AD-783 408

# MACHINE INDEPENDENT DATA MANAGEMENT SYSTEM (MIDMS) SYSTEM SPECIFICATIONS. CHAPTER 4 - RETRIEVAL AND OUTPUT DOCUMENTATION - PART I. CHAPTER 5 - RETRIEVAL AND OUTPUT DOCUMENTATION - PART II

Defense Intelligence Agency
Washington, D. C.

1 July 1974

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

| 4. TITLE (and Subtitle) | 5. TYPE OF REPORT & PERIOD COVERED |
|---|---|
| Machine Independent Data Management System (MIDMS) System Specifications<br>CHAPTER 4-RETrieval and Output Documentation<br>CHAPTER 5-Retrieval and output Documentation-PART II | PART I<br>User Documentation<br>6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s)<br>Defense Intelligence Agency (DIA) | 8. CONTRACT OR GRANT NUMBER(s) |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS<br>Defense Intelligence Agency<br>ATTN: Information Processing Division (DS-5)<br>Washington, D.C. 20301 | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS<br>Same as 9 | 12. REPORT DATE<br>1 July 1974<br>13. NUMBER OF PAGES<br>496 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) | 15. SECURITY CLASS. (of this report)<br>UNCLASSIFIED<br>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DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE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

# 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

Enclosures A, B, C + D - HD-783409

# CHAPTER 4

## RETRIEVAL AND OUTPUT DOCUMENTATION

## PART I

Subsystem Structure and Program Identification:

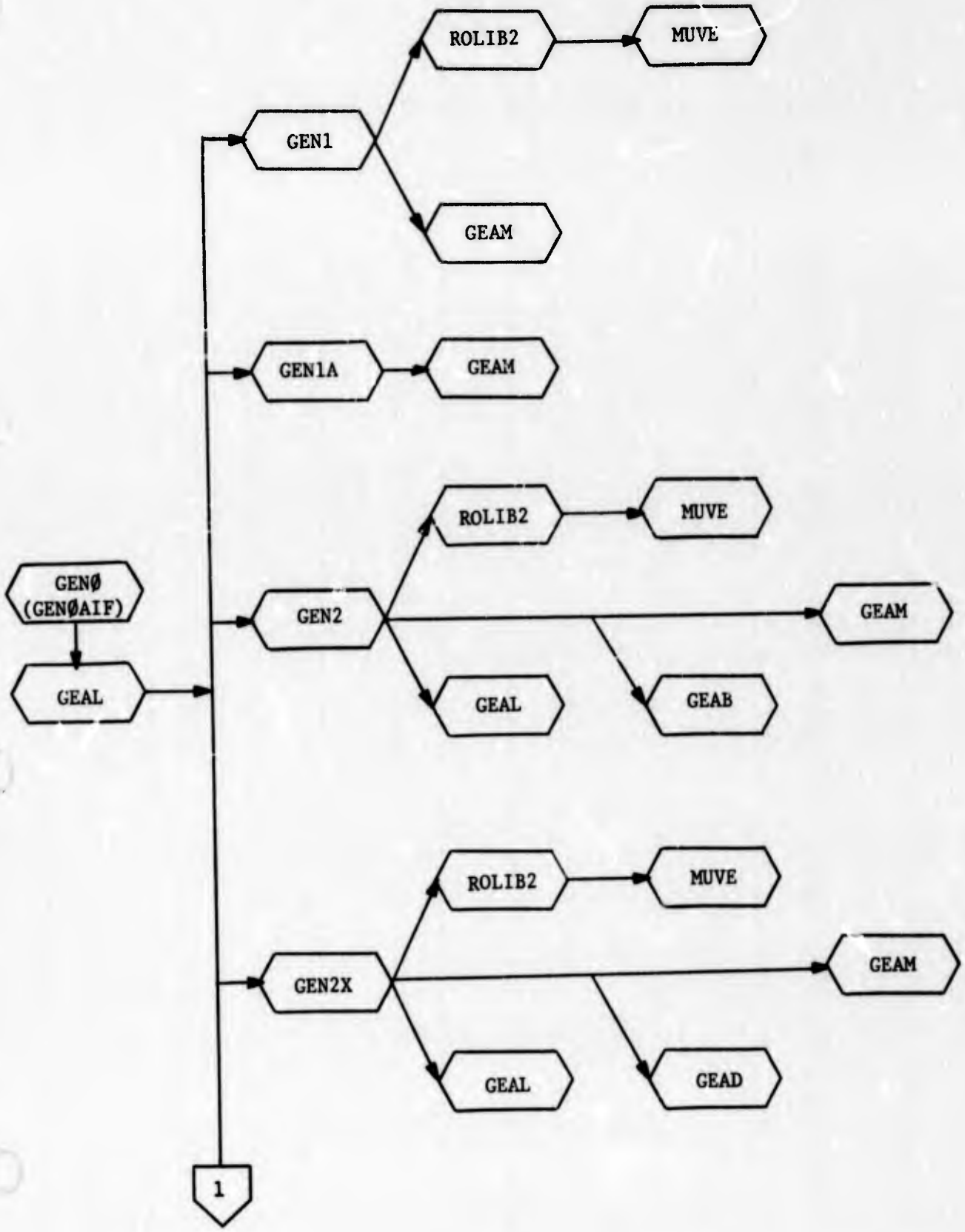
a. Subsystem Structure.

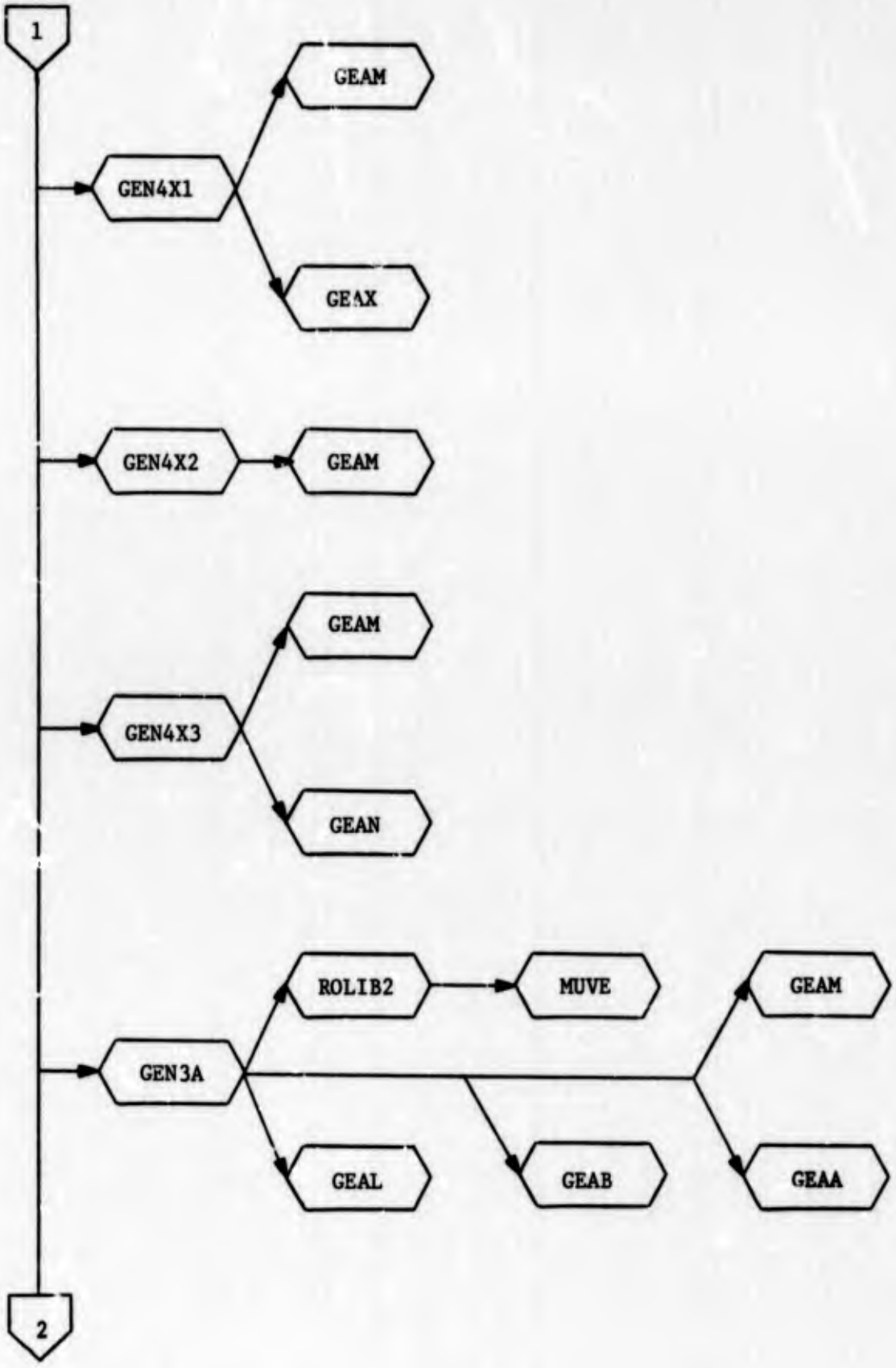
(1) The Retrieval and Output portion of the Machine Independent Data Management System (MIDMS) is made up of seventeen COBOL modules and fourteen ALC subroutines.

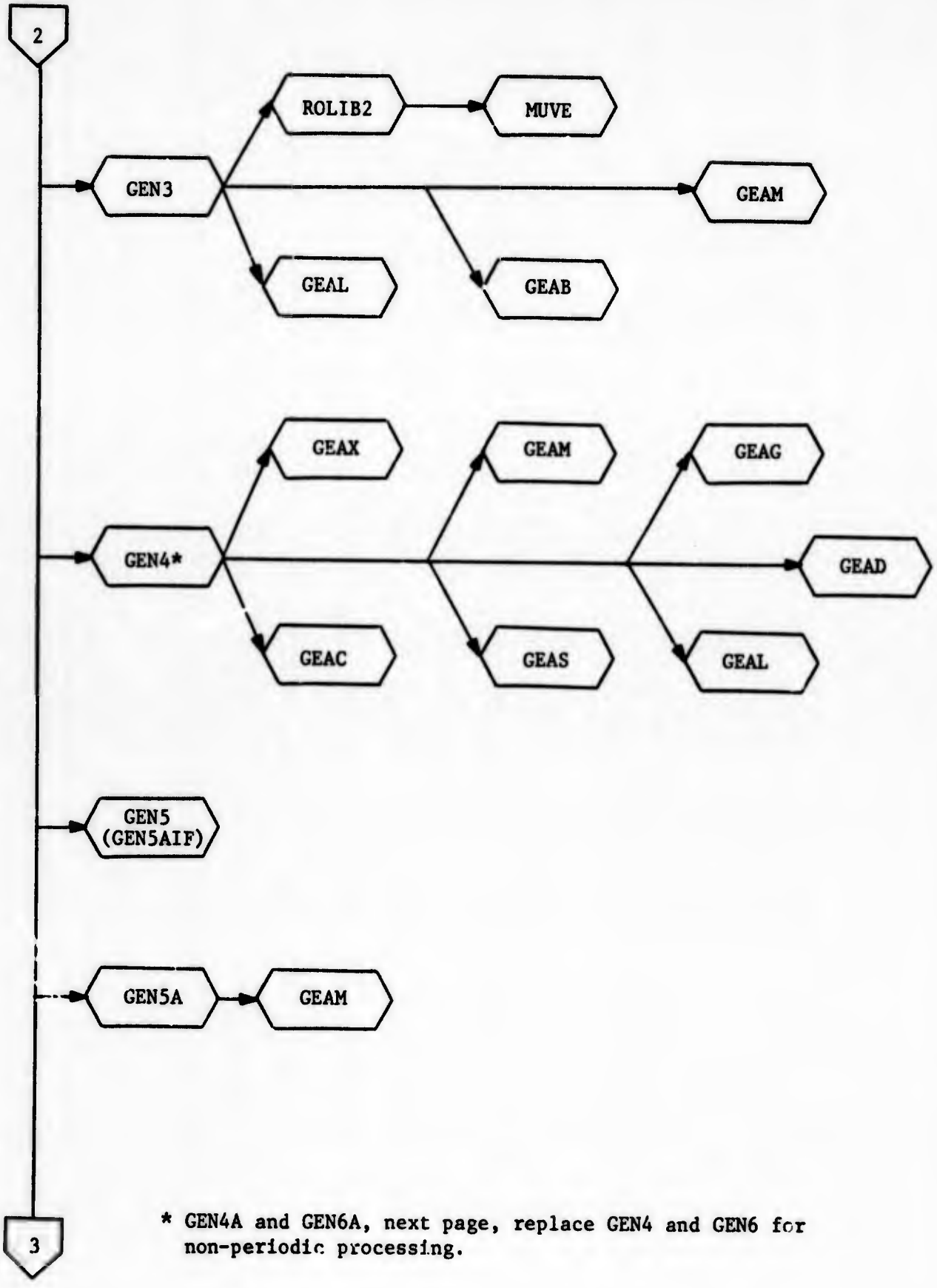
(a) COBOL Programs.

| | | |
|---|---|---|
| 1. | GEN0 | Supervisor |
| 2. | GEN1 | Batch Build |
| 3. | GEN1A | Source Statement Sort |
| 4. | GEN2 | Retrieval Source Statement Compiler |
| 5. | GEN2X | Non-Sequential Source Statement Compiler |
| 6. | GEN4X1 | Non-Sequential Logic Processor, analyzes lists according to users' source statements. |
| 7. | GEN4X2 | Non-Sequential Work File Sort |
| 8. | GEN4X3 | Non-Sequential Logic Processor, determines records to be selected and creates a summary file when directed. |
| 9. | GEN3A | SHL Output Source Statement Compiler |
| 10. | GEN3 | Output Source Statement Compiler |
| 11. | GEN3B | OP Library Maintenance Program |
| 12. | GEN4 | Retrieval Logic Processor (Periodic) |
| 13. | GEN4A | Retrieval Logic Processor (Non-Periodic) |
| 14. | GEN5 | Output Data Sort |
| 15. | GEN5A | Work File Sort |
| 16. | GEN6 | Output Logic Processor (Periodic) |
| 17. | GEN6A | Output Logic Processor (Non-Periodic) |

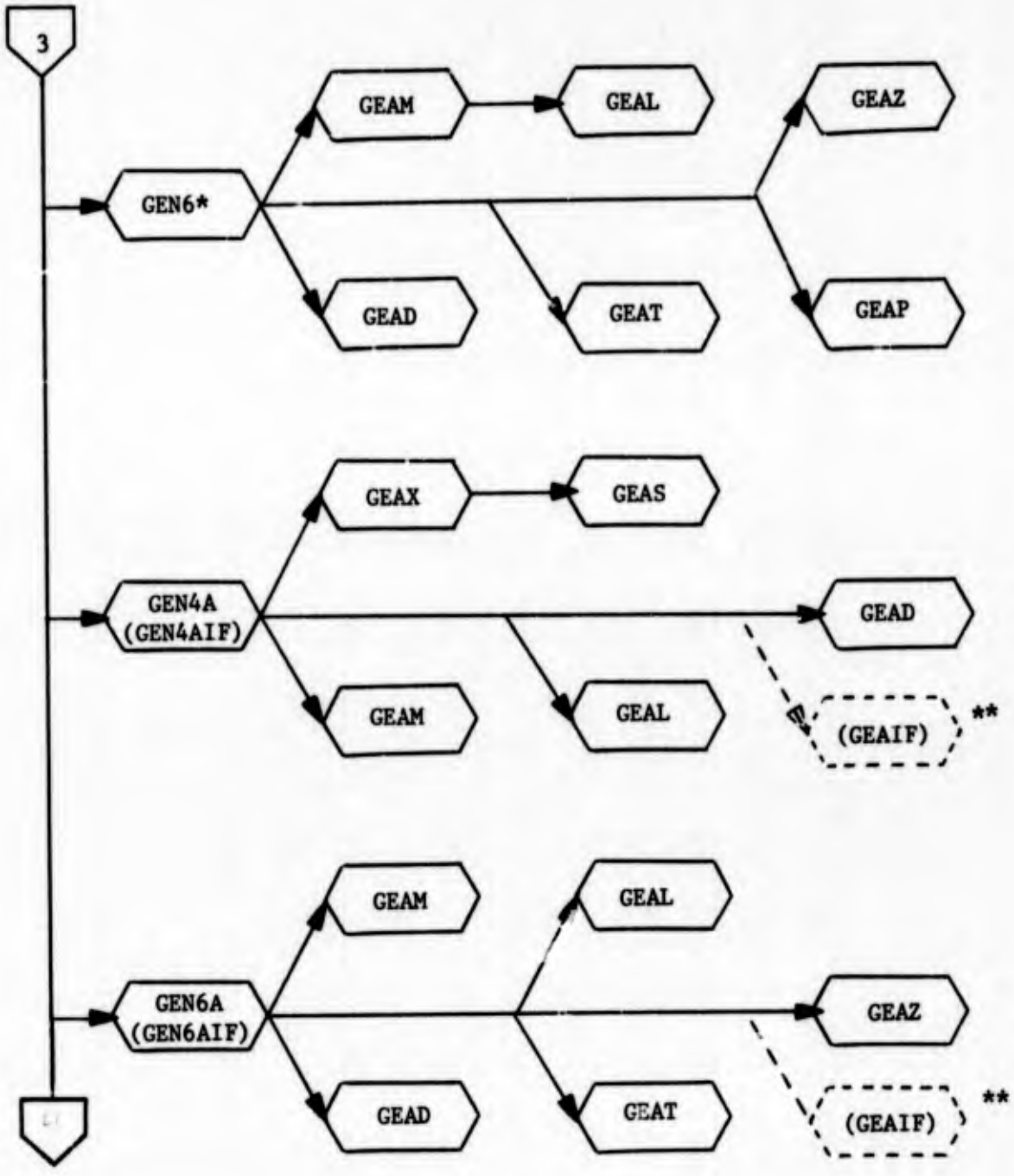
## RETRIEVAL AND OUTPUT SYSTEM FLOW




1
GEAM
GEN4X1
GEAX
GEN4X2
GEAM
GEAM
GEN4X3
GEAN
ROLIB2
MUVE
GEAM
GEN3A
GEAL
GEAB
GEAA
2

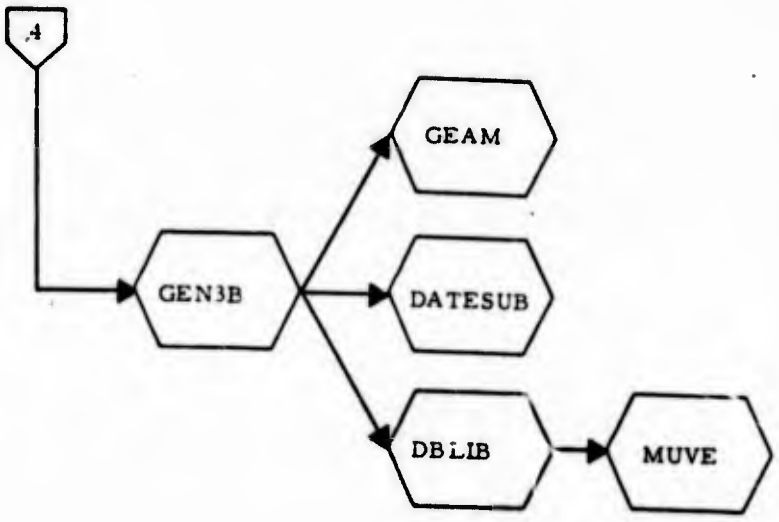



* GEN4A and GEN6A, next page, replace GEN4 and GEN6 for non-periodic processing.




* GEN4A and GEN6A replace GEN4 and GEN6 for non-periodic processing.

**AIF processing calls the routines GENØAIF, GEN4AIF, GEN5AIF, and GEN6AIF instead of GENØ, GEN4A, GEN5A, and GEN6A. The corresponding programs are virtually identical except for data record description. GEN4AIF and GEN6AIF include the same subroutines as do GEN4A and GEN6A, and in addition, include the AIF record expansion subroutine GEAIF. No further mention is made of AIF processing.


A
GEAM
GEN3B
DATESUB
DBLIB
MUVE

(b) ALC Subroutines.

| | | |
|---|---|---|
| 1. | GEAA | Provides System Date |
| 2. | GEAB | Locates Called Load Modules |
| 3. | GEAC | Decodes Periodic Set Control |
| 4. | GEAD | Loads Reusable Subroutines |
| 5. | GEAE | Set Exit List |
| 6. | GEAG | Flags Periodic Data Sets |
| 7. | GEAL | Link Routine |
| 8. | GEAM | Moves Logical String |
| 9. | GEAN | Moves a Name Parameter to DDname location of a random file DCB. |
| 10. | GEAP | Performs Periodic Subset Sort |
| 11. | GEAS | Builds Sort Key |
| 12. | GEAT | Locates Excess Core Storage for Tables |
| 13. | GEAX | Compares Two Fields |
| 14. | GEAZ | Executes Output Object Statements |

(c) The ALC routines are small and each performs a special function. They would be relatively easy to recode in another machine's assembly language.

(2) The currently operational Retrieval and Output modules include the following general capabilities:

(a) Machine transferability - Retrieval and Output has been implemented on both the IBM 360 and the HIS-635.

(b) Retrieval and Output support variable length and fixed length records, either blocked or unblocked, periodic or non-periodic.

(c) They will process sequential files on drum, disc, data cell, magnetic tape, card, paper tape, or any other device supported by that operating system.

(d) They include a multi-file retrieval and output capability that will process up to 60 files in a single query.

(e) The maximum record size is 10,000 characters.

(f) Retrieval and Output will automatically batch up to 60 independent queries and search each file only once.

(g) They will accept explicit and implicit convert routines. A convert routine is a user-written subprogram or subroutine that contains an algorithm to convert data from one format to another format - an explicit convert routine is specified by name by the user at execution time. An implicit convert routine on the other hand, is identified within the File Format Table so when that defined field is compiled, this convert routine will automatically be invoked.

(h) The user's language is simple to use and easy to learn. It is an explicit language that simplifies learning and use by a non-programmer.

(i) There are only a few reserved words in Retrieval and Output: SUBQUERY, SUBQUERYX, SORTKEY, ZERO, ZEROS, ZEROES, SPACE, SPACES, TABLE, and MATRIX. These reserved words allow a user to gain access to the subquery number within a query and to the contents of the sort key, to express the value zero or space, and to define an internal table or matrix. All other words may be used freely without restriction.

(j) Queries and reports are easy to modify to suit the user's changing requirements. There is a skeleton query capability and a skeleton report that provides the capability to change a stored query or report.

(k) Retrieval will accept user-written special operators to satisfy unique requirements that cannot be met by the standard language operators. These special operators may be written in any language supported by that machine.

(l) The system is fast and efficient, particularly in a batching environment.

(m) Up to nine levels of parentheses may be used to accommodate virtually any conceivable query logic.

(n) There are three stages of conditional statements available. The primary conditional statements operate against the data file itself to select certain records. The secondary conditional statements are applied against the record already selected by the primary conditional statements. Additionally, there is a capability for conditional statements within Output to further condition those records that have gone through primary and/or secondary conditional logic.

(o) The design of Retrieval and Output is modular to simplify conversion, system maintenance, and expansion.

(p) The system runs on a 128K 360/40.

(q) Blanks in a numeric field are edited and processed as zeros.

(r) The user is able to perform groups of statements from different paragraphs of a report to eliminate redundant coding. This perform capability is particularly useful in conjunction with the IF COMPLETE/IF CHANGE and MATRIX operators.

(s) The user has a comprehensive subscripting capability.

(t) The system has a non-sequential capability. A user utilizes this capability by writing standard MIDMS retrieval source statements under the non-sequential designator, SUBQUERYX. The statements are used to query the appropriate reference lists to determine the records satisfying the retrieval request. The selected records are directly retrieved.

(3) The design concept in Retrieval and Output is functionally interpretive. There are three basic elements in this process: the compiler, the linkage editor, and the logic processor. The compiler and linkage editor are not to be confused with the COBOL compiler and the OS linkage editor. The compilers in this system compile the user's language and generate an object code with the COBOL programs that make up the system. This object code is then processed by the linkage editor which essentially resolves the addressing within core. From there it goes into execution of the logic processor.

(4) Retrieval and Output exist as a single module. The execution of both is a one-step job. The following describes the sixteen COBOL programs that make up most of the system along with a description of the ALC routines that they call.

(a) The Main Control Program is a supervisor and it calls other COBOL programs in sequence by using the ALC subroutine GEAL. This subroutine provides the actual linkage between the COBOL programs. GEN0 enables the Retrieval and Output system to execute in a single step. This has enabled the system to interface with the Remote Inquiry Control System (RICS) and provide the on-line query and report capability. GEN1 through GEN6 are the actual working parts of the system.

(b) The Task Builder is the first processing program called. It reads the source program and attaches an 18-character sort key to the beginning and a 10-character trailer at the end of each record. It edits and special processes certain control statements (LIBRARY, QUERY, SUBQUERY, SUBQUERYX, REPORT, MODIFY, DEFINE).

(c) GEN1A sorts all the queries and subqueries together by file name so that all the statements for a particular file are grouped together.

(d) GEN2 is the Sequential Retrieval Source Statement Compiler. It edits and compiles the query and subqueries, and outputs object records containing: a-field, b-field, operation code, retrieval mode, and other information It also outputs all the constants found in the retrieval source language, and error vectors for invalid source statements. This program also executes any retrieval convert routines applicable, either explicit or implicit.

(e) GEN2X is the Non-Sequential Source Compiler. It edits and compiles the non-sequential queries (SUBQUERYX). The result of the compilation is a series of object vectors, constant pool strings, and an error vector. The object vectors contain the a-field, b-field, and operation code, and are sorted so that all vectors for any list appear together (a list is only read once). The constant strings represent the values used in the retrieval expression. The compiler provides any necessary conversion of constants. The error vector indicates any errors if they are found.

(f) GEN4X1 is the first non-sequential logic processor. GEN4X1 analyzes the appropriate lists for the user's retrieval criteria. If the query that contains the non-sequential statements contains an error, the remaining portion of the non-sequential process is bypassed.

(g) GEN4X2 is the Non-Sequential Work File List. GEN4X2 sorts the keys which were retrieved using the list processor GEN4X1.

(h) GEN4X3 is the second non-sequential logic processor. It accepts the sorted output from GEN4X2, and analyzes the sorted output to determine the acceptable records (according to the user's retrieval request) and when directed, it creates a summary file by directly retrieving the acceptable records.

(i) GEN3A is the Shorthand Language source statement generator. It performs error detection functions on source shorthand statements and generates equivalent standard output statements. The shorthand compiler also creates internal save areas for counts, totals, etc., and extracts the current date from the operating system.

(j) GEN3 is the Output Source Statement Compiler. It edits and compiles the reports and outputs the same three types of records as the Retrieval Source Statement Compiler.

(j.1) GEN3B is the OP Library Maintenance Program. It stores OP reports in source and object form on the MIDMS library.

(k) GEN4, the Retrieval Logic Processor, uses the output from the Retrieval Source Statement Compiler. Queries that contained any errors are bypassed. GEN4 performs two functions; it is a retrieval linkage editor and is also the retrieval logic processor. The linkage editor builds a set of tables, called vector arrays, which the logic processor uses to retrieve against the data files. The vector arrays are built only once for each job and they are structured to execute drop out logic against the records in the input buffer. This is also the subprogram in which special operators are executed. All qualifying records from all queries in the job are written on an answer file. The only exception to this is the summary file. In case the summary file is being generated, the records are written directly onto the new file and do not go into an answer file. GEN4 calls seven ALC subroutines to help in its retrieval function.

1. GEAM - moves a string of characters from one area of core storage to another.

2. GEAX - compares two fields and returns a code showing if the a-field is less than, equal to, or greater than the b-field.

3. GEAG - examines periodic data sets and flags the ones that meet the specified search mode and user requirements.

4. GEAC - decodes the periodic set control words which contain the number of subsets that exist for that set in a particular record and the starting address of the set.

5. GEAS - builds the 85-character user-specified sort key and returns it to the calling program.

6. GEAL - loads and links user-specified special operators.

7. GEAD - causes reusable special operators to remain in core for the duration of the job when adequate core is available.

GEN4A is similar to GEN4 except that it only processes non-periodic files. GEAC and GEAG are not used by GEN4A. Either GEN4 or GEN4A will be called, but not both in the same job.

(l) GEN5 is the Output Data Sort. It uses the 15-character system-built key and the 85-character user-built key to sort the answer file and restore the records to the matching report.

(m) GEN5A sorts work file records to accommodate flysheets.

(n) The Output Logic Processor is the last program called. It uses the output from the Output Source Statement Compiler and the sorted data from GEN5 to generate the required output reports. Any queries that contained errors are skipped. The program is the output linkage editor and logic processor. In addition, it executes output convert routines and table lookups. Up to 60 batched reports can be printed, punched, or written as a single job. GEN6 calls six ALC subroutines to assist in the output function:

1. GEAM - moves a string of characters from one area of core storage to another.

2. GEAZ - processes virtually all logical statements (arithmetic statements, TMOVE, EMOVE, etc.).

3. GEAT - searches areas of core storage such as the statement area, special operators area, and constant pool area for unused portions to fill up with load module tables.

4. GEAL - loads and links user-specified convert routines.

5. GEAD - causes reusable convert routines to remain in core for the duration of the job when sufficient core is available.

6. GEAP - executes a sort of periodic subsets within a record.

GEN6A is similar to GEN6 except that it only processes non-periodic fields. GEAP is not used by GEN6A. Either GEN6 or GEN6A will be called, but not both in the same job.

b. Subprogram Identification and Description.

(1) GEN0.

(a) Abstract of Driver.

1. Function. GEN0's function is to call the other COBOL routines in the proper order.

2. Calling Sequence. GEN0 calls the ALC subroutine GEAL using three paremeters:

| | Length | Type | |
|---|---|---|---|
| Module name | 8 | A | Name of Called Routine |
| SAVEREG | 20 FULLWORDS | B | Save Area for Registers |
| QUERY-COMMUNICATIONS | | | |
| AREA-ZERO | 120 | N | Error Vector |
| AREA-DD | 24 | A | Word File Names |
| AREA-COUNT | HALFWORD | B | Occurs 120 times<br>OP Statement Count |
| NSAX-AREA | 136 | MIXED | Non-Sequential Parameters |

(b) Description. GEN0 is the main COBOL program of the MIDMS System. Its purpose is to call the twelve MIDMS COBOL routines (GEN1, GEN1A, GEN2, GEN3, GEN3A, GEN3B, GEN4, GEN4A, GEN5, GEN5A, GEN6, or GEN6A) in the proper order through the ALC subroutine GEAL. If GEN2 references periodic fields, GEN4 is called, otherwise GEN4A. If GEN3 references periodic data, GEN6 is called, otherwise GEN6A.

(c) Limitations. The PNAME or name of the called program must be eight characters or less.

(d) I/O Data Sets. None.

(2) GEN1.

(a) Abstract of Task Builder.

