are located.

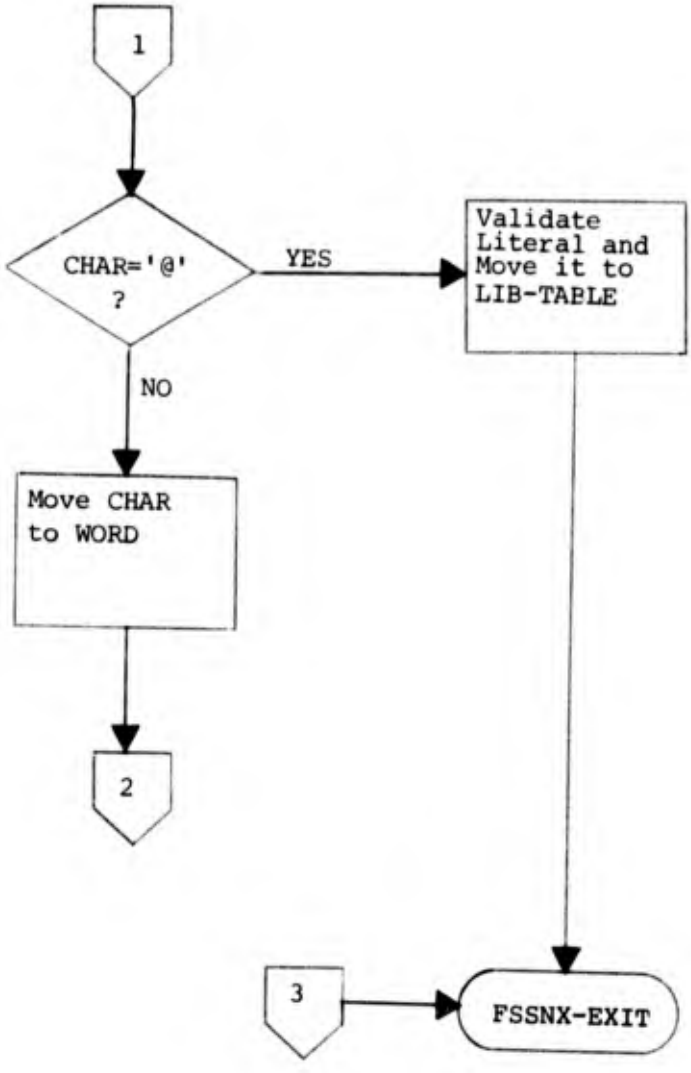
LIT-MOVE/NO-LIT. The number of characters between two at-signs is put into a count area and the literal itself is put in a literal hold area. It also checks to see if the maximum size of 132 for a literal has been exceeded. If only one at-sign is found, minus one (-1) is entered into the literal count area. If the two at-signs are adjacent, zero is moved to the literal count area. If more than 132 characters are found between the two at-signs, -999 is entered into the literal count area and only the first 132 characters are moved into the literal hold-areas. If none of the

above, the actual number of characters found between the at-signs is entered into the literal count area and the literal is moved into the literal hold area.

LEAVE-SUB/FSSN-EXIT. Initialize area counters and switches for the next card. Return control to calling subroutine.

(3) FSSNX Flowchart.


FSSNX
Initialize Counters
Perform Reverse Scan to Locate last character on Card
Perform forward scan to test eacn character
2
CHAR=',' ?
YES
Move word to WORD-TABLE
NO
CHAR='.' ?
YES
3
NO
1


1
CHAR='@'
?
YES
Validate
Literal and
Move it to
LIB-TABLE
NO
Move CHAR
to WORD
2
3
FSSNX-EXIT

c. FSPRX (PRINT/ERROR SUBROUTINES).

(1) Summary. FSPRX is the print subroutine for File Structuring. It contains three entry points--FSPR, FSER and FSML. An entry into FSPR simply allows a single line of data to be printed. An entry into FSER results in the selection of a single error message from a table of messages and the printing of the message. A entry into FSML allows the multiple listing of a FFT as specified by the user.

(a) Function. Handles all FS printing requirements.

(b) Calling Sequence.

```
ENTRY  'FSER'  USING  FS-DD
ENTRY  'FSPR'  USING  FS-DD
ENTRY  'FSML'  USING  FS-DD
```

(2) Description.

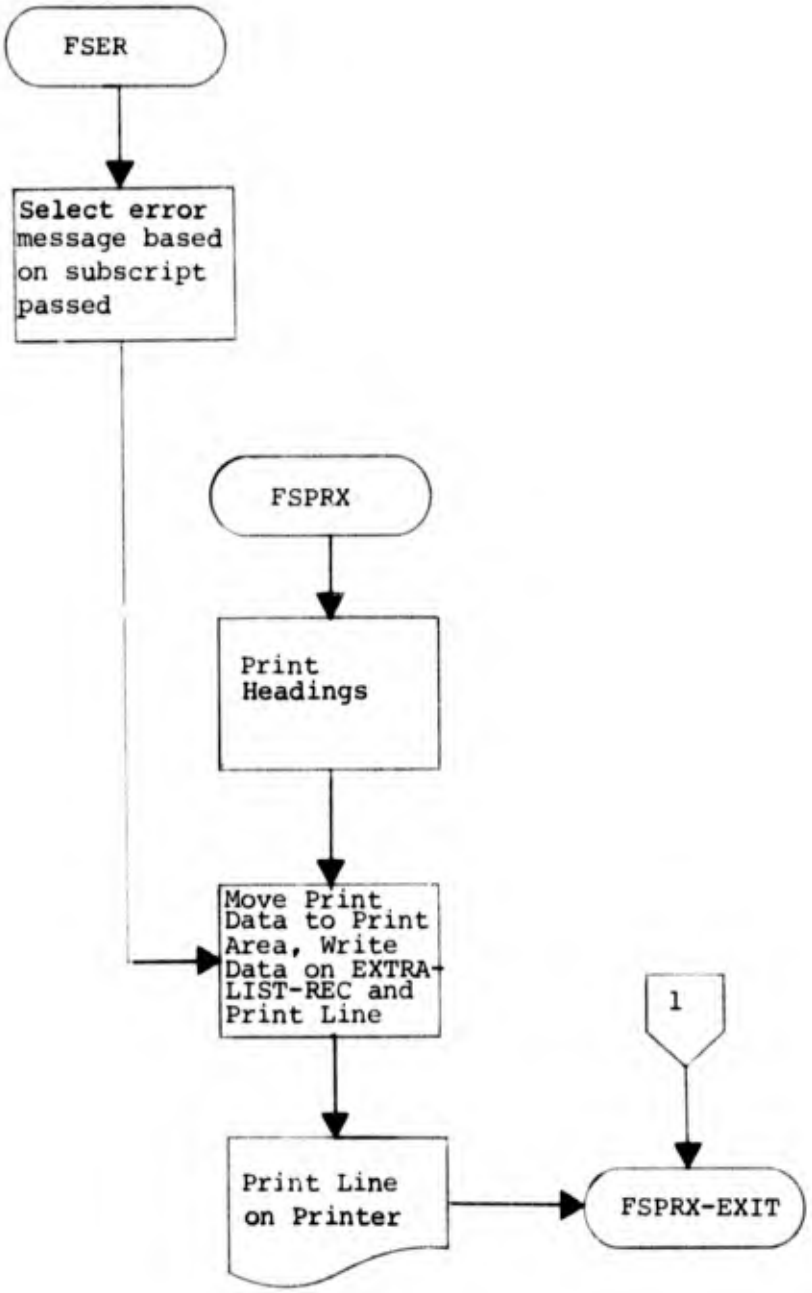
PROCEDURE DIVISION/FSER. Receives control from calling subroutine for entry into ERROR MESSAGE selection. Based on the value of ERR-CODE an error message is taken from a table of messages.

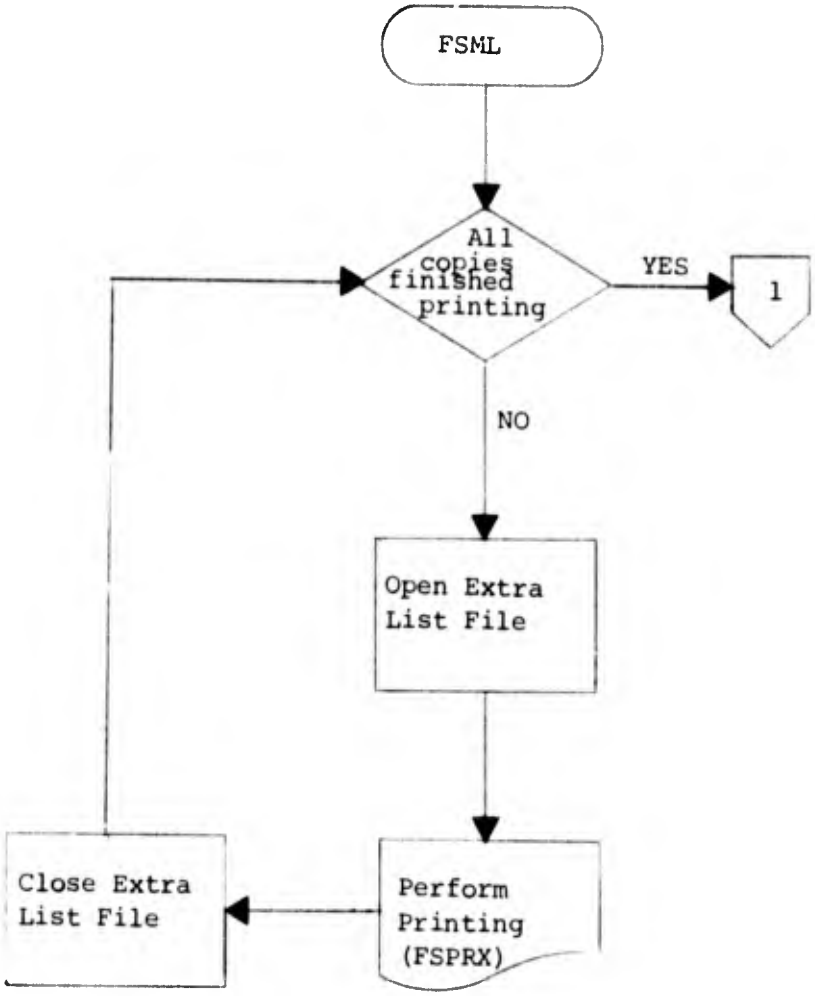
PRINTITN. Moves error message to print line for later printing. Contains entry point to printing subroutine.

FSPR/PRT-HDG. Prints heading at top of page after making necessary spacing requirement. Prints a line of data and writes the same line on extra listing file. (Record data for use with extra copy option.)

FSML/MULTI-LIST. Entry point to multi-listing coding. Opens extra list file reads and writes records until end of file is encountered. Closes file and tests LIST-SW to see if all copies have been printed. If not, opens file and repeats the operation.

(3) FSPRX Flowchart.


FSER
Select error message based on subscript passed
FSPRX
Print Headings
Move Print Data to Print Area, Write Data on EXTRA-LIST-REC and Print Line
1
Print Line on Printer
FSPRX-EXIT


FSML
All copies finished printing
YES
1
NO
Open Extra List File
Perform Printing (FSPRX)
Close Extra List File

d. FSJBX (JOB CARD).

(1) Summary. FSJBX is the first subroutine to be executed for the generation of an FFT. It initializes all logical records and a large number of hold areas, switches and counters used by itself and other subroutines. It also validates the file name, sets switches for other subroutines for user specified options FSJOB determines whether the mode of operation is TESTIT, (which allows for the debugging of the file's format definition deck) or whether the mode is CREATE which indicates the FFT is to be a new version of the MIDMS TABLE LIBRARY.

(a) Function. Processes the FSJOB card.

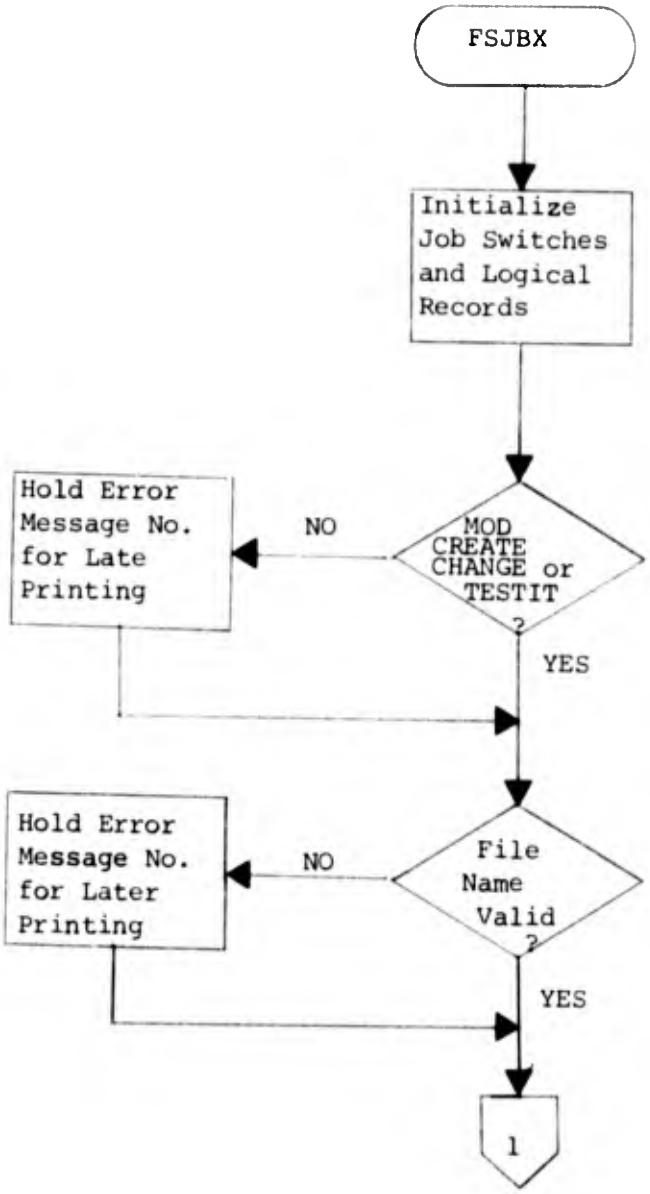
(b) Calling Sequence.

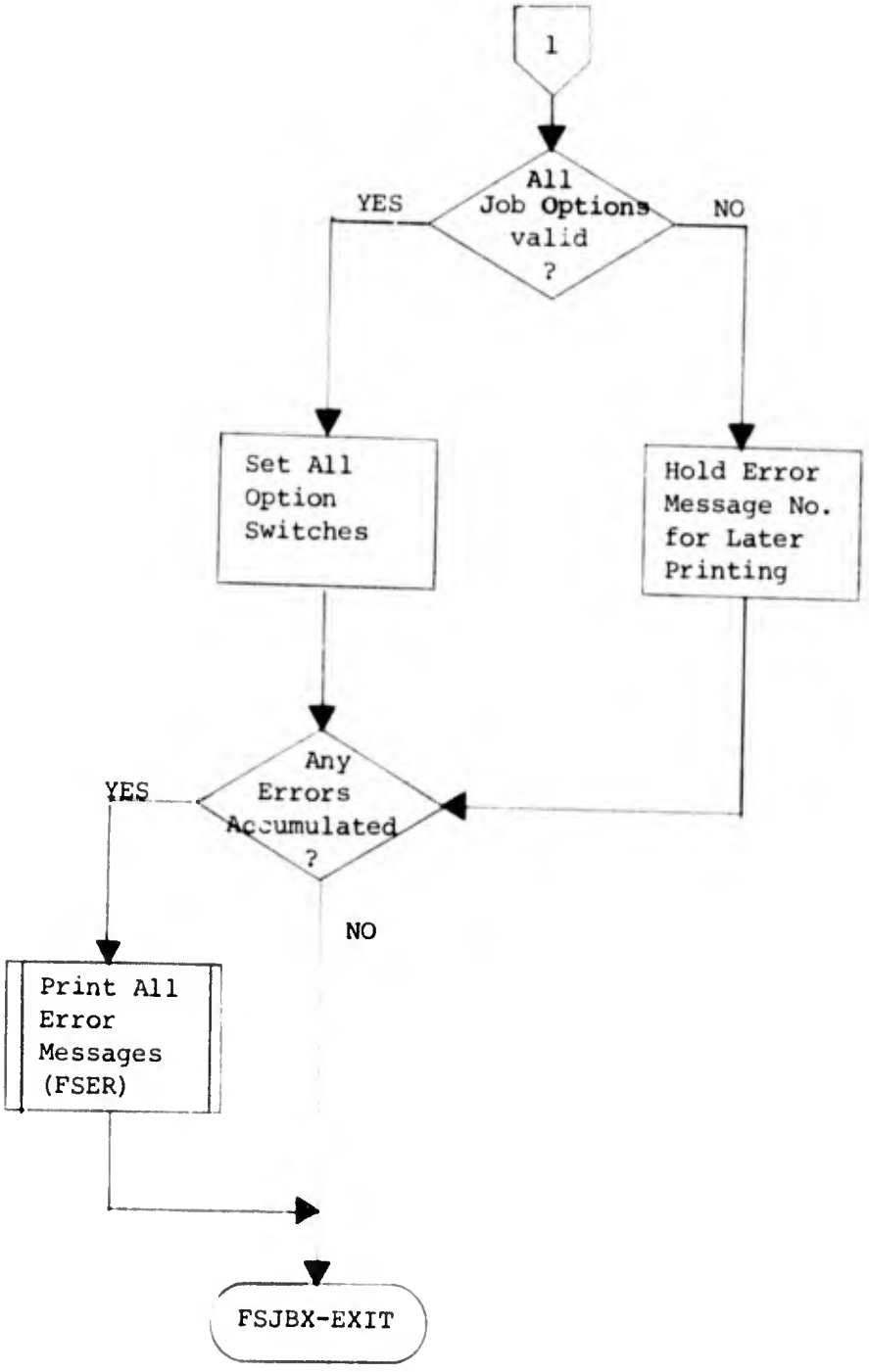
```
ENTRY  'FSJB'  USING  FS-DD
```

(2) Description.

FSJB thru CLEAR-TOTALS. Initialize counters, switches, and hold areas.
TEST-MODE. Checks to see if the FFT to be built is in CREATE, CHANGE, or TESTIT mode.
MOVE4A. Prints error message if mode is incorrect.
FILE-NAME. Checks validity of file name.
BAD-FILE-NAME. Prints error message if file name is invalid.
FILE-NAME-1 thru EXIT1. Checks to see if all options are valid and sets appropriate switches. The indicators for History and Data Exchange are in LR1.
JOB-PRINT/FSJB-EXIT. Sets switches and controls the printing of error messages which were found on the job card. After the printing of error messages, control is returned to the calling subroutine.

(3) FSJBX Flowchart.


FSJBX
Initialize Job Switches and Logical Records
MOD CREATE CHANGE or TESTIT ?
NO
Hold Error Message No. for Late Printing
YES
File Name Valid ?
NO
Hold Error Message No. for Later Printing
YES
1


1
All Job Options valid ?
YES
NO
Set All Option Switches
Hold Error Message No. for Later Printing
Any Errors Accumulated ?
YES
NO
Print All Error Messages (FSER)
FSJBX-EXIT

e. FSEDX (EDITS).

(1) Summary. The editing function of MIDMS is performed during output time; however, the FSEDX subroutine validates the name and the EDIT word length and determines the data size which the EDIT can handle. The edit mask consists of characters between the first and last at-signs. Using continuation cards, the edit mask may be as long as 132 characters. The EDIT name must not have embedded blanks and special characters.

(a) Function. Processes the EDIT card.