1. Function. The Retrieval and Output Task Builder reads from the card reader the MIDMS source statements. These statements can refer to stored queries on the library and they can temporarily modify the stored queries according to user specifications. An 18-character sort key is built and placed at the beginning of every retrieval and output source statement. A 10-character trailer is also added to each record giving it a new length of 108 characters. A syntax check is performed on special control statements: COMP-OP or COMP OP, LIB or LIBRARY, QUERY, SUBQUERY, REPORT, MODIFY, and DEFINE.

2. Calling Sequence. The module calls two subroutines:

a. GEAM.

| | Length | Type | |
|---|---|---|---|
| parm1 | HALFWORD | B | Base Address of Sending Area |
| parm2 | HALFWORD | B | Displacement of Sending Area |
| parm3 | HALFWORD | B | Base Address of Receiving Area |
| parm4 | HALFWORD | B | Displacement of Receiving Area |
| parm5 | HALFWORD | B | Length of String to be Moved |

b. ROLIB2.

| | Length | Type | |
|---|---|---|---|
| CALL-SEQ | | | |
| OPERATION | 1 | A | Value R for Read |
| ITEM-NAME | 5 | A | Library Member Name |
| SUFFIX-NAME | 2 | N | Member Number |
| FILLER | 1 | A | Not Used |
| ITEM-DATE | 5 | A | Not Used |
| ITEM-SIZE | 4 | N | Number of Characters Brought In |
| Receiving Area | 80 | A | Occurs 125 Times |

(b) Description. In G1 input cards are read into a working storage area called CARDIN. In G2 the first eight characters of the source input record are checked for one of the nine special control words which must be processed further. Control words with the exception of DEFINE must start in column 1 or they will be processed as an 'ALL OTHER' source statement.

If the control word is COMP-OP or COMP OP the program sets the COMP-OP switch ('CP'), moves the card image into the working storage area called COMP-JCL and scans the card to determine the COMP-OP name. If the name is not 5 characters in length and/or does not end in 'R', appropriate error messages are generated and displayed. If all is correct, the system will set the sort keys (COPOUT) so the COMP-OP card will be the first card in the input stream. Control is then returned to G1 and a new card is read. If one of the next cards is a LIB or LIBRARY card and the first card was a COMP-OP (CP=1), the card image is moved to the working storage area and TYP-ST (2) is set to 8 to indicate that both source and object reports are to be stored. If a subsequent LIB or LIBRARY card is used to obtain a complete source REPORT (or SHL-RPT) from the library, TYPE-ST (2) will be set to 7 to indicate that only an object report is to be stored.

If the control card is a LIB or LIBRARY and was not preceded by a COMP-OP card, the program checks the next word which should be 5 characters long and end in Q, R, F, or J. An error message is generated if the conditions are not met (G13). This is the name of the stored

member on the library. The subroutine LB is called to retrieve the first record from the named member (G15). If the named member is not on the library, an error message is generated and further library processing is bypassed for this card (G20). If the library name is located, the length of the member must be a multiple of 80 and there must be 125 or less card images or appropriate error messages are generated (G16), unless the program is processing on object report (name ends in 'J') in which case the length must be a multiple of 108 and there must be 92 or less images. The SWT1 switch is set to 1 to inform GEN1 to read the next input record from the library rather than the card input stream (G16). The SWT2 switch is set to the number of card images in the stored query (G17). The first card image from the library is moved to CARDIN and the process of checking for a control word is initiated (G19).

If the control word is QUERY, the TYP indicator is set to 1 and the REPORT-SWT indicator is set to spaces to indicate that a QUERY control card has been processed. The QUERY counter in the sort key is incremented by 1. The FILENAMEA and FILENAMEB are set to spaces and the SUBQUERY counter and card sequence counter CARD-NUM are set to zeros. This record is written to SIGEF1 which is the input file to the card procedure. When sorted, the key is now set so that this record will be the header card for the query.

If the control word is SUBQUERY, the TYPE indicator is checked to determine if a QUERY card was processed (G30). If TYPE is not equal to 1, a QUERY card is generated by the system, TYPE is set to 1, and an error message is written (G30). The constant COUNTER keeps track of the card position (CC 1 to CC 72) as the program scans across the card image. GEN1 looks for the next word, skipping commas, blanks and the word FILE if encountered (G32). The next word will be the file name and the first 8 characters are moved to FILENAMEA and FILENAMEB (G37). The counter SUBQUERY is incremented by 1 and the sequence counter CARD-NUM is initialized to zero. This record is written to zero. This record is written to SIGEF1, the card input file and CARD-NUM is set to 1000 (G34).

If the control word is REPORT, the REPORT-SWT indicator is checked to determine if a QUERY card was processed (G70). If REPORT-SWT is not equal to spaces, a QUERY card is generated by the system and an error message is written (G70). FILENAMEA and FILENAMEB are set to 99999999, the SUBQUERY and CARD-NUM counters are set to zero, and the TYPE indicator is set to 5 to indicate a report. The output record is written to SIGEF1 and CARD-NUM is set to 1000 (G71).

If the control word is SHL REPORT or SHL RPT, a switch is turned on and the TYP indicator is set to 4 to indicate a shorthand language (SHL) report.

If the control word is MODIFY, the program scans for the next word, skipping commas and blanks if encountered, looking for either SUBQUERY or REPORT. If SUBQUERY is found (G44), the program checks the file name (G50-G52A) and the subquery number entries to see if they are valid. FILENAMEA and FILENAMEB are set to the value on the card images (G53), the TYPE indicator and CARD-NUM counter are set to 1 (G54) and the SUBQUERY counter is set to the value found in the card image (G46). The record is then written to SIGEF1. If the word REPORT was found, FILENAMEA and FILENAMEB are set to 99999999, the TYPE indicator is set to 5, CARD-NUM is set to 1, the SUBQUERY counter is initialized to zero and the record is written to SIGEF1 (G43). If the words SUBQUERY or REPORT are not found or if the file name or subquery number entries are invalid, the indicators are reset and the record is written to SIGEF1 with an error message.

If the control word is DEFINE, CARD-NUM is incremented by 2 (G6) and the record is written to S1GEF1 (G3).

In all other source statements, CARD-NUM is incremented by 20 (G7) and the records are written to S1GEF1 (G3).

After each record is processed, the program checks to see if the next input record should come from a card or the library (G18). The next record is read and processing proceeds in G2.

(c) Limitations. None.

(d) I/O Data Sets.

1. CARD-INPUT. This is an 80 character fixed length input file used as the source statement input.

2. CARD-OUT. This is an output work file used for sequential processing.

3. CARD-OUTX. This is an output work file used for non-sequential processing.

4. FILE-OUT. If no sort is required after GEN1, the CARD-OUT file is copied to the FILE-OUT work file to maintain synchronization of work files. If a sort is needed, FILE-OUT is not used.

(3) GEN1A.

(a) Abstract of Driver.

1. Function. A COBOL sort is performed to group all subqueries together by file name.

2. Calling Sequence. The program calls one subroutine, GEAM.

| | Length | Type | |
|---|---|---|---|
| parm1 | HALFWORD | B | Base Address of Sending Area |
| parm2 | HALFWORD | B | Displacement of Sending Area |
| parm3 | HALFWORD | B | Base Address of Receiving Area |
| parm4 | HALFWORD | B | Displacement of Receiving Area |
| parm5 | HALFWORD | B | Length of String to be Moved |

(b) Description. GEN1A's only function is to sort user source statements and check for a sort error.

(c) Limitations. None.

(d) I/O Data Sets.

1. CARD-OUT. This is the work file created by GEN1. It is used as input to the sort and subsequently as output from the sort.

2. SORT-REC. This is the record being sorted. The sort work files are variable in number and as described in the JCL.

(4) GEN2.

(a) Abstract of Retrieval Compiler.

1. Function. The Retrieval Source Statement Compiler (GEN2) performs syntax editing of the formatted retrieval source statements and writes an error message for the first error encountered in each retrieval source statement. GEN2 builds vector arrays for each retrieval source statement, a constant pool string containing constants, literals, and defined values, and a query vector containing non-zero entries for those queries which contain an error in the source statements. The program passes all of its output along with the output from the Batch Build (GEN1) and the input source statements to the Output Compiler (GEN3) for further processing.

2. Calling Sequence. The program calls four subroutines:

a. ROLIB2.

| | Length | Type | |
|---|---|---|---|
| CALL-SEQ | | | |
| OPERATION | 1 | A | Value R for Read |
| ITEM-NAME | 5 | A | Library Member Name |
| SUFFIX-NAME | 2 | N | Overflow Counter |
| FILLER | 1 | A | Not Used |
| ITEM-DATE | 5 | A | Not Used |
| ITEM-SIZE | 4 | N | Number of Characters Brought In |
| Receiving Area | 80 | A | Occurs 125 Times |

b. GEAL.

| | Length | Type | |
|---|---|---|---|
| module name | 8 | A | Name of Called Routine |
| SAVEREG | 18 FULLWORDS | B | Save Area for Registers |
| parameter(s) | Variable | Variable | Parameter(s) being passed to called routine but not used by GEAL |

c. GEAB.

| | Length | Type | |
|---|---|---|---|
| SNAME | 8 | A | Name of Routine Being Checked |
| RET-CODE | HALFWORD | B | Return Code from Routine |

d. GEAM.

| | Length | Type | |
|---|---|---|---|
| Sending Area | | N | Address of Sending Area |
| Sending Area Offset | 2 | N | Character Subscript (Base 1) of Sending Area |
| Receiving Area | | N | Address of Receiving Area |
| Receiving Area Offset | 2 | N | Character Subscript (Base 1) of Receiving Area |
| Number of Chara ters | 2 | N | Number of Characters to Move |

(b) Description. Records are initially read in the NEW-STATE SECTION. The read statement is skipped if the MISSING-SWT equals 1 in which case the program has read a card previously, found that the record was continued by a character in cc. 72, read the next card and found that it was not the continuation card. An error message is generated and the read statement is bypassed bacause the next record has already been read. Extensive error checks occur throughout GEN2 including syntax, logic, system limits, and Retrieval control parameters. The error messages generated by the program are intended to be self-explanatory. When the exact card columns of the error are known, they are specified in the error message. The input records are checked to make sure they are in ascending sequence by filename and a copy of the input record is written to the output file. If the record type is REPORT (TYPE-IN = 5) further processing is bypassed.

NEW-STATE1 checks for any change in the SUBQUERY field of the sort key and various counters, switches, and work areas are initialized in INITIAL-REC.

The first word, including control words, found in the retrieval source statement is decoded by FIRST-WORD and if it is in error, a message is generated. The FATAL-ERROR paragraph is executed when input source statements are not in ascending order.

Paragraphs C-SHL1 and C-SHL1A search all source statements for SHL statements (TYPE = 9). If one is found EXTRACT-WORD is performed to search for either LIST or FILE SHL statement.

Paragraphs C-SHL4 through C-SHL6 provide data from FFT tables 2, 7, 8, and 9 required when processing the LIST statement in this module.

The processing required when the subquery entry in the sort key changes is contained in SUBQ-HK through SH1. A test is made to insure that the parentheses for the preceding subquery are balanced. Likewise, the processing required when the filename field of the sort key change is contained in C-POOL through CP1. A test is made to insure that the literal pool does not exceed 9000 characters.

When the end of the input file is reached, END-STEP, ES1, and ES2 are executed. If unbalanced parentheses or a constant pool overflow occurred, error vectors are written.

The retrieval and modification, if required, of the FFT tables specified by the file name on the subquery card is accomplished by paragraphs C-SUBQUERY through S6S.

In C-SORT thru CS51 the sort control information is generated specifying ascending or descending for each field to be sorted in the record.

Error messages that contain no card column identifications of where the error occurred are written out by the WRITE-ERR1 SECTION and the error messages that do contain card column ID are written out by the WRITE-ERR SECTION.

If a continuation card is encountered while executing the EXTRACT-WORD SECTION, the READ-OP SECTION is performed.

The C-LIST paragraphs are executed when a LIST card is encountered. The paragraphs in C-LIST do not edit the items but create define statements which contain internal labels, edit mask or convert routines for each FFT name found in the LIST statement. Logical records 7 and 8 are used to retrieve the data for the labels. Logical records 7 and 9 are used for edit mask or convert routines. It is possible to have a label and an edit mask or a convert routine, but not edit and convert together with a label. To prevent a user from using these define statements, the letters LL$, EE$, and CC$ are used to prefix the FFT define name.

In the WRITE-STATE SECTION, edits are performed on each compiler statement before it is outputted. The WRITE-NOP SECTION is performed each time a group of left parentheses comprising a parenthetical expression is encountered.

The FLD-PART SECTION processes partial notation following a valid A-NAME or B-NAME.

The identification of the operation code encountered in logical statements and placing of corresponding entries in the OP field of the compiler statement record is accomplished in the DECODE-OP SECTION.

The OP codes are listed below for the different valid operators:

| | |
|---|---|
| LESS | 1 |
| EQUAL | 2 |
| GREATER THAN | 3 |
| BETWEEN | 4 |
| SATISFIES | 5 |
| SPECIAL OP | 6 |
| NOT LESS | 7 |
| NOT EQ | 8 |
| NOT GT | 9 |

| | |
|---|---|
| NOT BET | 10 |
| STOP | 11 |
| NOT SPEC OP | 12 |
| RECEIVE | 13 |
| SELECT | 14 |
| FLAG | 15 |
| SORT | 16 |
| MERGE | 17 |
| NOP | 18 |
| SET-DECODE | 19 |
| TRUE /GO/ | 20 |
| FALL /NOGO/ | 21 |
| LIMIT | 22 |
| KEEP | 23 |
| LOAD | 24 |
| CONTAINS (F&P) | 25 |
| CONTAINS (VAR) | 26 |

The LMODE codes are as follows:

| | |
|---|---|
| ANY NAME | 4 |
| ALL NAME | 5 |
| ANY HIT NAME | 6 |
| ALL HIT(S) NAME | 7 |

The DECODE-A SECTION determines whether an A-NAME is a periodic field, fixed field, or a defined field. A check is made for duplicate DEFINE statements, and if one is encountered for which a corresponding modify flag was not set, an error message is generated. Fields A1, A2, A3, and A4 in the output compiler statement are filled.

Multiple b-fields are processed in the DECODE-B SECTION. Fields B1, B2, B3, and B4 in the compiler statement are filled. Before literals and constants are added to the constant pool, the SNAME is checked to see if a conversion routine must be executed utilizing the literal or constant as input.

The EXTRACT-WORD SECTION scans a card picking up all the characters between delimiters and moving them to a hold area. This section is executed each time the next word in a source statement is required. If a # is encountered as a delimiter, spaces are moved to SNAME which suppresses conversion. If the word does not exceed 14 characters, the hold area is padded with trailing spaces until the sum of the word and trailing spaces equal 14.

In the LEAD-ZERO SECTION, leading zeros are placed before constants so that the lengths of the A-NAME and constant are equal and likewise the TRAIL-SPACE SECTION places trailing spaces after literals to match the length of A-NAME.

Numeric data is edited in the EXTRACT-NUM SECTION. The RIGHT-PARNI SECTION decodes right parenthesis used in parenthetical expressions and the TYPE-TEST SECTION checks to insure that the A-NAME and B-NAME have the same data type, i.e., numeric.

HLD-REC SECTION is performed when a LIST statement continues to another card.

S806-TRAP SECTION performs a load module validation on all convert routine names.

KEEPP SECTION performs validation on all field-names referenced in KEEP statements prior to being written out at the end of the subquery as KEEP object vectors.

WRITEV SECTION creates the KEEP object vectors at the end of each subquery.

END-STEP SECTION generates the "AT END" object vectors required for each subquery.

(c) Limitations. The limitations of the Retrieval Compiler are enumerated below:

1. Numeric a-fields and b-fields cannot exceed 15 digits.

2. Alphanumeric a-fields and b-fields cannot exceed 360 characters.

3. SORT FLAGGED statements must be the last statements in a subquery and may reference fields from only one periodic set.

4. The length of the constant pool comprising literals, constants, and defined data for all subqueries against the same file cannot exceed 9000 characters.

5. User convert routines cannot exceed 10,000 bytes of core storage.

6. The total number of batched retrieval source statements pertaining to the same data file, including those internally generated cannot exceed 500.

7. Only one retrieval source statement is allowed per card.

(d) I/O Data Sets.

1. FILE-IN. This is an input work file that was the output work file in the previously called load module.

2. FILE-OUT. This is the output work file and will be used as input by the next-called load module.

(5) GEN2X.

(a) Abstract of Driver.

1. Function. GEN2X is called when GEN1 finds non-sequential source statements. GEN1 separates sequential and non-sequential source statements into two separate work files. The non-sequential work file is input to GEN2X.

GEN2X is the non-sequential counterpart of GEN2. GEN2X reads the non-sequential source card images and generates the appropriate object vectors and constant pool strings.

The output of GEN2X is an ordered list of object vectors, constant pool strings, edited source images with any error messages or warnings and the query error vector (if an error is discovered in the source statements the query containing the non-sequential statement is set to 1).

The edited source statements and GEN2X messages are passed to GEN5A to be merged with the standard sequential query data.

The object and constant vectors are passed to GEN4X1.

2. Calling Sequence. The program calls four subroutines:

a. ROLIB2.

| | Length | Type | |
|---|---|---|---|
| CALL-SEQ | | | |
| OPERATION | 1 | A | Value R for Read |
| ITEM-NAME | 5 | A | Library Member Name |
| SUFFIX-NAME | 2 | N | Overflow Counter |
| FILLER | 1 | A | Not Used |
| ITEM-DATE | 5 | A | Not Used |
| ITEM-SIZE | 4 | N | No. of Characters Brought In |
| Receiving Area | 80 | A | Occurs 125 Times |

b. GEAB.

| | Length | Type | |
|---|---|---|---|
| SNAME | 8 | A | Name of Routine Being Checked |
| RET-CODE | HALFWORD | B | Return Code from Routine |

c. GEAL.

| | Length | Type | |
|---|---|---|---|
| module name | 8 | A | Name of Called Routine |
| SAVEREG | 18 FULLWORDS | B | Save Area for Registers |
| parameter(s) | Variable | Variable | Parameter(s) being passed to called routine but not used by GEAL. |

d. GEAM.

| | Length | Type | |
|---|---|---|---|
| Sending Area | | N | Address of Sending Area |
| Sending Area Offset | 2 | N | Character Subscript (Base 1) of Sending Area |
| Receiving Area | | N | Address of Receiving Area |
| Receiving Area Offset | 2 | N | Character Subscript (Base 1) of Receiving Area |
| Number of Characters | 2 | N | Number of Characters to Move. |

(b) Description. GEN2X is a subset of GEN2. Two types of modifications are made to GEN2. The first type of modification stripped GEN2 of the functions and options which are not appropriate to the level of processing which was proposed for non-sequential processing. This type of modification eliminated logic modes, parenthetical expressions, special operators, sorting, etc.

The second type of modification changed the limits of certain types of fields and added the final auxillary processing to sort the object vectors.

The final process of GEN2X is to sort the object vectors which it has stored in an array. The array can contain 100 object vectors which is one of the new limits on the retrieval expression. The sort groups all object vectors which are associated with the same list.

The other portions of GEN2X are directly from GEN2. Refer to the GEN2 for a further description.

(c) Limitations. The same limitations as GEN2 plus the following:

1. No more than 100 generated object statements (generally one object statement per retrieval statement).

2. No parenthesis.

3. Operators: LESS, EQUAL, GREATER, NOT LESS, NOT EQUAL, NOT GREATER.

4. The b-field of an object vector cannot refer to an FFT item.

(d) I/O Data Sets. Same file structure as GEN2 except with different names.

(6) GEN4X1.

(a) Abstract of Driver.

1. Function. GEN4X1 reads the object vectors created by GEN2. The object vectors are used to query the list files. GEN4X1 reads the list files one at a time and executes the group of vectors that concern each list. The result of GEN4X1 is the key file (DDname - KEYFL). The key file is the cumulative result of the keys (of each list) that satisfied the object vectors for a particular list. The key file is passed to GEN4X2.

2. Calling Sequence. GEN4X1 calls two subroutines.

a. GEAM

| | Length | Type | |
|---|---|---|---|
| Sending Area | | N | Address of Sending Area |
| Sending Area Offset | 2 | N | Character Subscript (Base 1) of Sending Area |
| Receiving Area | | N | Address of Receiving Area |
| Receiving Area Offset | 2 | N | Character Subscript (Base 1) of Receiving Area |
| Number of Characters | 2 | N | Number of Characters to Move |

b. GEAX.

| | Length | Type | |
|---|---|---|---|
| Record Area | | N | Address of Record |
| Constant Area | | N | Address of Constant Area |
| Vector Area | | N | Address of Object Vector |

(b) Description. GEN4X1 reads all the sorted object vectors from GEN2X for one list. The list is opened, read, and analyzed according to the object vectors.

As each vector is processed, the keys for the list values which satisfy the object vector are written in a sort file. When the list is exhausted or the object vectors have been satisfied, GEN4X1 begins reading another set of object vectors.

When all object vectors have been satisfied, GEN4X1 returns control to GEN0.

(c) Limitations. Not applicable.

(d) I/O Data Sets.

1. NSW1. NSW1 is the vector/constant file created by GEN2X. NSW1 also includes the original source/error records.

2. NSW2. All type one records, source and error records, from NSW1 are written in NSW2. Any error messages from GEN4X1 are written on this file.

3. LISTFILE. Dummy DCB for the list files.

4. NSAX. NSAX is the key file. When a list entry satisfies the vector which has examined it, the key to that entry is written in the NSAX file. NSAX file is passed to a sort, GEN4X2.

(7) GEN4X2.

(a) Abstract of Driver.

1. Function. GEN4X2 sorts the key file which was created by GEN4X1.

2. Calling Sequence. GEN4X2 calls one program; GEAM.

| | Length | Type | |
|---|---|---|---|
| Sending Area | | N | Address of Sending Area |
| Sending Area Offset | 2 | N | Character Subscript (Base 1) of Sending Area |
| Receiving Area | | N | Address of Receiving Area |
| Receiving Area Offset | 2 | N | Character Subscript (Base 1) of Receiving Area |
| Number of Characters | 2 | N | Number of Characters to Move |

(b) Description. GEN4X2 sorts the key file which was created by GEN4X1. The completion code of the sort process is tested after the sort has terminated. If the completion code indicates an unsuccessful sort, NSW1 and NSW2 files are opened. NSW1 is an input file and contains the output source/error file from GEN4X1. All records in NSW1 plus an error message (bad sort) are written on file NSW2.

(c) Limitations. Not applicable.

(d) I/O Data Sets.

1. KEYFL. KEYFL contains the keys from GEN4X1 which are to be sorted and replaced in the KEYFL.

2. NSW1. Input source image file. Only opened to transmit an error in the SORT procedure.

3. NSW2. Output file for source/error images. Only opened to transmit an error in the SORT procedure.

(8) GEN4X3.

(a) Abstract of Driver.

1. Function. GEN4X3 reads the sorted key file, determines the acceptable record numbers and when directed, creates a summary file by directly returning those records.

2. Calling Sequence. GEN4X3 calls two programs.

a. GEAM.

| | Length | Type | |
|---|---|---|---|
| Sending Area | | N | Address of Sending Area |
| Sending Area Offset | 2 | N | Character Subscript (Base 1) of Sending Area |
| Receiving Area | | N | Address of Receiving Area |
| Receiving Area Offset | 2 | N | Character Subscript (Base 1) of Receiving Area |
| Number of Characters | 2 | N | Number of Characters to Move |

b. GEAN.

| | Length | Type | |
|---|---|---|---|
| P-FNME | 8 | A/N | DDname |
| File name | 8 | A/N | File name |

(b) Description. GEN4X3 analyzes the sorted key output from GEN4X2. If a sequence of keys satisfies the user's request, a check is made to determine if a summary file is to be created. If the summary is to be made, the record is retrieved and written on the summary file.

When all the sorted keys have been processed, GEN4X3 transmits a number which is the number of acceptable records.

(c) Limitations. Not applicable.

(d) I/O Data Sets.

1. KEYFILE. KEYFILE is the sorted output from GEN4X2. It contains the keys which were retrieved by GEN4X1.

2. SUMMFL. SUMMFL is the dummy file which is used to write the summary file.

3. INFILE. INFILE is a packed random file. Standard sequential MIDMS files are packed into fixed length random records. The key to retrieve a standard MIDMS record from the packed record is contained in the POINTR file. The POINTR file documents which packed random record(s) contain the standard record.

4. POINTR. The POINTR file documents where a particular standard record has been placed in the packed file.

5. NSW1. NSW1/NSW2 are the input/output files respectively. If GEN4X3 discovers an error (i.e., in the pointer of packed files), NSW1 records are written in NSW2 with an error message which describes the error.

6. NSW2. (Refer to 5).

(9) GEN3A.

(a) Abstract of Shorthand Language (SHL) Output Compiler.

1. Function. This module performs the syntax editing of a free formatted Shorthand Language output source statements; creates output object vectors (TYPE = 5s) required by the Output logic processor; builds constant pool strings containing: (1) constants, literals, and defined values encountered in SHL source statements, (2) literal labels, edit fields, and convert routines found in FFT tables, and (3) COUNT and SUM work areas as specified in SHL source statements; builds a query vector containing non-zero entries for those queries that contain errors in either retrieval and/or standard output source statements or SHL output source statements; writes an error message for the first error encountered in each invalid SHL output source statement; passes to the retrieval and logic processors all output from the Retrieval Compiler and the SHL Output Compiler.

2. Calling Sequence. GEN3A calls five programs.

a. POLIB2.

| | Length | Type | |
|---|---|---|---|
| CALL-SEQ | | | |
| OPERATION | 1 | A | Value R for Read |
| ITEM-NAME | 5 | A | Library Member Name |
| SUFFIX-NAME | 2 | N | Overflow Counter |
| FILLER | 1 | A | Not Used |
| ITEM-DATE | 5 | A | Not Used |
| ITEM-SIZE | 4 | N | No. of Characters Brought In |
| Receiving Area | 80 | A | Occurs 125 Times |

b. GEAL.

| | Length | Type | |
|---|---|---|---|
| module name | 8 | A | Name of Called Routine |
| SAVEREG | 18 FULLWORDS | B | Save Area for Registers |
| parameter(s) | Variable | Variable | Parameter(s) being passed to called routine but not used by GEAL. |

c. GEAB.

| | Length | Type | |
|---|---|---|---|
| SNAME | 8 | A | Name of Routine Being Called |
| RET-CODE | HALFWORD | B | Return Code from Routine |

d. GEAA.

| | Length | Type | |
|---|---|---|---|
| ARG | | | |
| RG1 | 6 | ID | Packed Decimal, Used by OS |
| RG2 | 6 | ID | Packed Decimal, Used by OS |
| OUTX | | | |
| DA | 2 | A | Day of Month |
| MO | 3 | A | Month |
| YR | 2 | A | Year |

e. GEAM.

| | Length | Type | |
|---|---|---|---|
| parm1 | HALFWORD | B | Base Address of Sending Area |
| parm2 | HALFWORD | B | Displacement of Sending Area |
| parm3 | HALFWORD | B | Base Address of Receiving Area |
| parm4 | HALFWORD | B | Displacement of Receiving Area |
| parm5 | HALFWORD | B | Length of String to be Moved |

(b) Description. All initial housekeeping is performed in HOUSE-KEEP.

Reinitialization of work areas, counters, and switches prior to processing of each new SHL report is performed in NEW-REPORT.

Input source statement records are read to determine when the last record in the input file (FILE-IN) has been encountered (TYPE = 4), when a SHL source statement has been encountered, and when each new SHL report has started. The latter tests are performed in NEW-STATE, NEW-STATE1, and NEW-STATE2.

Extensive error checks occur throughout GEN3A including syntax, logic, system limits, and SHL output control statements. The error messages generated are intended to be self-explanatory. When the exact card columns of the error are known they are specified in the error message.

This program processes SHL source statements (TYPE-IN = 9) which are changed to TYPE-IN = 5 just prior to being copied to the output file.

Paragraphs NS1B, NS1C, NS1E, and NS1CC through NS1Z contain the processing required for the previous SHL report when the query number associated with each SHL report changes and when the end of file condition occurs for the input file.

The END-ROUT through XEND-ROUT paragraphs are executed at the end of each query. These paragraphs produce the user header and trailer lines only if the CLAS-SWT equals one. The movement of data comes from the saved vectors produced by C-CLASS, HED-HN, and TRL-TN in the hold area called SHL-VECT. If the CLAS-SWT equals zero a fatal error occurs with a message at the end of the query.

Reinitialization required for each new SHL source statement is performed in paragraphs INITIAL-REC and PPP.

The first word, including control words, found in each SHL output source statement is decoded by FIRST-WORD.

In LAST-ONE through L02, the error vector record (TYPE = 4) is updated to indicate those queries for which errors were encountered in the SHL report.

Paragraphs C-FILE through S8S retrieve and modify logical records 1, 2, and 3 for each unique file name specified on a FILE statement.

Paragraphs which start processing the appropriate SHL source statements are identified by C-, i.e., C-FINAL, C-IF, C-AND, C-OR, C-OMIT, C-STOP, C-RESET, C-SET, etc.

In paragraph C-LIST through LST5A, syntax checking is performed on the LIST statement; when partials are encountered partial data including pointers to the partial data are processed. If the data fields and partials in a LIST statement are valid, processing of the data field entries starts and proceeds in the sequence: define names, fixed FFT fields, periodic FFT fields, variable FFT fields.

The WARN-MESSAGE paragraph writes caution messages without flagging the query number in the error vector.

The C-LSIZE paragraphs are executed when an LSIZE or LINESIZE card is encountered. The numbers or letters (see line size format) after LSIZE indicate the number of positions required for an output line. The system defaults to 132 if no LINESIZE card is present. If the number for line size is 20 or less the system will give an error message. If CRT is used 72 positions are allocated; if TTY is used 100 positions are allocated, and if PRT is specified 132 positions are moved to the line size area. In the case of numeric data EXTRACT-NUM is used for verification of digits.

Paragraphs C-CLASS through PRT-TOP-LINE7 are executed when compiler GEN3A receives a CLASS statement. A brief description of each paragraph is as follows:

C-CLASS picks up and verifies the literal within quotes using CLA-HED section. C-XXX1 loads the data into the constant pool using MOV-CH section. CENT-CLASS centers the literal between a specified or default line size (see LINE SIZE). CL-1 positions the data for the output area and moves the vector to a hold area using WRT-VECT and in addition sets CLAS-SWT to a value of 1. MOV-PAGE moves "PAGE" to the output area and loads "PAGE" into the constant

pool and inserts the vector into a hold area using WRT-VECT. ADD-PAGENO adds one to the reserved word "PAGENO" and moves the result to the constant pool using the LOAD-CONSTANT section and moves the vector into a hold area using WRT-VECT. EMOV-PAGENO suppresses leading zeros in the page number and moves the vector into the hold area using WRT-VECT. CALL-SYSDATE, using the ALC routine "DATETIME", converts the Julian date to day, month, year. MOV-SDATE moves data to the constant pool using CXXX1 and moves the data to the hold area using WRT-VECT. SPACE-26 creates the vector for spacing before printing and moves the vector to a hold area using WRT-VECT. PRT-TOP-LINE7 creates the vector for printing the first line of output and moves the vector to the hold area using WRT-VECT and returns to NEW-STATE. The paragraph WRT-VECT adds one to the index (SHL-IDX) and moves vectors to the hold area called SHL-VECT.

When compiler GEN3A encounters the H1 through H10 cards, paragraphs HED-HN through HN-3 are executed for creation of heading line vectors. The paragraphs verify the literal using EXTRACT-WORD section. If numeric data is found after the literal, EXTRACT-NUM verifies this data. If no data is found CNTER-HN paragraph centers the literal for output depending on the line size. Finally in HN-3 the vector is moved to a hold area using WRT-VECT and returns to NEW-STATE.

When T1 through T4 are received paragraphs TRL-TN and TN-1 are executed for outputting trailer vectors. In paragraph TRL-TN a switch TN-SWT is set to one and goes to HED-HN. In HN-3 a check is made on TN-SWT; if equal to one the program goes to paragraph TN-1 which moves the vector to a hold area and returns to NEW-STATE.

When a DISPLAY card is received in GEN3A the program immediately goes to C-DISPLAY. A DISPLAY statement can contain literals, define names, SPn and/or a numeric number. The paragraphs that follow C-DISPLAY perform validation for correct format, use, and positioning of data for output. If SPn (SPACE n) is used between data elements the "n", a numeric 1 to 132, will be checked using EXTRACT-NUM or if a numeric starting position is specified the number will be verified using EXTRACT-NUM; all others will use EXTRACT-WORD for validation. Since the FINAL-OUTPUT can have conditional statements a period check is made at the end of each SPn, numeric number, define name, and literal. If present, the indicator XFALSE will be initialized to 1. In order to move a 1 to XFALSE in paragraph DP-1A, a 2 is moved to SWTX and when WRT-REC is performed and SWTX equals 2, a 1 is moved to XFALSE and the record is written out by using WRITE-STATE.

Paragraphs C-COUNT through CS18 process both the COUNT and SUM statements. Syntax checking is performed to insure valid FFT names are referenced in the latter type of statements, i.e., data type is numeric. A search is performed through define names to determine if an internal COUNT or SUM work area has already been set up for each FFT name encountered. If none are found, internal work areas with appropriate define names are created.

Error messages that require no card column identification of where the error occurred are written out by WRITE-ERR1 SECTION and error messages that do require a card column identification are written out in WRITE-ERR SECTION.

The identification of the operation code encountered in logical statements and placing of corresponding entry codes in the OP field of the compiler statement record is accomplished by the DECODE-OP SECTION.

The OP codes are listed below for the different valid operators:

| | |
|---|---|
| LESS, LT, BEFORE, EARLIER | 1 |
| EQUAL, EQ, EQUALS | 2 |
| GREATER, GT, LATER, AFTER | 3 |
| NOT (LESS, LT, BEFORE, EARLIER) | 4 |
| NOT (EQUAL, EQ, EQUALS) | 5 |
| NOT (GREATER, GT, LATER, AFTER) | 6 |

The FLD-PART SECTION processes partial notation following a valid A-NAME or B-NAME field.

The DECODE-A SECTION determines whether an A-NAME is a periodic field, fixed field, or a defined field. Duplicate define statements are checked for and if one is encountered for which a corresponding modify flag was not set, an error message is generated. Fields A1, A2, A3, and A4 in the output compiler are filled.

Multiple b-fields found in IF, AND and OR statements are processed in the DECODE-B SECTION. Fields B1, B2, B3, and B4 in the compiler statement are filled. Before literals and constants are added to the constant pool, the SNAME is checked to see if a conversion routine must be executed utilizing the literal or constant as input. Load module validation is also performed on SNAME.

Object vectors completed in DECODE-A for conditional statements not found in the FINAL-OUTPUT SECTION are saved in a hold area rather than being immediately written out.

If a continuation card is encountered while executing the EXTRACT-WORD SECTION, the READ-OP SECTION is performed. The EXTRACT-WORD SECTION scans a card picking up all the characters between delimiters and moving them to a holding area. This section is executed each time the next word in a source statement is required. If a # is encountered as a delimiter, spaces are moved to SNAME which suppresses conversion. If the word does not exceed 14 characters the hold area is padded with trailing spaces.

In LOAD-CONSTANT, literals are moved to the constant pool.

WRITE-CDYN SECTION creates constant records (TYPE = 7) that contain most recent 90 characters of literals processed.

EXTRACT-NUM SECTION scans a field for non-numeric data.

FIX-F SECTION modifies logical record 2 entries and loads a table with values identifying FFT fields as fixed, periodic, or variable.

TYPE-TEST checks A3 and B3 in required object vectors to determine that they have the same data type, i.e., numeric.

USER-ERROR SECTION outputs any error messages generated during the execution of a user's subroutine.

S806-TRAP SECTION performs a load module validation on convert routines.

DUP-CHIC SECTION determines if a define name has been created that has a mnemonic identical to an FFT name.

EXTRACT-PART SECTION is used in the processing of the LIST statement. It validates all partials, processes data for each partial, and saves this data in a hold area which is addressable by pointers.

LINE-POINTER SECTION determines the number of related types of data names, i.e., fixed, that are to be written on one line and the corresponding starting position of each associated label and data contents.

PARTIAL-SEARCH SECTION is used to determine what field-names in a LIST statement had a partial associated with it.

DEFINE-SEARCH SECTION searches the table of define names for a system generated internal label, edit mask, or current routine name for each field-name found in a LIST statement.

WRITE-OBJ SECTION determines the LINE and BEGIN system generated paragraphs required to output the labels and data fields associated with each field-name found in a LIST statement. Tests are performed to determine where appropriate conditional and set object vectors are required and their correct placement in each generated LINE and BEGINS paragraph.

LOAD-AA SECTION loads the mnemonic name of a FFT field name in the constant pool which will be used as the label when processing the LIST statement.

GEN-LABELS SECTION generates the object vectors required in the system generated LINE paragraph to output labels associated with each field-name found in a LIST statement and when the BLP statement option has not been specified.

LOAD-B SECTION fills B1, B2, B3, and B4 fields for object vectors created to write labels.

GEN-34 SECTION is performed when LINE and BEGIN paragraphs are generated during the processing of the LIST, SUM, and COUNT statements.

For every object statement that is created, WRITE-STATE SECTION fills in the following fields: LMODE, TRUE, FILENAME, and CARD-CTR.

GEN-SET SECTION generates the object vectors required for periodic set selection specified by the user when the appropriate LINE and BEGINS paragraphs are created for periodic data.

GEN-LOGIC SECTION is performed for LIST, COUNT, and SUM statements which are conditioned. Conditional object vectors are written at the appropriate positions in LINE and BEGINS paragraphs required for processing the latter statements.

GEN-DATA SECTION generates the object vectors to output the data contents of all field-names referenced in a LIST statement.

LOAD-A SECTION loads the FFT name into the constant pool when it is required as a label. This is required only when it an FFT field-name does not have an internal label.

GEN-OVERFLOW SECTION is performed when a single data field's contents exceed the line size specified in the SHL report.

GEN-SPACE SECTION is performed when a space object vector is required.

The PERIOD-SCAN SECTION looks for a period at the end of a sequence of characters.

SUM-SEARCH SECTION searches the list of define names for SUM and COUNT work areas.

LOAD-DEFINE SECTION adds SUM and COUNT work area names to the list of define names.

The following sections are used to fill A1, A2, A3, A4, B1, B2, B3, B4 entries in object vectors (TYPE = 6): MOVE-DB, MOVE-FA, MOVE-DA, MOVE-AB.

WRITE-SC-OBJ SECTION generates the object vectors required to add occurrences of data fields to the corresponding COUNT work areas and the contents of numeric fields to the corresponding SUM work areas.

SC-OUTPUT SECTION generates the object vectors to write out the contents of all the SUM and COUNT work areas in FINAL-OUTPUT.

(c) Limitations. The limitations of the Output Source Statement Compiler are listed below:

1. Alphanumeric field-names cannot exceed 360 characters.

2. Numeric field-names cannot exceed 15 digits.

3. Only one output source statement is allowed per card.

4. IF COMPLETE and IF CHANGE operators not supported.

5. Table lookup not supported.

6. System formatted output. LIST statement processes FFT field-names according to the sequence in which they appear in FFT tables. All data fields are printed in this order: DEFINE, FIXED, PERIODIC, and VARIABLE.

7. FFT field names may not have both an internal edit mask and convert routine name.

(d) I/O Data Sets.

1. FILE-IN. This is an input work file that was the output work file in the previously called load module.

2. FILE-OUT. This is the output work file and will be used as input by the next-called load module.

(10) GEN3.

(a) Abstract of Output Compiler.

1. Function. The Output Source Statement Compiler (GEN3) performs the syntax editing of the free formatted output source statements; builds vector arrays for each output source statement; builds constant pool strings containing constants, literals, and defined values encountered in output source statements; builds a query vector containing non-zero entries for those queries that contain errors in either a retrieval or output source statement; writes an error message for the first error encountered in each invalid output source statement; passes to the Retrieval Logic Processor all output from Retrieval and Output compilers.

2. Calling Sequence. No parameters are passed to this module when it is initially called. The program calls four subroutines:

a. ROLIB2.

| | Length | Type | |
|---|---|---|---|
| CALL-SEQ | | | |
| OPERATION | 1 | A | Value R for Read |
| ITEM-NAME | 5 | A | Library Member Name |
| SUFFIX-NAME | 2 | N | Overflow Counter |
| FILLER | 1 | A | Not Used |
| ITEM-DATE | 5 | A | Not Used |
| ITEM-SIZE | 4 | N | Number of Characters Brought In |
| Receiving Area | 80 | A | Occurs 125 Times |

b. GEAL.

| | Length | Type | |
|---|---|---|---|
| module name | 8 | A | Name of Called Routine |
| SAVEREG | 18 FULLWORDS | B | Save Area for Registers |
| parameter(s) | Variable | Variable | Parameter(s) being passed to called routine but not used by GEAL. |

c. GEAB.

| | Length | Type | |
|---|---|---|---|
| SNAME | 8 | A | Name of Routine Being Called |
| RET-CODE | HALFWORD | B | Return Code From Routine |

d. GEAM.

| | Length | Type | |
|---|---|---|---|
| parm1 | HALFWORD | B | Base Address of Sending Area |
| parm2 | HALFWORD | B | Displacement of Sending Area |
| parm3 | HALFWORD | B | Base Address of Receiving Area |
| parm4 | HALFWORD | B | Displacement of Receiving Area |
| parm5 | HALFWORD | B | Length of String to be Moved |

(b) Description. Records are initially read in the NEW-STATE paragraph. The read statement is skipped if the MISSING-SWT equals 1 in which case the program has read a card previously, found that the record was continued by a character in cc 72, read the next card and found that it was not the continuation card. An error message is generated and the read statement is bypassed because the next record has already been read. Extensive error checks occur throughout GEN3 including syntax, logic, system limits, and output control parameters. The error messages generated by the program are intended to be self-explanatory. When the exact card columns of the error are known they are specified in the error message. The input records are checked to make sure they are in ascending sequence by file name and a copy of the input record is written to the output file. The section only processes records whose TYPE-IN equals 5 (REPORT RECORDS).

NEW-STATE1 tests for a change in the query field of the sort key.

Paragraphs NS1B through NS1Z contain the processing required for the previous report when the query field value changes.

Reinitialization of work areas, counters, and switches prior to processing of a new report is done in NEW-REPORT.

The first word, including control words, found in the output source statement is decoded by FIRST-WORD and if it is an error, a message is generated.

In LAST-ONE through LO2, the error vector record (TYPE = 4) is updated to indicate those queries for which output source errors were encountered.

Paragraphs C-FILE thru S6S retrieve and modify the FFT tables for the filename specified on the file card.

In the C-STOP paragraph, the STOP parameter is examined when the STOP is used in conjunction with a conditional; the output can be terminated when it reaches a certain user-specified limit.

The MOVE-WARNING and WARN-MESSAGE paragraphs write warning messages without flagging the query in the error vector.

Error messages that contain no card column identification of where the error occurred are written out by the WRITE-ERR1 SECTION and the error messages that do contain card column identification are written out by the WRITE-ERR SECTION.

The identification of the operation code encountered in logical statements and placing of corresponding entry codes in the OP field of the compiler statement record is accomplished by the DECODE-OP SECTION.

The OP codes are listed below for the different valid operators:

| | |
|---|---|
| LESS, LT, BEFORE, EARLIER | 1 |
| EQUAL, EQ, EQUALS | 2 |
| GREATER, GT, LATER, AFTER | 3 |
| NOT (LESS, LT, BEFORE, EARLIER) | 4 |
| NOT (EQUAL, EQ, EQUALS) | 5 |
| NOT (GREATER, GT, LATER, AFTER) | 6 |

The FLD-PART SECTION processes partial notation following a valid A-NAME or B-NAME field.

The DECODE-A SECTION determines whether an A-NAME is a periodic field, fixed field, or a defined field. Duplicate define statements are checked for and if one is encountered for which a corresponding modify flag was not set, an error message is generated. Fields A1, A2, A3, and A4 in the output compiler are filled.

Multiple b-fields found in IF, AND, and OR statements are processed in the DECODE-B SECTION. Fields B1, B2, B3, and B4 in the compiler statement are filled. Before literals and constants are added to the constant pool, the SNAME is checked to see if a conversion routine must be executed utilizing the literal or constant as input. Load module validation is also performed on SNAME.

In the WRITE-STATE SECTION, edits are performed on each compiler statement before it is outputted.

If a continuation card is encountered while executing the EXTRACT-WORD SECTION, the READ-OP SECTION is performed. The EXTRACT-WORD SECTION scans a card picking up all the characters between delimiters and moves them to a holding area. This section is executed each time the next word in a source statement is required. If a # is encountered as a delimiter, spaces are moved to SNAME which suppresses conversion. If the word does not exceed 14 characters the hold area is padded with trailing spaces.

In the LEAD-ZERO SECTION leading zeros are placed before constants and likewise the TRAIL-SPACE SECTION places trailing spaces after literals that require padding.

Numeric data is edited in the EXTRACT-NUM SECTION and the TYPE-TEST SECTION checks to insure that the A-NAME and B-NAME have the same data type, e.g., numeric.

In the USER-ERROR SECTION, invalid data returned from a convert routine is written out along with an error message.

The PERIOD-SCAN SECTION looks for a period at the end of a sequence of characters. If one is found, PERIOD-SWT is set, a space is written over the period, and a one is moved to XFALSE.

(c) Limitations. The limitations of the Output Source Statement Compiler are listed below:

1. Alphanumeric a-fields and b-fields cannot exceed 360 characters.

2. Numeric a-fields and b-fields cannot exceed 15 digits.

3. The program does not process FFT labels.

4. All retrieval must be performed before output is generated.

5. Implicit convert routines are not available (must be defined explicitly).

6. ZERO and SPACEs are invalid as a b-field entry.

7. Only one output source statement per card.

(d) I/O Data Sets.

1. FILE-IN. This is an input work file that was the output work file in the previously called load module.

2. FILE-OUT. This is the output work file and will be used as input by the next-called load module.

(10.1) GEN3B

(a) Abstract of OP Library Maintenance

1. Function. The OP Library Maintenance program maintains reports (RIT's) in object form on the MIDMS Library. Since the source version of each object report will also be maintained on the library, GEN3B will automatically add or replace a source report (if necessary) each time a corresponding object report is updated (i.e., added or replaced). Source and their corresponding object reports always appear on the library with names of the form XXXXR and XXXXJ, respectively. In addition to updating the MIDMS library, this program provides a standard RTOP listing plus messages giving the names of the object (and possibly source) reports which were added/replaced. At most, one object and one (corresponding) source report will be added/replaced during a single execution of GEN3B.

2. Calling Sequence. The program calls three subroutines:

a. GEAM

| | Length | Type | |
|---|---|---|---|
| parm1 | HALFWORD | B | Base Address of Sending Area |
| parm2 | HALFWORD | B | Displacement in Sending Area |
| parm3 | HALFWORD | B | Base Address of Receiving Area |
| parm4 | HALFWORD | B | Displacement in Receiving Area |
| parm5 | HALFWORD | B | Length of String to be Moved |

b. DATESUB

| | Length | Type | |
|---|---|---|---|
| parm1 | 5 BYTES | N | Current Date (YYDDD) |

c. DBLIB

| | Length | Type | |
|---|---|---|---|
| parm1 | 18 BYTES | A | Address of Parameter Buffer, as follows: |
| | 1 BYTES | A | Operation Code (Read, Write, etc.) |
| | 5 BYTES | A | Name of Report (XXXXR or XXXXJ) |
| | 2 BYTES | A | Member number (01, 02, etc.) |
| | 1 BYTE | A | Unused |
| | 5 BYTES | A | Date member written to library |
| | 4 BYTES | A | Size of member |
| parm2 | 10K BYTES | A | Address of Library Read/Write Buffer |

(b) Description. Program initialization consists of opening the input and output data sets and printing the "MIDMS START PROCESSING" page. TYPE-S (2) is then tested to determine the run type. If TYPE-S (2) equals 7 or 8 a COMP-OP card has been encountered. TYPE-S (2) equal 9 implies a normal MIDMS RTOP execution and GEN3B should not be called; this is a fatal error and processing terminates. If GEN1 detected an error in the COMP-OP card the next record read will be the error message (Q-IN equal Ø and TYPE-IN equal 3); otherwise, a QUERY card must be found. The "QUERY" page is printed, DUPLICATE-CHECK (see below) is performed and the report processing commences.

As the input data set is read it is sequence checked. Source records (TYPE-IN equal 4 or 5) are placed in the library buffer (if the source report is to be replaced) and printed. Object records (TYPE-IN equal 6 or 7) are always placed in the library buffer since object reports are always updated. Updating is controlled by TYPE-S (2) which is set by GEN1: if TYPE-S (2) equals 7 only the object report is updated; if TYPE-S (2) equals 8 both the source and object reports are updated. Since library member lengths may not exceed 10,000 bytes multiple writes to the library for a source or object report may be required. No library updating occurs if the error indicator (Q-SWT (1)) is non-zero. When the end of the input file is reached, the library buffer is flushed, messages indicating the names of the reports which were added/replaced are printed, the "QUERY" and and "END MIDMS PROCESSING" pages are printed, the data sets are closed and control is returned to GENØ.

There are two major subroutines included within GEN3B. They are WRITE-LIBRARY and DUPLICATE-CHECK. WRITE-LIBRARY is executed each time a member is to be written to the library. MCOUNT is incremented by one at each execution to maintain a count of, and uniquely name, each

member written to the library. For example, source reports exceeding 10,000 bytes will be stored in successive library members having names of the form XXXXRØ1, XXXXRØ2, XXXXRØ3, etc. The purpose of DUPLICATE-CHECK is to delete existing library members if a replace operation is to occur. Successive calls to the librarian subroutine are made (incrementing MEMBER-SUFFIX at each call) until all members constituting a single report are deleted.

(c) Limitations. The limitations of this program are:

1. A maximum of one source and one object report may be updated in a single execution.

2. Library members may not exceed 10,000 bytes.

3. Reports cannot by deleted using GEN3B.

(d) I/O Data Sets.

1. OPSTATE. This is the input data set; it contains a COMP-OP record, a QUERY record, a source report (either Standard or SHL) and an object report. If errors were detected by GEN1 or the language compilers, the error messages are included on this data set. Record types on this data set are Standard Language (TYPE-IN equal 5), Shorthand Language (TYPE-IN equal 4), Instruction vectors (TYPE-IN equal 6) and Constant Vectors (TYPE-IN equal 7).

2. REPORTOP. This is the printer output file used for writing the hardcopy report.

(11) GEN4.

(a) Abstract of Retrieval Periodic Logic Processor.

1. Function. The Retrieval Logic Processor retrieves and processes data records using vector arrays and constant pool strings created in the Retrieval Compiler. The queries that were flagged due to errors in either retrieval or output compilation are bypassed and corresponding vector arrays and constant pool strings are not processed. All subqueries referencing the same file and found in queries that contain no retrieval or output source errors are loaded into a hold area. A link edit is performed on these statements, bypassing those which logically do not contribute to final go or no-go decision. For each subquery for which the data record satisfies the retrieval logic, a copy of that record is written out.

2. Calling Sequence. The program calls seven subroutines:

a. GEAL.

| | Length | Type | |
|---|---|---|---|
| module name | 8 | A | Name of Called Routine |
| SAVEREG | 18 FULLWORDS | B | Save Area for Registers |
| parameter(s) | Variable | Variable | Parameter(s) Being Passed to Called Routine but not Used by GEAL |

b. GEAM.

| | Length | Type | |
|---|---|---|---|
| parm1 | HALFWORD | B | Base Address of Sending Area |
| parm2 | HALFWORD | B | Displacement of Sending Area |
| parm3 | HALFWORD | B | Base Address of Receiving Area |
| parm4 | HALFWORD | B | Displacement of Receiving Area |
| parm5 | HALFWORD | B | Length of String to be Moved |

c. GEAX.

| | Length | Type | |
|---|---|---|---|
| RX | 9999 | A | Data Record |
| CONX | 9000 | A | Constant Pool |
| STATE-FORM | 12 HALFWORDS | B | Object Statement |
| N | HALFWORD | B | Subscript of Active Statement |

d. GEAG.

| | Length | Type | |
|---|---|---|---|
| RX | 9999 | A | Data Record |
| SET-TABLE | 12 | A | Occurs 50 Times - Contains Description of Each Set in Record |
| SL | HALFWORD | B | Subscript of Last Used Occurrence in SET-TABLE |
| ABM | HALFWORD | B | Number of Flags in Buffer |
| FLAG-AREA | | | Address of Flag |
| OP | HALFWORD | B | Operation Code for Current Statement |
| SWTGO | HALFWORD | B | Return Code |

e. GEAC.

| | Length | Type | |
|---|---|---|---|
| RX | 9999 | A | Data Record |
| SET-W-TBL | 12 | A | Occurs 50 Times - Contains Description of Each Set in Record |
| SL | HALFWORD | B | Subscript of Last Used Occurrence in SET-W-TBL |

f. GEAS.

| | Length | Type | |
|---|---|---|---|
| RX | 9999 | A | Data Record |
| CONX | 9000 | A | Constant Pool |
| STATE-FORM | 12 HALFWORDS | B | Object Statement |
| KSR1 | 85 | A | Sort Key |

g. GEAD.

| | Length | Type | |
|---|---|---|---|
| SNAME | 8 | A | Subroutine Name |
| SAVEREG | 18 FULLWORDS | B | Save Area for Registers |
| AREALINKED IN-LENGTH | FULLWORD | B | Load/Delete Switch |

(b) Description. Processing starts by opening CON-FILE, the data set created by the language processors. This file contains source statements, generated tables from the source statements, constant pools, error and warning messages. From CON-FILE, GEN4 picks up an error vector (QUERY-SWT) which indicates which queries contain source statement errors and are not to be processed. GEN4 then selects all table and constant pool records from all error-free queries for the first file to be processed. The GEN4 subsystem then determines the structures of the data file from tables that were created by file structuring. The data file could be a MIDMS file or another type of file, such as the Binary Coded Decimal Automated Intelligence File (BCD AIF). The actual record structure of a file, whether fixed or variable length, periodic or non-periodic, is of no consequence to the subsystem, as long as there is a description of the data format available to the system. This means that virtually any existing data file can be processed by the retrieval subsystem if its record structure has been described to MIDMS.

The operator is notified on the console that processing is starting of a particular data file. If the file is on-line, processing will commence immediately, otherwise the system will go into a wait state, and a mount command will be issued on the console for the particular volumes needed. When the volumes are mounted, processing will begin. To avoid this wait state, the user may omit the DEFER MOUNTING parameter in the JCL. In this case, the volumes will be asked for when the system goes into execution, thereby allowing the operator more time to get the volumes mounted, however, this means the system will tie up the peripheral resources of the installation; if that is not acceptable, the DEFER MOUNTING parameter may be used.

The data record will be read in and the periodic control words, if any, will be extracted from the data record and saved in an area called SET-W-TBL. Then the location, in the case of the first subquery program, is obtained from the Q3 table. Subquery statements are processed individually and in an order dictated by the logic indicated by the user. If the logic of this subquery is satisfied by the data in the record, this particular record will be written out as an intermediate answer file (OUT-FILE). If the user has specified that he wants a summary file the record will be written on the particular volume specified by the user rather than on the intermediate answer file.

If there is another subquery directed against this record the location of the next subquery is obtained from the Q3 table and the cycle is repeated until all subqueries have been processed against this data record. Then a new data record is obtained from the input data file and processing of all subqueries will be repeated as indicated before.

There is a statement, STOP, which may be conditioned with logical statements. If such a statement is satisfied in a particular subquery, at that point the said subquery will be purged from the module and processing will continue with the remaining subqueries. There is a special case when all subqueries may be purged before the end of data file has been reached. In this case, GEN4 will simulate an end of file condition for the input data file and will proceed with the next data file, if any are left to be processed. If GEN4 does, in fact, encounter an end of volume for the input data file and the user has specified as concatenated another data volume in his JCL, the system will issue a mount command to the operator on the console for that particular volume. If there are no concatenated volumes left to process and the volume was of a direct access type, GEN4 will proceed to the next data file to be processed. However, if the volume were to be on magnetic tape, GEN4 will interrogate the

operator on the the console to determine whether there is another reel left to be processed for this particular data file. The operator may answer YES or NO. If the answer is YES, GEN4 will issue a mount command for that particular reel and processing will continue. If the answer is NO, the system will proceed to process the next data file. This next data file may be an old file or a summary file just previously created in this same run. This cycle of processing will continue until all input data files are processed.

In processing a subquery, the user may indicate, with the LIMIT operator, the maximum number of answers desired from a particular file and a particular subquery. This limit applies only to this subquery and will not affect the rest of the subqueries being processed against said data file. The LIMIT operator will not terminate processing as does the STOP operator but instead will bypass writing of answer records after the maximum has been written. Consequently, at the end of job the user will be provided with the number of answer records, up to the maximum indicated in the LIMIT operator, and also with the total number of records in the input data file and the count of how many answer records he would have had if no LIMIT operator was used. This is a simple way to obtain counts and to limit the output to a small number of answers, even zero.

The answers produced by GEN4 are of two kinds, intermediate answers and summary files. They are very different in nature and format. The intermediate answer records are of a MIDMS system format that is not used externally. Records from each file are reformatted to this MIDMS system format and sort keys are attached to the records. This makes it possible for the sort phase of the system to sort and merge answers from different files in a sequence desired by the user. Having them in such a sequence, the output processor will be able to produce reports from multiple data files in a sequence based on the values of the data fields of the records rather than files. All the data elements of the input record are available to the output subsystem, including free text type fields. The summary file type of records are of the same format as the input data file from which the records originate with the exception of the blocking factor if the user desires a different one. Each subquery within a query may ask for a summary file as output. Since a summary file created in a subquery may be the input to another subquery of the same query it is possible to create up to 60 summary files in a single query. Also, a summary file created in a subquery may be passed as input data to another subquery of a different query in the same run. Since there could be 60 queries batched into a single run, there could theoretically be 3600 work files or permanent files within a single run. This

number cannot be practically achieved with the present hardware since there is not enough core for the operating system to maintain the pointers required for so many files attached to a single job, but it can be said that the MIDMS system will handle as many input and/or output files as the particular operating system or a particular hardware is capable of supporting, not to exceed 3600 such data files. The summary file created in a job may be saved as output data files and/or work files for further processing within the same job. The same summary file created in a job may be used as input to several subqueries within the same job. The same volume that the summary file was created on may be used for another summary file after the first summary has been used as input to other subqueries. This allows the user to utilize limited external storage for larger volumes of data to be processed in the same run. The summary files created in GEN4 are also available as source direct input to GEN6. Since GEN6 is capable of producing a report from 58 different files, i.e., a line of print, a card, a record, some of the data items retrieved from a file can very easily be used as values to qualify records from another input data file. This is not a true interfile logic, but with some skill on the part of the user it very rapidly approaches the capabilities of interfile logic as presently known. The design of this system is oriented toward data base processing rather than file processing. Interfile logic has not been achieved in the present system due to the strict requirement of being able to operate within 128K bytes of core memory, including the operating system with all its functions. At the time the core restriction is eliminated the system will operate upon data bases rather than files and the interfile logic capability will be achieved with ease in such an environment. In fact the intermediate answer file created by GEN4 is the embryo of such a data base.

(c) Limitations. The limitations of the Retrieval Logic Processor are listed below:

1. The program does not process inter-file logic.

2. Only 500 retrieval source statements can be processed against any one file.

3. Only BCD sequential files are processed by GEN4.

4. There are four reserved words:

SPACE (S)
ZERO (S,ES)
SUBQUERY
SORTKEY

5. The sortkey is 85 characters in length.

(d) I/O Data Sets.

1. OPSTATE. This is the output work file and will be used as input by the next-called load module.

2. SUMMFILE. This file is used as output for a summary file creation. Otherwise it is not used.

3. CONFILE. This is an input work file that was the output work file in the previously called load module.

4. INFILE. This is the input data file used for reading variable length MIDMS data file records.

5. OUTFILE. This is the answer record output file.

(12) GEN4A.

(a) Abstract of Retrieval Non-Periodic Logic Processor.

1. Function. Same as GEN4, except that only non-periodic data is processed by GEN4A.

2. Calling Sequence. Same as GEN4, except that GEAG and GEAC are not used by GEN4A.

(b) Description. Same as GEN4.

(c) Limitations. Same as GEN4.

(d) I/O Data Sets. Same as GEN4.

(13) GEN5.

(a) Abstract of Data Sort.

1. Function. GEN5 is a program that does nothing but sort data. The data records which are sorted include the retrieved records that satisfied the user-specified logic requirements, the copies of the retrieval and output source statements, the error message records, and the constant pool strings. The data is sorted on the first 100 characters of the 412 character records. The first characters contain the control fields (query counter, type, subquery counter, and card counter). The last 85 characters of the sort key contain user-specified sort fields.

2. Calling Sequence. This routine calls no subroutines.

(b) Description. The sort statement is the first significant statement in the program. The input procedure INSRT SECTION reads the records to be sorted and releases them to the sort procedure. It also opens and closes the input file. The output procedure OUTSRT SECTION gets the sorted records and writes them to the output file. The standard COBOL sort/merge is used in GEN5, and the records can be sorted in either ascending or descending order.

(c) Limitations. This program uses the standard OS COBOL sort routine and all limitations that apply to the SORT package also apply to this program.

(d) I/O Data Sets.

1. CARD-OUT. This is the data record answer file created by the previous load module. It is used as input to the sort and subsequently as output from the sort.

2. SORT-REC. This is the record being sorted. The sort work files are variable in number and are as described in the JCL.

(14) GEN5A.

(a) Abstract of Work File Sort.