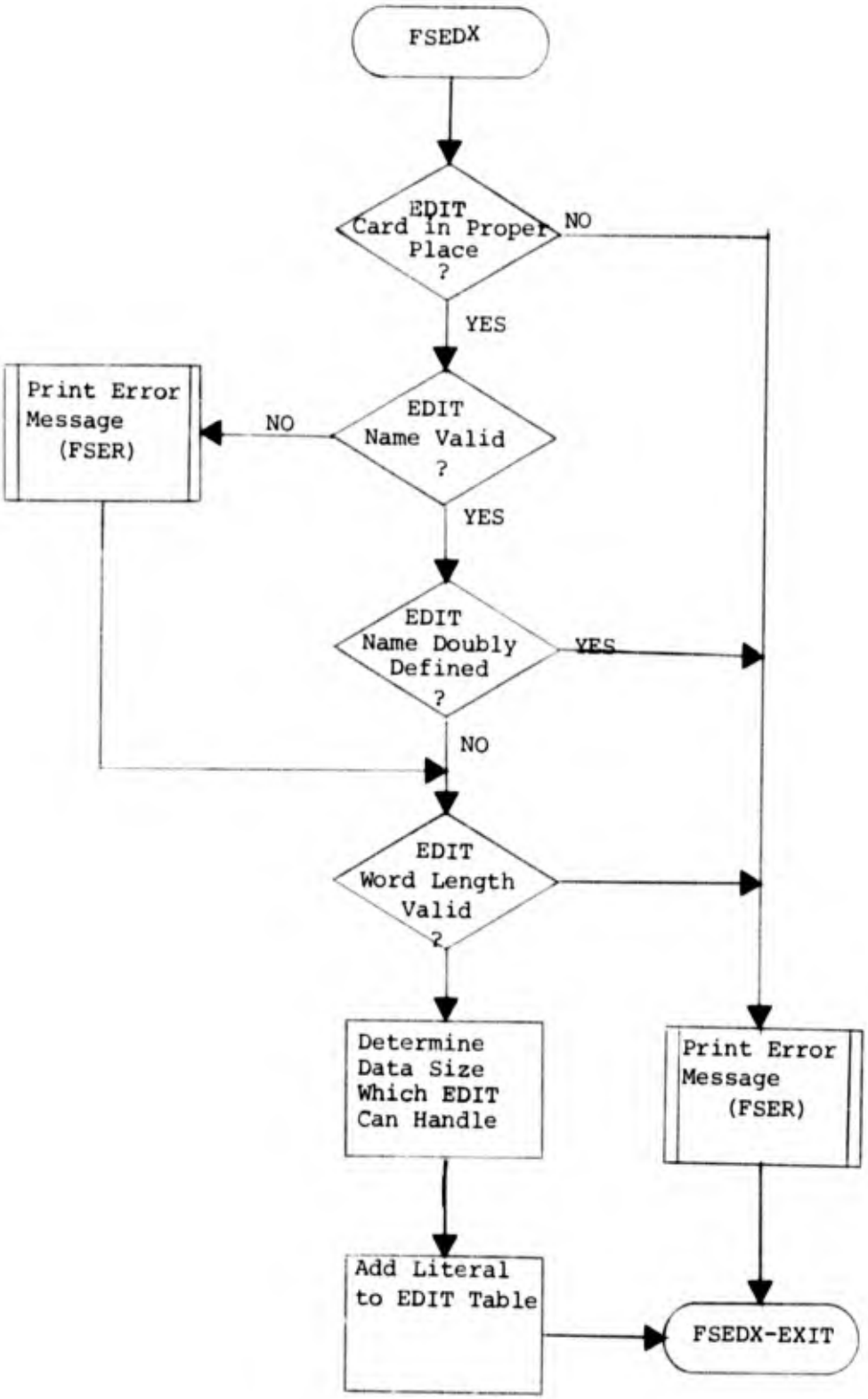
(b) Calling Sequence.

```
ENTRY  'FSED'  USING  FS-DD
```

(2) Description.

FSED. Checks to see if an EDIT is legal at this point.
TEST-NAME-CHAR. Checks the name to see if there are no embedded blanks, special characters and to see that the fifth character is not "S."
NAME-TST/TEST-NAME. Scans Edit Table, checking for doubly-defined name.
TEST-LITCNT. Validates the literal count. It must be greater than zero but less than 133 characters.
TEST-WORDCNT. Checks to see if there is an additional word other than the EDIT word. If there is, the extra word or parameter is ignored by FSED.
COUNT-CHARS/HOLES. Determines the number of zeros and blanks in the EDIT word.
EDIT-MOVE/FSED-EXIT. Builds the Edit Table, first by building the EDIT character-by-character, then by moving the EDIT to a string of EDITs.

(3) FSEDX Flowchart.


FSEDX
EDIT Card in Proper Place ?
NO
YES
EDIT Name Valid ?
NO
Print Error Message (FSER)
YES
EDIT Name Doubly Defined ?
YES
NO
EDIT Word Length Valid ?
Determine Data Size Which EDIT Can Handle
Print Error Message (FSER)
Add Literal to EDIT Table
FSEDX-EXIT

f. FSSBX (SUBROUTINE AND TABLE).

(1) Summary. FSSB processes the Subroutine and Table cards. FSSBX makes no distinction between SUB and TAB cards. The types are provided for file designers' purposes only. SUB and TAB should not be used to specify input conversion. This is done with a DSD. The program validates the name, the size and the output of SUB and TAB cards. The program builds the Subroutine Table and the Out Table, which contains a listing of all output sizes.

(a) Function. Processes the subroutine and table cards.

(b) Calling Sequence.

```
ENTRY  'FSSB'  USING FS-DD
```

(2) Description.

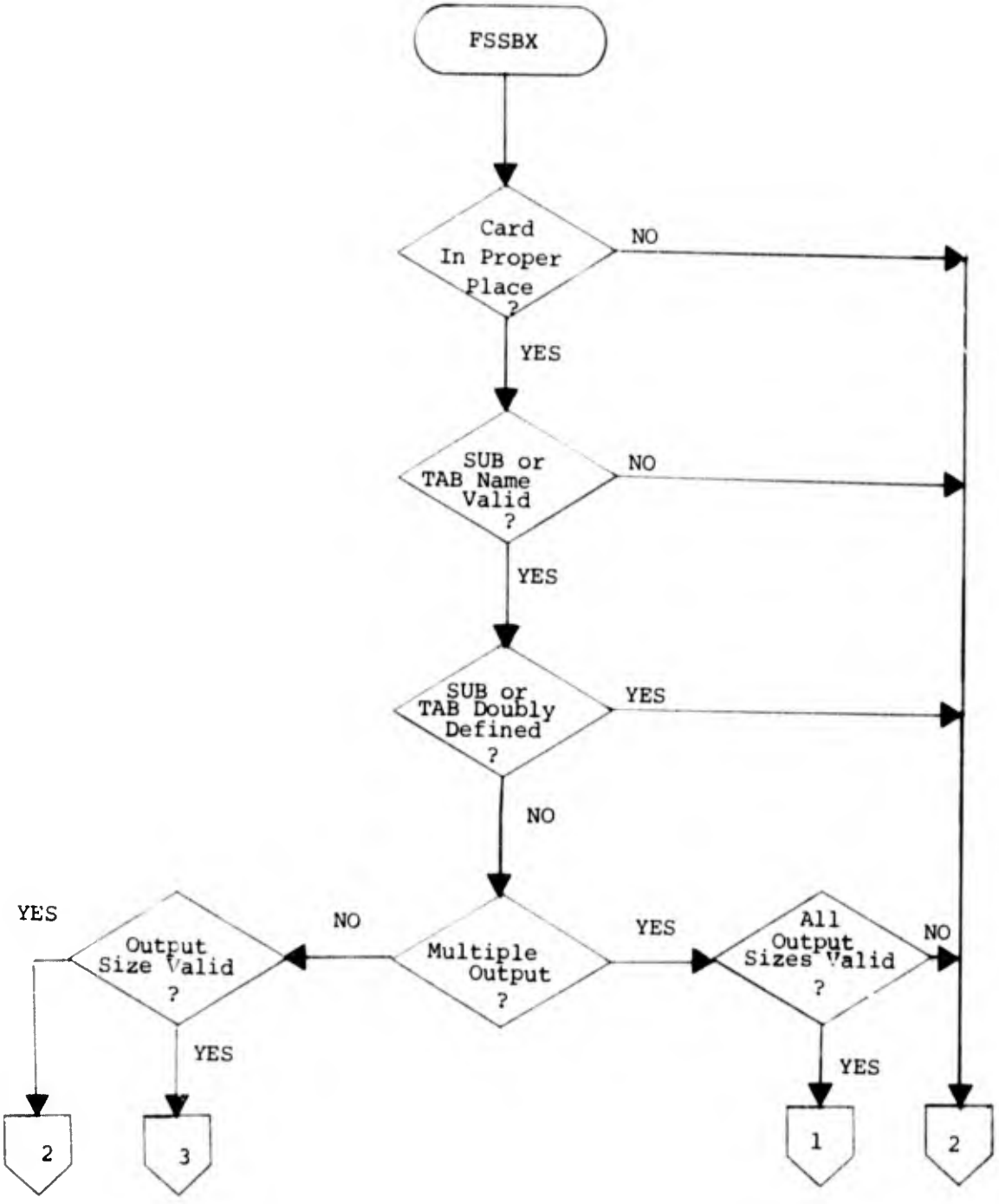
FSSB. Determines if the SUB or TAB card is legal at this point. All EDIT, SUB, and TAB cards must precede the FIELD and GROUP cards. Validates the name by checking the first and fifth characters for an alpha and "S" character respectively. Also, checks for special characters.

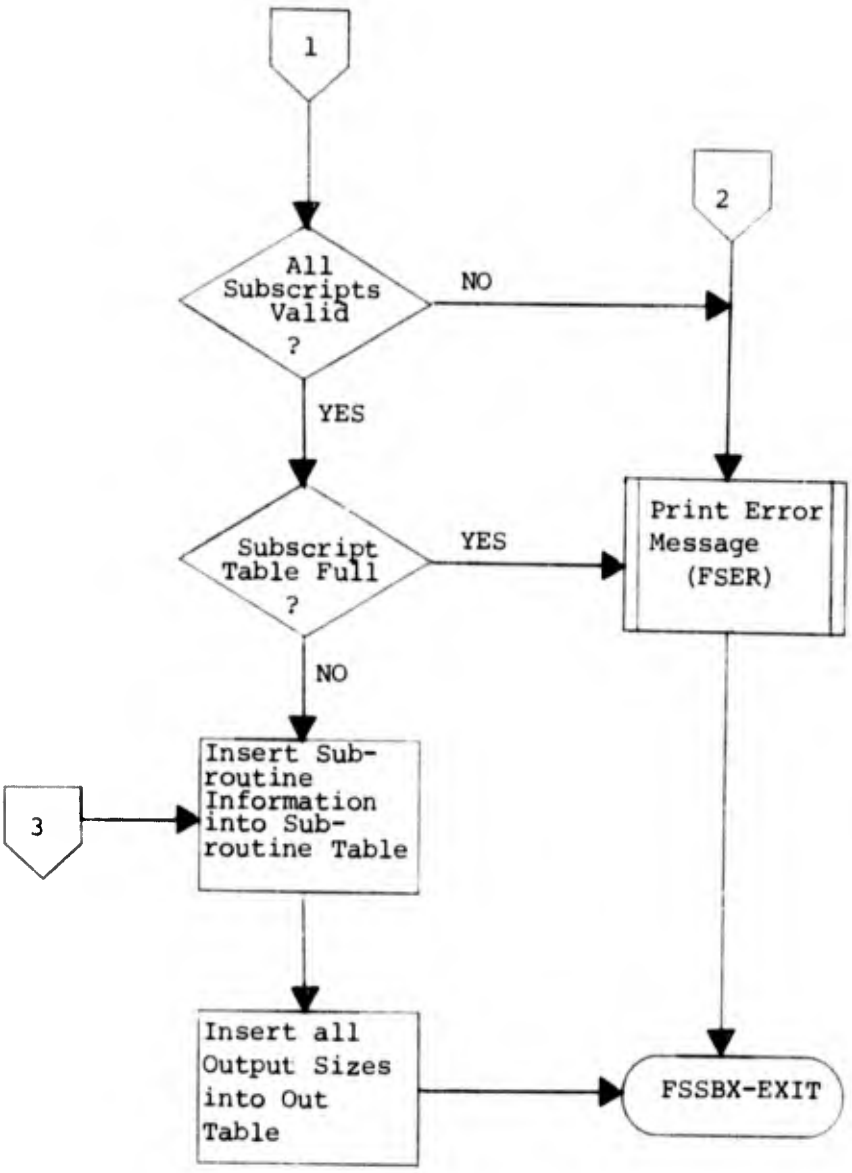
NAME-CHECK. Checks for doubly-defined names.

WORD-CHK-FSSB/LIT-CHECK-FSSB. Validates output size and determines whether there should be single or multiple output. Makes a check to see if a literal is present on the card. If so, the literal is ignored by FSSB.

TAB-BUILD/FSSB-EXIT. Makes entries into the Subroutine Table and inserts all output sizes into the Output Table.

(3) FSSBX Flowchart.


FSSBX
Card In Proper Place ?
NO
YES
SUB or TAB Name Valid ?
NO
YES
SUB or TAB Doubly Defined ?
YES
NO
Multiple Output ?
NO
YES
Output Size Valid ?
YES
YES
All Output Sizes Valid ?
NO
YES
2
3
1
2


1
2
All Subscripts Valid ?
NO
YES
Subscript Table Full ?
YES
Print Error Message (FSER)
NO
3
Insert Sub-routine Information into Sub-routine Table
Insert all Output Sizes into Out Table
FSSBX-EXIT

g. FSFLX (FIELD Card).

(1) Summary. FSFLX performs initial processing of the FIELD card. It validates the name and the size, and checks for duplicate defined names. FSFLX checks the set type specification to see whether the field should be placed with control, fixed or periodic data.

(a) Function. Processes the FIELD card.

(b) Calling Sequence.

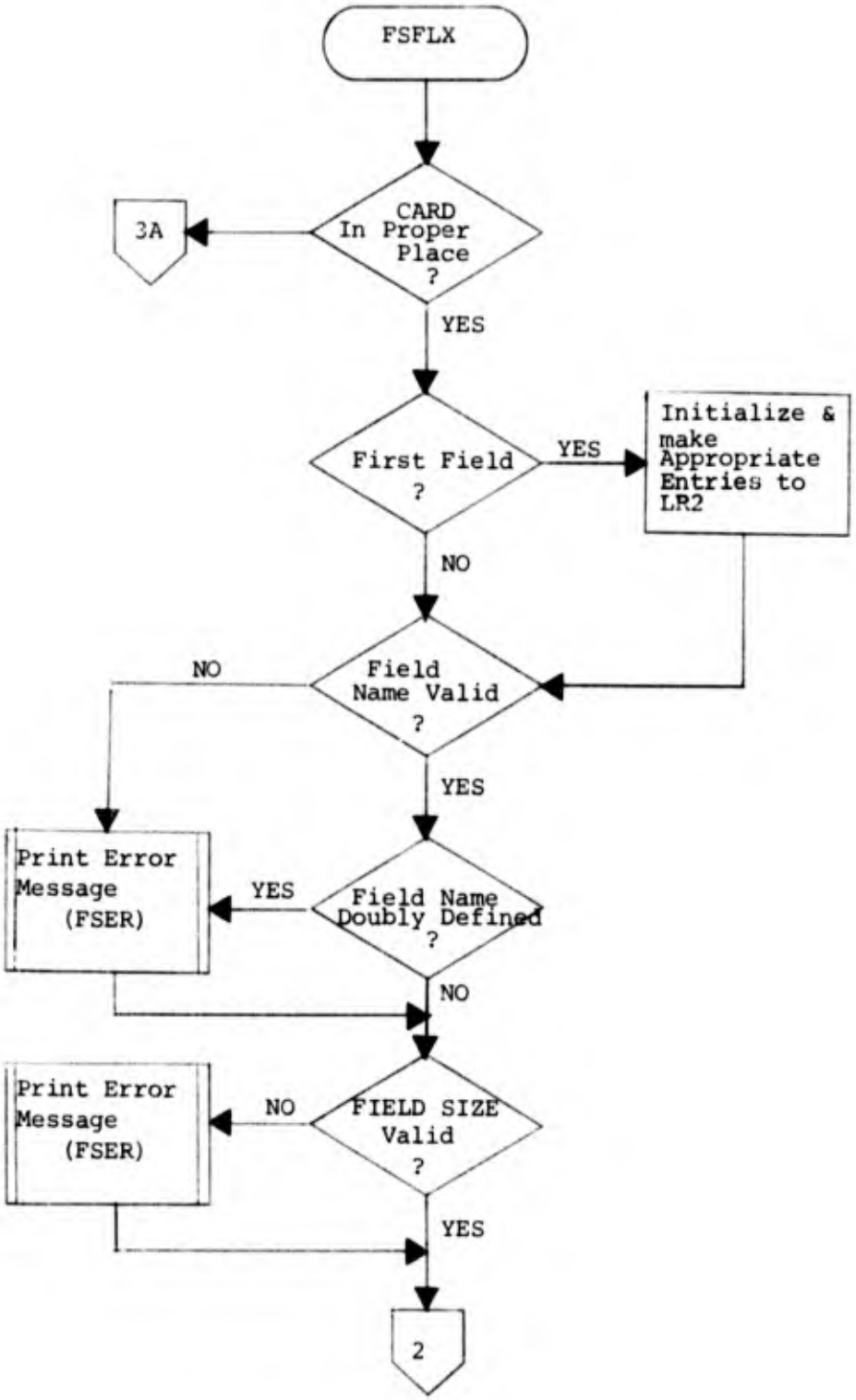
```
ENTRY  'FSFL'  USING  FS-DD
CALL   'FSDT'  USING  FS-DD
CALL   'FSIO'  USING  FS-DD
CALL   'FSGO'  USING  FS-DD
```

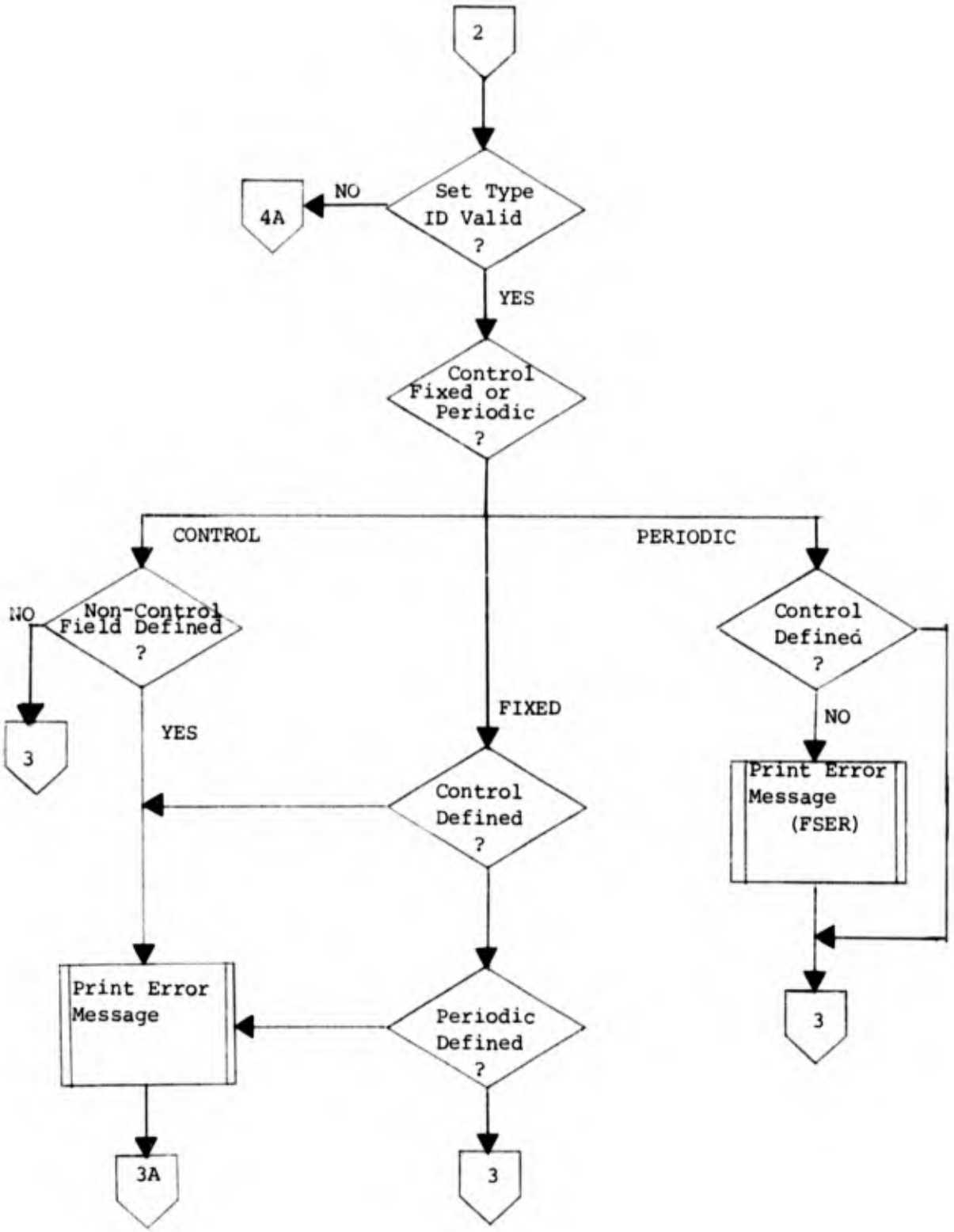
(2) Description. Procedure Division receives control from calling subroutines.