1. Function. GEN5A sorts the work file prior to output processing to accommodate multiple fly-sheets.

2. Calling Sequence. This routine calls one subroutine, GEAM.

| | Length | Type | |
|---|---|---|---|
| parm1 | HALFWORD | B | Base Address of Sending Area |
| parm2 | HALFWORD | B | Displacement of Sending Area |
| parm3 | HALFWORD | B | Base Address of Receiving Area |
| parm4 | HALFWORD | B | Displacement of Receiving Area |
| parm5 | HALFWORD | B | Length of String to be Moved |

(b) Lescription. The program GEN5A sorts and resequences the source and object statements in the work file and determines whether fly-sheet source statements are to be printed.

(c) Limitations. This program uses the standard OS COBOL sort routine and all limitations that apply to the SORT package also apply to this program.

(d) I/O Data Sets.

1. OPSTATE. This is the work file created by the previously called retrieval logic processor. It is used as input to the sort and subsequently as output from the sort.

2. SORT-REC. This is the record being sorted. The sort work files are variable in number and are as described in the JCL.

(15) GEN6.

(a) Abstract of Output Periodic Logic Processor.

1. Function. The Output Logic Processor processes retrieved data records using vector arrays and constant pool strings created in the Output Compiler. Queries that were flagged due to errors in either retrieval or output compilation are bypassed and corresponding vector arrays and constant pool strings are not processed. This program is the report generator for the system. The report formats are set up according to the user-specified requirements in the output source statements and are printed, punched, or written onto user-specified devices such as tape or disc.

2. Calling Sequence. No parameters are initially passed to the Output Logic Processor. The program calls six subroutines:

a. GEAM.

| | Length | Type | |
|---|---|---|---|
| parm1 | HALFWORD | B | Base Address of Sending Area |
| parm2 | HALFWORD | B | Displacement of Sending Area |
| parm3 | HALFWORD | B | Base Address of Receiving Area |
| parm4 | HALFWORD | B | Displacement of Receiving Area |
| parm5 | HALFWORD | B | Length of String to be Moved |

b. GEAZ.

| | Length | Type | |
|---|---|---|---|
| N | HALFWORD | B | Subscript of Active Statement |
| RX | 9999 | A | Data Record |
| MATRIX | Variable | Variable | Matrix Area, If Used |
| STATE-FORM | 12 HALFWORDS | B | Object Statement |

c. GEAT.

| | Length | Type | |
|---|---|---|---|
| parm1 | 29117 | A | Base Address of Receiving Area |
| parm2 | FULLWORD | B | Displacement of Receiving Area |
| parm3 | FULLWORD | B | Size of Receiving Area |
| STATE-FORM | 12 FULLWORDS | B | Object Statement |
| RX | 9999 | A | Data Record |

d. GEAL.

| | Length | Type | |
|---|---|---|---|
| module name | 8 | A | Name of Called Routine |
| SAVEREG | 18 FULLWORDS | B | Save Area for Registers |
| parameter(s) | Variable | Variable | Parameter(s) Being Passed to Called Routine but not Used by GEAL |

e. GEAD.

| | Length | Type | |
|---|---|---|---|
| SNAME | 8 | A | Subroutine Name |
| SAVEREG | 18 FULLWORDS | B | Save Area for Registers |
| AREALINKED IN-LENGTH | FULLWORD | B | Load/Delete Switch |

f. GEAP.

| | Length | Type | |
|---|---|---|---|
| R | 1 | A | Address of Record |
| S3 | HALFWORD | B | Number of Subsets |
| STATE | 24 | A | Object Statement |
| IX | HALFWORD | B | Number of Fields to Sort |
| S2 | HALFWORD | B | Length of Subset |

(b) Description. The output logic processor performs the report generating function of a query. The inputs used by this module are: (1) compiled object vectors (TYPE 6 and 8 records), (2) constant pool strings (TYPE 7 records), (3) a single error vector indicating what queries had errors during processing of the retrieval and output compiler (TYPE 4 records), (4) retrieval and output source statements (TYPE 1 and 5 records), (5) data records that were retrieved during execution of the retrieval logic processor (TYPE 9 record), and (6) data records, although not retrieved during execution of the retrieval logic processor, but made available through use of the SOURCE DIRECT option specified in the report section of a query.

Processing starts with general housekeeping, opening as input both Opstate File (consisting of TYPE 1, 2, 3, 4, 5, 6, 7, and 8 records) and Answer File (consisting of TYPE 1, 5, 8, and 9 records), opening as output Report File (which is generally assigned to a printer) and the printing of a header page containing the message, MIDMS START PROCESSING.

The following narrative is performed on each report that is processed by this module.

The Opstate File is read until the first TYPE 5 record for a query is found. Next, the Answer File is read until the first source statement of a query is found (TYPE 1 or 5 record). At this point a header page for the report is printed containing the first source statement in a query. Reading of the Answer File continues during which time all TYPE 1 and 5 records are printed providing the user with a listing of both his retrieval and source statements. This continues until a TYPE 8 or 9 record is encountered having a query number equivalent to the query number of the last record read from the Opstate File or a record is found containing a different query number.

If the latter occurs this indicates that an error occurred during compilation of the source statements; no data is available for processing; the logic flow goes back to processing the next report in the input job stream. Note that the error vector

(TYPE 4 record) is used primarily as an end of file indicator for the Stateop File and is not used to test each compiled report statement to determine if it comes from a query containing an error as was done in the Retrieval Logic Processor.

Once the first data record available for processing (TYPE 9 record) or a record indicating that source direct option must be performed (TYPE 8 record) is found in the Answer File, control goes back to the reading of the Opstate File. As the Opstate File is being read, TYPES 1, 2, 3, 4, and 5 records are bypassed for the query being processed. When a TYPE 6 record is encountered it is moved into a working storage area, STATE. The working storage area can hold up to 1000 compiled report statements. Also, all TYPE 7 records which are encountered are moved into another working storage area, CON-IN. The latter working storage area can hold up to 9000 characters of constants, literals, and defined values. During reading of the TYPE 6 record the following occurs: (1) pointers are set indicating where in the holding area (STATE) the REPORT Section, TRAILER Section, and FINAL REPORT Section begin, (2) pointers are set to determine where in the holding area, FILEX, the file name associated with each FILE Section, and (4) stores statement numbers to later determine the length of each FILE Section. Since FILEX can hold up to 58 file names, a report may contain up to 58 unique FILE Sections.

Reading of the Opstate File continues until a record is read with a different query number which indicates that the last TYPE 6 or 7 record for the report being processed has been read and control now goes to the link-edit portion of the CTL-RET Section, LINK-EDIT through LOOP-E.

Within LINK-EDIT a test is being performed to determine if any output is required to be written. If so, the output file, Fileout, is opened with the DCB parameters modified to reflect information found in both the format report source statements and the JCL parameters supplied by the user. During LINK-EDIT, the TYPE 6 record are processed. The true pointer is used to indicate what statement should be executed if the logic found in the current statement is satisfied. The false pointer serves the same function but is used when the logic found in the statement being processed is not met.

Upon completion of the LINK-EDIT, control goes to STEP2 Section. It is from this point in the program that the TYPE 6 statements in State are executed and data from the Answer File, if retrieval occurred, or data from another file of the SOURCE DIRECT option was specified, is made available for processing.

In processing the data records, the program logic determines functionally what operation should be performed next. At the beginning of output the HEADER Section of the report, if provided, is executed. The HEADER pointers previously created direct the program to execute those TYPE 6 records in the holding area that comprise the HEADER Section of the report. Upon completion of this HEADER Section, pointers are reset to start processing these TYPE 6 records that pertain to the appropriate FILE Section for which data records are being processed. These latter statements are executed in sequence until an end of data file is encountered or the number of body lines printed exceeds the page size parameter for printed output.

When the page size is exceeded, pointers are reset to that portion of the holding area (State) which contains those TYPE 6 records comprising the TRAILER Section. Upon completion of the TRAILER Section, pointers are reset to execute those TYPE 6 records comprising the HEADER Section. When the latter is completed, a check is made to see if any label-lines are required, i.e., output was specified in the LINE paragraph of the appropriate FILE Section. (The first time the TYPE 6 records, comprising the LINE paragraph of a FILE Section are executed, the generated output from these statements is also saved in another holding area, LAB-LINES. LAB-LINES provides space for 6 body lines of print output.) If LABEL-LINES are available and required, they are written out and the pointers are reset to execute the TYPE 6 records comprising the BEGINS Section of the report. If periodic data is processed in any one of the TYPE 6 records comprising the BEGINS Section, pointers are reset to access the next subset of data to be processed and those TYPE 6 records comprising the BEGINS Section are executed again. Output continues until all data in subsets requested are exhausted.

When executing the BEGINS Section of the report, should the number of body lines exceed the pagesize parameter, pointers are reset to execute the TRAILER and HEADER Sections, respectively. Any required label lines are printed and pointers are set back to continue executing those statements just prior to when the pagesize was exceeded. When the processing of data for a report is completed, pointers are set to execute the TYPE 6 records that comprise the FINAL-REPORT Section. At the end of each individual report, a trailer page is written which consists of the same data as appeared in the header page of the report.

Execution of the TYPE 6 records comprising the various File Sections continues with the execution of headers, trailers, and label-lines when appropriate until the end of the data file is encountered when using the source direct option and when the last retrieved data record from the Answer File corresponding to the current report query no. has been processed, or when a TYPE 6 record with an Operation Code of 28 is executed.

Should any of the latter conditions occur, processing of the current data file is stopped. Immediately following this, the TRAILER and FINAL-OUTPUT Sections of the report are executed, a trailer page is printed, intermediate housekeeping is performed which closes the Output File or the Source Direct File if these options were used, and the logic flow of the program returns to start processing the next report. The termination of a report can occur before the end of file condition occurs in any of the input files if the user utilizes the STOP statement in his report.

(c) Limitations. The program does not process implicit output convert routines. All convert routines used for output must be executed by a CMOVE or DEFINE statement.

(d) I/O Data Sets.

1. MATRIS. This is a dummy file used to acquire core dynamically for MATRIX processing. It is only used when a MATRIX is called for.

2. GEWORK3. This is a dummy file used to acquire space dynamically for loading the output object vectors.

3. FILEDIR. This file is used as the SOURCE DIRECT input data file when required.

4. OPSTATE. This is the input work file created by a previous load module.

5. ANSWERIN. This is the input data record answer file created by a retrieval logic processor.

6. REPORTOP. This is the printer output file used for writing the reports.

7. PUNCHOP. This is the output punch file, used as required for punching decks.

8. FILEOUT. This file is used as required for writing an output file with variable length records onto tape or disk.

9. FILEOUTF. This file is used as required for writing an output file with fixed length records onto tape or disk.

(16) GEN6A.

(a) Abstract of Output Non-Periodic Logic Processor.

1. Function. Same as GEN6, except that only non-periodic data is processed by GEN6A.

2. Calling Sequence. Same as GEN6, except that GEAP is not used by GEN6A.

(b) Description. Same as GEN6.

(c) Limitations. Same as GEN6.

(d) I/O Data Sets. Same as GEN6.

(17) GEAA.

(a) Current System Date.

1. Function. This ALC subroutine extracts from the Julian date the current system date and returns the result to the calling program.

2. Calling Sequence. No data is passed and no programs are called but does return via the call the data described below:

| Length | Type | Description |
|---|---|---|
| 2 | N | Day of the Month |
| 3 | A | Abrr. Month (e.g., Aug) |
| 2 | N | Current System Year |

(b) Description. This subroutine is executed only when a CLASS statement is encountered in the GEN3A compiler. In GEN3A the CLASS statement is mandatory. Omission of the IS card causes a fatal error.

(c) Limitations. None.

(d) I/O Data Sets. None.

(18) GEAB.

(a) Abstract of Build.

1. Function. The ALC subroutine GEAB determines whether or not a user-specified subroutine is available to the system.

2. Calling Sequence. GEAB calls no programs. The data passed and returned is described below:

| | Length | Type | |
|---|---|---|---|
| SNAME | 8 | A | Name of Routine Being Checked |
| RET-CODE | HALFWORD | B | Return Code from Routine |

(b) Description. The program searches the system load module library indexes to confirm the existence of a load module of the name requested prior to actually calling it. If a non-existent load module was called, MIDMS would ABEND.

(c) Limitations. GEAB only determines the existence of a named load module; it does not determine whether the load module is executable.

(d) I/O Data Sets. None.

(19) GEAC.

(a) Abstract of Control.

1. Function. The subroutine GEAC decodes periodic set control words and checks to see if the file has any periodic sets.

2. Calling Sequence. GEAC calls no programs. The data passed and returned is described below:

| | Length | Type | |
|---|---|---|---|
| RX | 9999 | A | Data Record |
| SET-W-TBL | 12 | A | Occurs 50 Times - Contains Description of Each Set in Record |
| SL | HALFWORD | B | Subscript of Last Used Occurrence in SET-W-TBL |

(b) Description. GEAC computes the address of the first periodic set control word and checks to see if the file has any periodic sets. If it does not, it returns control back to the calling program, GEN4. If it does contain periodic information, it processes the control word and goes on to compute the address of the next Periodic Set Control Word. The program is executed once per data record.

(c) Limitations. None.

(d) I/O Data Sets. None.

(20) GEAD.

(a) Abstract of Subroutine Loader.

1. Function. The subroutine GEAD loads subroutines into core and considers them reusable. It also deletes subroutines when they are no longer needed.

2. Calling Sequence. The data passed and returned is described below:

| | Length | Type | |
|---|---|---|---|
| SNAME | 8 | A | Subroutine Name |
| SAVEREG | 18 FULLWORDS | B | Save Area for Registers |
| AREALINKED IN-LENGTH | FULLWORD | B | Load/Delete Switch |

(b) Description. GEAD is intended for use under operating system MFT, as a normal call to a subroutine under that system would cause the subroutine to be loaded each time it is executed. This routine causes the called subroutine to be loaded only once and subsequently to be reused in core. MVT provides this service.

(c) Limitations. The called subroutine must be reusable.

(d) I/O Data Sets. None.

(21) GEAG.

(a) Abstract of Flag.

1. Function. The ALC subroutine GEAG flags the periodic subsets which satisfy the retrieval logic and search mode specified by the user in the Retrieval Source Language.

2. Calling Sequence. GEAG calls no programs. The data passed and returned is described below:

| | Length | Type | |
|---|---|---|---|
| RX | 9999 | A | Data Record |
| SET-TABLE | 12 | A | Occurs 50 Times - Contains Description of Each Set in Record |
| SL | HALFWORD | B | Subscript of Last Used Occurrence in SET-TABLE |
| ABM | HALFWORD | B | Number of Flags in Buffer |
| FLAG-AREA | | | Address of Flag |
| OP | HALFWORD | B | Operation Code for Current Statement |
| SWTGO | HALFWORD | B | Return Code |

(b) Description. GEAG processes the periodic sets of the input data. The programs check the type of search mode being used and then check each part of the periodic set against the user logic to determine if flags should or should not be set. The program is called after each execution of a subquery for each subset.

(c) Limitations. None.

(d) I/O Data Sets. None.

(22) GEAL.

(a) Abstract of Link.

1. Function. The ALC subroutine GEAL provides the link between the called program, its resulting output, and the calling COBOL program.

2. Calling Sequence. This subroutine calls a subprogram depending on the SNAME parameter which is passed to the routine by the calling program. The data passed and returned is described below:

| | Length | Type | |
|---|---|---|---|
| module name | 8 | A | Name of Called Routine |
| SAVEREG | 18 FULLWORDS | B | Save Area for Registers |
| parameter(s) | Variable | Variable | Parameter(s) Being Passed to Called Routine but not Used by GEAL |

(b) Description. GEAL links the MIDMS main control program GENO to each of the COBOL routines. GENO passes the name of the routine being called to MARINL and the ALC routine provides the actual link to the called program. This routine is also used to establish linkage with user-specified special operators and convert routines through the standard calling sequence.

(c) Limitations. The limitations of GEAL are listed below:

1. The called program name cannot exceed 8 characters.

2. A maximum of 360 characters of output can be received from the called program.

(d) I/O Data Sets. None.

(23) GEAM.

(a) Abstract of MOVE.

1. Function. The ALC subroutine GEAM moves a logical string of characters from one area to another.

2. Calling Sequence. GEAM calls no programs. There are five parameters passed.

| | Length | Type | |
|---|---|---|---|
| parm1 | HALFWORD | B | Base Address of Sending Area |
| parm2 | HALFWORD | B | Displacement of Sending Area |
| parm3 | HALFWORD | B | Base Address of Receiving Area |
| parm4 | HALFWORD | B | Displacement of Receiving Area |
| parm5 | HALFWORD | B | Length of String to be Moved |

(b) Description. GEAM starts with the address generated by the first base address and displacement passed and moves it to the address generated by the record base and displacement passed. It continues moving until the passed length parameter has been satisfied.

(c) Limitations. A maximum of 32768 bytes of core storage can be moved.

(d) I/O Data Sets. None.

(24) GEAN.

(a) Abstract of Random File DCB Override.

1. Function. GEAN moves a ddname parameter into the random file DCB.

2. Calling Sequence.

| | Length | Type | |
|---|---|---|---|
| DDname | 8 | A/N | Name to be Moved |
| Filename | HALFWORD | B | File Name, DEC Address |

(b) Description. GEAN moves a name to the DCB of a random file. COBOL passes the address of the DEC when the filename of a random file is used. The DCB address is one of the parameters of the DEC (offset 8 bytes).

(c) Limitations. None.

(d) I/O Data Sets. None.

(25) GEAP.

(a) Abstract of Periodic Sort.

1. Function. The subroutine GEAP sorts the subsets of a periodic set within a single record according to the sort key identified in the calling sequence.

2. Calling Sequence. The data passed is described bel w:

| | Length | Type | |
|---|---|---|---|
| R | 1 | A | Address of Record |
| S3 | HALFWORD | B | Number of Subsets |
| STATE | 24 | A | Object Statement |
| IX | HALFWORD | B | Number of Fields to Sort |
| S2 | HALFWORD | B | Length of Subset |

(b) Description. GEAP sorts in either ascending or descending sequence. No data is returned, as the periodic set remains in its original location in the record.

(c) Limitations. None

(d) I/O Data Sets. None

(26) GEAS.

(a) Abstract of Sort Key Builder.

1. Function. The ALC subroutine GEAS builds the sort key specified by the user in the retrieval source language.

2. Calling Sequence. GEAS calls no programs. The data passed and returned is described below:

| | Length | Type | |
|---|---|---|---|
| RX | 9999 | A | Data Record |
| CONX | 9000 | A | Constant Pool |
| STATE-FORM | 12 HALFWORDS | B | Object Statement |
| KSRT | 85 | A | Sort Key |

(b) Description. GEAS builds the sort key for the data that has met the search requirements. The actual sort takes place in GEN5. The program gets the sort fields from either the record buffer pool or the constant pool. If the descending SORT/MERGE Option is specified, the data is inverted by binary translation. The sort field is built according to user specifications and is returned in the 85-character KSRT parameter.

(c) Limitations. The maximum length of the user sort field is 85 characters. An inverted sort key is not printed.

(d) I/O Data Sets. None.

(27) GEAT.

(a) Abstract of Table-Loader.

1. Function. Attempts to find unused core storage areas in which to place module tables.

2. Calling Sequence. GEAT calls no programs. The data passed and returned is described below:

| | Length | Type | |
|---|---|---|---|
| parm1 | 29117 | A | Base Address of Receiving Area |
| parm2 | FULLWORD | B | Displacement of Receiving Area |
| parm3 | FULLWORD | B | Size of Receiving Area |
| STATE-FORM | 12 FULLWORDS | B | Object Statement |
| RX | 9999 | A | Data Record |

(b) Description. GEAT attempts to find unused core storage. It checks the partition of core where the program is located and the subpartitions such as the statement area, the special operators area, and the constant pool. If it finds enough space, it loads in a load module table and passes the starting address back to the calling program.

(c) Limitations. It does not chain small parts of core together. It only loads a table if it finds enough contiguous unused core.

(d) I/O Data Sets. None.

(28) GEAX.

(a) Abstract of Compare.

1. Function. The ALC subroutine GEAX compares two alphanumeric or numeric fields and passes the results of the compare back to the calling program.

2. Calling Sequence. GEAX calls no programs. The data passed and returned is described below:

| | Length | Type | |
|---|---|---|---|
| RX | 9999 | A | Data Record |
| CONX | 9000 | A | Constant Pool |
| STATE-FORM | 12 HALFWORDS | B | Object Statement |
| N | HALFWORD | B | Subscript of Active Statement |

(b) Description. The program determines from the passed parameters the type of fields being compared (alphanumeric or numeric). It also determines whether the a- and b-fields are in the record buffer pool or the constant pool. It then compares the two fields and sets a switch to show if the a-field is greater then, equal to, or less than the b-field.

(c) Limitations. Alphanumeric fields cannot exceed 360 characters in length, and numeric fields cannot exceed 15 digits.

(d) I/O Data Sets. None.

(29) GEAZ.

(a) Abstract of Edit.

1. Functions. The ALC subroutine GEAZ is called by the Output Logic Processor (GEN6) and is used to individually execute most of the logical statement op-codes.

2. Calling Sequence. GEAZ calls no programs. The data passed and returned is described below:

| | Length | Type | |
|---|---|---|---|
| N | HALFWORD | B | Subscript of Active Statement |
| RX | 9999 | A | Data Record |
| MATRIX | Variable | Variable | Matrix Area, If Used |
| STATE-FORM | 12 HALFWORDS | B | Object Statement |

(b) Description. GEAZ is called by the Output Logic Processor to execute most logical op-codes, including arithmetic statements, TMOVE and EMOVE. The program does not execute any I/O statements. It does not duplicate the function of any other routine in GEN6, but it processes all remaining op-codes.

(c) Limitations. Numeric fields cannot exceed 15 characters.

(d) I/O Data Sets. None.

c. Module Error Messages.

ABNORMAL COMPLETION OF SORT IN GEN4X2 RETURN CODE XX

GEN4X2

XX = return code of sort utility. User action - rerun job; if error re-occurs submit a discrepancy report to the MIDMS support group.

AFIELD AND SORT KEY SIZE INCONSISTENT

GEN2

The length of the sort key space a-field is moved to must be equal to the length of the field being moved.

A-NAME AND B-NAME LENGTHS DIFFER

GEN2

Alphanumeric a-field and b-fields must be of the same length.

A-NAME AND B-NAME LENGTHS DIFFER

GEN3

Alphanumeric a-field and b-fields must be of the same length.

A-NAME CONVERSION SUPPRESSED

GEN3

The convert routine associated with an FFT field name was suppressed utilizing the # sign around a literal b-field. This is a caution.

A-NAME EXCEEDS 15 DIGITS

GEN2

A-field is numeric and cannot exceed 15 digits.

A-NAME EXCEEDS 15 DIGITS

GEN3

A-field is numeric and cannot exceed 15 digits.

A-NAME EXCEEDS 15 DIGITS

GEN3A

A-field is numeric and cannot exceed 15 digits.

A-NAME EXCEEDS 360 CHARACTERS

GEN3

The a-field data name exceeded 360 characters. Use partials if data must be extracted from data fields that physically exceed 360 characters.

A-NAME HAS AN IMPLICIT CONVERT ROUTINE

GEN2

There is an FFT designated convert routine associated with this data field.

A-NAME HAS AN IMPLICIT CONVERT ROUTINE

GEN3

There is an FFT designated convert routine associated with this data field.

A-NAMF HAS AN IMPLICIT CONVERT ROUTINE

GEN3A

Advisory message. The file format table will cause conversion to take place unless overridden.

A-NAME NOT NUMERIC

GEN3

A-field is not numeric as required by the statement format corresponding to the first word.

A-NAME NOT PERIODIC FIELD

GEN2

The statement format corresponding to the first word requires a periodic a-field.

A-NAME NOT PERIODIC FIELD

GEN3

The statement format corresponding to the first word requires a periodic a-field.

A-NAME TOO LONG

GEN2

FFT field names cannot exceed 5 characters and defined names cannot exceed 8 characters.

A-NAME TOO LONG

GEN3

FFT field names cannot exceed 5 characters and defined names cannot exceed 8 characters.

A-NAME TOO LONG

GEN3A

FFT field names cannot exceed five characters and defined names cannot exceed 8 characters.

A-NAME UNDEFINED

GEN2

Encountered an a-field that is neither a valid 5-character mnemonic associated with an FFT field for a previous subquery nor is equivalent to a previously encountered defined name.

A-NAME UNDEFINED

GEN3

Encountered an a-field that is neither a valid 5-character mnemonic associated with an FFT field for a previous File Section nor is equivalent to a previously encountered defined name.

A-NAME UNDEFINED

GEN3A

Encountered an a-field that is neither a valid 5-character mnemonic associated with an FFT field for a previous File Section nor is equivalent to a defined name.

A PROGRAM MUST APPEAR WITHIN A SECTION

GEN3

The PROGRAM statement must be within a FILE SECTION after all LINEs and BEGINs.

BLOCK IN MIDMS LIB NOT A MULTIPLE OF 80

GEN1

The size of a card-image source statement data block on the Library must be a multiple of 80. Resubmit. If error persists, submit a discrepancy report to the MIDMS support group.

B-NAME CONTAINS NONNUMERIC DATA

GEN2

The b-field contains an alphabetic or special character.

B-NAME EXCEEDS 15 DIGITS

GEN2

B-field is numeric and should not exceed 15 digits.

B-NAME EXCEEDS 15 DIGITS

GEN3

B-field is numeric and should not exceed 15 digits.

B-NAME EXCEEDS 15 DIGITS

GEN3A

B-field is numeric and should not exceed 15 digits.

B-NAME MISSING

GEN2

The statement format as defined by the first word requires the use of a b-field. See the Users Manual for proper statement formats.

B-NAME MISSING

GEN3

The statement format as defined by the first word requires the use of a b-field. See the Users Manual for proper statement formats.

B-NAME MISSING

GEN3A

The statement format as defined by the first word requires the use of a b-field. See the Users Manual for proper statement formats.

B-NAME NOT NUMERIC

GEN3

A numeric a-field requires a numeric b-field.

B-NAME TOO LARGE

GEN2

Alphanumeric a-field and b-fields must have the same lengths.

B-NAME TOO LARGE

GEN3

Alphanumeric a-field and b-fields must have the same lengths.

B-NAME TOO LARGE

GEN3A

Alphanumeric a-field and b-fields must have the same lengths.

CAN APPEAR ONLY IN FILE, LINE OR BEGIN PARAGRAPH

GEN3

This statement may only be used in the file oriented paragraphs, not in HEADERS or TRAILERS.

CAN APPEAR ONLY IN FILE, LINE OR BEGIN PARAGRAPH

GEN3A

This statement may only be used in the file oriented paragraphs, not in HEADERS or TRAILERS.

****** CAUTION, NO SUMMARY FILE WILL BE CREATED ******

GEN2X

A summary card was not included in the job, only a count will be made.

COMMA OR BLANK MISSING

GEN2

A required comma or blank was not found.

COMMA OR BLANK MISSING

GEN3

A required comma or blank was not found.

COMMA OR BLANK MISSING

GEN3A

A required comma or blank was not found.

COMPARE WORD MISSING OR INVALID

GEN2

The compare operator was either misspelled or is missing.

COMPARE WORD MISSING OR INVALID

GEN3

The compare operator was either misspelled or is missing.

COMPARE WORD MISSING OR INVALID

GEN3A

The compare operator was either misspelled or is missing.

*COMP-OP AND LIBRARY NAME DO NOT MATCH

GEN1

The names on the COMP-OP card and on the library card are not the same.

COMP-OP CARD MISSING - FATAL ERROR

GEN3B

COMP-OP card must be the first card in a COMP-OP run.

*COMP-OP NAME DOES NOT END IN -R-*

GEN1

The fifth character of the name for the COMP-OP card must be an "R."

CONFLICTING DATA TYPES

GEN2

A-field and b-field must both be either alphanumeric or numeric.

CONFLICTING DATA TYPES

GEN3

A-field and b-field must both be either alphanumeric or numeric.

CONFLICTING DATA TYPES

GEN3A

A-field and b-field must both be either alphanumeric or numeric.

CONSTANT A-NAME NOT ALLOWED

GEN3

Constants cannot be used as a-fields in this statement format.

CONSTANT A-NAME NOT ALLOWED

GEN3A

Constants cannot be used as a-fields in this statement format.

CONSTANT B-NAME NOT ALLOWED

GEN3

The constant in the b-field cannot be used in the statement format corresponding to the first word.

CONSTANT EXCEEDS A-NAME

GEN2

The length of the constant is greater than the length of the a-field data name. Alphanumeric a-field and b-fields must have the same lengths.

CONSTANT EXCEEDS A-NAME

GEN3

The length of the constant is greater than the length of the a-field data name. Alphanumeric a-field and b-fields must have the same lengths.

CONSTANT EXCEEDS A-NAME

GEN3A

The length of the constant is greater than the length of the a-field data name. Alphanumeric a-field and b-fields must have the same lengths.

CONSTANT POOL OVERFLOW OCCURRED IN THIS QUERY

GEN2

The length of all characters comprising defined values, constants, and literals exceeds 9,000 characters.

CONSTANT POOL OVERFLOW OCCURRED IN THIS REPORT

GEN3

The length of all characters comprising defined values, constants, and literals exceeds 9,000 characters.

CONSTANT POOL OVERFLOW OCCURRED IN THIS REPORT

GEN3A

The length of all characters comprising defined values, constants, and literals exceeds 9,000 characters.

CONVERSION NOT ALLOWED

GEN3

Improper use of the conversion routine option. See the Users Manual.

CONVERSION NOT ALLOWED

GEN3A

Convert routine/special operator not permitted with this statement type.

CONVERT OUTPUT EXCEEDS LINESIZE

GEN3A

Output from convert routine exceeds line size specified in SHL report - truncation will occur.

CONVERT ROUTINE MISSING

GEN3

A required convert routine enclosed in asterisks was not found in the statement.

CONVERT ROUTINE NOT ALLOWED

GEN3

The use of a convert routine is not permitted in the statement format corresponding to the first word.

CONVERT ROUTINE RETURNS LENGTH EXCEEDING 360 CHARACTER

GEN2

A convert routine may not return more than 360 characters of data in the OUT-DATA field. The OUT-LENGTH parameter exceeded this limit.

CONVERT ROUTINE RETURNS LENGTH EXCEEDING 360 CHARACTER

GEN3

A convert routine may not return more than 360 characters of data in the OUT-DATA field. The OUT-LENGTH parameter exceeded this limit.

CONVERT ROUTINE RETURNS LENGTH EXCEEDING 360 CHARACTERS

GEN3A

A convert routine may not return more than 360 characters of data in the OUT-DATA field. The OUT-LENGTH parameter exceeded this limit.

CONVERT ROUTINES EXCEEDED

GEN3A

A maximum of 25 convert routines are permitted.

DATA FOUND AFTER PARENTHESES

GEN2

Right parentheses used in parenthetical expressions must be the last entry on a source statement.

DATA FOUND AFTER PERIOD

GEN3

A period, when used, must be the last character on a source statement card.

DATA FOUND AFTER PERIOD

GEN3A

A period, when used, must be the last character on a source statement card.

DATA IN S/M STATEMENTS NOT FROM THE SAME SET

GEN2

Periodic fields being sorted or merged should be from the same periodic set.

DATA NOT NUMERIC

GEN2

An alphabetic or special character was found in a numeric field.

DATA NOT NUMERIC

GEN3

An alphabetic or special character was found in a numeric field.

DATA NOT NUMERIC

GEN3A

An alphabetic or special character was found in a numeric field.

DATA NOT PROPERLY STORED ON LIBRARY

GEN1

An improper card type has been retrieved from the library.

DEFINE LENGTH EXCEEDED

GEN2

The length parameter in the DEFINE statement is not consistent with the length of the constant or literal following it.

DEFINE LENGTH EXCEEDED

GEN3

The length parameter in the DEFINE statement is not consistent with the length of the constant or literal following it.

DEFINE LENGTH EXCEEDED

GEN3A

The length parameter in the DEFINE statement is not consistent with the length of the constant or literal following it.

DEFINE NAME ILLEGAL

GEN3A

Define names may not be referenced in a COUNT or SUM statement.

DEFINE OR FFT NAMES NOT ALLOWED

GEN2

A defined or FFT name is not allowed in this statement.

DEFINE OR FFT NAMES NOT ALLOWED

GEN3

Notify the MIDMS support group.

DEFINED MATRIX EXCEEDS 32K

GEN3

The product of the element length, number of rows, and number of columns in the defined matrix may not exceed 32,000.

DEFINED NAME REQUIRED

GEN2

A defined anme, up to 8 characters in length, is missing in the DEFINE statement.

DEFINED NAME REQUIRED

GEN3

A defined name, up to 8 characters in length, is missing in the DEFINE statement.

DESCENDING MUST PRECEDE A-NAME

GEN2

The descending parameter must precede the a-field.

DIFFERENT DATA TYPE

GEN2

This is a caution message. A defined name's data type (ALPHANUMERIC/NUMERIC) was changed by utilizing the Modify Subquery Option.

DIFFERENT DATA TYPE

GEN3

This is a caution message. A defined name's data type (ALPHANUMERIC/NUMERIC) was changed by utilizing the Modify Report Option.

DIFFERENT DATA TYPE

GEN3A

This is a caution message. A defined name's data type (ALPHANUMERIC/NUMERIC) was changed by utilizing the Modify Report Option.

DUPLICATE DEFINE STATEMENT

GEN2

A duplicate defined name was encountered.

DUPLICATE DEFINE STATEMENT

GEN3

A duplicate defined name was encountered.

DUPLICATE DEFINE STATEMENT

GEN3A

A duplicate defined name was encountered.

DUPLICATE FILE SECTIONS

GEN3

Duplicate file sections were submitted.

DUPLICATE FILE SECTIONS

GEN3A

Duplicate file sections were submitted.

DUPLICATE PROGRAM NAME

GEN3

Each program must have a unique number.

END OF PAGE MARGIN CANNOT EXCEED 9 LINES

GEN3

EJECT ON n statement does not allow the value of n to be greater than 9.

ERROR CODE RETURNED FROM CONVERT ROUTINE _____

GEN2

An error was encountered while processing a user convert routine specified in the error message. If this occurred in other than a DEFINE statement check to see what convert routine is associated with the FFT field name used as the a-field entry.

ERROR CODE RETURNED FROM CONVERT ROUTINE _____

GEN3

An error was encountered while processing a user convert routine specified in the error message. If this occurred in other than a DEFINE statement check to see what convert routine is associated with the FFT field name used as the a-field entry.

ERROR CODE RETURNED FROM CONVERT ROUTINE _____

GEN3A

An error was encountered while processing a user convert routine specified in the error message. If this occurred in other than a DEFINE statement check to see what convert routine is associated with the FFT field name used as the a-field entry.

****** ERROR ENCOUNTERED WRITING (name) TO LIBRARY CONTENTS INVALID - RELOAD REQUIRED ******

GEN3B

An error was encountered while trying to write the named item to the library. Recheck set-up and rerun. If error persists, restore library from backup tape before proceeding.

*** ERROR *** MORE THAN ONE SUBQUERYX

GEN1

Non-sequential subqueries may not be batched.

EXPECTED QUERY CARD MISSING

GEN1

A QUERY card is expected to be the first card of each user's job, regardless of whether the remainder of that job is on the library or SOURCE DIRECT is used.

FATAL ERROR, BFIELD CANNOT REFER TO AN FFT ITEM

GEN2X

B-fields cannot refer to FFT item - remove source statement.

FATAL ERROR GEN4X1, TOO MANY OBJECT VECTORS

GEN4X1

User action - submit a discrepancy report to the MIDMS support group.

FATAL ERROR IN GEN4X1, ILLEGAL OPCODE OF XX

GEN4X1

XX = opcode processed by GEN4X1. User action - submit a discrepancy report to the MIDMS support group.

FATAL ERROR IN GEN4X3 AAAAAAAA AND BBBBBBBB FILE DATA

GEN4X3

Data in either file AAAAAAAA or BBBBBBBB is inaccurate or erroneous; please check file content.

FATAL ERROR, MORE THAN 100 GENERATED STATEMENTS

GEN2X

User's retrieval expression contains too many statements - reduce source statements.

FATAL ERROR, ONLY ONE SUBQUERYX CARD PER JOB IS PERMITTED

GEN2X

At least two non-sequential subqueries were processed; only one is allowed.

FATAL ERROR, ONLY ONE SET AND FIXED SET ARE PERMITTED

GEN2X

User has referred to fields located in different sets, only one set and fixed set can be used in a subqueryx - alter request.

FATAL ERROR, OVERFLOW GEN4X1 CONSTANT POOL

GEN4X1

User action - submit a discrepancy report to the MIDMS support group.

FATAL ERROR, PARENTHESIS ARE NOT ALLOWED

GEN2X

Parentheses were encountered in source statements - remove parentheses.

FFT AND DEFINE ARE THE SAME

GEN3

The DEFINE statement contains a mnemonic identical to an FFT field-name. Change mnemonic name in DEFINE statement.

FFT AND DEFINE ARE THE SAME

GEN3A

The DEFINE statement contains a mnemonic identical to an FFT field-name. Change mnemonic name in DEFINE statement.

FFT LR7 MISSING

GEN2

FFT Logical Record 7 could not be found for the file name found in the current File Section.

FFT LR8 MISSING

GEN2

FFT Logical Record 8 could not be found for the file name found in the current File Section.

FFT LR9 MISSING

GEN2

FFT Logical Record 9 could not be found for the file name found in the current File Section.

FFT NAMES NOT ALLOWED

GEN3A

The DEFINE statement cannot contain FFT names.

FFT TABLE 2 MISSING

GEN2

FFT Logical Record 2 could not be found for the file name specified in the current subquery.

FFT TABLE 2 MISSING

GEN3

FFT Logical Record 2 could not be found for the file name specified in the current File Section.

FFT TABLE 2 MISSING

GEN3A

FFT Logical Record 2 could not be found for the file name specified in the current File Section.

FFT TABLE 3 MISSING

GEN2

FFT Logical Record 3 could not be found for the file name specified in the current subquery.

FFT TABLE 3 MISSING

GEN3

FFT Logical Record 3 could not be found for the file name specified in the current File Section.

FFT TABLE 3 MISSING

GEN3A

FFT Logical Record 3 could not be found for the file name specified in the current File Section.

FIELD NAME IN FILE REQUIRED

GEN2

A field name from the data file is required.

FIELD NAME IN FILE REQUIRED

GEN3

An FFT field name is required.

FIFTH CHARACTER NOT-A

GEN2

The file name specified is incorrect.

FILE NAME EXCEEDS 8 CHARACTERS

GEN3

A file name can vary in length from 5 to 8 characters, inclusive.

FILE NAME EXCEEDS 8 CHARACTERS

GEN3A

A file name can vary in length from 5 to 8 characters, inclusive.

FILE NAME NOT AT LEAST FIVE CHARACTERS

GEN3

The file name specified cannot be less than 5 characters.

FILE NAME NOT AT LEAST FIVE CHARACTERS

GEN3A

The file name specified cannot be less than five characters.

FILE NAME NOT VALID

GEN2

File names must be at least 5 characters long but cannot exceed 8 characters. The fifth character must be an "A".

FILE SECTIONS EXCEED 60

GEN3

Up to 60 unique file sections may be used in any report.

FILE SECTIONS EXCEED 60

GEN3A

Up to 60 unique file sections may be used in any report.

FINAL-OUTPUT CARD MISSING

GEN3A

DISPLAY statements not preceded by FINAL OUTPUT statement.

FIRST CHARACTER MUST BE NUMERIC

GEN3

An alphabetic or special character was found in a numeric field.

FIRST CHARACTER MUST BE NUMERIC

GEN3A

Notify the MIDMS support group.

FOLLOWING LOAD MODULE NOT EXECUTABLE _____

GEN2

A convert routine or special operator cannot be executed due to an uncorrectable I/O error.

FOLLOWING LOAD MODULE NOT EXECUTABLE _____

GEN3

A referenced convert routine or table lookup name cannot be executed due to an uncorrectable I/O error.

FOLLOWING LOAD MODULE NOT EXECUTABLE _____

GEN3A

A referenced convert routine or table lookup name cannot be executed due to an uncorrectable I/O error.

FOLLOWING LOAD MODULE NOT FOUND _____

GEN2

Could not find a convert routine or special operator on library.

FOLLOWING LOAD MODULE NOT FOUND _____

GEN3

Could not find the referenced convert routine or table lookup in the library.

FOLLOWING LOAD MODULE NOT FOUND _____

GEN3A

Could not find the referenced convert routine or table lookup in the library.

HEADER/TRAILER LINES EXCEEDED

GEN3A

A maximum of 9 header lines are permitted. A maximum of 3 trailer lines are permitted.

ILLEGAL A-NAME

GEN3

The data name is incorrect.

ILLEGAL FIRST CHARACTER

GEN3

The first character for all data names must be alphabetic.

ILLEGAL FIRST WORD

GEN2

No retrieval statements start with this word.

ILLEGAL FIRST WORD

GEN3

The first word encountered at the beginning of the statement is not a valid first word.

ILLEGAL FIRST WORD

GEN3A

The first word encountered at the beginning of the statement is not a valid first word.

IMPROPER STATEMENT FORMAT

GEN2

The format of the statement is not consistent with the options required by the first word of the statement. See the Users Manual.

IMPROPER STATEMENT FORMAT

GEN3

Refer to the Users Manual, Appendix I, for proper format.

IMPROPER STATEMENT FORMAT

GEN3A

The format of the statement is not consistent with the options required by the first word of the statement. See the Users Manual.

INCOMPLETE STATEMENT

GEN2

The statement as written is not complete. Refer to the Users Manual for proper formats.

INCOMPLETE STATEMENT

GEN3

The statement as written is not complete. Refer to the Users Manual for proper formats.

INCOMPLETE STATEMENT

GEN3A

The statement as written is not complete. Refer to the Users Manual for proper formats.

INCORRECT PLACEMENT OF PERFORM PARAGRAPH

GEN3

Notify the MIDMS support group.

INCORRECT PLACEMENT OF PERFORM STATEMENT

GEN3

Refer to the Users Manual for proper use of the PERFORM statement.

INCORRECT PLACEMENT OF WRITE OR PUNCH STATEMENT

GEN3

WRITE and PUNCH statements may not be used in HEADERS or TRAILERS.

INVALID ALPHANUMERIC MOVE

GEN3

A-field and b-field must both be alphanumeric.

INVALID ALPHANUMERIC MOVE

GEN3A

Notify the MIDMS support group.

INVALID A-NAME

GEN2

The data name is not defined and is not an FFT field name.

INVALID A-NAME

GEN3

The data name is not defined and is not an FFT field name.

INVALID CALL TO GEN3B - FATAL ERROR

GEN3B

Probable hardware or system error.

INVALID CHANNEL VALUE

GEN3

The value of the channel option field must be between 2 and 11, inclusive.

INVALID COMBINATION OF PARENTHESES

GEN2

The number of left parentheses does not equal the number of right parentheses.

INVALID COMPARE

GEN2

The COMPARE operator is misspelled or multiple COMPAREs were encountered.

INVALID COMPARE

GEN3

Invalid use of the COMPARE operators.

INVALID COMPARE

GEN3A

Invalid use of the COMPARE operators.

INVALID DELIMITER

GEN2

The expected delimiter must be a space.

INVALID END OF FILE

GEN2

The error vector record is missing. Submit a discrepancy report to the MIDMS support group.

INVALID END OF FILE

GEN3

The error vector record is missing. Submit a discrepancy report to the MIDMS support group.

INVALID END OF FILE

GEN3A

The error vector record is missing. Submit a discrepancy report to the MIDMS support group.

INVALID FIELD LENGTH

GEN2

An alpha field must have a length between 1 and 360, inclusive.

INVALID FIELD LENGTH

GEN3

The field length is incorrect.

INVALID FIELD LENGTH

GEN3A

The field length is incorrect.

INVALID FLAGGING OPTION

GEN2

Incorrect statement format. Refer to the Users Manual for valid flagging options.

INVALID LENGTH

GEN2

A constant value is expected as a length parameter.

INVALID LENGTH

GEN3

The length of the data field was specified incorrectly.

INVALID LENGTH

GEN3A

The length of the data field was specified incorrectly.

INVALID LIMIT NUMBER

GEN2

Limit number does not contain numeric data.

INVALID LINE SIZE

GEN3A

Incorrect parameter in line size statement.

INVALID LITERAL

GEN2

Notify the MIDMS support group.

INVALID LITERAL

GEN3

The literal contained data inconsistent with the data type of a-field.

INVALID LITERAL

GEN3A

The literal contained data inconsistent with the data type of a-field.

INVALID MATRIX SUBSCRIPT

GEN3

The matrix subscript must be of the form MATRIX (r,c).

INVALID MODIFY STATEMENT

GEN1

The MODIFY card must identify the subquery or report it applies to. See the Users Manual for proper statement formats.

*INVALID NAME

GEN1

The name of a member stored on the library must have a fifth character equal to Q, R, or F.

INVALID NUMBER

GEN3

Either a flysheet number or a program number is not in the range 1-99.

INVALID NUMBER OF B-FIELDS

GEN2

To many b-fields were used.

INVALID NUMBER OF SUBSETS

GEN2

A negative subset number was specified.

INVALID OPERATION _____

GEN4

A compiler error has occurred. Rerun the job. If error persists, submit a discrepancy report to the MIDMS support group.

INVALID OUTPUT AREA

GEN3

The boundary of your output field is incorrect.

INVALID PARTIAL

GEN2

The partial parameters are used improperly.

INVALID PARTIAL

GEN3

The partial parameters are used improperly.

INVALID PARTIAL

GEN3A

The partial parameters are used improperly.

INVALID PROGRAM NUMBER IN PERFORM

GEN6

The number of the PROGRAM being performed may not be less than one nor more than 99. Correct and resubmit.

INVALID PROGRAM NUMBER IN PERFORM

GEN6A

The number of the PROGRAM being performed may not be less than one nor more than 99. Correct and resubmit.

INVALID PROGRAM NUMBER IN PERFORM

GEN6AIF

The number of the PROGRAM being performed may not be less than one nor more than 99. Correct and resubmit.

INVALID RECORD SIZE

GEN3

The output-field recordsize cannot exceed 9999 characters nor be less than 15 characters.

INVALID RETURN CODE FROM CONVERT ROUTINE _____

GEN2

An invalid return code was received from a user convert routine. Correct the convert routine to comply with MIDMS standard calling sequence conventions as specified in the Users Manual.

INVALID RETURN CODE FROM CONVERT ROUTINE _____

GEN3

An invalid return code was received from a user convert routine. Correct the convert routine to comply with MIDMS standard calling sequence conventions as specified in the Users Manual.

INVALID RETURN CODE FROM CONVERT ROUTINE _____

GEN3A

An invalid return code was received from a user convert routine. Correct the convert routine to comply with MIDMS standard calling sequence conventions as specified in the Users Manual.

INVALID SET/SUBSET NUMBER

GEN3

The number specified for the set or subset is incorrect.

INVALID SET/SUBSET NUMBER

GEN3A

The number specified for the set or subset is incorrect.

INVALID SORT KEY

GEN2

The sort key specified in this statement is incorrect.

INVALID SOURCE DIRECT STATEMENT

GEN3

The SOURCE DIRECT statement must immediately follow the FILE SECTION card.

INVALID SOURCE DIRECT STATEMENT

GEN3A

The SOURCE DIRECT statement must immediately follow the FILE SECTION card.

INVALID SOURCE QUERY STATEMENT

GEN3

The SOURCE QUERY statement must be of the form SOURCE QUERY nn, where nn is the number of the query in the answer file used by this report.

INVALID STATEMENT

GEN2

Invalid statement format associated with the first word in the statement.

INVALID STATEMENT

GEN3

Invalid statement format associated with the first word in the statement.

INVALID STATEMENT USING VARIABLE DATA

GEN3

A-field may be variable only in MOVE or VMOVE statements. B-field can never be variable.

INVALID STATEMENT USING VARIABLE DATA

GEN3A

A-field may be variable only in MOVE or VMOVE statements. B-field can never be variable.

INVALID S8Ø6-TRAP RETURN CODE _____

GEN2

An invalid return code was received from a system routine. Submit a discrepancy report to the MIDMS support group.

INVALID S8Ø6-TRAP RETURN CODE _____

GEN3

An invalid return code was received from a system routine. Submit a discrepancy report to the MIDMS support group.

INVALID USE OF ( CHAR

GEN3

Subscripts may not be used with the operator in this statement.

INVALID USE OF OPTIONS

GEN2

An invalid combination of options was encountered for the SORT/MERGE statement. Refer to the Users Manual for valid options.

INVALID USE OF OPTIONS

GEN3

An invalid combination of options was encountered for this statement. Refer to the Users Manual for valid options.

INVALID USE OF SATISFY

GEN2

Incorrect statement format or improper use of statement. Refer to Users Manual for proper format.

INVALID USE OF VARIABLE DATA

GEN2

The variable data specified was used incorrectly.

INVALID USE OF VARIABLE DATA

GEN3

A-field may be variable only in a MOVE or VMOVE statement.

INVALID USE OF VARIABLE DATA

GEN3A

A-field may be variable only in a MOVE or VMOVE statement.

INVALID WORD

GEN2

The data name is invalid for the statement format corresponding to the first word.

INVALID WORD

GEN3

A word was found that is not in the statement format corresponding to the first word.

INVALID WORD

GEN3A

A word was found that is not in the statement format corresponding to the first word.

LABEL TOO LONG

GEN3A

Labels may not exceed linesize.

KEEP ON FIXED FIELD INVALID

GEN2

The KEEP option of the SORT FLAGGED statement applies only to periodic and variable sets.

LABEL EXCEEDS LINESIZE

GEN3A

Label exceeds line size specified in SHL report - truncation will occur.

LAST CHARACTER NOT -- A

GEN3

In the file name the fifth character is not an A.

LAST CHARACTER NOT -- A

GEN3A

In the file name the fifth character is not an A.

LAST WORD NOT PROPERLY DEFINED

GEN2

Notify the MIDMS support group.

LAST WORD NOT PROPERLY DEFINED

GEN3

Notify the MIDMS support group.

LEFT PARENTHESES ARE EXCEEDED

GEN2

The number of right parentheses exceeds the number of left parentheses in the retrieval logic.

LEFT PARENTHESES EXCEED 9

GEN2

There are more than 9 left parentheses used in the retrieval logic.

LENGTH EXCEEDS 12 DIGITS

GEN3

Notify the MIDMS support group.

LENGTH RETURNED BY CONVERT ROUTINE EXCEEDS A-NAME

GEN2

The length returned by a convert routine exceeds the size of the alphanumeric a-field.

LENGTH RETURNED BY CONVERT ROUTINE EXCEEDS A-NAME

GEN3

The length returned from a convert routine is greater than the length of the alphanumeric a-field.

LENGTH RETURNED BY CONVERT ROUTINE EXCEEDS A-NAME

GEN3A

The length returned from a convert routine is greater than the length of the alphanumeric a-field.

LIB CARD CANNOT BE STORED

GEN1

A LIB or LIBRARY card may not be stored on the library.

*LIBRARY NAME DOES NOT END IN AN F OR AN R

GEN1

The five character library name must end in either an F or and R in a COMP-OP run.

*LIBRARY NAME IN ERROR OR IS MISSING

GEN1

The library name in a COMP-OP run must be five characters in length and be separated from the word COMP-OP by at least one space.

LINE PARAGRAPH NOT PRECEEDED BY FILE SECTION OR BEGINS PARAGRAPH

GEN3

Line paragraph not immediately following File Section or BEGINS paragraph is omitted.

LINE SIZE EXCEEDED

GEN3A

Data larger than specified line size.

LITERAL A-NAME NOT ALLOWED

GEN3

Literals cannot be used as a-fields in this statement format.

LITERAL A-NAME NOT ALLOWED

GEN3A

Literals cannot be used as a-fields in this statement format.

LITERAL B-NAME NOT ALLOWED

GEN3

The literal found in the b-field is not permitted in the statement format corresponding to the first word.

LOWER LIMIT EXCEEDS UPPER LIMIT

GEN2

The lower limit is greater than the upper limit.

MARGIN PARAMETER MISSING

GEN3

The EJECT ON nn statement requires the inclusion of the nn parameter.

MISSING A LINE OR BEGIN SECTION IN REPORT

GEN3

Each LINE paragraph requires a corresponding BEGINS paragraph - even if the latter paragraphs consist of only a LINE card or BEGINS card, respectively.

MISSING CLASS STATEMENT

GEN3A

The CLASS statement is always required in the shorthand language. If not, CLASS lines are to be printed; the value associated with the CLASS statement should be blank.

MISSING CONSTANT OR LITERAL

GEN2

This statement is missing a constant or a literal.

MISSING CONSTANT OR LITERAL

GEN3

This statement is missing a constant or a literal.

MISSING CONTINUATION CARD

GEN2

Column 72 contained a non-blank character but a continuation card did not follow.

MISSING CONTINUATION CARD

GEN3

Column 72 contained a non-blank character but a continuation card did not follow.

MISSING CONTINUATION CARD

GEN3A

Column 72 contained a non-blank character but a continuation card did not follow.

MISSING DELIMITER

GEN2

The second delimiter enclosing a literal was missing.

MISSING DELIMITER

GEN3

The second delimiter enclosing a literal was missing.

MISSING DELIMITER

GEN3A

The second delimiter enclosing a literal was missing.

MISSING FILE NAME

GEN2

The file name is missing from this statement.

MISSING FILE NAME

GEN3

The file name is missing from this statement.

MISSING FILE NAME

GEN3A

The file name is missing from this statement.

MISSING OUTPUT FILE NAME

GEN3

The report specified the WRITE option but output file name was not provided.

MISSING THE WORD - IF

GEN2

The word IF is missing.

MISSING THE WORD - IN

GEN2

In the SORT statement the word IN is required.

MULTIPLE B-NAMES NOT ALLOWED

GEN2

Only one b-field can be used in the statement format corresponding to the first word.

MULTIPLE B-NAMES NOT ALLOWED

GEN3

Only one b-field can be used in the statement format corresponding to the first word.

MULTIPLE B-NAMES NOT ALLOWED

GEN3A

Only one b-field can be used in the statement format corresponding to the first word.

MULTIPLE DATA FIELD

GEN2

This field has previously been defined.

MULTIPLE DATA FIELD

GEN3

A DEFINE statement cannot have more than one constant or literal.

MULTIPLE DATA FIELD

GEN3A

A DEFINE statement cannot have more than one constant or literal.

MULTIPLE MATRICES NOT ALLOWED

GEN3

Only one MATRIX may be defined in a report.

NAME EXCEEDS 8 CHARACTERS

GEN2

Defined and file names cannot exceed 8 characters. FFT field names cannot exceed 5 characters.

NAME EXCEEDS 8 CHARACTERS

GEN3

A defined name cannot exceed 8 characters.

NAME EXCEEDS 8 CHARACTERS

GEN3A

A defined name cannot exceed 8 characters.

NAME IDENTICAL TO FFT NAME

GEN2

A defined name was encountered which is identical to an FFT name.

NAME IDENTICAL TO FFT NAME

GEN3

The define name used is the same as an FFT name.

NAME IDENTICAL TO FFT NAME

GEN3A

The define name used is the same as an FFT name.

NAME PREVIOUSLY DEFINED

GEN2

Encountered a defined name identical to a previous defined name for which the Modify Option was not used. Each defined name must be unique unless the Modify Subquery Option is used.

NAME PREVIOUSLY DEFINED

GEN3

Encountered a defined name identical to a previous defined name for which the Modify Option was not used. Each defined name must be unique unless the Modify Report Option is used.

NAME PREVIOUSLY DEFINED

GEN3A

Encountered a defined name identical to a previous defined name for which the Modify Option was not used. Each defined name must be unique unless the Modify Report Option is used.

NAME REDEFINED

GEN2

This is only a caution. A defined name's value was changed by utilizing the Modify Subquery Option.

NAME REDEFINED

GEN3

This is a caution message. A defined name's value was changed by utilizing the Modify Report Option.

NAME REDEFINED

GEN3A

This is a caution message. A defined name's value was changed by utilizing the Modify Report Option.

NEGATIVE PAGESIZE

GEN3

A negative pagesize was found.

NEGATIVE PAGESIZE

GEN3A

A neagtive pagesize was found.

NO DATA FOLLOWING COMP-OP CARD - FATAL ERROR

GEN3B

COMP-OP card not followed by required information - check deck set-up.

NO DATA ON INPUT TAPE - FATAL ERROR

GEN3B

No data on input tape - check deck set-up.

NO ENTRIES IN FFT TABLE 7

GEN2

This is only a caution. Logical Record 7 has been referenced but there are no entries in that table.

NO FFT TABLES FOUND

GEN2

Unable to find FFT Logical Record 1 for the file name found in the current File Section.

NO FFT TABLES FOUND

GEN3

Unable to find FFT Logical Record 1 for the file specified in the File Section.

NO FFT TABLES FOUND

GEN3A

Unable to find FFT Logical Record 1 for the file specified in the File Section.

NO FURTHER COMPILATION FROM THIS POINT

GEN3

The table cannot be processed and further compilation would be meaningless.

*NO LIBRARY NAME ON COMP-OP CARD

GEN1

COMP-OP card must contain a 5 character name ending in R and separated by at least one space from COMP-OP.

NO LIMIT NUMBER PROVIDED

GEN2

This is only a caution. Limit number not provided; system will default to 0 records.

NO OF GENERATED STATEMENTS EXCEED 490 AGAINST THIS FILE

GEN2

A maximum of 490 generated object statements per file are permissable.

NO PERFORM STATEMENT ASSOCIATED WITH PROGRAM = _____

GEN3

This is a caution. Unless referenced by a PERFORM statement, a PROGRAM will not be executed.

NO QUERY CARD BAD OP PROCEDURES

GEN1

The first card in each user's job is expected to be a QUERY card. Omission of that card is a bad operating procedure.

NO QUERY FOR THIS REPORT

GEN1

Each REPORT must be preceded by a QUERY.

NON-NUMERIC DATA RETURNED BY CONVERT ROUTINE

GEN2

An alphabetic or special character was returned by the convert routine in a statement which has a numeric a-field.

NON-NUMERIC DATA RETURNED BY CONVERT ROUTINE

GEN3

An alphabetic or special character was returned by the convert routine in a statement which has a numeric a-field.

NOT ALLOWED IN THIS PARAGRAPH

GEN3

The EJECT statement cannot be used in the HEADER or TRAILER paragraphs.

NUMBER EXCEEDS ONE DIGIT

GEN3

The number in the statement format corresponding to the first word must be one digit.

NUMBER EXCEEDS 4 DIGITS

GEN2

The value of the constants in a SATISFY operator may not exceed 9999.

NUMERIC A-NAME EXCEEDS 15 DIGITS

GEN3

All numeric data cannot exceed 15 digits.

NUMERIC A-NAME EXCEEDS 15 DIGITS

GEN3A

All numeric data cannot exceed 15 digits.

NUMERIC FIELD EXCEEDS 15 DIGITS

GEN2

The numeric field specified is greater than 15 digits.

NUMERIC FIELD EXCEEDS 15 DIGITS

GEN3

The numeric field is larger than 15 digits.

NUMERIC FIELD EXCEEDS 15 DIGITS

GEN3A

The numeric field is larger than 15 digits.

OCCURS AREA EXCEEDS CONSTANT POOL

GEN3

The table definition has caused overflow. The combined length of all defined space and values may not exceed 9000 characters.

ONLY VALID IN TRAILER PARAGRAPH

GEN3

The SKIP statement can only be used in the TRAILER Section.

OP4-SWT

GEN2

Compiler error. Resubmit the job. If error persists, submit a discrepancy report to the MIDMS support group.

OP5-SWT

GEN2

Compiler error. Resubmit the job. If error persists, submit a discrepancy report to the MIDMS support group.

OPTION TO ABORT ON ANY ERROR HAS BEEN EXERCISED

GEN6

One or more of the queries or reports in the batch have a compilation error, therefore all were prevented from executing.

OPTION TO ABORT RUN ON ANY ERROR HAS BEEN EXERCISED

GEN6A

One or more of the queries or reports in this batch have a compilation error, therefore all were prevented from executing.

OPTION TO ABORT RUN ON ANY ERROR HAS BEEN EXERCISED

GEN6AIF

One or more of the queries or reports in this batch have a compilation error, therefore all were prevented from executing.

OUTPUT FILE PREVIOUSLY DEFINED

GEN3

This is a caution message. The output file parameter was changed utilizing the Modify Report Option.

PAGESIZE PREVIOUSLY DEFINED

GEN3

This is a caution message. The pagesize parameter was changed utilizing the Modify Report Option.

PAGESIZE PREVIOUSLY DEFINED

GEN3A

This is a caution message. The pagesize parameter was changed utilizing the Modify Report Option.

PARAGRAPH 1 IS MISSING

GEN3

The BEGINS paragraph does not immediately follow the LINE paragraph.

PARTIAL EXCEEDS FIELD LENGTH

GEN2

Incorrect length was specified for a partial field.

PARTIAL EXCEEDS FIELD LENGTH

GEN3

The partial parameter is too large.

PARTIAL EXCEEDS FIELD LENGTH

GEN3A

The partial parameter is too large.

PARTIAL NOTATION OUTSIDE FIELD BOUNDARIES

GEN2

The partial notation specified is outside the field size.

PARTIAL NOTATION OUTSIDE FIELD BOUNDARIES

GEN3

The partial notation is greater than the field size.

PARTIAL NOTATION OUTSIDE FIELD BOUNDARIES

GEN3A

The partial notation is greater than the field size.

PROGRAM NUMBER EXCEEDS 2 DIGITS

GEN3

The program number must be between 1 and 99, inclusive.