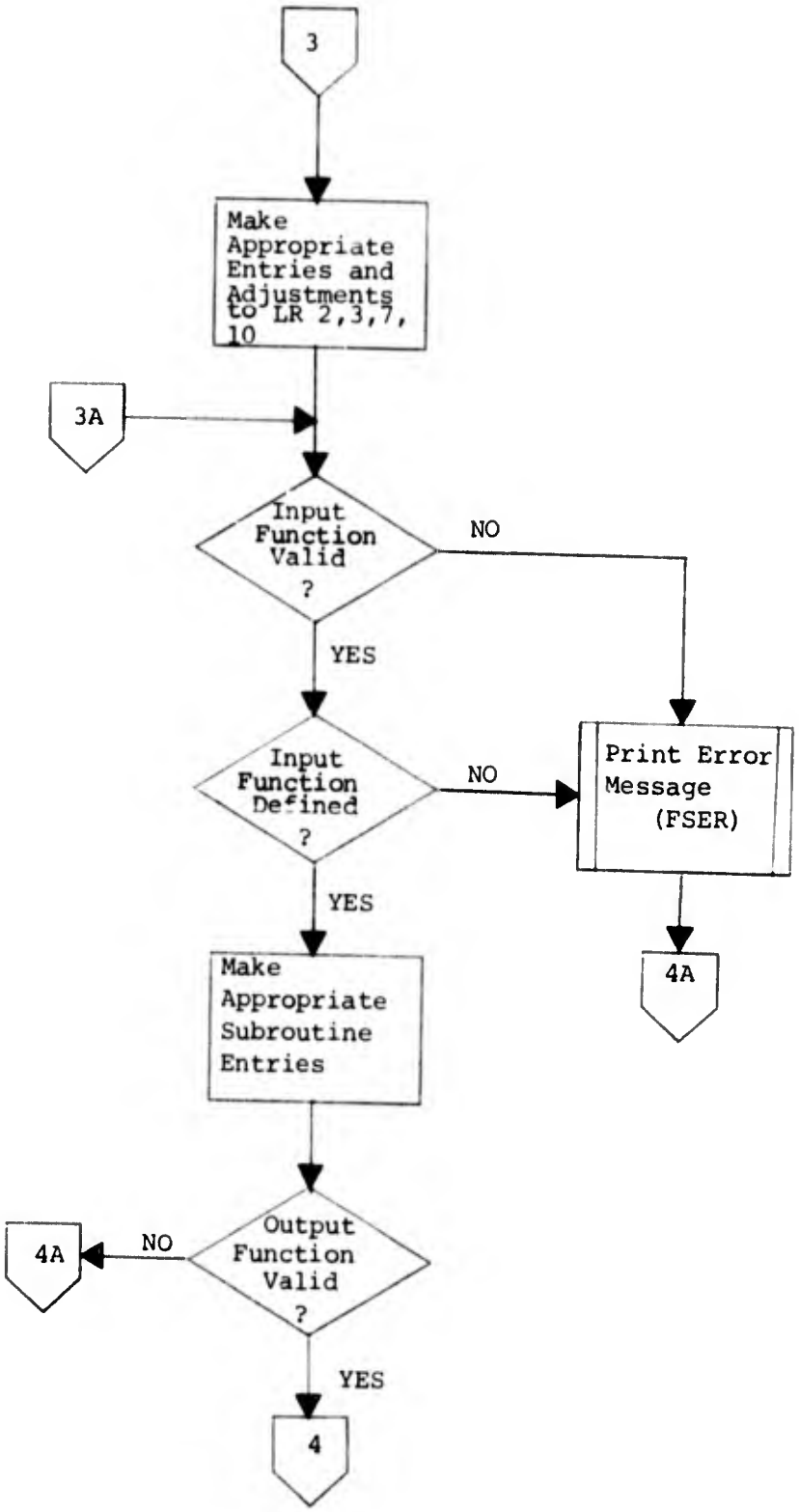
FSFL. Determines if the FIELD card is legal at this point. If so, and it is the first FIELD card, (a) put RECCT field entries into LR2 and LR7, (b) put first entry into LR3, and (c) put first three entries into LR10.
RECCT-MOV-PFRM/RECCT-MOVE. Moves output label for RECCT to LR8.
CHECK-FIELD. Initializes D-FFT-LR2 and D-FFT-LR1 (dummy logical records). Validates field name.
NOT-IN1/LR2-LU. Checks for duplication of field names in LR2.
NOT-IN2. Moves field name to dummy logical records and validates field size.
FLSIZ-ERR. Prints an error message concerning the invalid size.
FL-FIELD-TYPE/FL-FIELD-TYPE. Checks for and validates different types of set ID formats.
CHKCTL-LEN/CNTL-SET. Checks to see if control fields have been defined.
MOVE7. Updates LR3 and dummy LR2.
FL-X. If "X" type card is legal at this point, make appropriate entries into dummy LR2 and set switches to indicate "X" type.
CTL-MOVE-XTYPE-MOVE. Adjust field sequence level numbers.
XTYPE-MOVE1/NODATE. Updates LR10 and test DATE-SW to see whether the date option was specified. If so FSDTX will be called to generate CREATE DATE and CHANGE DATE fields.
FL-IP/FIELD-ID. Checks to see whether 'N,' 'S,' 'I,' 'T,' or a blank (Mark III FFS Retrieval Mode Specification) is on the card. If one of the above specifications is on the card, a pointer is set to skip over the position. FSIOX is than called to process the input and output functions.
WORD-MOVER/CALL-FSGO. Controls the initial and subsequent calls to FSGOX.

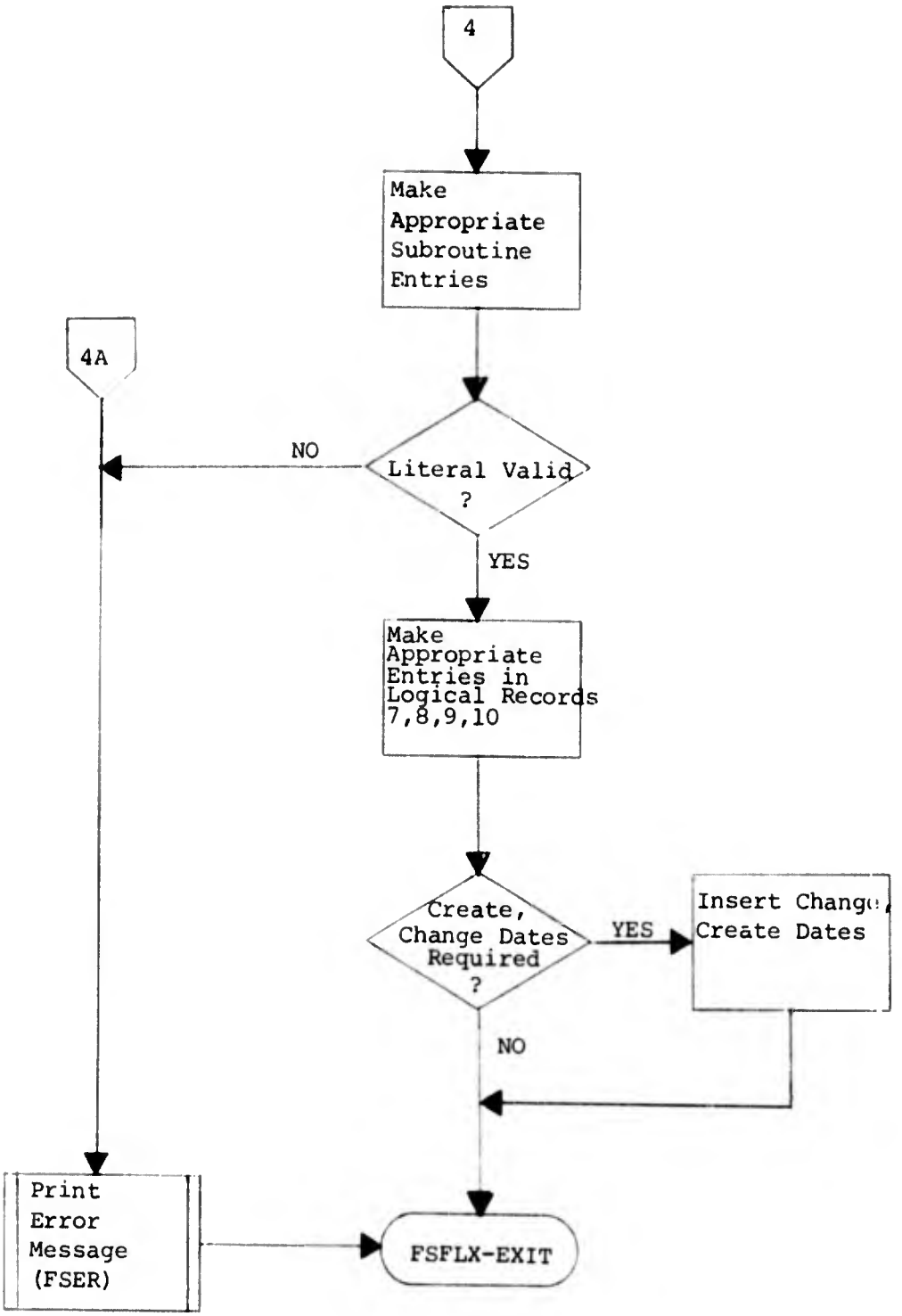
FSFLX-EXIT. Return control to calling subroutine.

(3) FSFLX Flowchart.


FSFLX
CARD In Proper Place ?
3A
YES
First Field ?
YES
Initialize & make Appropriate Entries to LR2
NO
Field Name Valid ?
NO
YES
Print Error Message (FSER)
YES
Field Name Doubly Defined ?
NO
Print Error Message (FSER)
NO
FIELD SIZE Valid ?
YES
2


2
Set Type ID Valid ?
NO
4A
YES
Control Fixed or Periodic ?
CONTROL
PERIODIC
FIXED
Non-Control Field Defined ?
NO
3
YES
Control Defined ?
Print Error Message
Print Error Message (FSER)
Control Defined ?
NO
3
Periodic Defined ?
3A
3


3
Make Appropriate Entries and Adjustments to LR 2,3,7, 10
3A
Input Function Valid ?
NO
YES
Input Function Defined ?
NO
Print Error Message (FSER)
4A
YES
Make Appropriate Subroutine Entries
Output Function Valid ?
4A
NO
YES
4


4
Make Appropriate Subroutine Entries
4A
NO
Literal Valid ?
YES
Make Appropriate Entries in Logical Records 7,8,9,10
Create, Change Dates Required ?
YES
Insert Change, Create Dates
NO
Print Error Message (FSER)
FSFLX-EXIT

h. FSIOX (INPUT OUTPUT FUNCTION).

(1) Summary. FSIOX validates the input function as well as the output function for FIELD and GROUP cards. It checks whether an input subroutine is being used for both input and output functions, whether the subroutine has been defined in the subroutine table, or whether the output of the input function is equal to field size. The subroutine also validates the extract label. An extract label is a name or title of a single data field, which can be used for outputting data with specified output names.

(a) Function. Validates the input and output functions of the field and group cards.

(b) Calling Sequence.

```
ENTRY  'FSIO'  USING  FS-DD
```

(2) Description.

PROCEDURE DIVISION. Receives control from calling subroutine and tests IO-SW to determine where processing should be continued.

FL-IPFUN. Determines whether I/O functions are specified. If so, are there any Mark III multi or single input function indicators (*$,*1) specified; if there are, skip to next word; otherwise go to IPOP-CHK to see if the present word is an I/O function.

MORE-IP. This paragraph handles the skipping over of multi-input functions (specified on Mark III FDD's only). MIDMS accepts only the first input function specified on a FIELD card. Tests are performed for the presence of an output function.

IPOP-CHK. Checks for presence of input and output functions.

CHK-ALLIPFUN. Checks and validates all input functions, ALPHA, NUMER, SGNUM, conversion subroutines and tables.

ALPHA-ASSUMED. If input subroutine is not specified, then assume alpha input function.

TEST-NUM-SIZE. Tests numeric field for size greater than 18, (COBOL limitation on maximum numeric field size).

IP-SUB-CHK/FL-IP-SIZ. Validates input subroutines and tables, for definition, for size of output, for subscripts accuracy, etc.

FL-OPFUN/OP-NOW. Checks to see whether multi-input functions are being processed, if so, continue with multi-input processing. Checks to see if an output function has been defined.

BLANK-ASSUMED. If output function is not defined, assume blank output function.

OP-RETURN. Is there an output conversion subroutine or edit specified.

OP-SBTBL-PFRM/FL-ENTRY1. Validates conversion subroutines, tables and edits making the following checks:

UNDEFINED OUTPUT FUNCTION
SUBROUTINE USE FOR BOTH INPUT AND OUTPUT
FUNCTION REQUIRES SUBROUTINE
ILLEGAL SUBSCRIPT

EXTRACT-CHK. Tests LITCOUNT for existence of extract label. If LITCOUNT is equal to -1 or LITCOUNT greater than 132, then the extract label is invalid; otherwise, it is an accepted literal.

FL-LAB-SIZE. Determines relative size of output label for dummy LR7.

FL-LAB-MOVE. Checks for LR8 overflow.

FL-LMOV-PFRM/FL-LABEL-MOVE. Moves output label to LR8.

MOVE-X. Go to FL-SUMMARY.

OP-EDIT/FLD-EDTBL. If output function can be an EDIT, check to see if EDIT has been defined. If so, make sure the edit will hold the data.

FL-LR7-FIN. Moves output edit information to dummy LR7. Makes entry into EDTBL to show that the edit has been used.

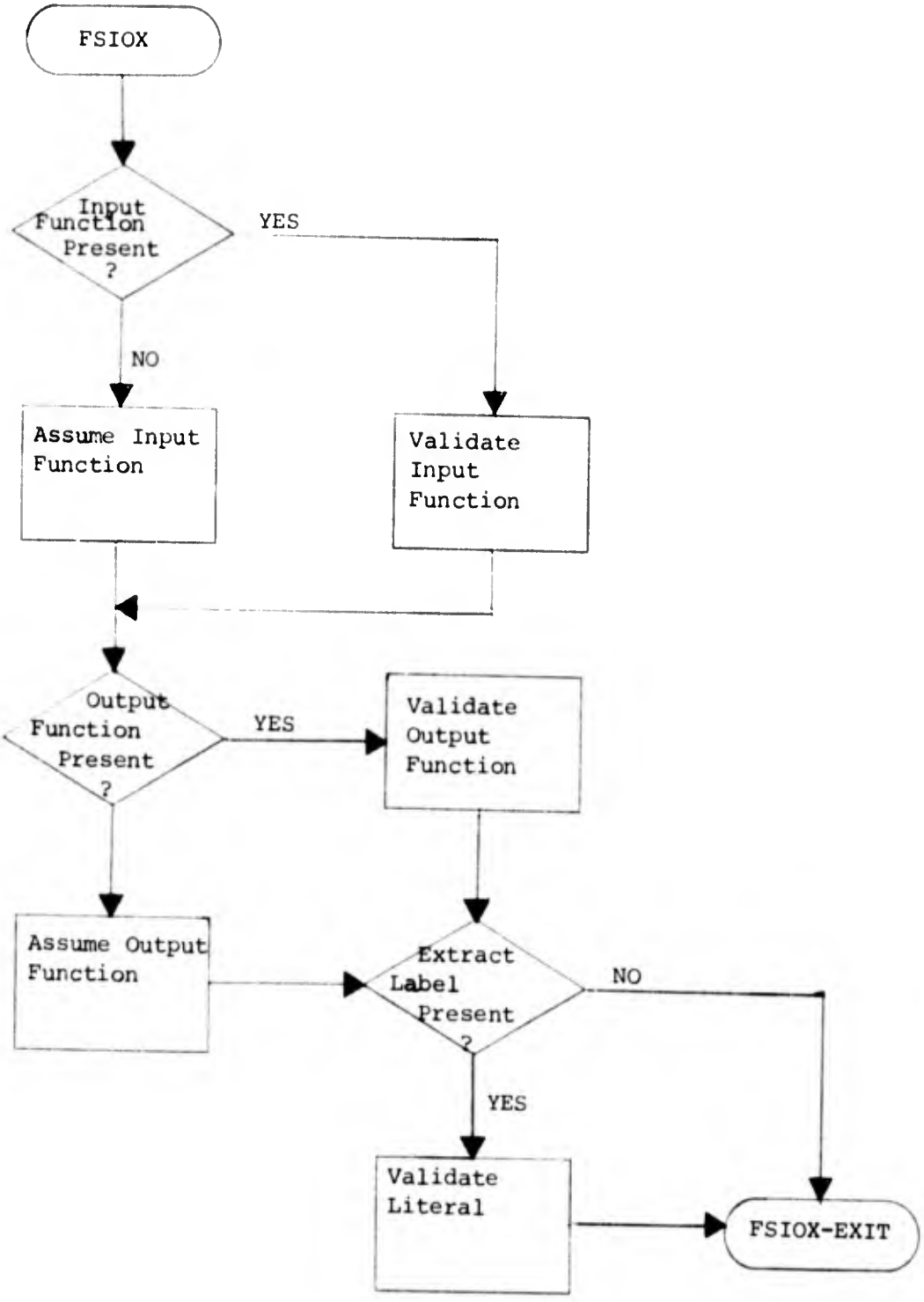
FL-ENTRY2. Moves edit information into dummy LR9.

FL-EDT-MOVE1. If LR9 will not overflow, move edit length into LR9.

FL-EDT-2/FL-EDIT-3. If LR9 will not overflow, move edit capacity into LR9.

FL-EDT-4/FL-EDT-5. If LR9 will not overflow, move the edit word into FL-SUMMARY. Checks for LR2 overflow. Checks to see if length of control fields is greater than 30 characters. Makes updates to LR10.

(3) FSIOX Flowchart.


FSIOX
Input Function Present ?
YES
NO
Assume Input Function
Validate Input Function
Output Function Present ?
YES
Validate Output Function
Assume Output Function
Extract Label Present ?
NO
YES
Validate Literal
FSIOX-EXIT

i. FSDTX (Management Date).

(1) Summary. If the date option is specified on the FSJOB card, FSDTX will be called by FSFLX and FSENX in order that CREDT and CHGDT entries may be added to logical record two, seven, eight and ten.

(a) Function. Generate CREDT and CHGDT fields.

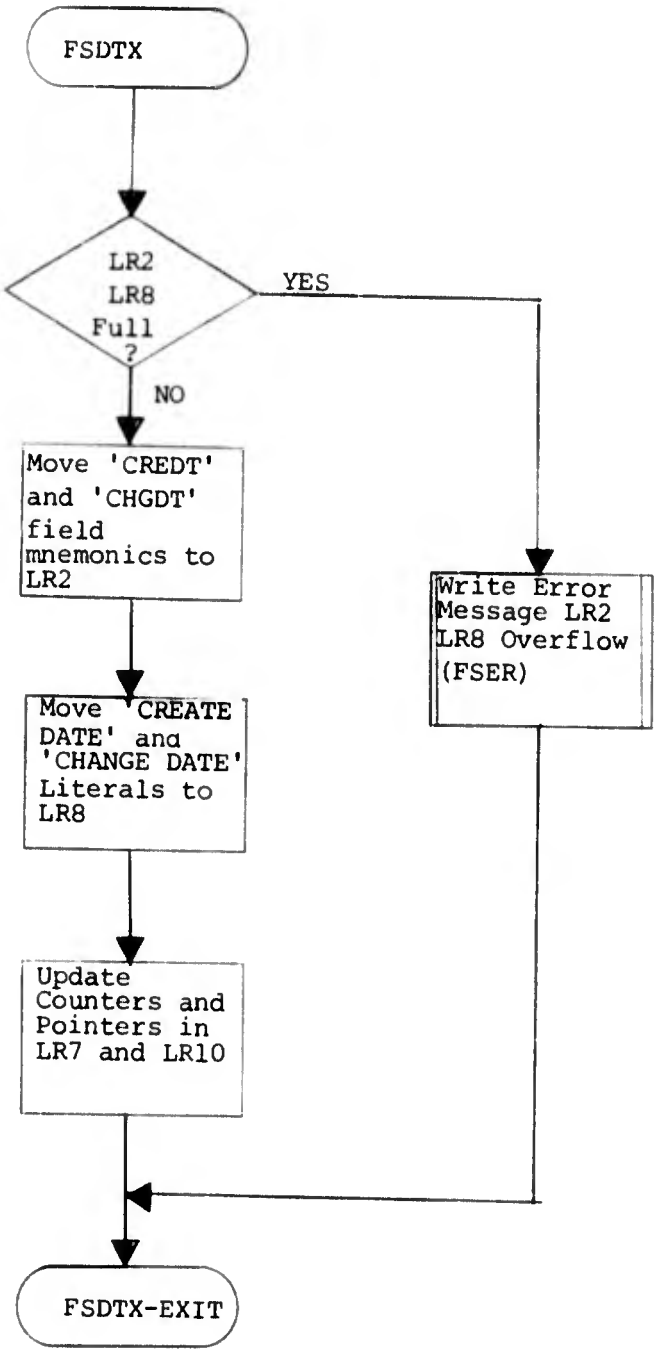
(b) Calling Sequence.

```
ENTRY  'FSDT'  USING  FS-DD
```

(2) Description.