PROGRAM NUMBER MISSING

GEN3

A program number, ranging in value from 1 to 99, must be included as part of the PROGRAM name. See the Users Manaul.

PROGRAM SECTION = _____ IS MISSING

GEN3

A referenced PROGRAM paragraph is either missing, improperly identified, or incorrectly located in the REPORT. See the Users Manual.

QUERY CARD MISSING - DUMMY CARD WILL BE GENERATED

GEN3B

This is a warning. No QUERY card was found and GEN3B generated one.

QUERY CARD SHOULD NOT BE STORED

GEN1

This is a caution. The QUERY card should not be stored on the source statement library.

RECED-ID value1 SET-ID value2 PSS-ID
value3 CHARACTER POS value4

GEN4

This message is generated upon receipt of a return code of 4 from a special operator. Value 1 is the record ID, value 2 the set number, value 3 is the PSS field, and value 4 is the character position within that subset of the data passed to the special operator.

RECED-ID value1 SET-ID value2 PSS-ID
value3 CHARACTER POS value4

GEN4A

This message is generated upon receipt of a return code of 4 from a special operator. Value 1 is the record ID, value 2 the set number, value 3 is the PSS field, and value 4 is the character position within that subset of the data passed to the special operator.

RECED-ID value1 SET-ID value2 PSS-ID
value3 CHARACTER POS value4

GEN4AIF

This message is generated upon receipt of a return code of 4 from a special operator. Value 1 is the record ID, value the set number, value 3 is the PSS field, and value 4 is the character position within that subset of the data passed to the special operator.

RECEIVING FIELD LONGER WILL ADD TRAILING SPACES

GEN3

This is only a caution. The receiving field was larger than the sending field. Contents of the sending field will be sent to the receiving field, left-justified, with trailing spaces added.

RECEIVING FIELD SHORTER - WILL TRUNCATE

GEN3

This is a caution message. The number of characters moved from the sending field will be equal to the length of the receiving field and will be left-justified.

RECORDSIZE PREVIOUSLY DEFINED

GEN3

This is a caution message. The recordsize parameter was changed utilizing the Modify Report Option.

REPORT OBJECT STATEMENTS EXCEED 1333

GEN2

The number of object statements generated from the user source statements found in the report exceed 1333. Report must be shortened. Multiple b-fields are expanded into one object statement for each b-field.

REPORT OBJECT STATEMENTS EXCEED 1333

GEN3A

The number of object statements generated from the user source statements found in the report exceeded 1333. Report must be shortened. Multiple b-fields are expanded into one object statement for each b-field.

RETRIEVAL LOGIC STATEMENTS FOLLOWING SELECTIVE LOGIC

GEN2

Once a secondary conditional statement is written no primary conditional statements may be written in the remainder of the subquery. See the Users Manual.

RETRIEVAL VECTOR MISSING - RERUN - IF ERROR PERSISTS CONSULT SE

GEN3

Submit a discrepancy report to the MIDMS support group.

RIGHT PAREN MISSING

GEN3

A right parenthesis is required to close subscript expressions.

RIGHT PARENTHESES EXCEED 9

GEN2

There are more than 9 right parentheses used in the retireval logic.

RIGHT PAPEN MISSING

GEN2

A right parenthesis is required to match the left parenthesis previously encountered.

RIGHT PAREN MISSING

GEN3

A right parenthesis is required to match the left parenthesis previously encountered.

ROUTINE EXCEEDS 8 CHARACTERS

GEN3

A special operator, convert routine, or table lookup name cannot exceed 8 characters.

ROUTINE EXCEEDS 8 CHARACTERS

GEN3A

A special operator, convert routine, or table lookup name cannot exceed 8 characters.

SAME AS DEFINE NAME

GEN2

A referenced FFT name is the same as a name previously DEFINEd.

SECOND WORD MISSING

GEN2

The second word in the statement format corresponding to the first word is missing.

SECOND WORD MISSING

GEN3

The second word in the statement format corresponding to the first word is missing.

SKIPPING TO NEXT REPORT

GEN3

Further compilation of this report is meaningless based on errors already encountered.

SKIPPING TO NEXT REPORT

GEN3A

Further compilation of this report is meaningless based on errors already encountered.

SKIPPING TO NEXT SUBQUERY

GEN2

Syntax checking was not performed on all retrieval statements following this error message up to the next subquery card.

****** SORT ERROR - LIBRARY CONTENTS MAY BE INVALID

GEN3B

Restore library from backup tape before proceeding.

SORT ERROR, ANSWER FILE NOT IN SEQUENCE

GEN6

Allocate additional space for the sort work files and resubmit. If error persists, submit a discrepancy report to the MIDMS support group.

SORT ERROR, ANSWER FILE NOT IN SEQUENCE

GEN6A

Allocate additional space for the sort work files and resubmit. If error persists, submit a discrepancy report to the MIDMS support group.

SORT ERROR, ANSWER FILE NOT IN SEQUENCE

GEN6AIF

Allocate additional space for the sort work files and resubmit. If error persists, submit a discrepancy report to the MIDMS support group.

SORT ERROR, JOB KAPUT, ERROR**********

GEN5A

Check JCL for sort work space. If adequate, submit a discrepancy report to the MIDMS support group.

SORT ERROR, JOB KAPUT, ERROR **********

GEN1A

Check JCL for sort work space. If adequate, submit a discrepancy report to the MIDMS support group.

SORT FIELD NOT PERIODIC

GEN3

The ASORT and DSORT field names must be periodic.

SORT KEY EXCEEDS 85 CHARACTERS

GEN2

This is only a caution. The sort key has exceeded 85 characters. Truncation will occur.

SORT/MERGE FLAGGED STATEMENT MISSING

GEN2

KEEP statements require a SORT/MERGE FLAGGED statement be present.

SORT/MERGE FLAGGED STATEMENT NOT AT END

GEN2

Retrieval sort or merge statements containing the flagged option must be the last physical statement in a subquery.

SORT/MERGE STATEMENTS NOT AT END OF SUBQUERY

GEN2

The SORT/MERGE statements must be at the end of the subquery and can be followed only by a SORT/MERGE statement with the FLAGGED option.

SORT STATEMENT NOT IN LINE PARAGRAPH

GEN3

The ASORT and DSORT operators may only be used in a LINE paragraph.

SPECIAL OP EXCEEDS 8 CHARACTERS

GEN2

The special operator name cannot exceed 8 characters.

STATEMENT NOT PROCESSED

GEN2

This statement was not processed because of misspelling or incorrect use.

STATEMENT NOT PROCESSED

GEN3

A previously encountered error condition precluded processing of this statement.

SUBQUERY CRD MISSING

GEN2

The only cards that may precede a SUBQUERY card within a QUERY are LIB cards.

SUMMARY CARD DOES NOT IMMEDIATELY FOLLOW SUBQUERY CARD

GEN2

The placement of the summary card is incorrect.

TABLE DEFINE MISSING

GEN3

Referenced table name has not been defined.

TABLE ELEMENT MISSING

GEN3

The control field containing the number of elements in the table has a greater value than the number of elements actually found.

TABLE ELEMENT TOO LONG

GEN3

The combined length of an argument and function exceeded 68 characters.

THE WORD, OF, IS MISSING

GEN2

The word OF is misspelled and is required in the statement format corresponding to the first word.

THE WORD, TO, IS MISSING

GEN3

The word TO is required in the statement format corresponding to the first word.

TOO MANY CONDITIONALS

GEN3A

Only nine conditionals may be used.

TOO MANY COUNTS/SUMS

GEN3A

More than 100 unique field names were found in SUM and COUNT statements of type 1 format in the SHL report.

TOO MANY DEFINES

GEN3A

The valid number of define names including the system generated ones for internal labels, edit masks, and implicit output convert routines was exceeded.

TOO MANY KEEP NAMES

GEN2

The maximum number of KEEPs is 50.

TRUNCATION IN SORT KEY

GEN2

This is only a caution. The rightmost characters in the sort key were truncated.

UNDEFINED B-NAME

GEN2

The data name in the b-field was not found in the list of defined names or in the list of FFT names associated with the current subquery. If intended to be a literal, the value must be enclosed in delimiters.

UNDEFINED B-NAME

GEN3

The data name in the b-field was not found in the list of defined names or in the list of FFT names associated with the current File Section. If intended to be a literal, the value must be enclosed in delimiters.

UNDEFINED B-NAME

GEN3A

The data name in the b-field was not found in the list of defined names or in the list of FFT names associated with the current File Section. If intended to be a literal, the value must be enclosed in delimiters.

UPPER LIMIT MISSING

GEN2

The upper limit in the SATISFIES operator cannot be found.

VALUE EXCEEDS 999

GEN2

The HIT parameter in SELECT statement exceeds 999.

VALUE EXCEEDS 999

GEN3

The value specified may not exceed 999. See the Users' Manual.

VARIABLE DATA NAME REQUIRED

GEN3

The VMOVE operator requires that the a-field be a variable set field name.

VMOVE OPERATOR CHANGED TO MOVE

GEN3

VMOVE may only be executed for a BEGINS paragraph.

YOU ARE IN A LOOP, FRIEND ...

GEN6

A paragraph of output statements has been executed without interruption more than 11111 times. This can be caused by an invalid control word or by illegal modification of a control word.

YOU ARE IN A LOOP, FRIEND ...

GEN6A

A paragraph of output statements has been executed without interruption more than 11111 times. This can be caused by an invalid control word or by illegal modification of a control word.

YOU ARE IN A LOOP, FRIEND ...

GEN6AIF

A paragraph of output statements has been executed without interruption more than 11111 times. This can be caused by an invalid control word or by illegal modification of a control word.

98 LOGIC ERROR - QUERY DELETED, ERROR

GEN4

A logic processor error has occurred. Rerun the job; if error persists, submit a discrepancy report to the MIDMS support group.

CHAPTER 5

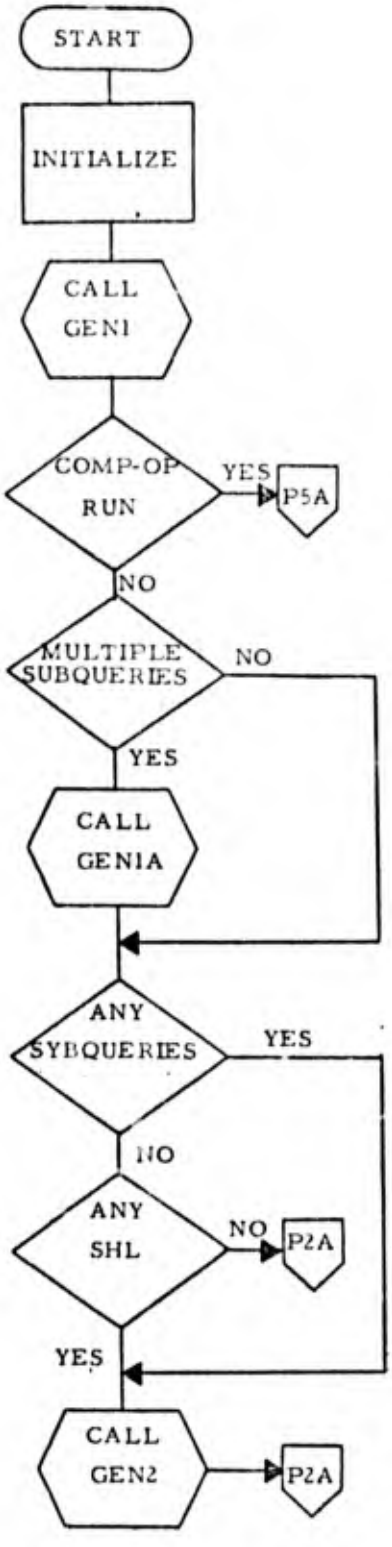
RETRIEVAL AND OUTPUT DOCUMENTATION

PART II

Flowcharts and Narratives:

a. Program Flowcharts

(1) GENO



CHAPTER 5

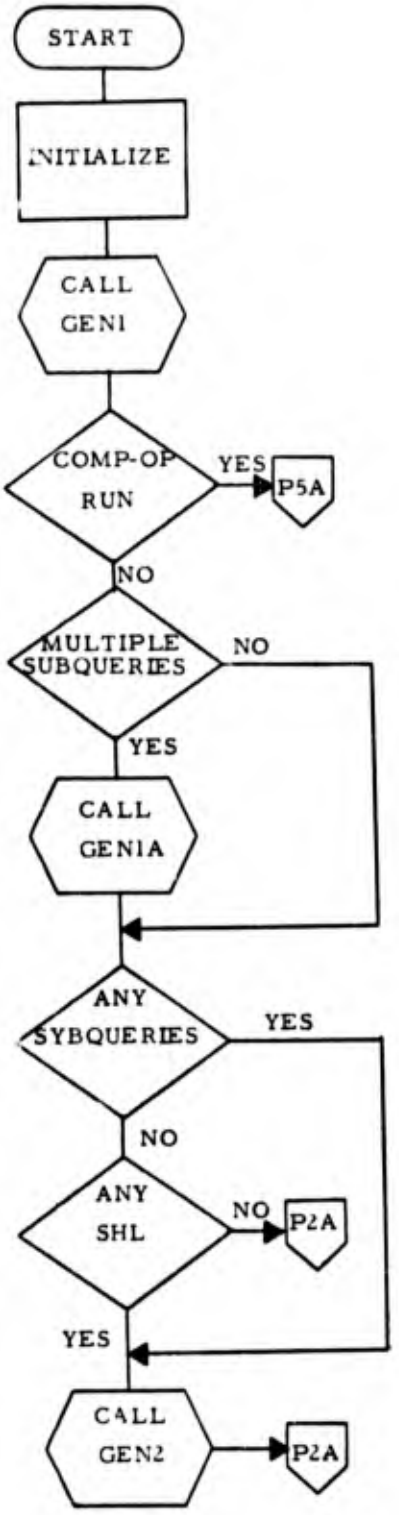
RETRIEVAL AND OUTPUT DOCUMENTATION

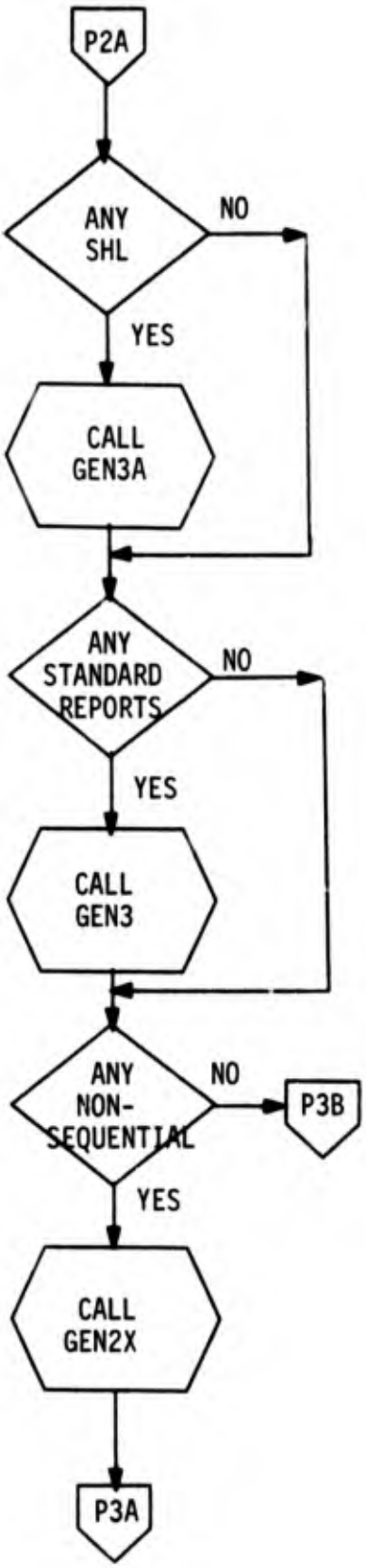
PART II

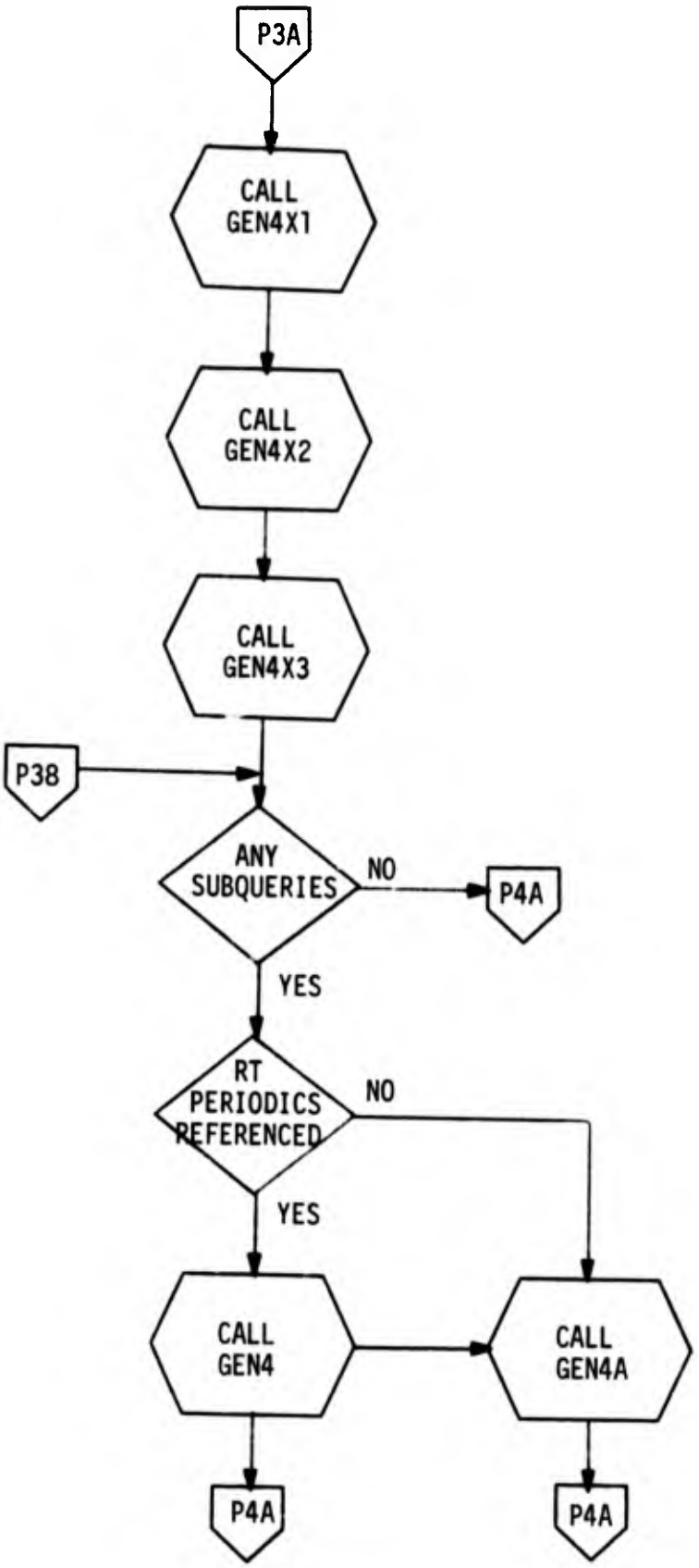
Flowcharts and Narratives:

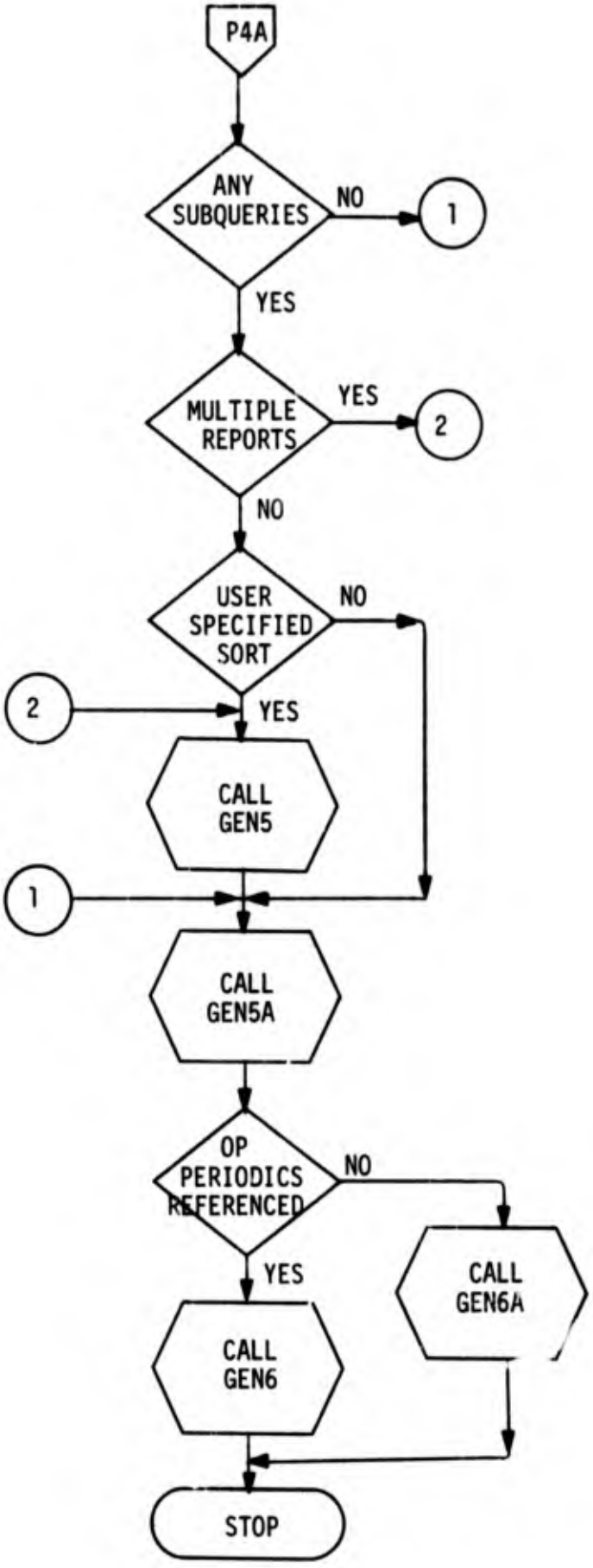
a. Program Flowcharts

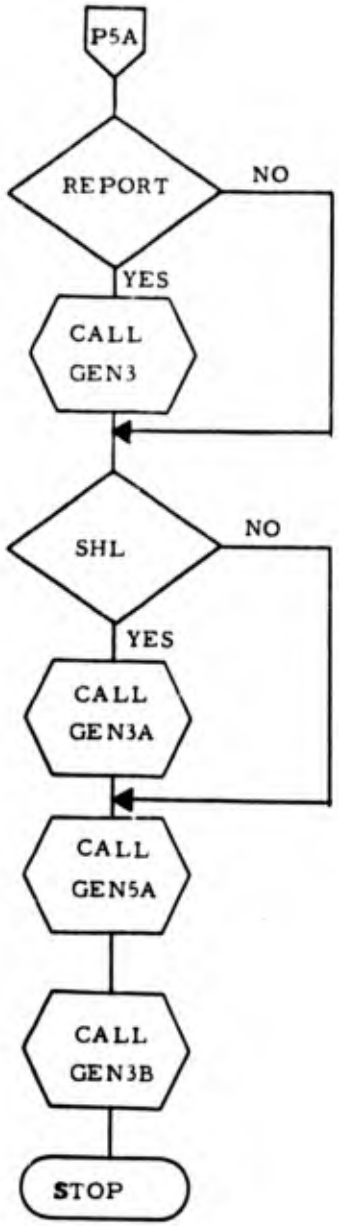
(1) GENO



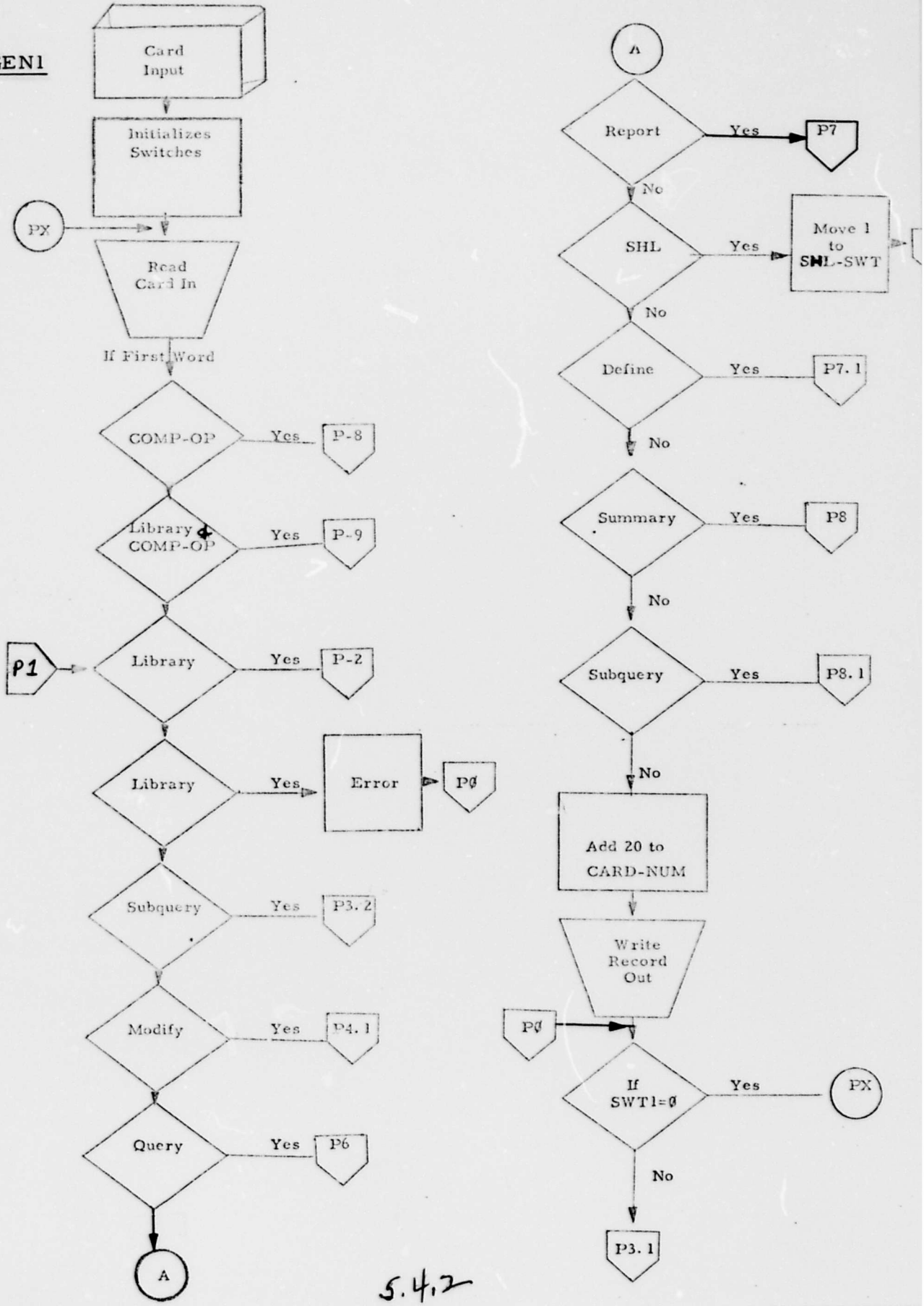

P2A
ANY SHL
NO
YES
CALL GEN3A
ANY STANDARD REPORTS
NO
YES
CALL GEN3
ANY NON-SEQUENTIAL
NO
P3B
YES
CALL GEN2X
P3A


P3A
CALL GEN4X1
CALL GEN4X2
CALL GEN4X3
P38
ANY SUBQUERIES
NO
P4A
YES
RT PERIODICS REFERENCED
NO
YES
CALL GEN4
CALL GEN4A
P4A
P4A


P4A
ANY SUBQUERIES
NO
1
YES
MULTIPLE REPORTS
YES
2
NO
USER SPECIFIED SORT
NO
2
YES
CALL GEN5
1
CALL GEN5A
OP PERIODICS REFERENCED
NO
YES
CALL GEN6
CALL GEN6A
STOP

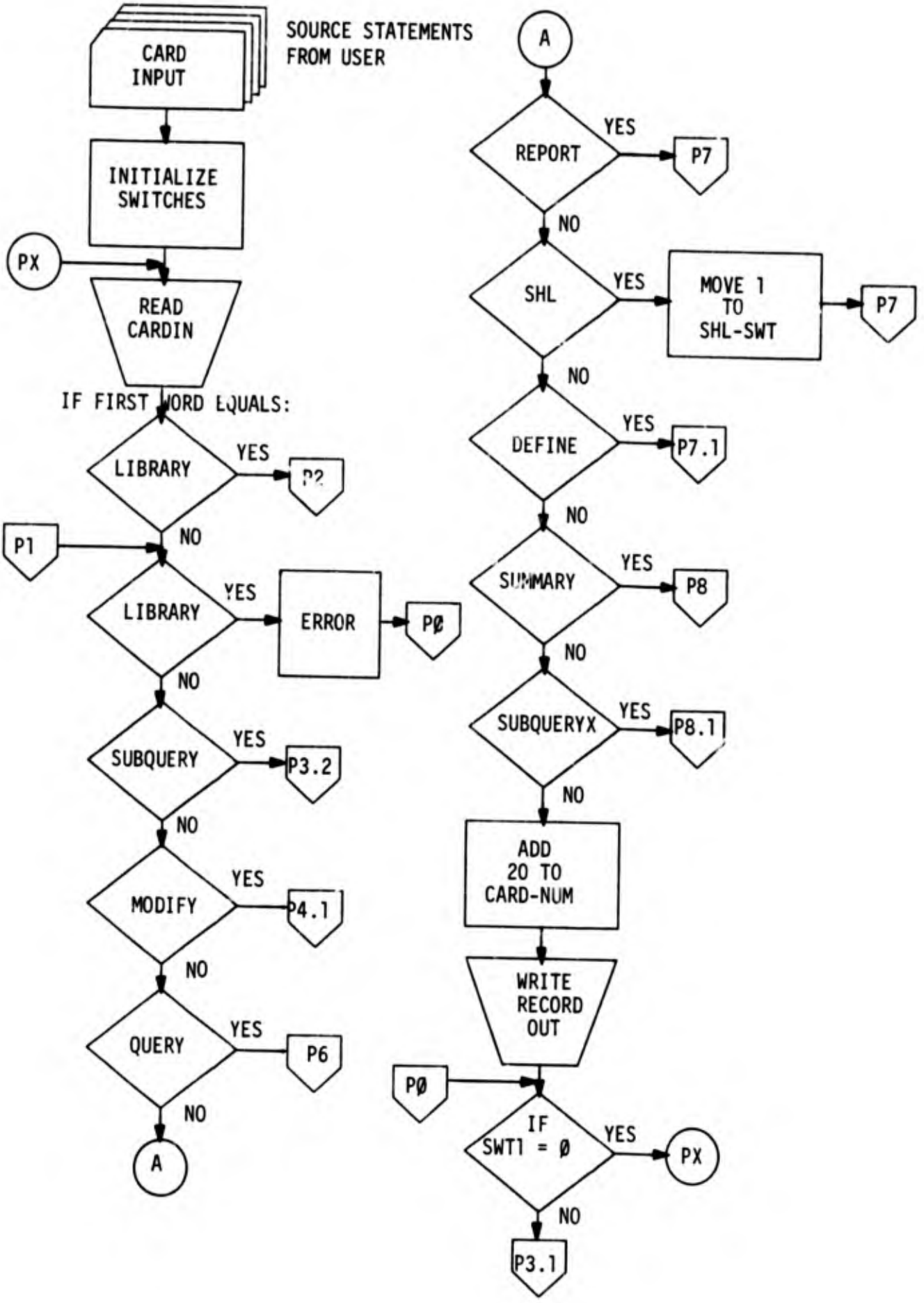

P5A
REPORT
NO
YES
CALL
GEN3
SHL
NO
YES
CALL
GEN3A
CALL
GEN5A
CALL
GEN3B
STOP

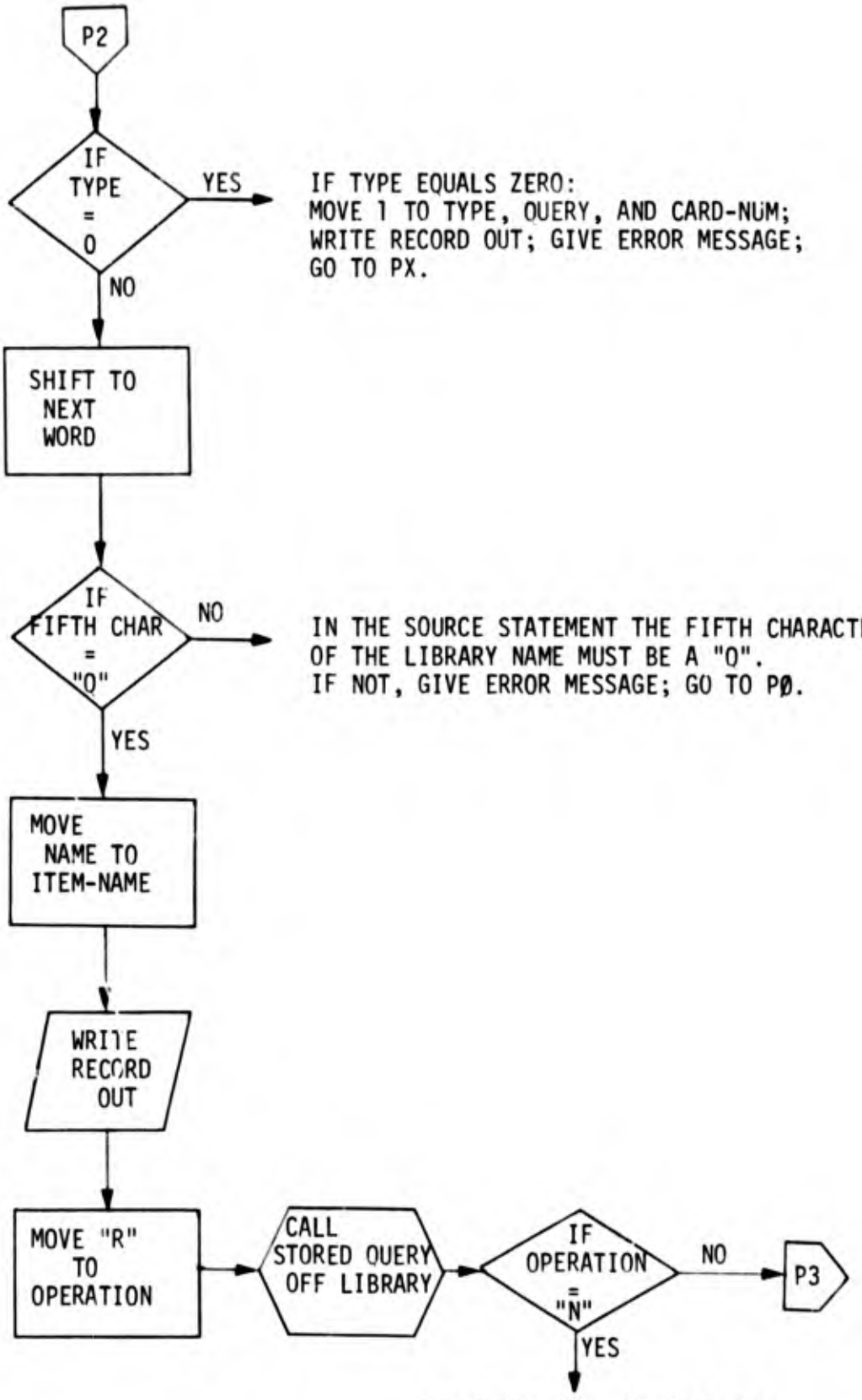
## 2. GEN1

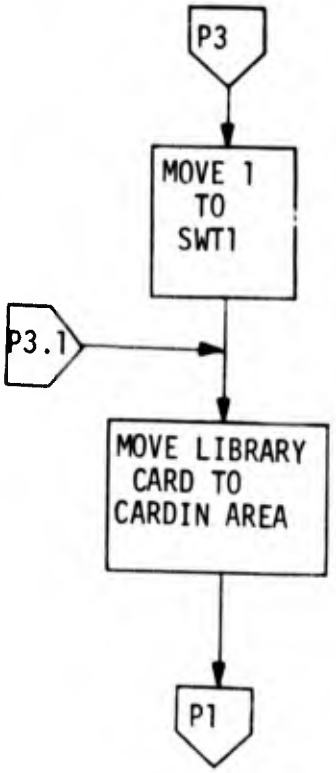

Card Input
Initializes Switches
PX
Read Card In
If First Word
COMP-OP
Yes
P-8
Library COMP-OP
Yes
P-9
P1
Library
Yes
P-2
Library
Yes
Error
PØ
Subquery
Yes
P3.2
Modify
Yes
P4.1
Query
Yes
P6
A
A
Report
Yes
P7
No
SHL
Yes
Move 1 to SHL-SWT
No
Define
Yes
P7.1
No
Summary
Yes
P8
No
Subquery
Yes
P8.1
No
Add 20 to CARD-NUM
Write Record Out
PØ
If SWT1=Ø
Yes
PX
No
P3.1

5.4.2

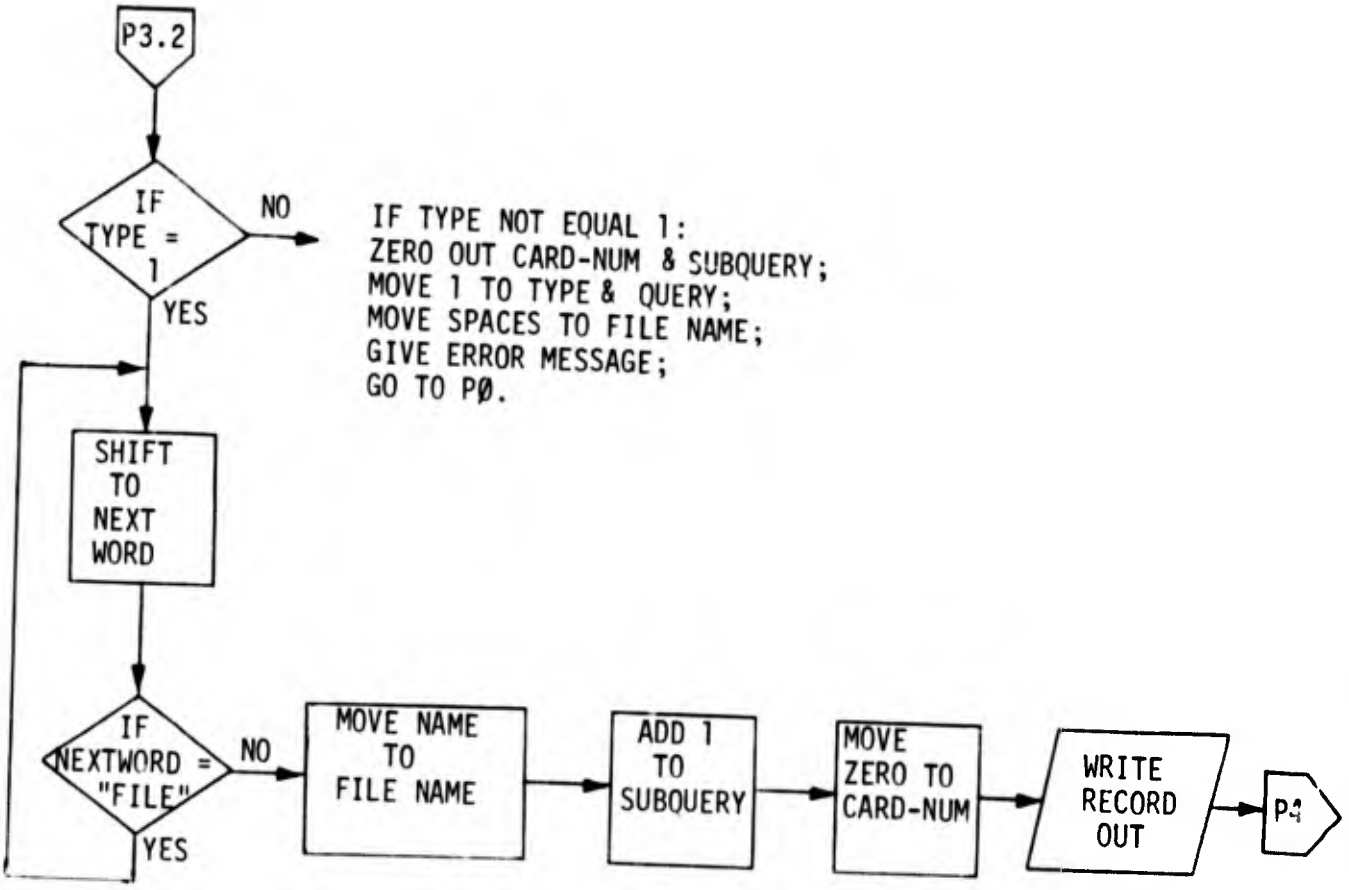
(2) GEN1.


CARD INPUT
SOURCE STATEMENTS FROM USER
INITIALIZE SWITCHES
PX
READ CARDIN
IF FIRST WORD EQUALS:
LIBRARY
YES
P2
NO
P1
LIBRARY
YES
ERROR
PØ
NO
SUBQUERY
YES
P3.2
NO
MODIFY
YES
P4.1
NO
QUERY
YES
P6
NO
A
A
REPORT
YES
P7
NO
SHL
YES
MOVE 1 TO SHL-SWT
P7
NO
DEFINE
YES
P7.1
NO
SUMMARY
YES
P8
NO
SUBQUERYX
YES
P8.1
NO
ADD 20 TO CARD-NUM
WRITE RECORD OUT
PØ
IF SWT1 = Ø
YES
PX
NO
P3.1

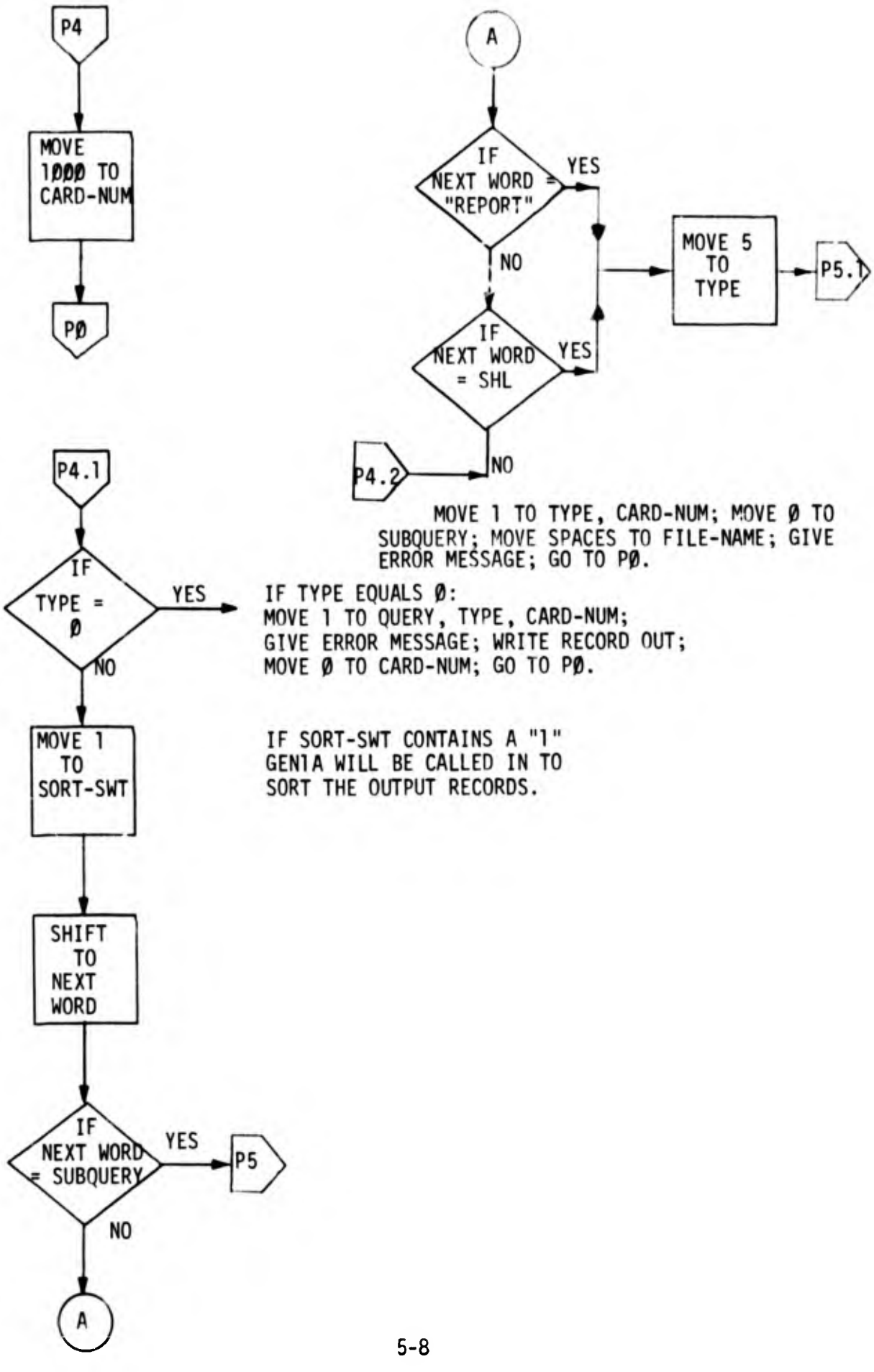

P2
IF TYPE = 0
YES
IF TYPE EQUALS ZERO:
MOVE 1 TO TYPE, QUERY, AND CARD-NUM;
WRITE RECORD OUT; GIVE ERROR MESSAGE;
GO TO PX.
NO
SHIFT TO NEXT WORD
IF FIFTH CHAR = "0"
NO
IN THE SOURCE STATEMENT THE FIFTH CHARACTER
OF THE LIBRARY NAME MUST BE A "0".
IF NOT, GIVE ERROR MESSAGE; GO TO PØ.
YES
MOVE NAME TO ITEM-NAME
WRITE RECORD OUT
MOVE "R" TO OPERATION
CALL STORED QUERY OFF LIBRARY
IF OPERATION = "N"
NO
P3
YES
AFTER THE CALL RETURNS AND
OPERATION CONTAINS AN "N",
GIVE ERROR MESSAGE; MOVE Ø
TO SWT1; GO TO PX.

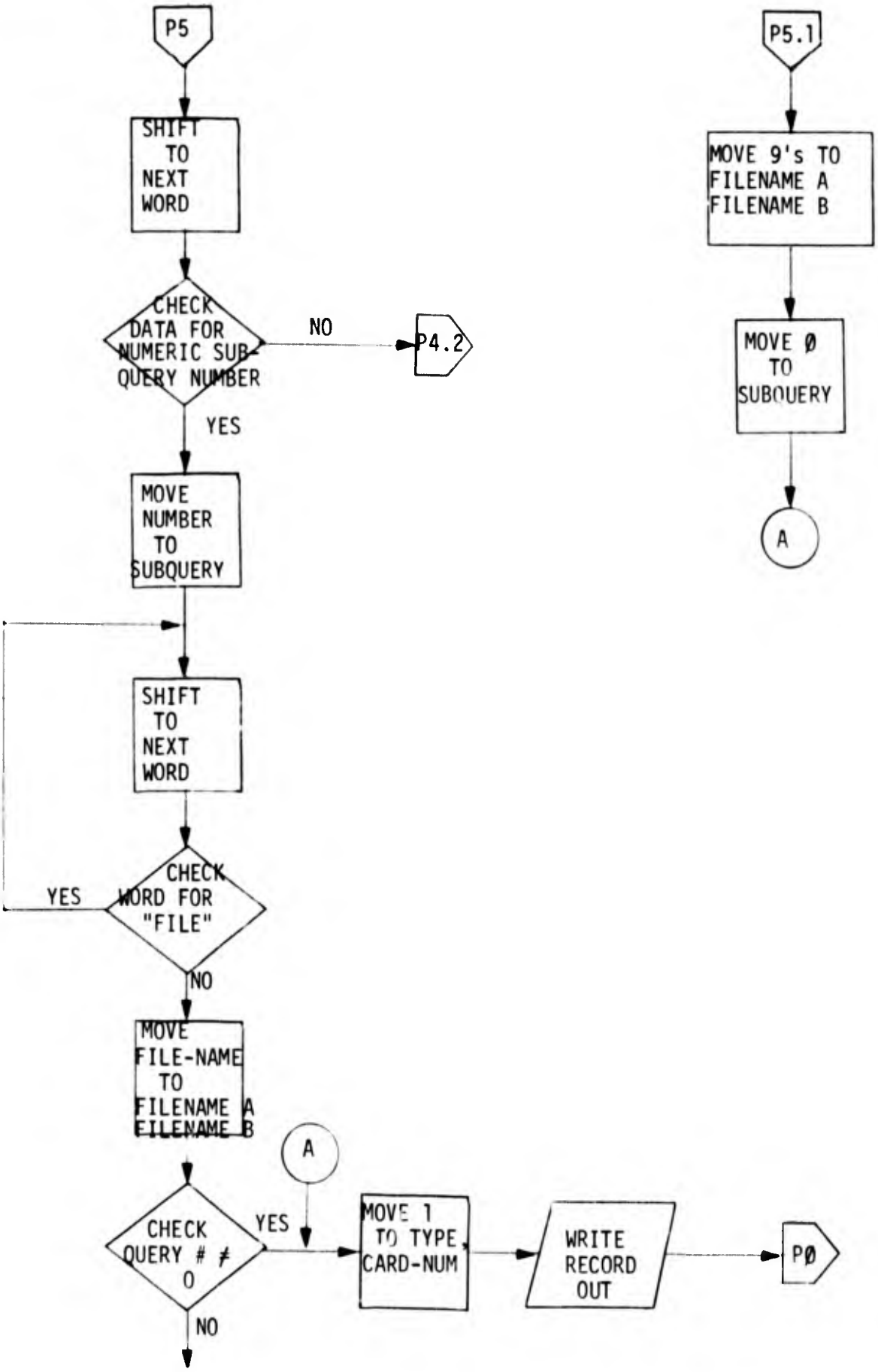



TO INDICATE SOURCE STATEMENTS
WILL BE COMING FROM LIBRARY;
OTHERWISE SWT1 REMAINS ZERO.

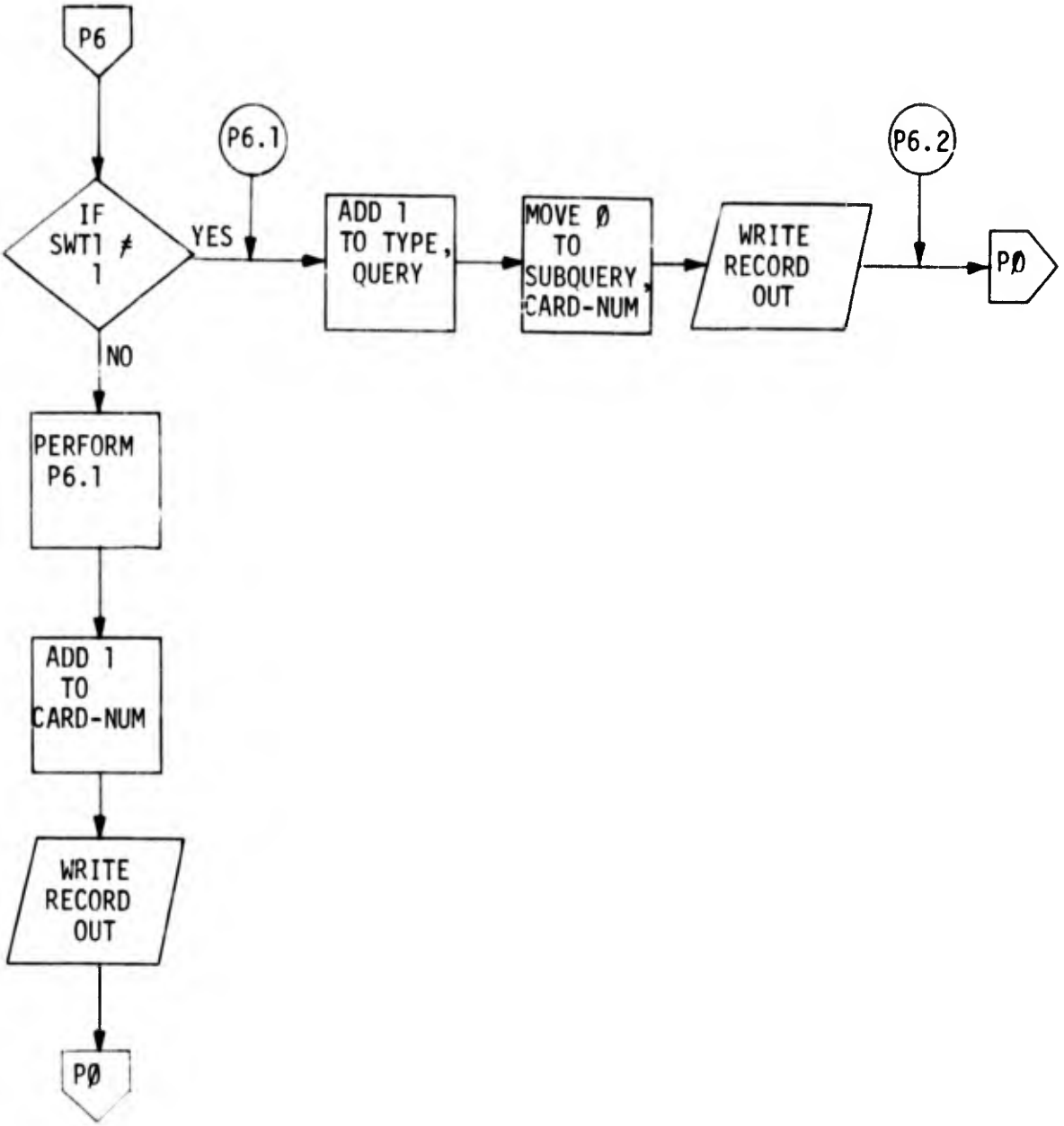


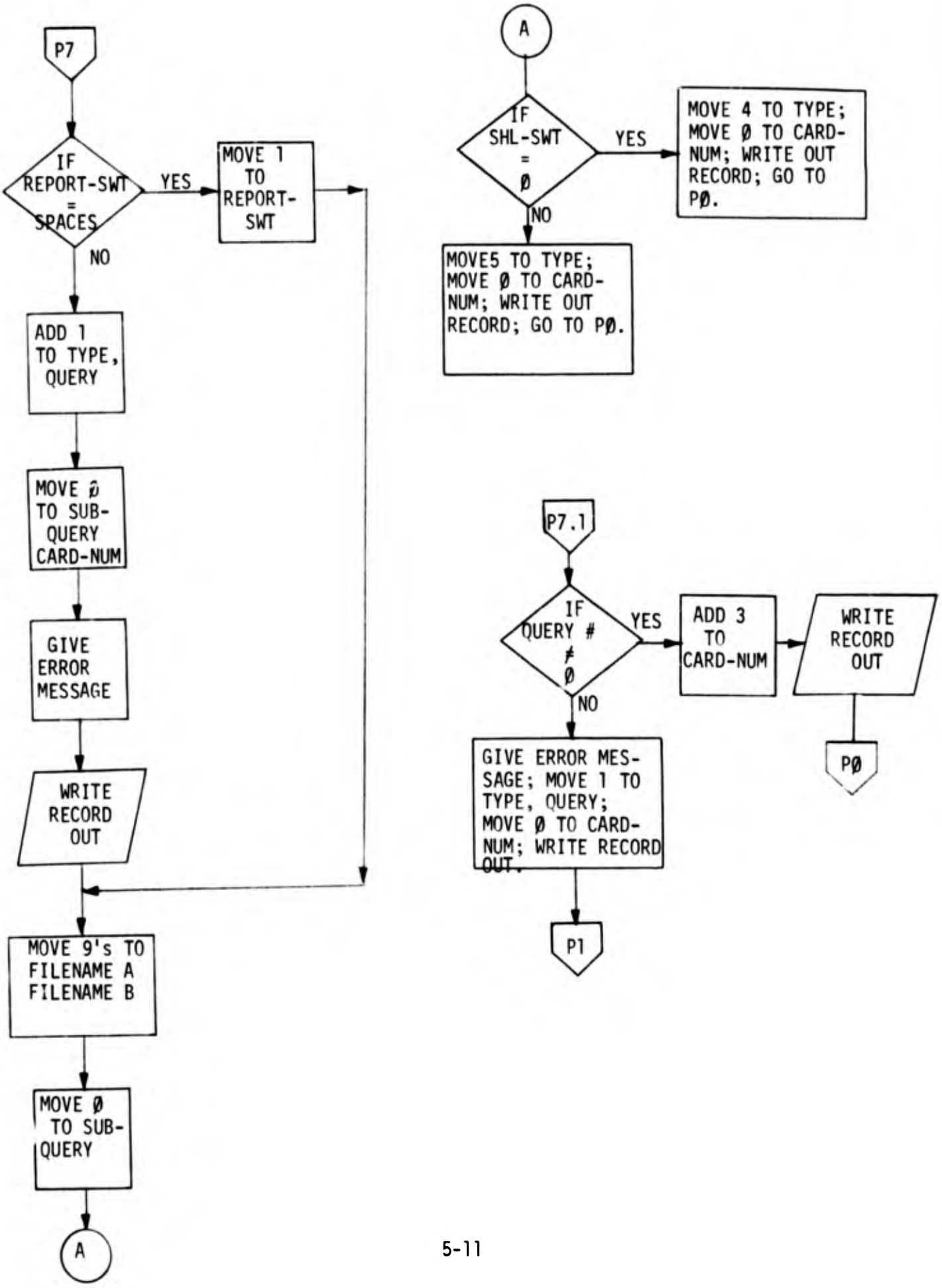

IF TYPE NOT EQUAL 1:
ZERO OUT CARD-NUM & SUBQUERY;
MOVE 1 TO TYPE & QUERY;
MOVE SPACES TO FILE NAME;
GIVE ERROR MESSAGE;
GO TO PØ.