PROCEDURE DIVISION. Receives control from calling sequence.
DATE-MOVE1/ADD-LR3-FIX. Moves 'CREAT DATE' and 'CHANGE DATE' literals to LR8. Makes necessary adjustments to LR2, LR7, and LR10.
FSDT-EXIT. Returns control to calling subroutine.

(3) FSDTX Flowchart.


FSDTX
LR2
LR8
Full
?
YES
NO
Move 'CREDT' and 'CHGDT' field mnemonics to LR2
Write Error Message LR2 LR8 Overflow (FSER)
Move 'CREATE DATE' and 'CHANGE DATE' Literals to LR8
Update Counters and Pointers in LR7 and LR10
FSDTX-EXIT

j. FSGOX (FIELD Card Continued).

(1) Summary. FSGOX performs further processing of the FIELD card. When the set type is missing, FSGOX assumes the set type based on the previous set defined. Entries are added to logical records two, three, seven, eight, nine and ten.

(a) Function. Continued processing of the field card.

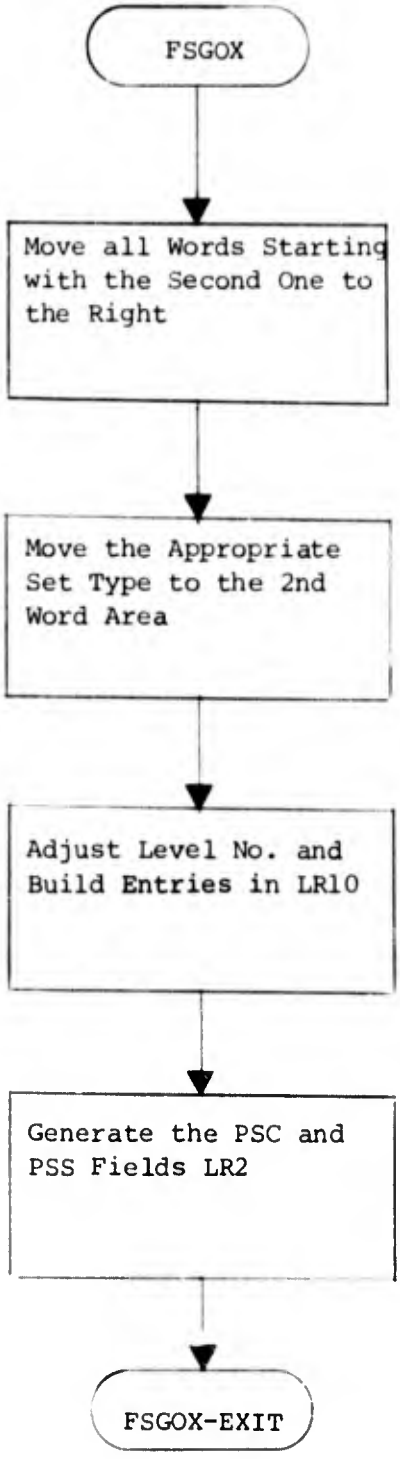
(b) Calling Sequence.

```
ENTRY  'FSGO'  USING  FS-DD
```

(2) Description.

PROCEDURE-DIVISION. Receives control from calling subroutine. Test for return paragraph within subroutine.
SHIFT-WRDTAB/SHIFT-IT. Moves words which are specified on the card down by one position to make room for the insertion of set type.
BUILD-SETYPE. Based on the previous card's set membership, a set type is generated and placed into the space created by the above paragraphs. Control is than returned to FSFLX to validate the set type.
FL-NUM/FL-TYPE-MOVE1. Validates and, if correct, saves periodic set ID with three character format.
GO-ON-1. Checks to see if periodic set ID is in its proper order.
FLDTYPE-MOVE. Updates LR10 if new periodic set.
FLDTYPE-MOVE1/FLTYPEMOVE1. Updates LR10.
FIRST-SET. If first set, make appropriate entries into LR10.
FLTYPEMOVE2. Updates LR10.
MOVE4. Checks for LR2 overflow. Updates size fields for LR2, LR7 and LR8 in LR1.
MOVE2. Saves periodic set ID and moves it to PSS and PSC entries in LR2.
MOVE3. Makes entries for PSS and PSC fields into LR2 and LR3.
GO-ON. Makes entries for PSS and PSC fields into LR2 and LR7. Checks for LR8 overflow.
OP-MOVE-1. Puts output label for PSC field into LR8.
GO-ON2/OP-MOVE-2. Puts output label for PSS field into LR8.
GO-ON3. Save LR2 and LR7 pointers.
MOVE6. Builds dummy LR3 entry.
FSGOX-EXIT. Returns control to calling subroutine.

(3) FSGOX Flowchart.


FSGOX
Move all Words Starting with the Second One to the Right
Move the Appropriate Set Type to the 2nd Word Area
Adjust Level No. and Build Entries in LR10
Generate the PSC and PSS Fields LR2
FSGOX-EXIT

k. FSGRX (GROUP Card).

(1) Summary. FSGRX processes the GROUP card specification until it exhausts the list of fields to be grouped. Entries are built for logical record two, seven, and ten. FSGRX determines that: (a) the group mnemonic is in the proper format and has not been previously defined and (b) the list of fields have been defined and they are adjacent. A single field may be grouped, the group may be grouped and that group may be grouped. Thus, one data item may be referenced by several different names to any practical level. The grouping scheme makes it possible to reference fields as well as Groups by several different names.

(a) Function. Processes the GROUP card.

(b) Calling Sequence.

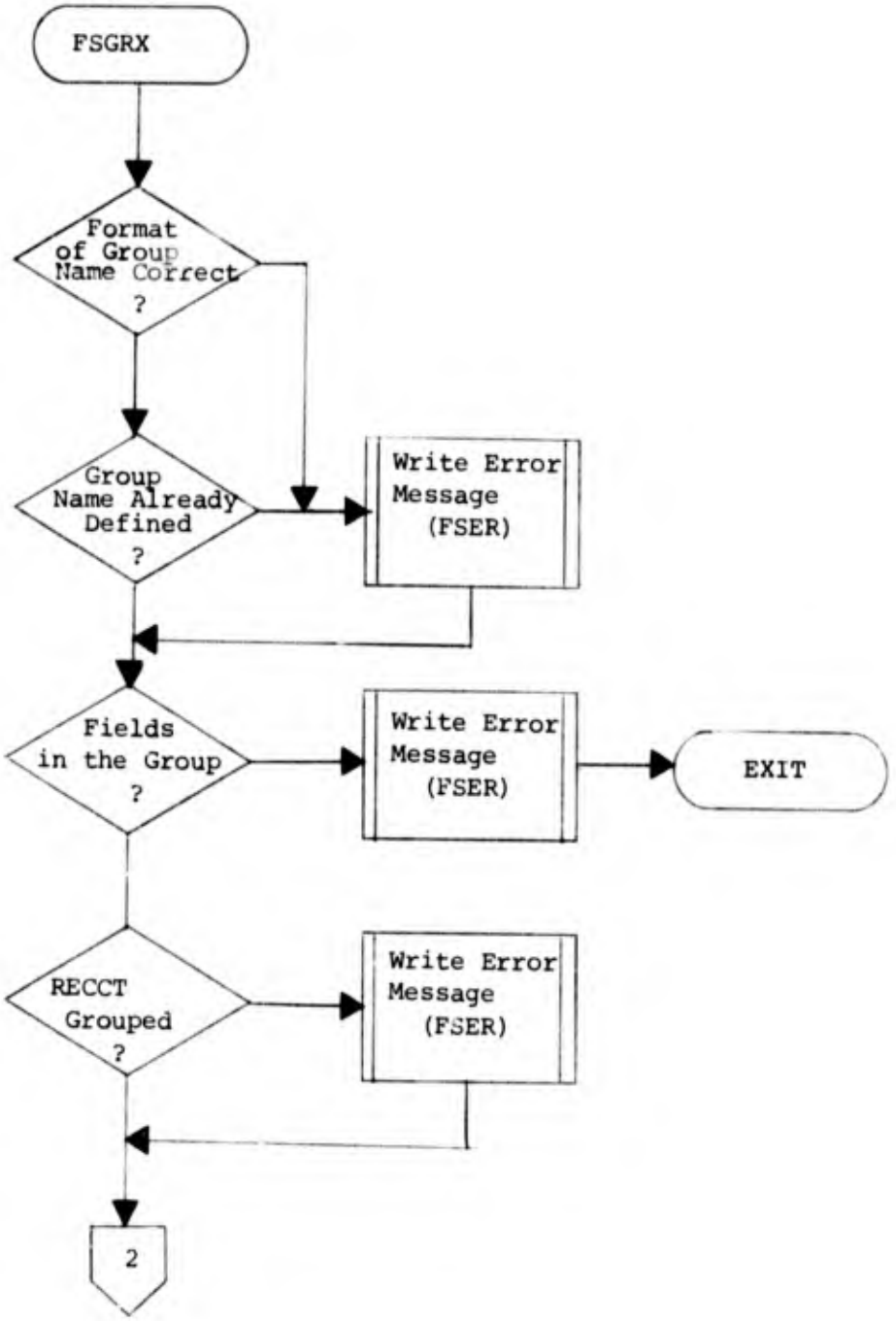
```
ENTRY  'FSGR'  USING  FS-DD
CALL   'FSIO'  USING  FS-DD
```

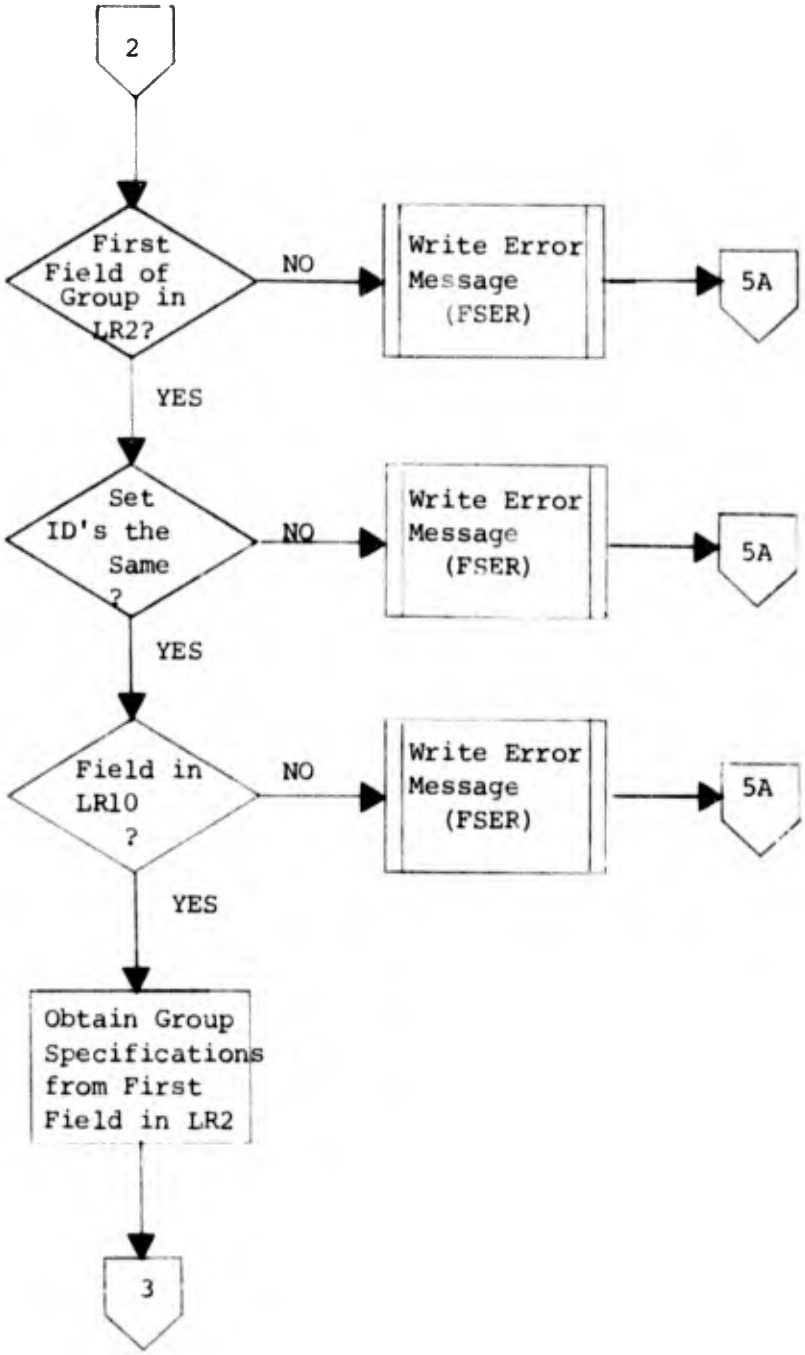
(2) Description.

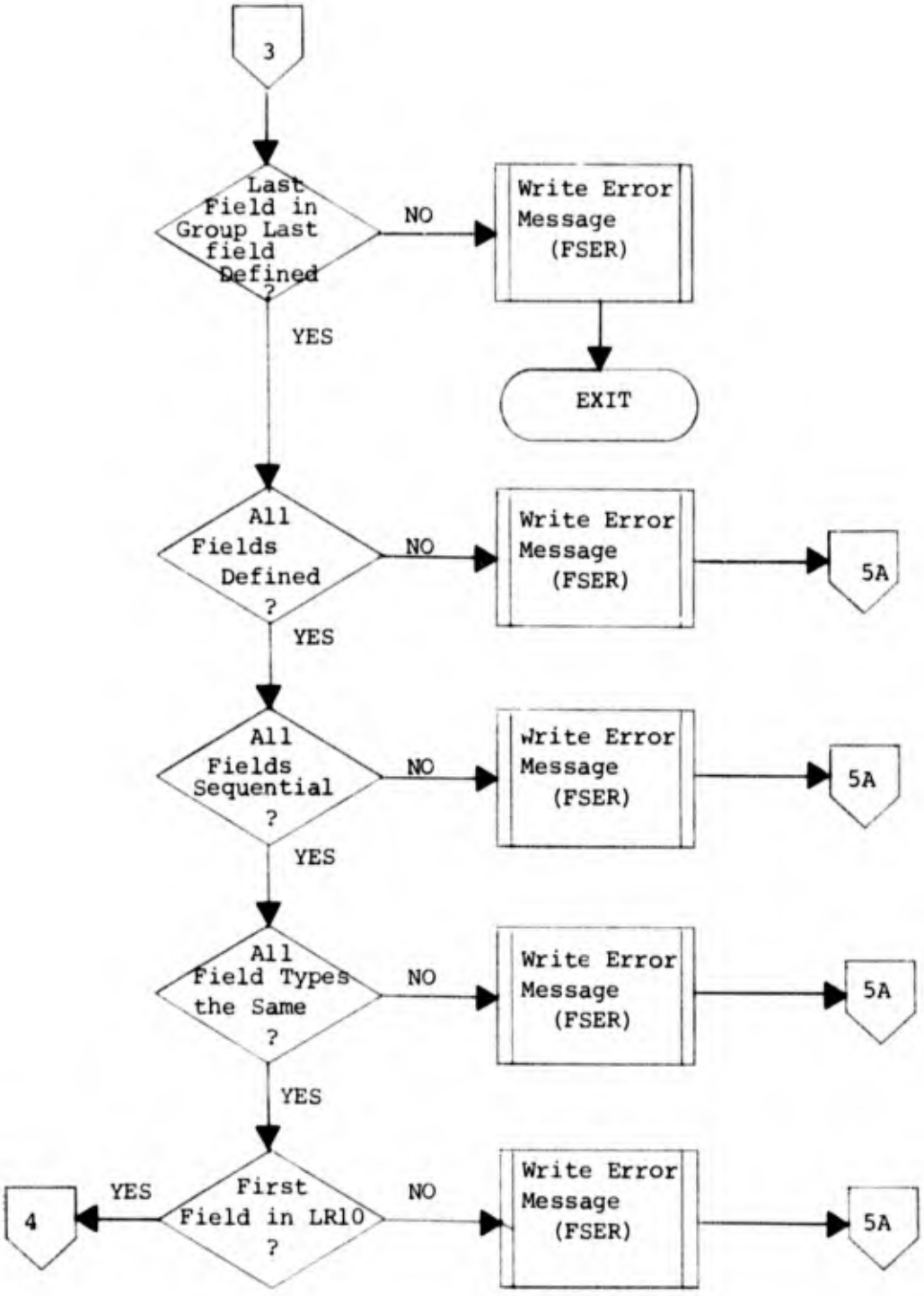
FSGR. Checks the format of the group mnemonic.
DUP-NAME-CHK. Verifies that the group mnemonic has not been previously defined.
GR-SET-CHK1. Verifies that RECCT has not been grouped.
GR-FLD-RETURN/GR-FIELD-FIND. Finds the first field definition in LR2.
GR-SET-CHK2. Verifies that all field types in the group are the same and locates the fields in LR10.
FIND-LAST-FIELD. Places a pointer at the last field in the list for the FIELD routine when an error is encountered which makes execution impossible.
GR-LAST-FLD. Finds the last field in the group list during processing.
LR10-SEQ-FIND. Finds the position of the first entry in LR2.
LR10-SEQ-CHK/GR-TYPE-TEST. Obtains specifications of the group from the first field specifications in LR2. Verifies that the group type is correct.
GR-SEQ-CHK. Checks if the last field in the group is the last field defined.
GR-FIELD-MOVE thru GR-HOP-CHECK. Checks to see if the fields are sequential, if they have been defined, and if their field types are the same.
LR10-CARD-CHK. Finds the field entries in LR10 and modifies the group specifications as necessary.
GR-LR2-BLD. Builds a dummy LR2 to pass to the FIELD routine.

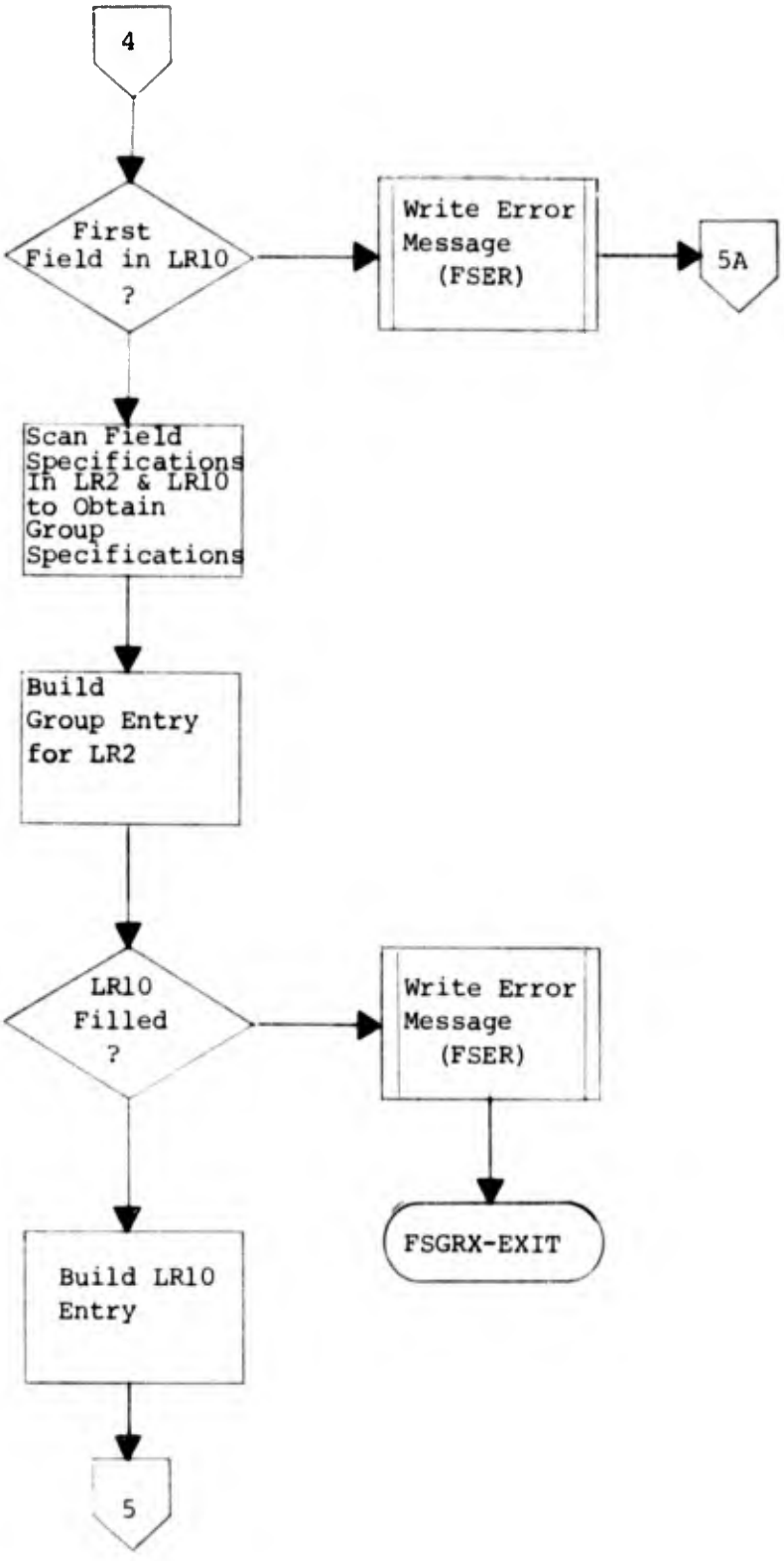
SPECIAL-GROUP thru FSGR-EXIT. Builds special group and adjusts sequence level numbers in LR10.

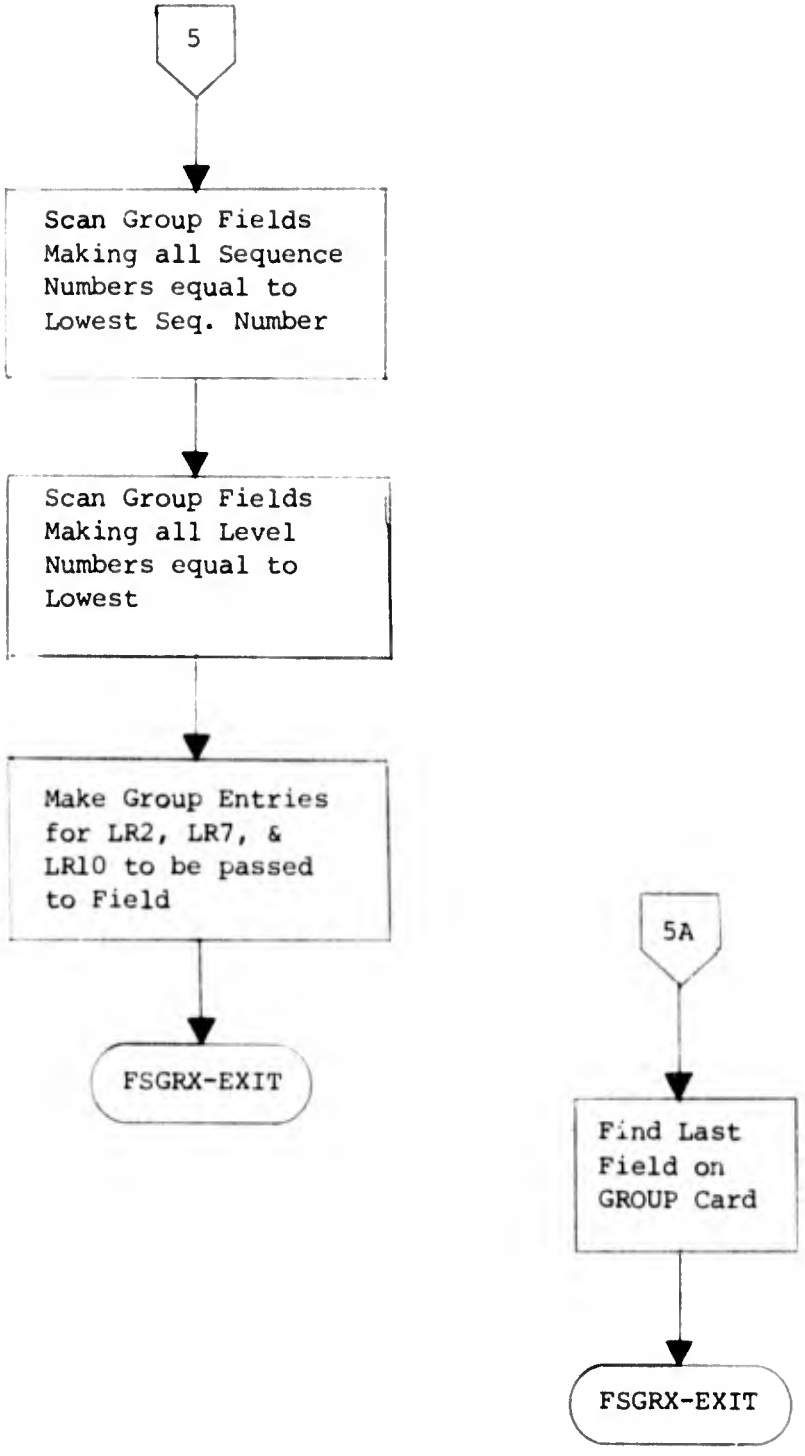
(3) FSGRX Flowchart.


FSGRX
Format of Group Name Correct ?
Group Name Already Defined ?
Write Error Message (FSER)
Fields in the Group ?
Write Error Message (FSER)
EXIT
RECCT Grouped ?
Write Error Message (FSER)
2


2
First Field of Group in LR2?
NO
Write Error Message (FSER)
5A
YES
Set ID's the Same ?
NO
Write Error Message (FSER)
5A
YES
Field in LR10 ?
NO
Write Error Message (FSER)
5A
YES
Obtain Group Specifications from First Field in LR2
3


3
Last Field in Group Last field Defined ?
NO
Write Error Message (FSER)
EXIT
YES
All Fields Defined ?
NO
Write Error Message (FSER)
5A
YES
All Fields Sequential ?
NO
Write Error Message (FSER)
5A
YES
All Field Types the Same ?
NO
Write Error Message (FSER)
5A
YES
First Field in LR10 ?
YES
4
NO
Write Error Message (FSER)
5A


4
First Field in LR10 ?
Write Error Message (FSER)
5A
Scan Field Specifications In LR2 & LR10 to Obtain Group Specifications
Build Group Entry for LR2
LR10 Filled ?
Write Error Message (FSER)
FSGRX-EXIT
Build LR10 Entry
5


5
Scan Group Fields Making all Sequence Numbers equal to Lowest Seq. Number
Scan Group Fields Making all Level Numbers equal to Lowest
Make Group Entries for LR2, LR7, & LR10 to be passed to Field
FSGRX-EXIT
5A
Find Last Field on GROUP Card
FSGRX-EXIT

FSGRX5

1. FSVSX (VSET Card).

(1) Summary. FSVSX processes the Variable Set card. The variable set consists of the defined variable field, which may appear only at the end of the data record. Up to 49 variable sets may be defined per file. Checks for correct format and length of the variable set. The VSET literal may not exceed 132 characters.

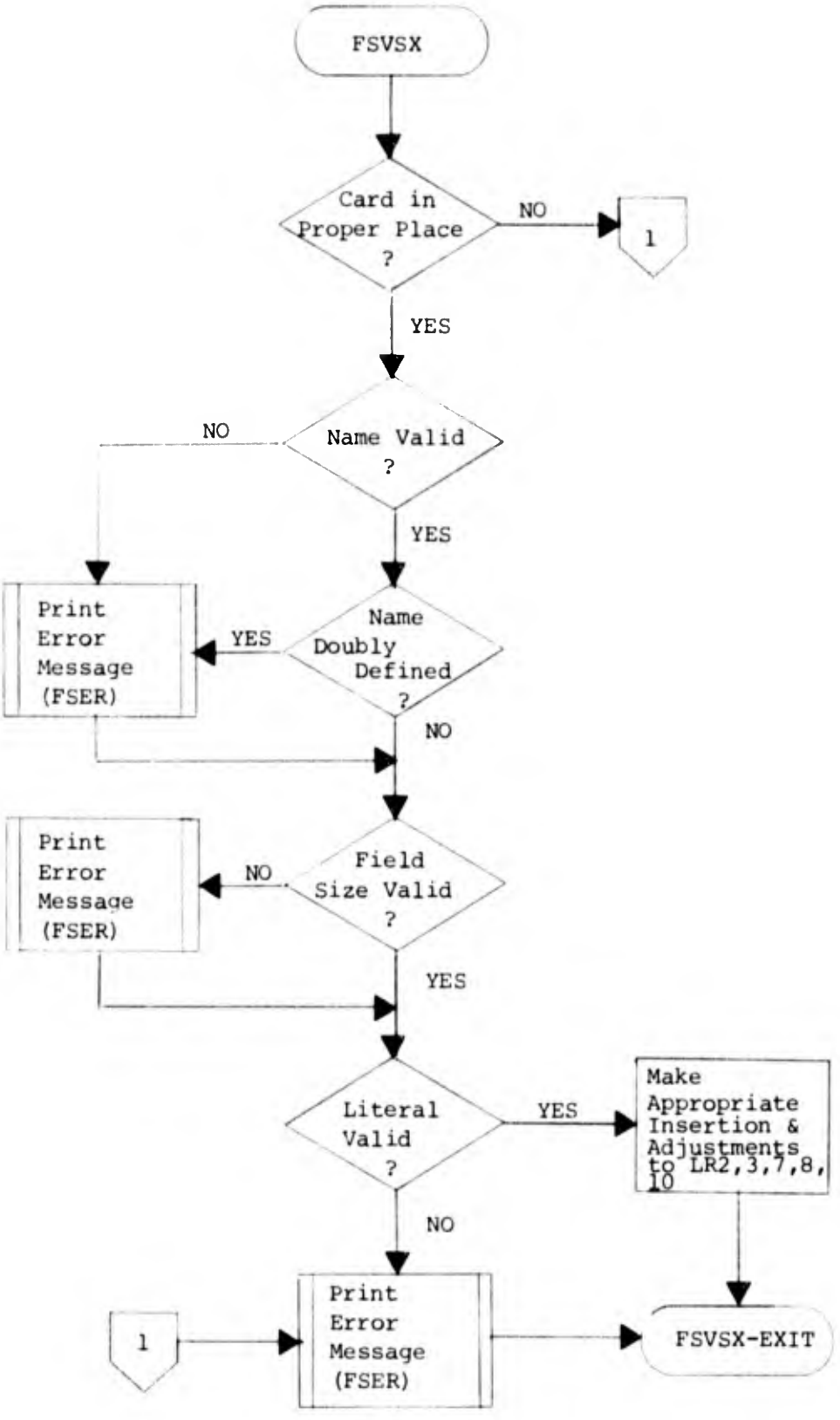
(a) Function. Processes the VSET card.

(b) Calling Sequence.

(2) Description.

FSVS. Determines if the VSET card is legal at this point.
FSVS-NAME-CHECK. Validates the format of the VSET name.
DUP-NAME-FSVS. Checks for doubly-defined name by performing a table lookup in LR2 for peviously used name.
WORD-CHK/WORD-CHK-RETURN. Checks for correct format of the VSET card and checks the length of the variable set literal for validity. If the literal size is not specified, a size of 100 will be assumed.
LIT-CHECK. Checks for valid output label.
ERROR-CHK. Verifies that the number of fields and/or the number of allowed sets has not been exceeded.
VSLR10-CHK/MIN-LEV-ASG. Assigns min-level to all field groups which are not contained within groups.
VSW-CHK/FS-VSET1. Sets switches for later use and assigns level numbers for date fields.
VS-ALL-CRD/VS-INSERT. Adjust sequence of LR10 items so that an item for VSC can be inserted.
VSLR10-BLD. Insert LR10 item for VSC (variable set).
LR2-BLD1/MOVE-LIT2. Generate LR2, 3, 7, and 8 items for variable set.
UPDATE-LR-LENGTH/FSVS-EXIT. Resets error switches.

(3) FSVSX Flowchart.


FSVSX
Card in Proper Place ?
NO
1
YES
Name Valid ?
NO
YES
Print Error Message (FSER)
YES
Name Doubly Defined ?
NO
Print Error Message (FSER)
NO
Field Size Valid ?
YES
Literal Valid ?
YES
Make Appropriate Insertion & Adjustments to LR2,3,7,8, 10
NO
1
Print Error Message (FSER)
FSVSX-EXIT

m. FSENX (ENDFS Card).

(1) Summary. FSENX completes the processing of certain fields in logical records 1, 2 and 7 and completes the fields of LR10 (which are at that time stored in the working storage WORK-LR10). FSENX generates logical record eleven (LR11) by performing an alphabetic sort of LR2's field mnemonics. The sorted results and two low values at the beginning of the table and two high values at the end of the table compose LR11. FSENX formats each of the eleven logical records for printing.

(a) Function. Processes the ENDFS card and controls the writing of Logical Records on the MIDMS Table Library.

(b) Calling Sequence.

```
ENTRY  'FSEN'  USING  FS-DD
CALL   'FSDT'  USING  FS-DD
CALL   'FSDD'  USING  FS-DD
MULTIPLE ENTRIES TO 'LB'
```

(2) Description.

X1/X4. Processing of LR10 is as follows:

Paragraph X1 examines the entries of LR10-ITEM in UNGROUPED-LIST. If any LR10-ITEM is not zero, the LR10-ITEM-Nth entry of LEV-NO is set equal to MIN-LEVEL.

Paragraph X2 sets LEV-NO(2) and LEV-NO(3) equal to MIN-LEVEL minus 1, provided CNTL-SW=W is not equal to one. If CNTL-SW is equal to one paragraph X2 is bypassed.

Paragraph X3/X4 moves the items of WORK-LR10 into FFT-LR10. Each item WORK-LR10-DATA is moved into that position of FFT-LR10-DATA which is shown only by the associated LR10-SEQ. For example:

WORK-LR10-DATA	LR10 SEQ	FFT-LR10-DATA
1 aaaaaaa	2	ddddddd
2 bbbbbbb	3	aaaaaaa
3 ccccccc	5	bbbbbbb
4 ddddddd	1	eeeeeee
5 eeeeeee	4	ccccccc

P1A/P1C1. Format and print LR1.
P2A/P2B1. Format and print LR2.
P3A/P3B1. Format and print LR3.
P4A/P4B1. Format and print LR4.
P5A/P5B1. Format and print LR5.
P6A/P6B1. Format and print LR6.
P7A/P7A1. Format and print LR7.
P8A/P8C. Format and print LR8.

Paragraph P8B first determines the length of the entry as the sum of the contents of fields REL-SIZE and OUTPUT-SIZE (from LR7). The size of the entry calculated is stored in the print area LR8-LI1-EN. The high-order position (leftmost position) of the entry is also stored in LR8-LI1-EN. This position is set at 1 for the first entry and then maintained by accumulating the total of all entry lengths.

P9A/P9C. Format and print LR9.

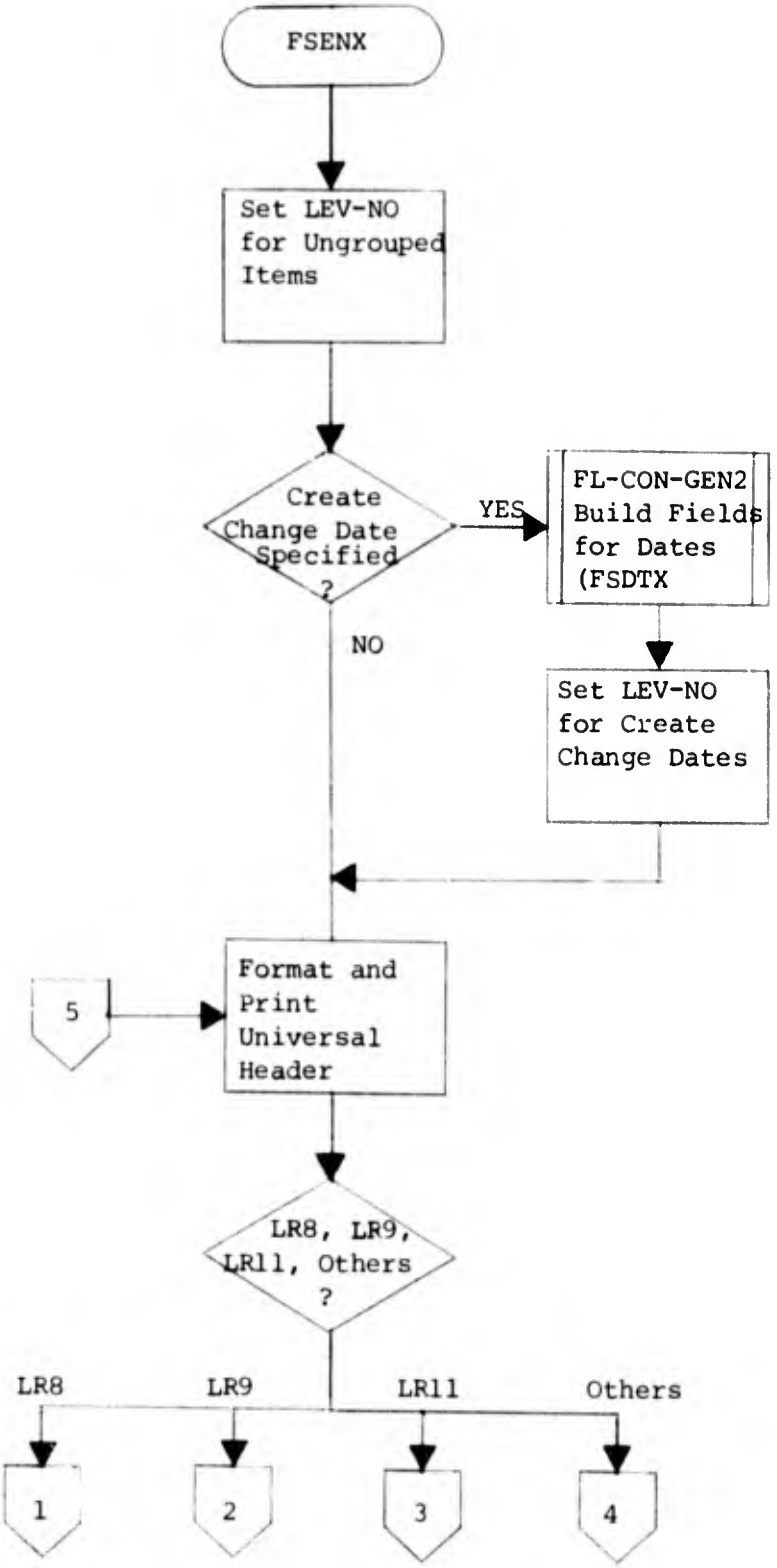
Paragraph P9A processes the entries of LR9 in much the same way as P8A works on LR8. The main difference here is that the length of the entry is found in decimal form as 3 characters preceding the entry proper. More precisely, the entry looks like this:

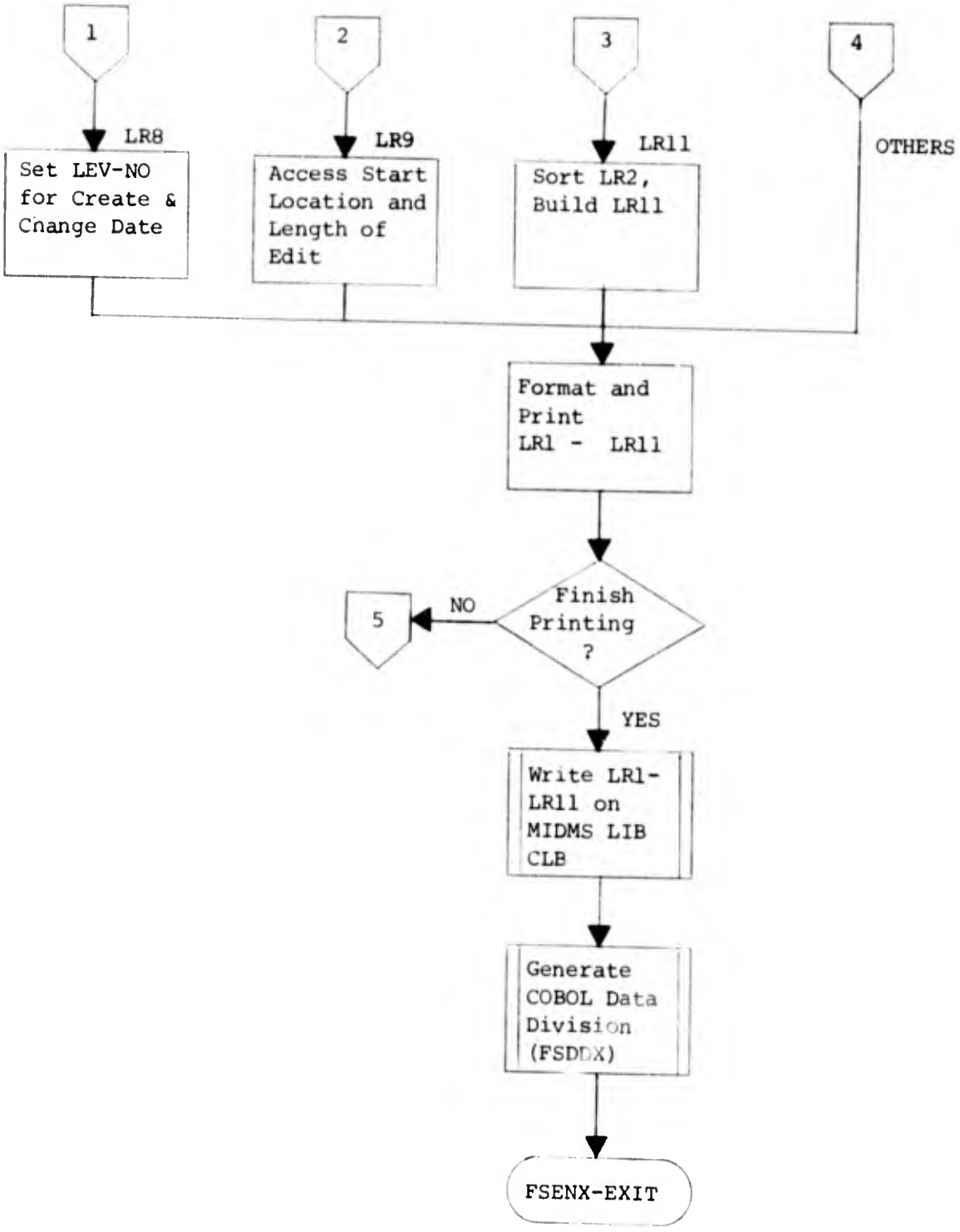
aaabbbxx...x

where aaa is the length of the string, xx...x (decimal form) and bbb is the length of the field to be processed by xx...x as a data-word. The number aaa must be transformed from a PICTURE XXX to a PICTURE 999 field in order to be used computationally as the length of field xx...x. This is done by moving aaa into field ALPHA-TRIPLET with PICTURE of xxx and redefining it as NUM-TRIPLET with PICTURE of 999.

P10A/P10C1. Format and print LR10.
LR11-SORT/LR11-SORTING. Sorting of LR2's field nmemonic to produce LR11.
P11A/P11C1. Format and print LR11.
LIB-FFT-WRITE/LIB-LR-EXIT. Handles the calling of Librarian subroutine in order to write structured FFT's on the MIDMS Table Library.
FFT-COUNT/FSEN-EXIT. These paragraphs calculate the total size of an FFT, print the size out and than return control to the calling subroutine.

(3) FSENX Flowchart.


FSENX
Set LEV-NO for Ungrouped Items
Create Change Date Specified ?
YES
FL-CON-GEN2 Build Fields for Dates (FSDTX
NO
Set LEV-NO for Create Change Dates
5
Format and Print Universal Header
LR8, LR9, LR11, Others ?
LR8
LR9
LR11
Others
1
2
3
4


1
2
3
4
LR8
LR9
LR11
OTHERS
Set LEV-NO for Create & Change Date
Access Start Location and Length of Edit
Sort LR2, Build LR11
Format and Print LR1 - LR11
5
NO
Finish Printing ?
YES
Write LR1-LR11 on MIDMS LIB CLB
Generate COBOL Data Division (FSDDX)
FSENX-EXIT

n. FSDDX (COBOL Data Division).

(1) Summary. FSDDX interprets logical record ten (LR10) and based on data found in LR10 a COBOL data division is produced and stored on MIDMS Table Library. When a Logical Maintenance run is made COBOL programs are generated to perform certain tasks. The COBOL programs use the generated COBOL data division which FSDDX has stored in the MIDMS Table Library.

(a) Function. Generates COBOL Data Division.

(b) Calling Sequence.

```
ENTRY  'FSDD'  USING  FS-DD
```

(2) Description.

PROCEDURE DIVISION. Receives control from calling subroutine.
FSDD. Opens the temporary Data Division file. Checks to see if the data division is already on the Library or whether this is the first creation of it.
READ-ITEM. Checks for end of input, checks for change of sets. Determines the type of Data Division statement to be generated.
CREAT-FILLER. When a filler is required, the appropriate level number, size and picture is moved to the filler print line.
FF/FN. Tests for the variations in the data division statements to be generated. Possible values are 'D' for DEFINE, 'N' for define of numeric group, 'S' for special define used for overlapping groups. 'R' for REDEFINE also used in overlapping groups and 'F' for filler.
DISPLAY1. Performs the writing of a generate statement and controls the reading of more data to create a new statement.
FFD/SE. Determines if picture is numeric or alphanumeric.
ADD-UU/ADD-P. Handles the adding of -U to a user defined item name and the adding of 'S,' 'R,' 'N,' or '.' to the item name area.
FSD/WX. Processes special defines (SD) and performs the generation of description for GROUP DEFINE (GD). Writes generated statement out.
FGS. This paragraph processes the group special define (GS). It must be noted that the -S must be added.
FGR/FGROUP-R. This coding processes the GROUP REDEFINE.
FGN. This paragraph is used to process GROUP numeric type card (GN). The action required is the generation of two cards--the first is simply a group card with a picture and the second is a redefinition of the field.

FPD/NOGEN. Controls the execution of a Periodic Define (PD). When the first PD item is encountered the following items must be generated:

```
46  PSC-INFO
  48  PSC-DATA OCCURS (CNT-LR3(3)-1) TIMES.
    49  PSC-CNT  PICTURE 9(4).
    49  PSC-HOP  PICTURE 9(4).
46  PSC-INFO-R REDEFINES PSC-INFO.
```

FFF/F-PICTURE. These two paragraphs generate a filler card required by the redefinition of a group. Filler cards are identified by FF in LR10.

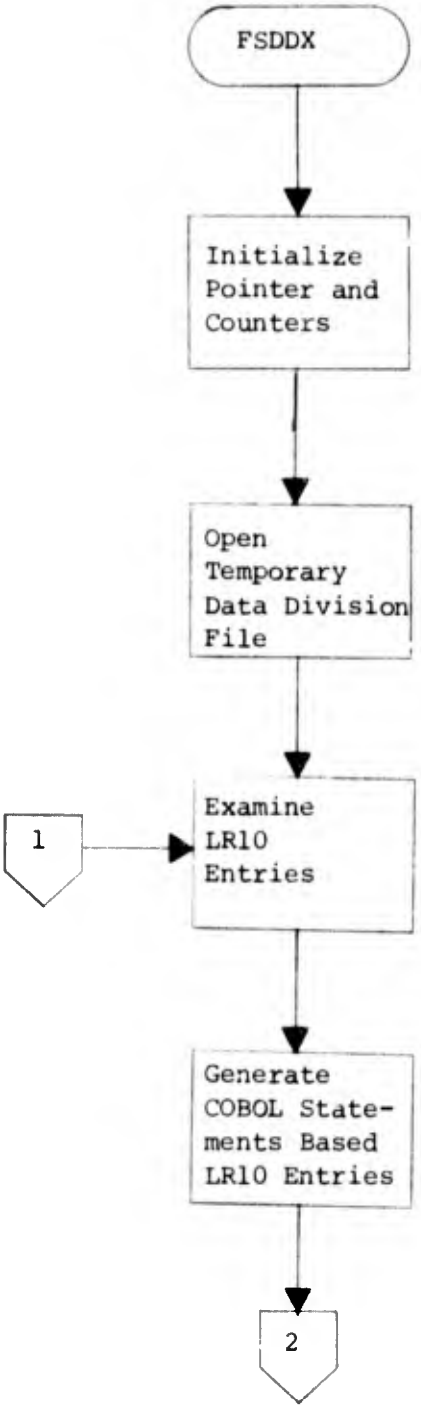
FLD. When the definition of the control field is detected, the record control group (RECID) is generated. The name 'REC-CTL' is move to the GROUP name. The level number is taken from LR10. The RECID will consist of all field/group definitions having a level number algebracally higher than the REC-CTL level number.

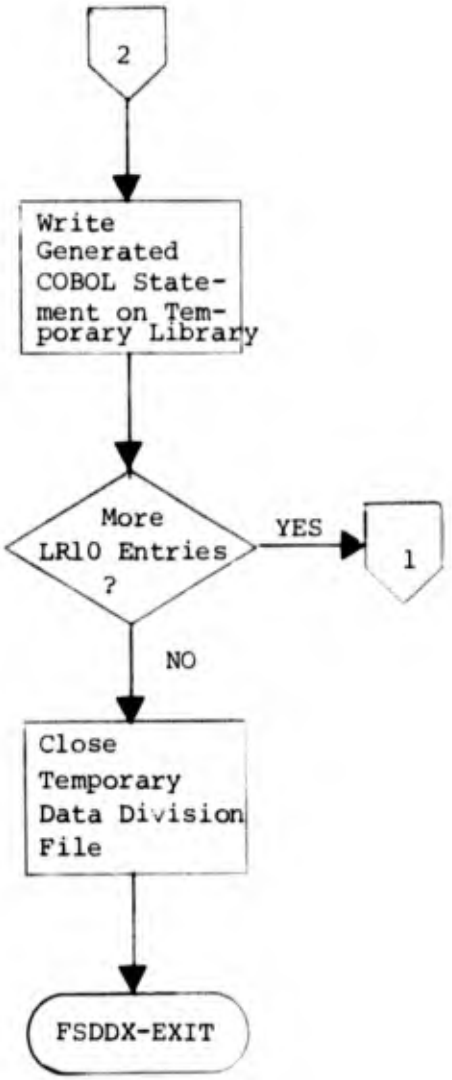
START-PSET. This paragraph handles the reinitializing of the counter used for the start of a new periodic set. It also determines if a variable set 'N' has been detected in which case the generation is terminated.

WRITEL/WRITET. Handles the writing of a generated record.

END-OF-FFT/FSDD-EXIT. Determines whether all items have been generated before returning control to the calling subroutine.

(3) FSDDX Flowchart.


FSDDX
Initialize Pointer and Counters
Open Temporary Data Division File
1
Examine LR10 Entries
Generate COBOL State-ments Based LR10 Entries
2


2
Write Generated COBOL State-ment on Tem-porary Library
More LR10 Entries ?
YES
1
NO
Close Temporary Data Division File
FSDDX-EXIT

4. FILE STRUCTURING ERROR MESSAGES.

a. 'ASSUMED ALPHA INPUT FUNCTION'

(1) FSIOX: ALPHA-ASSUMED.

(2) No input function was specified on the field or group card. System assumed the data to alphabetic. Check for correctness.

b. 'ASSUMED BLANK OUTPUT FUNCTION'

(1) FSIOX: BLANK-ASSUMED.

(2) No output function was specified on a FIELD/GROUP card. The system assumed that no output conversion is needed. Check for correctness.

c. 'ASSUMED CONTROL FIELD'

(1) FSGOX: BUILD-SETYPE

(2) No set type specified and the card was either the first FIELD in the Format Definition Deck or the previous defined field contained control data.

d. 'ASSUMED FIXED FIELD'

(1) FSGOX: BUILD-SETYPE

(2) No set type was specified and previous defined field contained fixed data. Check for correctness.

e. 'ASSUMED PERIODIC SET'

(1) FSGOX: BUILD-SETYPE

(2) "N" set type specification or omission of the set type resulted in the automatic assignment of periodic set numbers. (The assigned set number is represented by the 'NN' of the advisory message.)

f. 'ASSUMED VSET SIZE, 100 CHARACTERS'

(1) FSVSX: WORD-CHK-RETURN

(2) There was no size specification on the VSET card. The system assumed a size of 100.

g. 'CARD ILLEGAL HERE'

(1) FSX: ED-CHECK-PERFORM
FSGRX: FSGR
FSFLX: FSFL
FSVSX: FSVS

(2) Card type is out of sequence, see the MIDMS User's Reference Manual, chapter 2, figure 15 (FS Deck Squence), for correct card type sequence.

h. 'CARD MUST PRECEED THE FIELD-GROUP CARDS'

(1) FSSBX: FSSB

(2) A SUB or TAB card is out of sequence. All subroutines and tables must be defined before any FIELD or GROUP cards are defined. Place cards in correct sequence and rerun job.

i. 'CNTL FLD MUST BE THE FIRST FLD DEFINED'

(1) FSFLX: CNTL-SET

(2) A control field was defined after fixed or periodic field was defined. Correct the card sequence and rerun job.

j. 'COLUMN 4 OF CARD MUST CONTAIN B OR F'

Not used at present.

k. 'CONTROL FIELD EXCEEDS 30 CHARACTERS'

(1) FSIOX: FL-SUMMARY

(2) The maximum size of the RECORD ID is 30 characters. Redesign the FDD to reflect the correct size and rerun job.

l. 'CREAT, CHANGE or TESTIT MODE REQUIRED'

(1) FSSBX: TEST-MODE

(2) Modes of operation acceptable to FS are CREATE, CHANGE or TESTIT. Check columns 14 through 19 of FSJOB card for correct mode. Correct and rerun job.

m. 'DOUBLY DEFINED EDIT FIELD ID'

(1) FSEDX: TEST-NAME

(2) A previously defined item has the name. Make the name unique and rerun the job.

n. 'DOUBLY DEFINED MNEMONIC'

(1) FSSBX: NAME-CHECK
FSGRX: DUP-NAME-CHK
FSFLX: LR2-LU
FSVSX: DUP-NAME-FVS

(2) A previously defined subroutine, table, field or group has the name. Make name unique and rerun job.

o. 'DOUBLY DEFINED SUBROUTINE ID'

(1) FSSBX: NAME-CHECK

(2) A previously defined item has the name. Make the name unique and rerun the job.

p. 'ELEMENT FOR FIND BASE EXCEEDS 20 CHARS'

Not used at present.

q. 'ENDFS CARD IS MISSING'

(1) FSX: END-OF-CARDS

(2) The ENDFS card was omitted. Place an ENDFS card after the last FDD in the job and rerun.

r. 'EXCEEDS MAXIMUM RECORD SIZE'

(1) FSFLX: MOVE7
FSGOX: MOVE6
FSDTX: ADD-LR3-FIX
FSVSX: LR3-BLD
FSENX: FIX-CHK

(2) The total of field sizes is greater than 10,000. Redesign FDD to reflect a size under 10K characters and rerun job.

s. 'EXCEEDS CAPACITY OF EDIT FIELD'

(1) FSIOX: FL-EDTBL

(2) The EDIT card is accepted by MIDMS (FS) for compatibility only. Correct card to reflect the proper edit or delete the EDIT card and all references to it in every FIELD or GROUP card.

t. 'EXTRA PARAMETERS ON STATEMENT IGNORED'

(1) FSEDX: TEST-WORDCNT
FSSBX: LIT-CHECK-FSSB
FSIOX: FL-ENTRY1
FSVSX: WORD-CHK

(2) All parameters for the particular card type have been satisfied. Additional words on card will be treated as comments by the system. Check for correctness.

u. 'FFT IN LIB. IT WILL BE CHANGED'

Not used at present.

v. 'FIELD NOT IN LR1O-PROGRAM ERROR'

(1) FSGRX: LR1O-SEQF-PRFM,LR1O-CRDCHK-PFRM

(2) See a MIDMS system programmer.

w. 'FIELD OUT OF SEQUENCE'

(1) FSGRX: GR-HOP-CHK

(2) The mnemonics specified in the field list of the GROUP card are not in the same sequence as the fields defined in the FDD. Correct and rerurn job.

x. 'FIELD SIZE INVALID'

(1) FSFLX: NOT-IN2, FLSIZ-ERR,
FSIOX: TEST-NUM-SIZE,
FSVSX: WORD-CHK-RETURN,LIT-CHECK

(2) Field size may be one, two, or three digits with leading zeros permitted. The field size of numeric or signed numeric data may not be greater than 18 digits.

y. 'FIELDS HAVE DIFFERENT SET ID'S'

(1) FSGRX: GR-SET-CHK2, GR-HOP-CHECK

(2) Two or more fields in the field list of the GROUP card are members of different sets. Correct and rerun job.

z. 'FIELDS HAVE DIFFERENT TYPE ID'S'

(1) FSGRX: GR-HOP-CHECK

(2) Two or more fields in the field list of the GROUP card have different type ID's. Correct and rerun job.

aa. 'FILE ID NOT AVAILABLE IN FICON TABLE'

Not used at present.

ab. 'FIND ID OUT OF SEQUENCE'

Not used at present.

ac. 'FUNCTION LENGTH CANNOT EXCEED 50'

Not used at present.

ad. 'FUNCTION REQUIRES SUBSCRIPT'

(1) FSIOX: FL-IP-SIZE, OP-SBTBL

(2) The subroutine or table specified should be a multiple output function. Specify the appropriate subscript on the function and rerun job.

ae. 'HAD SAME ID AS SUBROUTINE'

Not used at present.

af. 'ILLEGAL CARD TYPE'

(1) FSX: WRONG-CARD

(2) See DIAM 65-9-9, MIDMS User's Reference Manual, chapter 1 for correct card types.

ag. 'ILLEGAL FILE NAME'

(1) FSJBX: BAD-FILE-NAME

(2) MIDMS file name must be five characters; the first character must be alphabetic and the fifth character must be an "A." The other characters ajj be letters or digits. Check columns 21 through 25 for correct file name. Correct and rerun job.

ah. 'ILLEGAL MNEMONIC'

(1) FSFLX: FSFL
FSEDX: TEST-NAME-CHAR
FSGRX: FSGR
FSVSX: FSVS-NAME-CHK

(2) A field mnemonic may be one to five characters; the first character must be alphabetic, no embedded blanks and no special characters are allowed. Correct and rerun job.

ai. 'ILLEGAL SET ID'

(1) FSGOX: Go-ON-1

(2) Set type specified is not periodic. Correct and rerun job.

aj. 'ILLEGAL SUB BASE'

Not used at present.

ak. 'ILLEGAL SUB NAME'

(1) FSSBX: FSSB

(2) Subroutine and table names must be five characters long, the first character must be alphabetic, the fifth character must be "S," and the other characters may be letters or digits. Correct and rerun job.

al. 'ILLEGAL SUBSCRIPT'

(1) FSSBX: M-SS-CHK
FSIOX: FL-IP-SZ, FL-SZ-9

(2) Check multiple output functions for valid pound signs (#) and for valid subscript sequence. Correct and rerun job.

am. 'INCORRECT GROUP TYPE-PROGRAM ERROR'

(1) FSGRX: LR10-SEQ-CHK, GR-TYPE-TEST

(2) See a MIDMS system programmer.

an. 'INVALID LITERAL'

(1) FSIOX: EXTRACT-CHK

(2) The FIELD or GROUP extract specification is incorrect or the literal on a VSET card is incorrect. See the MIDMS User's Reference Manual, chapter 2, paragraphs 5.c.(5), (6), and (7), for correct specification and rerun the job.

ao. 'INVALID OPTION'

(1) FSJBX: OPTION-ERROR
FSVSX: LIT-CHECK

(2) Each FSJOB card is exactly four characters long and is separated by a preceding comma. Valid options may be in any sequence, i.e. DATE, LSTn XCHG, HFIX, HALL, etc. See MIDMS User's Reference Manual, chapter 2, paragraph 5.c.(1), for detailed explanation. Check options for validity starting with column 26 of the FSJOB card. Correct and rerun job.

ap. 'JOB NOT COMPLETED BECAUSE OF ERRORS'

(1) FSX: JOB-ERROR

(2) The FDD contains errors, correct and rerun job.

aq. 'LAST FIELD IS NOT LAST FIELD DEFINED'

(1) FSGRX: GR-SEQ-CHK

(2) The last name in the field list specification of the GROUP card is the immediately previous field or group that was defined. Correct and rerun job.

ar. 'LR2 OVERFLOW'

(1) FSDTX: FL-CON-GEN2
FSGOX: MOVE4
FSIOX: FL-SUMMARY
FSVSX: ERROR-CHK

(2) More than 300 fields, groups and system generated mnemonics have been placed in logical record two. Redesign FDD to reflect the proper number of mnemonics by adding (1 for the RECORD COUNT, and 2 if the DATE option is used on FSJOB card, 2 for each periodic set defined, and 1 for each variable set defined) to the number of fields and groups defined in the FDD.

as. 'LR3 OVERFLOW'

(1) FSGOX: MOVE3
FSVSX: ERROR-CHK

(2) More than the maximum of 50 sets have been defined. Correct FDD to reflect the proper number of sets and rerun job.

at. 'LR4 OVERFLOW'

Not used at present.

au. 'LR5 OVERFLOW'

Not used at present.

av. 'LR6 OVERFLOW'

Not used at present.

aw. 'LR8 OVERFLOW'

(1) FSDTX: FL-CON-GEN2
FSIOX: FL-LAB-MOVE
FSGOX: GO-ON
FSVSX: LR8-BLD1

(2) More than 3000 characters have been placed in Output Extract Label table. Shorten the output extract label on each FIELD or GROUP card and rerun job.

ax. 'LR9 OVERFLOW'

(1) FSIOX: FL-EDT-MOVE1, FL-EDT-3

(2) More than 3000 characters of edit words have been put into logical record nine. Remove all edit specifications and rerun job.

ay. 'LR10 OVERFLOW'

(1) FSDTX: F1-CON-GEN2
FSGOX: FLTYPEMOVE2
FSIOX: FL-SUMMARY
FSVSX: VSLR10-BLD
FSGRX: LR10-BLD

(2) LR10 contains information necessary to build COBOL Data Division. Redesign FDD to reduce the number of COBOL statements to be generated and rerun job.

az. 'MAX NUMBER OF FIELDS PER GROUP EXCEEDED'

(1) FSGRX: FSGR

(2) There are FIELD specifications in the field list of the GROUP card than what is allowed. See the MIDMS User's Reference Manual, chapter 2, paragraph 5.c.(6), for correct specification.

ba. 'MISSING ASTERISK'

Not used at present.

bb. 'MISSING QUOTE ON EDIT LITERAL'

(1) FSEDX: TEST-LITCNT

(2) A quote has be omitted from the literal on the EDIT card. Correct edit literal and rerun the job.

bc. 'MORE THAN THREE CONTINUATION CARDS'

(1) FSX: ERROR-PRINT

(2) Only the original and three additional cards are used for continuation purposes.

bd. 'NEXT JOB CARD OUT OF ORDER'

(1) FSX: ED-CHK-PERFORM

(2) FSJOB signals the beginning of a group of input cards containing the data for constructing an FFT. There must be one FSJOB card for each FFT.

be. 'NOT IN LIBRARY'

Not used at present.

bf. 'O/P OF I/P FUNCT. NOT EQUAL TO FLD SIZE'

(1) FSIOX: FL-IP-Sz-CHK, FL-IP-SIZ

(2) The field size is greater than the output size of the subroutine or table specified. Correct and rerun job.

bg. 'OUTPUT SIZE IN ERROR'

(1) FSSBX: WORD-CHK-FSSB, MULTI-OUT-CHK

(2) An output size may be one, two or three digits. Correct and rerun job.

bh. 'RECCT CANNOT BE GROUPED'

(1) FSGRX: GR-SET-CHK1

(2) "RECCT" is specified in the field list of the GROUP card. Delete "RECCT" specification and rerun job.

bi. 'SET ID OUT OF SEQUENCE'

(1) FSGOX: CHKCTL-LEN, FL-X, GO-ON-1
FSFLX: FL-X

(2) Set types are placed in an arrangement other than C,X,Ø1,.... See the MIDMS User's Reference Manual, chapter 2, figure 19 (FIELD Card), for detailed explanation.

bj. 'SUBROUTINE OR TABLE DEFINED BUT NOT USED'

(1) FSX: FS-ENDFS

(2) The subroutine or table was not referenced by a FIELD or GROUP card.

bk. 'SUBROUTINE TABLE OVERFLOW'

(1) FSSBX: TAB-BUILD

(2) Exceeds maximum table size of 100. Redesign FDD to reflect proper number of subroutines and rerun the job.

bl. 'SUBROUTINE USED FOR BOTH I/P AND O/P'

(1) FSIOX: FL-IP-Sz, OP-SBTBL

(2) The same subroutine or table is used for the input function as well as the Output functions. Make correct specification and rerun job.

bm. 'THE 4TH CHAR OF SUBNAME MUST BE N OR T'

Not used at present.

bn. 'THIS CARD FORMAT CANNOT BE FIRST IN SET'

Not used at present.

bo. 'UNDEFINED FIELD/GROUP ID OR OPR CARD'

Not used at present.

bp. 'UNDEF.NED FIELD ID'

(1) FSGRX: GR-SET-CHK1, GR-FLD-FIND-PFRM, GR-FLD-PFRM

(2) The name in the field list of a GROUP card has not been defined as a field.

bq. 'UNDEFINED INPUT FUNCTION'

(1) FSIOX: UNDEF-INPUT

(2) The input function specified is not "ALPHA," "NUMER" or "SGNUM" and it is not a valid subroutine or table. See the MIDMS User's Reference Manual, chapter 2, figure 19 (FIELD Card), for correct specification. Rerun the job.

br. 'UNDEFINED OUTPUT FUNCTION'

(1) FSIOX: UNDEF-OUTPUT

(2) Subroutine or table specified was not found in the subroutine table. Correct and rerun the job.

bs. 'UNDEFINED SUBRT SPECIFIED FOR BASE'

Not used at present.

5. QUANTITATIVE LIMITS. One of the considerations in the design of a data file is the quantitative limits placed on the various data elements. These limits have been set, based on the limits of an earlier system (1410 Formatted File System Mark III). MIDMS programs have been written in a manner which allows several of these limits to be altered without the necessity of significant reprogramming (assuming that the computer configuration can accommodate the change). MIDMS has the following restrictions currently (with absolute limitations indicated, based on conditions which cannot be easily changed).

a. 9999 Characters/Data Record (depending on storage device).

b. 50 Set/Record - 1 Fixed, 49 Periodic or Variable (can be increased to 99 Periodic plus 99 Variable).

c. 600 Subsets/Set (cannot be increased due to FM convention).

d. 299 Fields or Groups/Record - including MIDMS-generated control fields (can be increased to 450).

e. 999 Characters/Alpha Field (Alpha groups have no size limitations other than that of the record size).

f. 18 Digits/Numeric Field or Group (COBOL limitation).

g. 30 Characters/Record Control Group (can be increased to 99).

h. 132 Characters/Extract Label or Edit Word (cannot be increased due to printer size, a lower maximum may be necessary).

i. 3000 Characters/Total of Extract Labels - including labels for MIDMS-generated control fields (can be increased to 9999).

j. 3000 Characters/Total of Edit Words and Control Information - six characters are added to each edit word (can be increased to 9999).

k. 45 Levels of Grouping Hierarchy/Set - groups within groups within groups (overlapping groups result in extra labels - cannot be increased).

l. 500 Lines in Generated COBOL Data Division/File - 3 lines/set, 1 line/field, at least 1 line/group depending on hierarchical relationships (can be increased to 1400).

m. 100 SUB or TAB Cards/FS Deck (can be increased to 900).

n. 300 Total Subscripts on SUB or TAB Cards/FS Deck - 1 to 9 per SUB or TAB card; 1, if no subscripts specified (can be increased to 999).

o. 100 EDIT Cards/FS Deck (can be increased to 450).

CHAPTER 2

Librarian

1. SUBSYSTEM STRUCTURE. The MIDMS Librarian is a collection of programs that act as the interface between the MIDMS functional programs, FS, FM, RTOP, LIBUT, etc. and a MIDMS library. Each functional program calls the main librarian module, LBLB, when it wishes to read from, write to or delete members on the library. The Librarian consists of COBOL program LBLB and several assembly language subprograms.

2. SUBPROGRAM IDENTIFICATION AND DESCRIPTION. The Librarian consists of one COBOL program LBLB and seven assembly language programs.

LBLB performs all the library operations, read, write, delete, and size and is the program called by the functional programs.

LBCMPRS compresses a string of data characters by replacing four or more consecutive blanks with a three character blank compression code.

LBEXPAND expands a compressed data string produced by LBCMPRS into its original format.

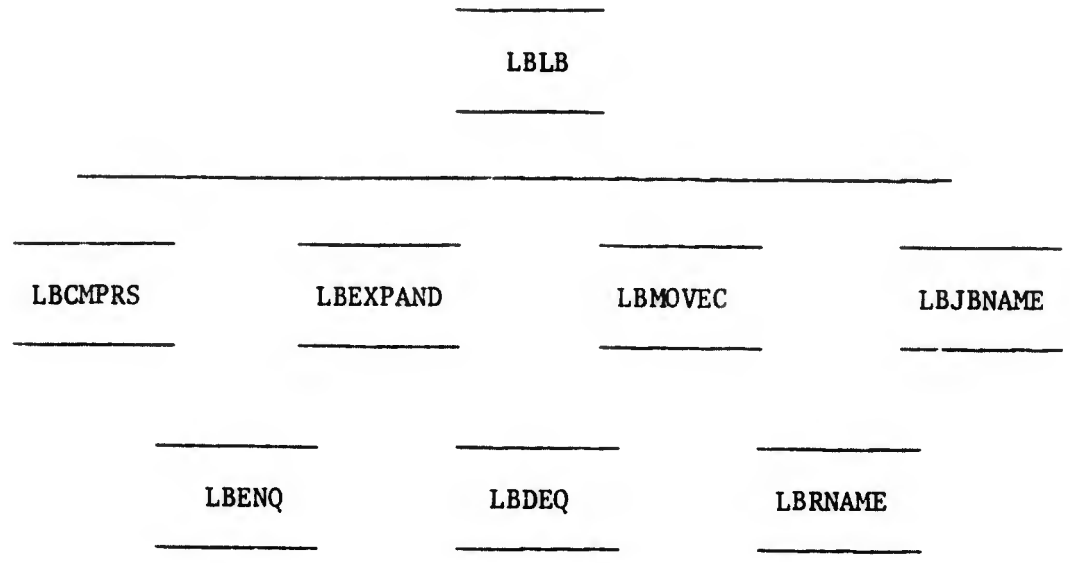
LBMOREC is used to increase the speed of character moves.

LBJBNAME obtains the job name specified in the // JOB card for inclusion in the directory record.

LBENQ obtains exclusive or shared control of the library being accessed.

LBDEQ releases control of a library when LBLB has finished processing.

LBRNAME provides the data set name of the library to the calling program.



a. LBLB.

(1) Abstract.

(a) Function. This program provides the interface between MIDMS processing modules and the MIDMS library. It services requests from these modules for reading, writing, and deleting members of the library. During its operation, LBLB calls several assembly language subroutines which perform special services. In addition to servicing MIDMS modules requests, it also maintains the STATISTICS Record (Record 0) of the library which contains a variety of data necessary for the library operation.

(b) Calling Sequence. This program is called at entry point 'LB' and is passed two parameters:

1. The address of an 18-byte area containing the following four fields:

a. A one-byte code which informs LBLB of the action to be performed, W=write, R=read, D=delete, S=size.

b. A four-byte size field which either contains the size in bytes of the entry to be written or will be filled in with the size of the entry which is read by LBLB.

c. An eight-byte name field which contains the name of the library entry to be affected.

d. A date field which either contains a date indicating when the entry was written into the library or which will be filled in with a date when an entry is read.

2. The address of a 10,000 byte area which either contains the entry to be written into the library or is filled in by LBLB with an entry from the library.

(2) Description.

(a) LB is called by a MIDMS processing module when it wishes interaction with a MIDMS library. The data fields described in paragraph 2.a.(1)2. are passed as parameters and determine the actions LBLB will perform. When entered LBLB immediately calls LBENQ to request control of the MIDMS library. This insures that no other LBLB program in another job will modify the library during the time LBLB is executing. After gaining control of the library, the switch "WAS-DELETED-BY-LB-DELETE" is blanked. This switch will be turned on if an entry is deleted by the LB-DELETE portion and will indicate the proper return code to be placed in LKOP when LBLB returns to the caller. The original LKOP is saved in LKOP-SAVE and examined during clean-up to indicate whether the statistics record 0 should be written to the library. The library file is then opened as a relative I/O file to be accessed by physical block number. The block number field "RECNO" is set to 0 indicating file physical record 0 which is then read into STAT-REC since record 0 in a MIDMS library is the statistics (see paragraph 2.a.(4) for the statistics record format). LKOP is now checked to determine the operation to be performed. LB-READ processes "R" and "S," LB-WRITE processes "W," "L," "A" and "B" and LB-DELETE processes "D." If LKOP is not one of these codes, a "not done" response, "N," is placed in LKOP and LBLB exits. The calling program determines successful or not successful operation of the library request by examining LKOP when LBLB returns control. If "N," LBLB could not successfully perform the operation requested and the calling program should take the appropriate action.

(b) LB-DELETE through LB-DELETE-EXIT deletes an entry from the library and adjusts the statistics record. It first moves the name of the entry to be deleted to DR-SEARCH-NAME and the first letter of the name to DR-SEARCH-LETTER and performs FIND-DIREC-ENTRY which searches the library directory records for an entry with that specific name. If RECNO equals 0 after FIND-DIREC-ENTRY has completed, a directory entry for the given name was not found (i.e., the given entry does not exist in the library). If LKOP equals "D," which means the calling program wishes to delete the entry, an "N" is placed in LKOP indicating the entry does not exist and LB-DELETE exits. If LKOP does not equal "D," then LB-DELETE was performed by LB-WRITE prior to writing an entry to the library and LB-DELETE exits without altering LKOP. If RECNO does not equal 0, then the directory entry for the member to be deleted was found and RECNO contains the record number of the

directory record containing the directory entry, the directory record is setting in DIREC-REC of LIB-FILE and I contains a number from 1 to 16 indicating the appropriate directory entry in the directory record. Since we will delete the member from the library, "WAS-DELETED-BY-LB-DELETE" is turned on to so indicate. The directory entry is moved to a work area, DIREC-ENTRY-WORK to save its contents, the entry slot in the directory is set to all '9' to indicate an emptry directory entry and the directory record is rewritten to the library. The operation has deleted the member's entry from the directory record but has not deleted the data portion of the member. The counter "STAT-DIREC-ENTRIES-USED" in the statistics record is decremented to indicate one less directory entry used. The field "DWPTR" in the directory record, which points to the first data record associated with the entry being deleted is moved to RECNO in order to free the data records. If this pointer is 0, that indicates there are no data records associated with the member, the delete process is finished and LB-DELETE exits. If the pointer is not 0, the data record is read, the next record pointer in the data record is saved in J, the entire data record is blanked out, the fields "THIS-REC" and "NEXT-REC" are placed back in the data record from RECNO and J, respectively, and an "F" is moved to REC-TYPE to indicate it is a free record. If the next record pointer is not equal 0 indicating that another data record follows the one being freed, one is subtracted from the number of data records counter, one is added to the free records counter in the statistics record, the freed record is rewritten to the library, the next record pointer is moved to RECNO from J and control returns to LBD-READ-DATA-REC to read and free the next data record. If the next record pointer is 0 indicating this is the last data record associated with the member being deleted, the data records and free records counters are decremented, the record number of the first free record on the free record chain is moved to the next record pointer field of the free record being worked on to effectively add the records just freed to the beginning of the free record chain, the current free record is rewritten to the library and the record number of the first record of the chain just deleted is placed in STAT-FREE-REC-PTR in the statistics record to complete adding the records freed to the free records chain. At this point the library member has been completely deleted and LB-DELETE exits.

(c) LB-READ through LB-READ-EXIT performs two functions for a calling program. It will return the size of a member of the library if LKOP is "S" and will read the member into a work area if LKOP is "R." The first action LB-READ performs is try to find the member's directory entry. It does this using FIND-DIREC-ENTRY as LB-DELETE did. After FIND-DIREC-ENTRY returns, if RECNO is 0, the directory entry was not found, "N" is placed in LKOP to indicate this and LB-READ exits. If RECNO is not 0,

the directory entry is moved to the directory entry work area, the size from the directory entry is moved to LKSIZE in the calling parameters and also saved in SIZEX for use later during possible moving of the data portion of the member. The date is moved to the calling parameter. LKOP is now checked to see which operation is to be performed. If LKOP is "S," only the size of the member is requested and LB-READ exits. If LKOP is not "S," then it is "R" and the member data is desired. If the size of the member is 0, then LB-READ exits since there is no data to be moved. If the size is no 0, the pointer to the first member data record is placed in RECNO and the data record is read. At this point SIZEX is checked to determine the type of data move which should be performed. If SIZEX is less than 501 indicating that this will be the last data moved for the member, subroutine LBMOVEC is called to move the remaining characters into the receiving area. If SIZEX is greater than 500, this means that the full 500 characters of the data record are to be moved and the data is moved to the 500-character segment indicated by the segment number contained in the dat record. The segment numbers begin at 1 and increase by 1 for each subsequent data record up to a maximum of 20. After the data has bee moved to the receiving area, SIZEX is decremented by 500. The next record pointer is moved to RECNO from the data record. If the pointer is 0, this is the last data record of the member and reading of data records ceases. If the pointer is not 0, the next record is read and moved. After the entire data portion of the member has been moved, the REC-FORMAT field is checked to determine the format of the data stored on the library. If it is "C," the data is stored in a compressed format with each four or more consecutive blanks replaced by a three character code. Before returning to the caller, the data has to be expanded to its original format. This is done by performing EXPAND which calls subroutine LBEXPAND. At this point, the read function is completed and LB-READ exits.

(d) LB-WRITE writes a member to the library. The data written is passed to LBLB by the calling program in an up to 10,000 character area. LB-WRITE first performs LB-DELETE to insure that the member being written does not exist on the library. When LB-DELETE returns, the member to be written is not on the library. An "E" is moved to the REC-FORMAT-X field to indicate a default expanded format for the data. This will be changed to "C" if the data is later compressed. If LKOP is "A" or "B," the data being written came from a backup tape and is already compressed. "C" is moved to a REC-FORMAT-X and the compress test is bypassed. If the fifth letter of the member is "Q" or "R" or "F" indicating that the member is a stored query, report as flysheet or it is a member indicating the member is a data division produced

by File Structuring, COMPRESS is performed which eliminates each four or more consecutive blanks and replaces them with a three character code. After the compression process has been done, the number of data records which will be written is computed. Each data record will contain 500 characters. The last data record will contain 500 or less characters. The number of data records required for this member plus one for a possible new directory record is compared to the number of free records in the library. If there are not enough free records left in the library to accommodate the new member, "N" is placed in LKOP and LB-WRITE exits. If there are enough free records, the directory entry for the new member is now created in the directory entry work area, DIREC-ENTRY-WORK. The name and size come from the calling parameters. The size is also placed in SIZEX for use later when the data is actually moved to data records. The date field is initially filled in from the LKDATE6 field of the calling parameters. If the date is legal, it is accepted; otherwise spaces are moved in. If the date is not legal, the current system date is used. If LKOP equals "W" or "A," the job name of the current job is placed in the job name field of the directory entry; otherwise, the job name field is filled in from the calling parameters. If the size is 0 indicating there is no date to be written, the date record pointer field is set to 0 and control is passed to add the directory entry to the library. If the size is not 0, data is to be written and the record number of the first record on the free record chain is placed in the data record pointer of the directory entry. This pointer is placed in RECNO for subsequent reading of the free records and I is initialized to 0 to be used as the data segment counter. The free record is not read and formatted. The segment indicator is incremented to indicate the current 500-character segment being processed. REC-TYPE is set to "X" to indicate a data record. The record format character is placed in the data record. The member name and segment number are also placed in the data record header. The data segment is now moved to the data portion of the record. If there are less than 500 characters left to move, LBMOVEC is called to move the characters; otherwise the entire 500-character segment is moved from the input area to the data record. SIZEX is decremented by 500 to indicate 500 characters have been moved and the number of data blocks counter is decremented by 1 to indicate one more data block has been processed. If this is not the last data block to be processed, the next record pointer in the data record is saved in J, the data record is written to the library, the statistics record is updated, the record number of the next free record is moved to RECNO for subsequent reading and control passes again through LBW-READ-DATA-REC to format the next data record. If this is the last data record to be processed, the free record pointed to by this data record will be the first free record on the free record chain and its record number is placed in the statistics record. The next record pointer in this data record is set to 0

to indicate that this is the last data record for the member being added. The data record is written to the library and the statistics record is updated. At this point all the data records have been written to the library and to complete the processing, the directory entry for the new member must be placed in an appropriate directory record on the library. The parameters used to find a free directory entry slot are set up by moving the first character of the member name to DR-SEARCH-LETTER and all '9' to the DR-SEARCH-NAME field. FIND-DIREC-ENTRY is not performed which searches the directory records associated with the first letter of the member name for an empty entry slot indicated by an entry name of all '9.' If RECNO is 0 when FIND-DIREC-ENTRY returns, a free directory entry was not foound. It is neccessary then to perform ADD-DIREC-REC-TO-CHAIN to create a new directory record for that letter which contains empty entries and add it to the directory record chain. When this routine returns or if FIND-DIREC-ENTRY did find an empty entry slot, RECNO will contain the record member of the directory record, I will contain an index to the empty entry in the record and the directory record will be in DIREC-REC. The new directory entry is then placed in the empty slot, the updated directory record is written to the library, the statistics record is updated to show that one more directory entry exists and LB-WRITE exits.

(e) LBLB Subroutines. The following subroutines are performed by the previously described main processing portions, LB-DELETE, LB-READ and LB-WRITE.

1. ADD-DIREC-REC-TO-CHAIN is performed by LB-WRITE when no free directory entries were found for the letter specified. Its purpose is to create a new directory record for the letter, format it with 16 empty entries and place it as the first record of the directory record chain. When entered, it determines the number to be used as an index in the statistics record to point to the proper DIREC-PTR for the specified letter. For example, letter C corresponds to number 3 and the third DIREC-PTR is a pointer to the first directory record for the letter. If the DIREC-PTR is 0, no directory records exist for that letter. The proper index is placed in I. Next the record number of the first record on the free record chain is moved to RECNO to become the new empty directory record. This record is read and the record it points to becomes the first free record on the chain. The free record counter in the statistics record is decremented. Now the directory record is formatted by moving all '9' to it, placing RECNO back into the THIS-REC field, moving the DIREC-PTR to the NEXT-REC field so that this new record points to the first record of the existing directory chain, moving THIS-REC to the DIREC-PTR so this record becomes the first record in the chain, making the REC-TYPE equal 'D," placing the specified letter in the REC-FORMAT field. The number of directory records counter in the statistics record is incremented by 1 and 1 is placed in I to

indicate to LB-WRITE that entry number one is is an empty entry. The routine then exits back to LB-WRITE.

2. COMPRESS is performed by LB-WRITE when it determines that the data to be written should be compressed. If the size is 500 or less, there will only be one data record and compressing is not necessary or desirable; COMPRESS immediately exits. If the size is more than 500, "C" is moved to REC-FORMAT-X to indicate the record is compressed and LBCMPRS is called to do the compression.

3. EXPAND is called by LB-READ when it finds the member read from the library to be in compressed format. It places the compressed size in SIZEX, calls "EXPAND" to expand the data and places the expanded size into the calling parameter field, LKSIZE.

4. FIND-DIREC-ENTRY is performed by LB-DELETE and LB-READ when this wish to find the directory entry of a particular member and by LB-WRITE when it wishes to find an empty entry in the directory records for a particular letter. When entered, DR-SEARCH-NAME contains the member name and DR-SEARCH-LETTER contains the letter of the directory record chain to be scanned. The routine first determines the index number corresponding to DR-SEARCH-LETTER. It then moved the appropriate DIREC-PTR pointing to the first directory record on the chain to RECNO. If RECNO equal 0, no directory records exist and control is returned to the calling routine. RECNO equal 0 indicates the directory entry was not found. If RECNO is not 0, the directory record is read and the entries are scanned. If a matching entry is found, I contains the index of the entry and the routine. If a matching entry is not found, the pointer to the next record is put in RECNO. If RECNO now equals 0, there are no more directory records and the routine exits; otherwise the next directory record is read and scanned.

5. NOOP is a dummy paragraph used by various routines for scanning purposes.

6. LB-EXIT is the common exit point when the main routines have finished processing. LKOP is checked to see if the statistics record has been altered. If it has, the statistics record is rewritten. If the member was deleted by LB-DELETE, "D" is placed in LKOP to so indicate. This could occur when the calling program passed "W" as LKOP and LBLB had to delete a member of the same name before writing the new version. The LIB-FILE is now closed. LBLEQ is called to relinquish control of the library and control is returned to the calling program.

(3) Limitations. The maximum size member which may be written on the library is 10,000 characters.

(4) Input and Output Data Set Description. A MIDMS library is a random access data set containing fixed length 532-byte record (Honeywell--540 character records). The first 12 characters of each library record contain four fields as follows:

1- 5 The record number of this record (the first is 00000).

6-10 The record number of the next record in a chain of records. This field contains 00000 if this is the last record in the chain.

11-11 A one-character code indicating the function of the record, F=free, S=statistics, D=directory, X=data.

12-12 A one-character code with different meanings for each record type:

Record Type	12-12 Meaning
F (free)	Blank
S (statistics)	1=first statistics record
D (directory)	A=this is a directory record for the letter A
X (data)	E=expanded data C=compressed data

(a) Free Record Format.

1-12 Record Header
- 1- 5 Record number of this record
- 6-10 Record number of the next record in the free record chain or 00000.
- 11-11 "F"
- 12-12 Blank

13-532 Blank (Honeywell:13-540)

(b) Statistics Record Format. The format of the STATISTICS record (record 00000) is:

```
 1-12 Record Header (0000000000S1)
13-17 Five-byte number of records in the library
18-22 Five-byte number of directory records in the library
23-27 Five-byte number of data records in the library
28-32 Five-byte number of free records in the library
33-37 Five-byte number of directory entries in the library
38-42 Five-byte record number of the first free record on the
      free record chain
43-172 Twenty-six five-byte record numbers of the first record
       number of each directory record chain
173-532 Not used (Honeywell:173-540)
```

(c) Directory Record Format. Each directory record contains 16 directory record entries of 32 bytes each.

```
 1-12 Record Header   1- 5 Record number of this record
                      6-10 Record number of the next record in
                           the directory record chain or 00000
                     11-11 "D"
                     12-12 An alphabetic letter A-Z

13-524 Sixteen directory entries of 32 bytes each
```

Directory Entry Format

```
 1- 8 Entry Name
 9-13 Entry Size
14-19 Entry Date
20-24 Record number of the first data record of
      this entry
25-32 Job name of the job that created this entry
      (Honeywell:not used, blank)

525-532 Not used (Honeywell:525-540)
```

(d) Data Record Format. Each data record contains the entry name, segment number and up to 500 bytes of data.

```
 1-12 Record Header   1- 5 Record number of this record
                      6-10 Record number of the next record in
                           the data record chain or 00000
                     11-11 "X"
                     12-12 "E" if the data is expanded (contains
                           spaces)
                           "C" if the data is compressed (four or
                           more spaces eliminated)
```

13-20 Entry name; matches an entry name in a directory record
21-22 Segment number; 1-20
23-32 Reserved; not used
33-532 500-byte data area

(Honeywell:533-540 not used)

(e) Initialization.

1. When created, the library is formatted as follows:

a. Record 00000 is made the STATISTICS record and its header is 000000000S1.

b. Records 0001 thru the last record of the library are formattted as free records and are chained together.

2. When created, the record headers would appear as follows for a 2000-record library:

```
000000000S1
0000100002F
0000200003F
   thru
0199801999F
0199900000F
```

3. The STATISTICS record contains the following information after the library is created (assume a 2000-record library).

1-12	"000000000S1"
13-17	"02000" number of library records
18-22	"00000" number of directory records
23-27	"00000" number of data records
28-32	"01999" number of free records
33-37	"00000" number of directory entries
38-42	"00001" record number of first free record in free record chain
43-172	"00000.....00000" there are no directory records.

(5) Tables, Switches and Major Work Areas.

(a) LIB-FILE.

1. LIB-REC describes the common 12-byte header of each library record.

2. DIREC-REC is the format of a directory record and each directory entry.

3. DATA-REC is the format of each data record.

(b) DIREC-ENTRY-WORK is the work area for constructing or using a directory entry.

(c) DR-ENTRIES-PER-DIREC defines the number of directory entries in a directory record.

(d) DR-SEARCH-LETTER and DR-SEARCH-NAME are parameters filled in prior to searching the library for a particular directory entry.

(e) LETTERS and LETTERS-A is used to convert an alphabetic letter into the corresponding numeric value (i.e., C=3).

(f) REC-FORMAT-X is used when writing a data record to the library to contain either "E" or "C" depending on whether the data is expanded or compressed.

(g) RECNO is used as the nominal key for the relative file LIB-FILE and contains the relative block number for all read and write operations.

(h) SIZEX is a work area used during moving of data and contains the number of remaining characters to be moved.

(i) STAT-REC is the definition of the statistics record in the library. The statistics record is read into this area each time LBLB is entered and it is updated and written back to the library during WRITE and DELETE operations.

(j) LKINFO and LKDATA are the definitions of the parameters passed to LBLB by the calling program.

b. LBCMPRS.

(1) Abstract.

(a) Function. This program compresses a string of data characters by replacing four or more consecutive blanks with a three character blank compression code.

(b) Calling Sequence. This program is entered at "LBCMPRS" or "COMPRESS" and is passed to two parameters: 1. the address of the beginning character of the string to be compressed and 2. the number of characters in the character string. When it returns to the calling program, the character string has been compressed in its original area and the compressed size is passed back in the original size field.

(2) Description.

(a) When entered, the program obtains the size of the data to be compressed, the address of the data and computes the address of the character past the end of the original data. It then begins scanning the data looking for a blank. If a blank is found, a counter, N spaces, is incremented. If the counter= 99 (99 consecutive spaces have occurred) the subroutine, SPACERTN, is executed to indicate a compression of 99 blanks up to this point. If NSPACES does not equal 99, the scan continues. If the data character is not a blank, the data character is checked for the code indicating compressed blanks, X'FF.' If the data character is X'FF,' the data is already compressed and the program returns to the calling program. If the data character is not X'FF,' NSPACES is checked to see if any blanks have occurred prior to this non-blank character. If there have been blanks, SPACERTN is executed to process them. In either case, the non-blank character is moved to its output position, the input and output character pointers are incremented and the next character is checked.

(b) When the character to be checked is past the end of the original data, the program branches to "FINISH." At this point any blanks remaining are processed, the compressed size is computed and returned to the calling program and control is returned to the calling program.

(c) When consecutive spaces have been found and the compression code is to be placed in its proper output location, subroutine SPACERTN is executed. If the number of consecutive blanks is 3 or less, actual blanks are placed into the output location since no compressional may occur for three or less blanks. If there are four or more blanks, the number in two-byte EBCDIC is placed in the second and third positions of "CODE" creating a

compression code (e.g., X'FF'73). This three-byte code is then placed in the proper output location and the blank counter, NSPACES, is zeroed.

(3) Limitations. None.

(4) Input and Output Data Set Descriptions. None.

(5) Tables, Switches and Major Work Areas. None.

c. LBEXPAND.

(1) Abstract.

(a) Function. LBEXPAND expands a compressed data string produced by program LBCMPRS into its original non-compressed format. It does this by scanning the string for the three-character blank suppression code and moving into the output location the appropriate number of blanks.

(b) Calling Sequence. This program is entered at "LBEXPAND" or "EXPAND" and is passed to two parameters: 1. the address of the compressed data string and 2. the address of the number of characters in the data string.

(2) Description. When entered, LBEXPAND moves the address of the compressed string and its length into storage areas. Then the data string is right-justified in the 10,000-byte area in which it is initially left-justified. This allows the data string to be expanded back into the left portion of the area. If the compressed data string is less then 5001 characters, LBMOVEC is called to move it to the right. If more than 5000, the data is moved byte-by-byte from the right end of the data to the right side of the area. When the right-justification is completed, R2 is initialized with the address of the first data string character, R3 with the address of the first character of the data area and R4 with the length of the compressed data. The program then begins scanning the compressed data string. If a character is not a blank code (i.e., X'FF'), the character is moved to the output location, the pointers are incremented by one, the length is decremented by one and the next character is checked. If the character is X'FF', the next two characters indicate the number of blanks that have been eliminated. The blanks are moved to the output area, the pointers adjusted, the length count reduced by three and the next character is checked. When the entire compressed data string has been scanned and the original data string reconstructed, the expanded length is computted and passed back to the calling program. Control is then returned to the calling program.

(3) Limitations. When called, the compressed data string must be left-justified in a 10,000-byte area because LBEXPAND will right-justify the compressed data and expand it into the left portion of this area overlaying the compressed data.

(4) Input and Output Data Set Descriptions. None.

(5) Tables, Switches and Major Work Areas. None.

d. LBMOVEC.

(1) Abstract.

(a) Function. LBMOVEC is an assembly language subroutine called by LBLB to speed up character moves.

(b) Calling Sequence. This routine may be called at either of two entry points, LBMOVEC or MOVEC, and is passed three parameters:

PARAMETER	
1	The address of the first byte of the sending field
2	The address of the first byte of the receiving field
3	The number of characters to be moved from the sending field to the receiving field.

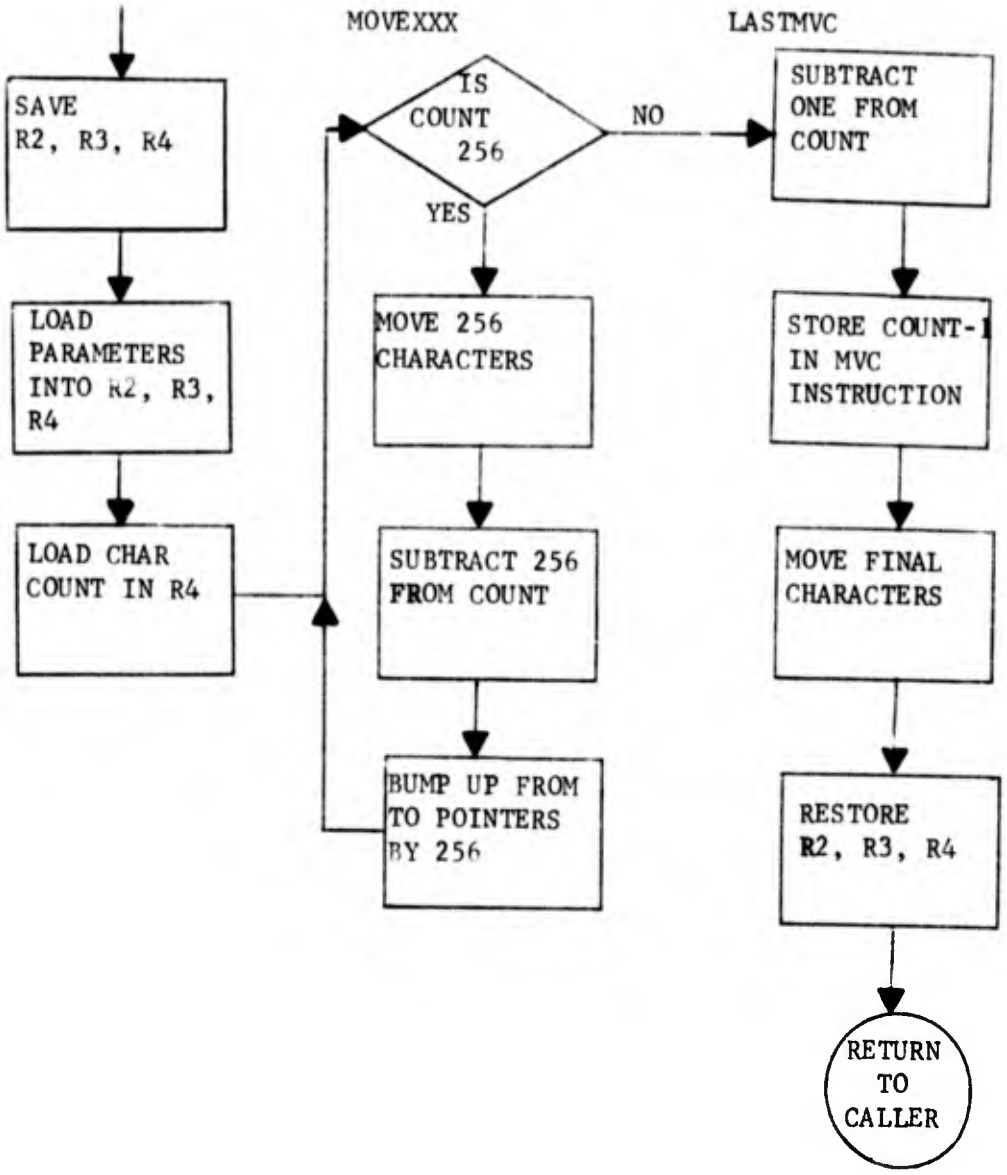
(2) Description. LBMOVEC is entered at either LBMOVEC or MOVEC and uses R15 as the base register throughout the program. R2, R3 and R4 are saved since the three parameters passed will be placed in them. The addresses of the sending and receiving areas and of the number of characters to be moved are placed in R2, R3, and R4, respectively. The actual length is then placed in R4. At "MOVEXXXX" the number of characters to be moved is checked. If greater than 256, 256 characters are moved, the count is reduced by 256, the sending and receiving field addresses are increased by 256 and control passes to "MOVEXXXX" to check the count. If less than 256 are to be moved, the count less one is stored in the move instruction (MVC), the last characters are moved, R2, R3, R4 are restored and control is returned to the calling program.

(3) Limitations. The numbers of characters to be moved may not be zero.

(4) Input and Output Data Set Descriptions. None.

(5) Tables, Switches and Major Work Areas. None.

(6) Subprogram Flowchart.



e. LBJBNAME.

(1) Abstract.

(a) Function. This program obtains the job name specified in the // JOB card from the Task Input Output Table and returns it to the calling program.

(b) Calling Sequence. This routine may be called at either entry points "LBJBNAME" or "JOBNAME" and is passed one parameter, the address of an eight-byte area in which LBJBNAME will place the job name.

(2) Description. When entered, LBJBNAME saves registers, obtains the address of the TIOT via the CVT and TCB, and moves the job name (the first eight bytes of the TIOT) to the receiving area provided by the calling program.

(3) Limitations. None.

(4) Input and Output Data Set Descriptions. None.

(5) Tables, Switches, and Major Work Areas. None.

f. LBENQ.

(1) Abstract.

(a) Function. This program allows the librarian, LBLB, to gain exclusive or shared control of a library preventing the librarian in another job from altering the library before LBLB has finished processing. This control is gained by using the ENQ macro with a QNAME of "MIDMS" and an RNAME of the library's data set name. See IBM System/360 Operating System Supervisor Services GC28-6646 for an explanation of the ENQ macro.

(b) Calling Sequence. This program is called at entry point "LBENQ" and is passed to two parameters: 1. the address of the library file's DECB and 2. the address of the LKOP field denoting the particular operation being performed.

(2) Description. The program moves the LKOP character into a save area. It then determines of the library's data set name has already been obtained and stored in the program. If location DSN does not contain X'58', then the 44-byte data set name has been obtained and resides at DSN. The program then branches to WHICHENQ to continue processing. If the data set name has not been obtained, the Job File Control Block is read into a 176-byte area at

JFCB. The first 44-byte of the JFCB is the data set name. The type of ENQ is now determined. If LKOP is "S" or "R," only shared control is required sincy the library will not be altered. However, if LKOP is "D" or "W," exclusive control is required. The appropriate ENQ macro is issued and the program returns to the caller.

(3) Limitations. None.

(4) Input and Output Data Set Descriptions. None.

(5) Tables, Switches and Major Work Areas. None.

g. LBDEQ.

(1) Abstract.

(a) Function. This program releases control of a library by issuing a DEQ macro to cancel a preceding ENQ issued by LBENQ.

(b) Calling Sequence. This program is entered at "LBDEQ" and is passed no parameters.

(2) Descriptions. When entered, the address of the 18-word save area (ENQSAVE) in the LBENQ module is obtained and used as the save area for this program. The addresses of the QNAME and RNAME are obtained and placed in registers 2 and 3 and the DEQ macro is issued to release control of the library. The program then returns to the calling program (LBLB).

(3) Limitations. LBENQ must have been called previously to gain control of the library being released.

(4) Input and Output Data Set Descriptions. None.

(5) Tables, Switches and Major Work Areas. None.

h. LBRNAME.

(1) Abstract.

(a) Function. This program provides the data set name of the library to the calling program.

(b) Calling Sequence. This program is entered at "LBRNAME" and is passed one parameter, the address of a 44-byte area for the library data set name.

(2) Description. When entered, the address of the 18-word save area (ENQSAVE) in the LBENQ module is obtained and used as the save area for this program. The address of the RNAME field in LBENQ which contains the library data set name is obtained and the name is moved into the calling programs receiving area.

(3) Limitations. LBENQ must have been called before this program is entered to insure that RNAME contains the data set name.

(4) Input and Output Data Set Descriptions. None.

(5) Tables, Switches and Major Work Areas. None.

3. ERROR MESSAGES. There are no error messages associated with the Librarian. Any abnormal conditions found during its execution are indicated to the calling program by placing a "N" in the operation field of the calling parameters before returning control to it. The meaning of the "N" depends on the request operation:

OPERATION	"N" MEANING
R (READ)	The library member was not found on the library.
W (WRITE)	Insufficient room is left in the library to write the requested member.
D (DELETE)	The member was not found on the library.
S (SIZE)	The member was not found on the library.