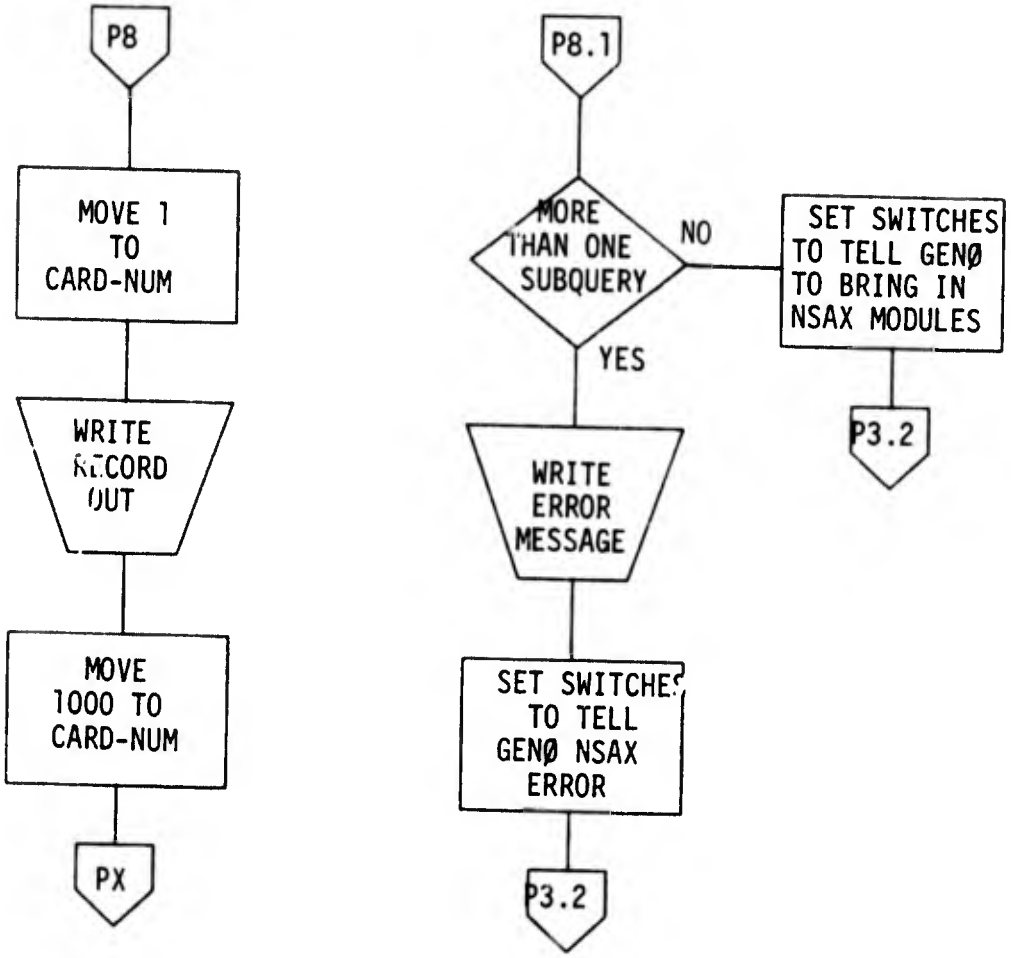

P4
MOVE 1000 TO CARD-NUM
PØ
A
IF NEXT WORD = "REPORT"
YES
NO
MOVE 5 TO TYPE
P5.1
IF NEXT WORD = SHL
YES
NO
P4.2
MOVE 1 TO TYPE, CARD-NUM; MOVE Ø TO SUBQUERY; MOVE SPACES TO FILE-NAME; GIVE ERROR MESSAGE; GO TO PØ.
P4.1
IF TYPE = Ø
YES
NO
IF TYPE EQUALS Ø:
MOVE 1 TO QUERY, TYPE, CARD-NUM;
GIVE ERROR MESSAGE; WRITE RECORD OUT;
MOVE Ø TO CARD-NUM; GO TO PØ.
MOVE 1 TO SORT-SWT
IF SORT-SWT CONTAINS A "1"
GEN1A WILL BE CALLED IN TO
SORT THE OUTPUT RECORDS.
SHIFT TO NEXT WORD
IF NEXT WORD = SUBQUERY
YES
P5
NO
A

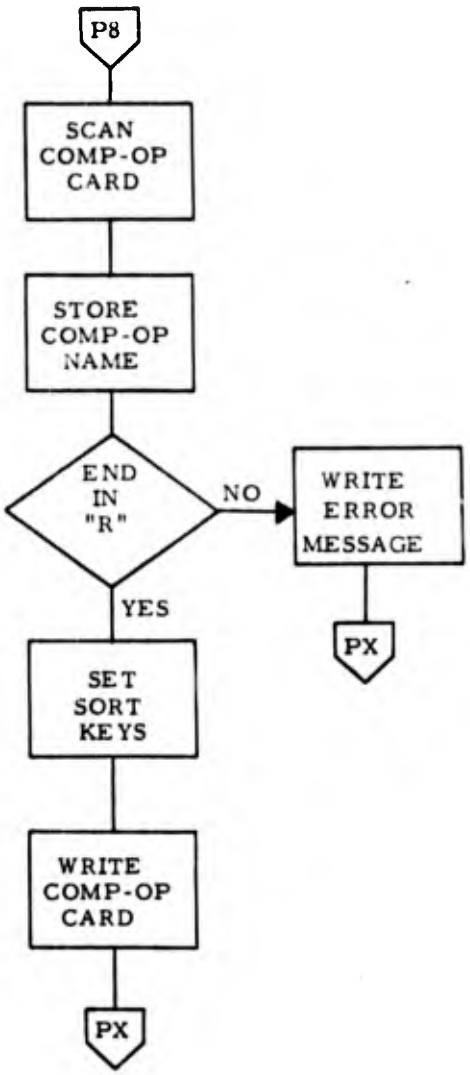


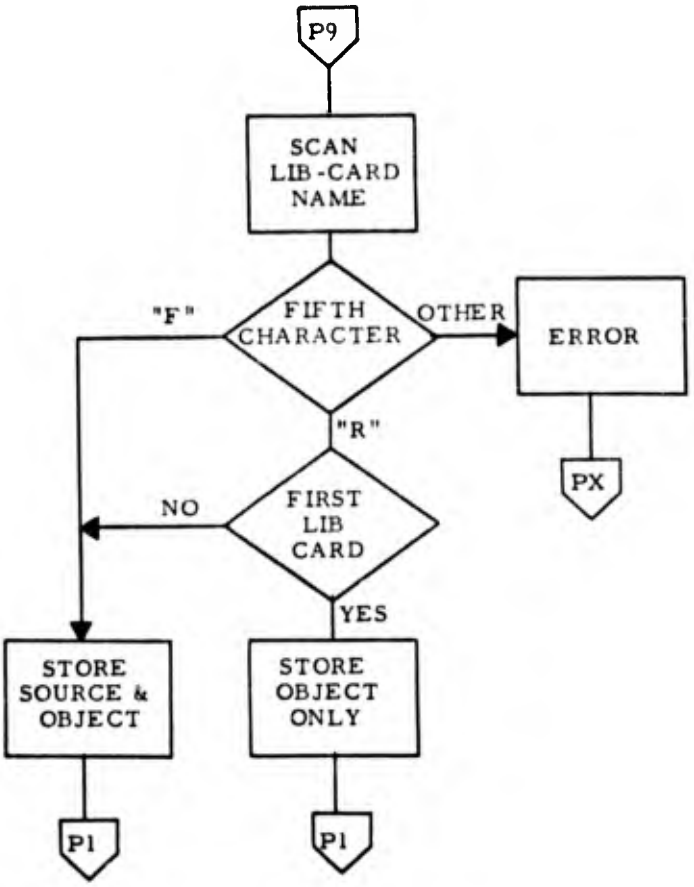

MOVE 1 TO TYPE, QUERY; MOVE Ø TO SUBQUERY, CARD-NUM; GIVE ERROR MESSAGE; MOVE CARD TO WORK AREA; GO TO P1.


P6
P6.1
P6.2
IF SWT1 ≠ 1
YES
NO
ADD 1 TO TYPE, QUERY
MOVE Ø TO SUBQUERY, CARD-NUM
WRITE RECORD OUT
PØ
PERFORM P6.1
ADD 1 TO CARD-NUM
WRITE RECORD OUT
PØ

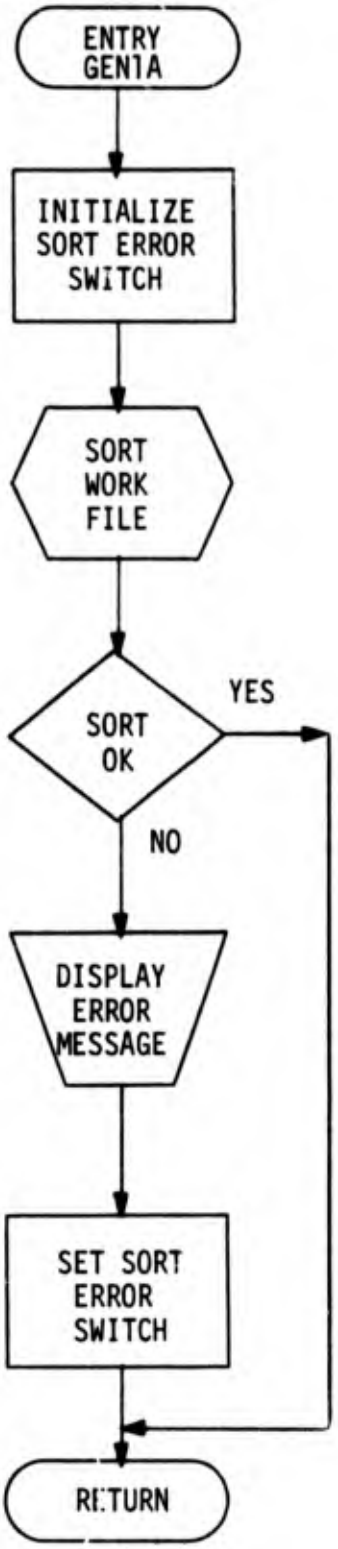

P7
IF REPORT-SWT = SPACES
YES
MOVE 1 TO REPORT-SWT
NO
ADD 1 TO TYPE, QUERY
MOVE Ø TO SUB-QUERY CARD-NUM
GIVE ERROR MESSAGE
WRITE RECORD OUT
MOVE 9's TO FILENAME A FILENAME B
MOVE Ø TO SUB-QUERY
A
A
IF SHL-SWT = Ø
YES
MOVE 4 TO TYPE; MOVE Ø TO CARD-NUM; WRITE OUT RECORD; GO TO PØ.
NO
MOVE5 TO TYPE; MOVE Ø TO CARD-NUM; WRITE OUT RECORD; GO TO PØ.
P7.1
IF QUERY # ≠ Ø
YES
ADD 3 TO CARD-NUM
WRITE RECORD OUT
PØ
NO
GIVE ERROR MESSAGE; MOVE 1 TO TYPE, QUERY; MOVE Ø TO CARD-NUM; WRITE RECORD OUT.
P1


P8
MOVE 1
TO
CARD-NUM
WRITE
RECORD
OUT
MOVE
1000 TO
CARD-NUM
PX
P8.1
MORE
THAN ONE
SUBQUERY
NO
SET SWITCHES
TO TELL GENØ
TO BRING IN
NSAX MODULES
P3.2
YES
WRITE
ERROR
MESSAGE
SET SWITCHES
TO TELL
GENØ NSAX
ERROR
P3.2

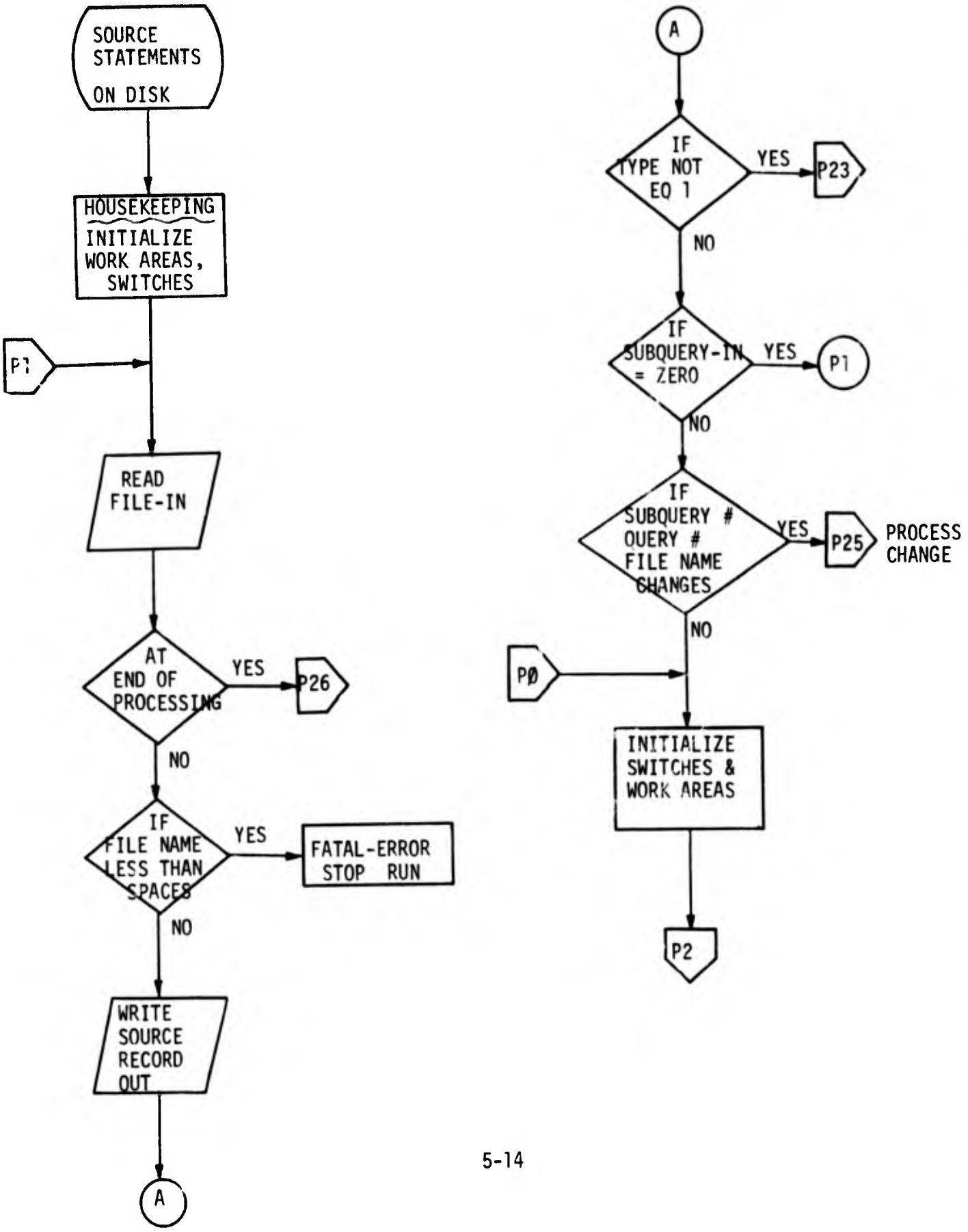

P8
SCAN COMP-OP CARD
STORE COMP-OP NAME
END IN "R"
NO
WRITE ERROR MESSAGE
PX
YES
SET SORT KEYS
WRITE COMP-OP CARD
PX

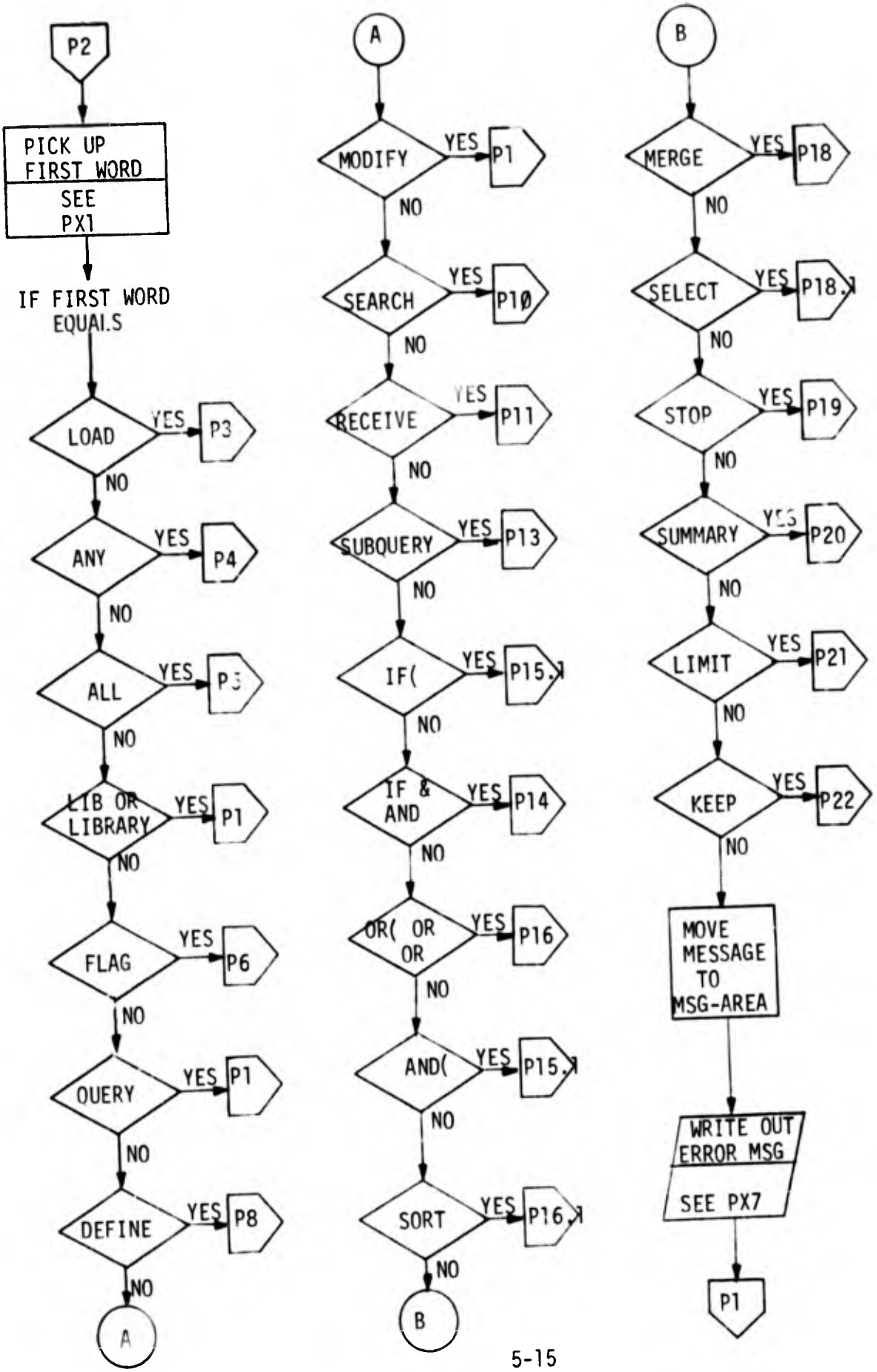

P9
SCAN LIB-CARD NAME
FIFTH CHARACTER
"F"
OTHER
ERROR
PX
"R"
NO
FIRST LIB CARD
YES
STORE SOURCE & OBJECT
STORE OBJECT ONLY
P1
P1

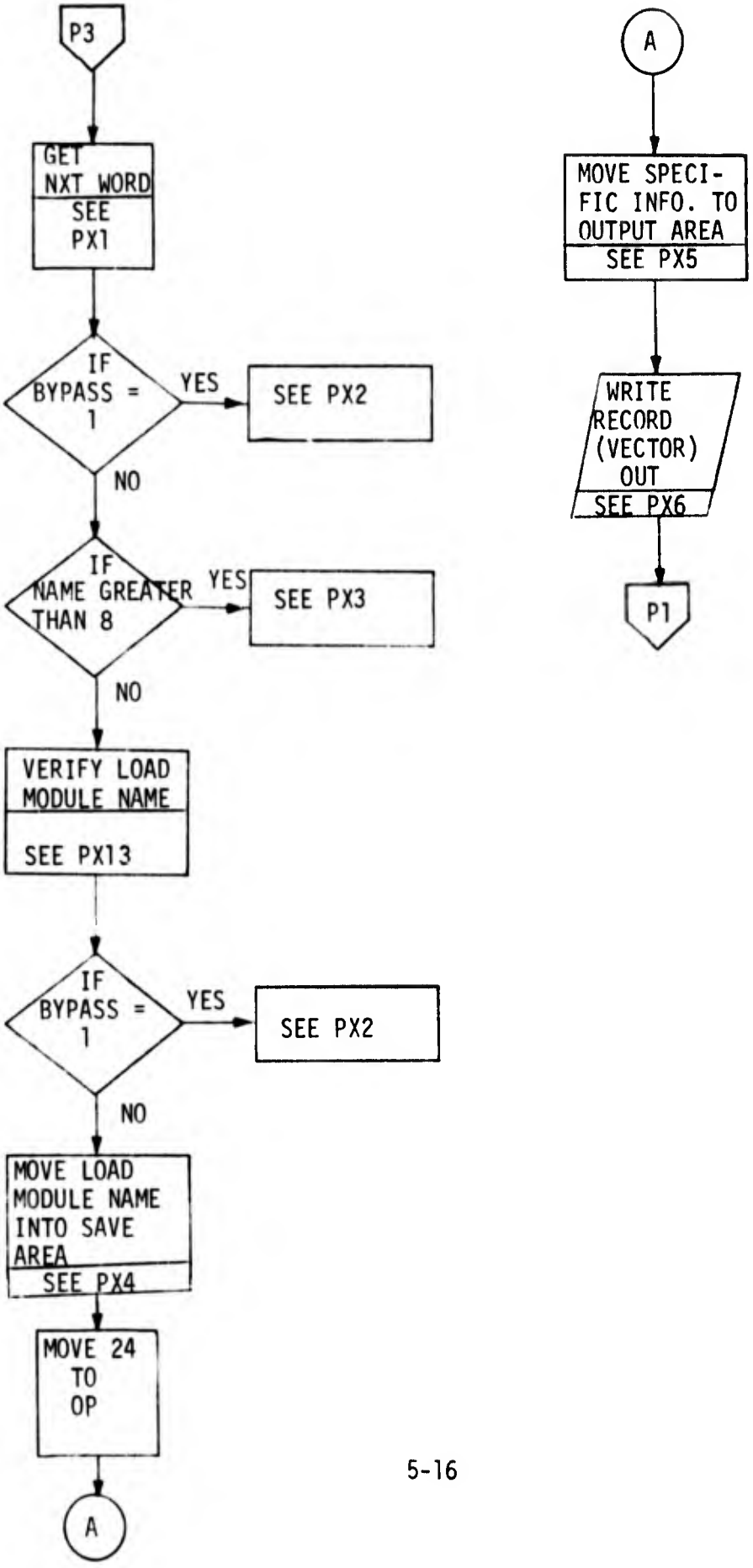
(3) GEN1A.

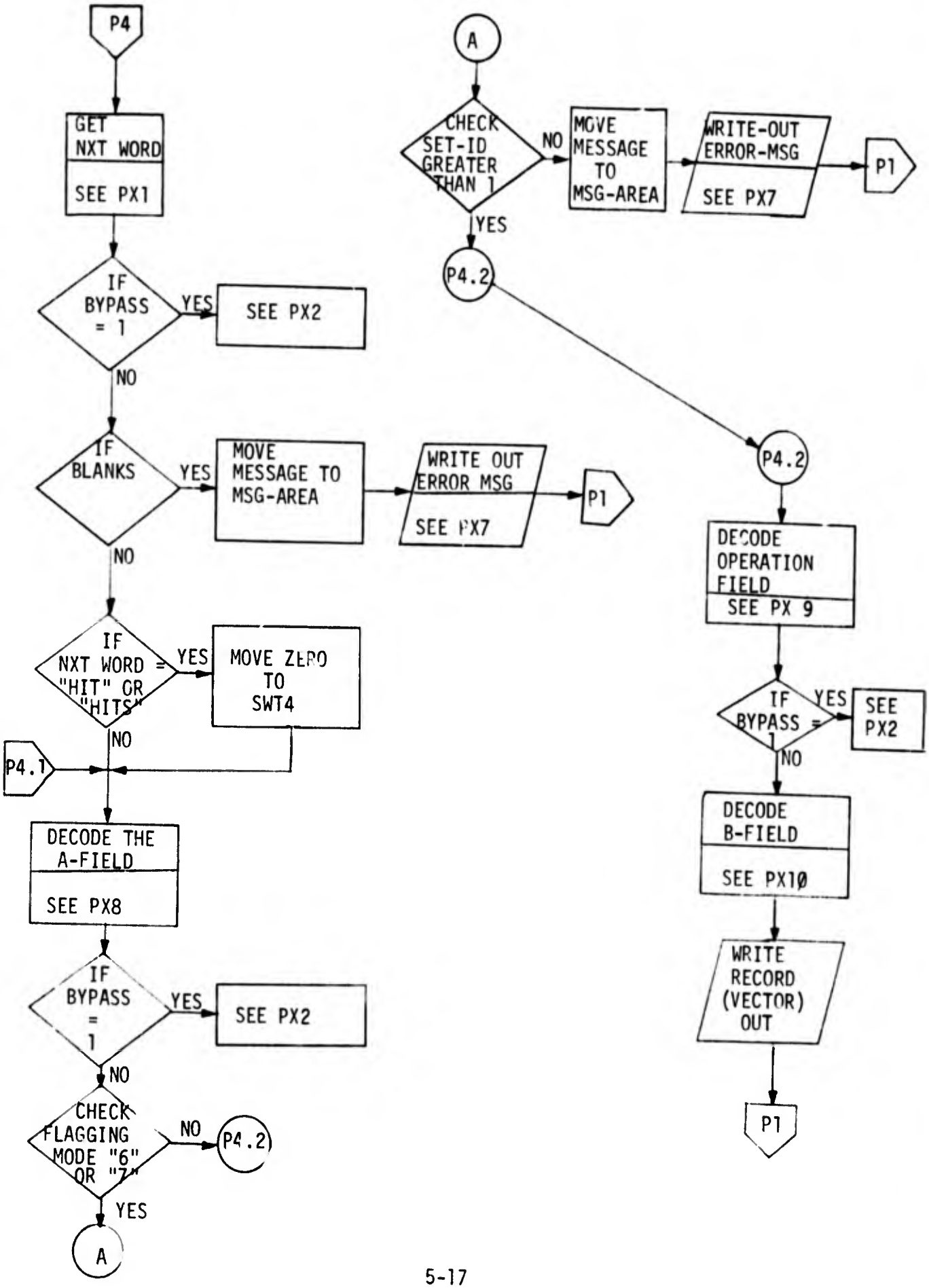

ENTRY
GEN1A
INITIALIZE
SORT ERROR
SWITCH
SORT
WORK
FILE
SORT
OK
YES
NO
DISPLAY
ERROR
MESSAGE
SET SORT
ERROR
SWITCH
RETURN

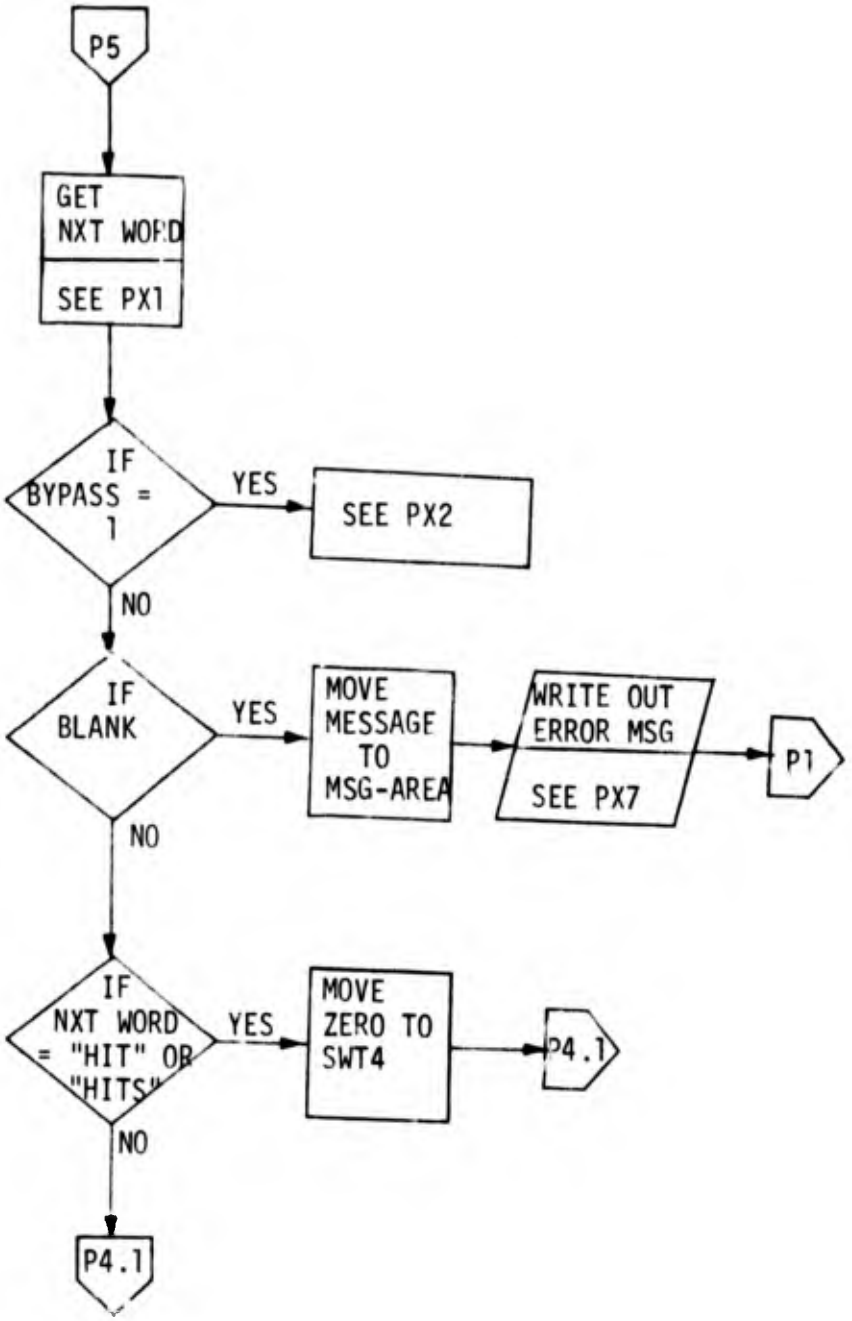
(4) GEN2.

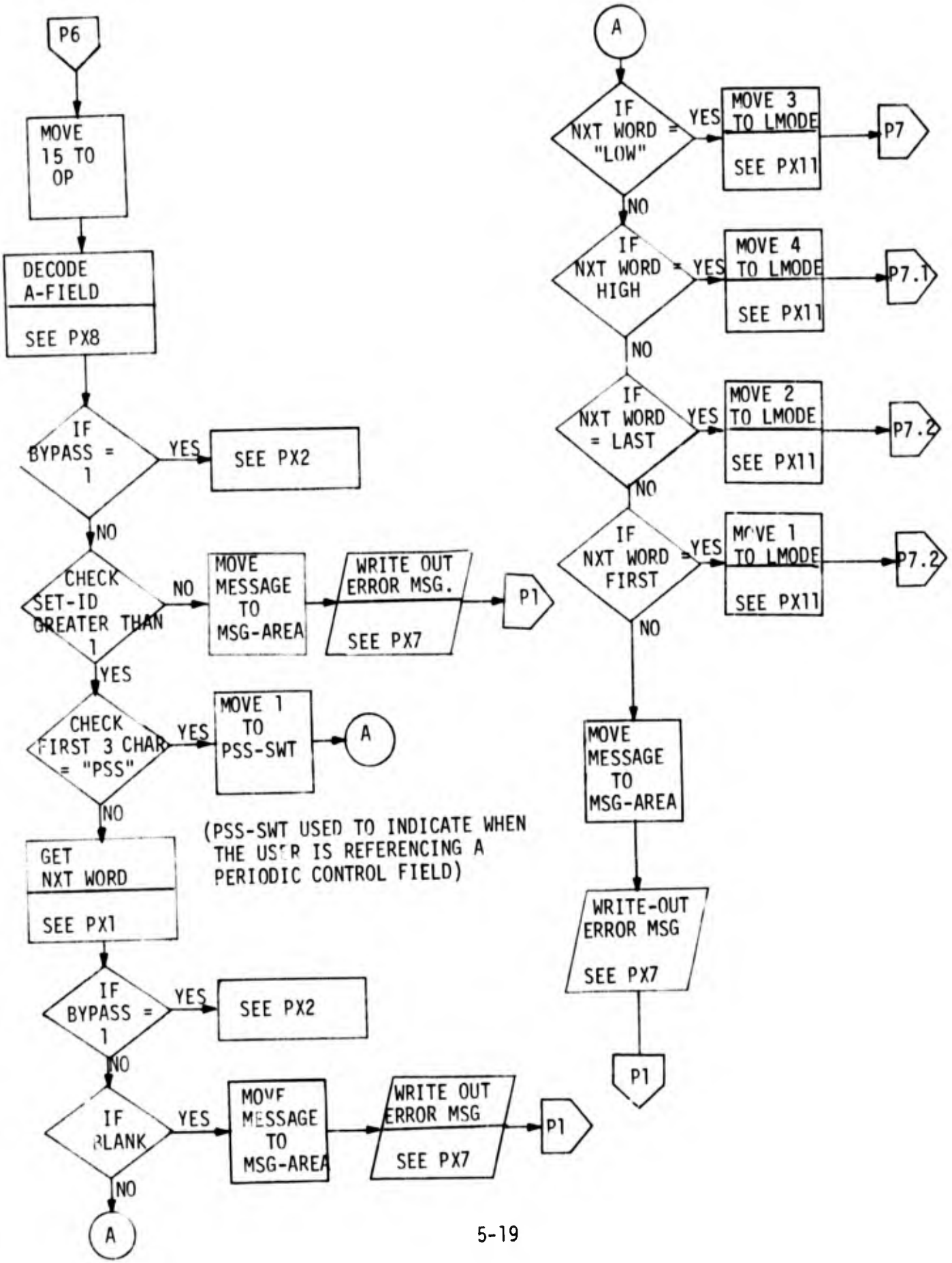

SOURCE STATEMENTS ON DISK
HOUSEKEEPING
INITIALIZE WORK AREAS, SWITCHES
P1
READ FILE-IN
AT END OF PROCESSING
YES
P26
NO
IF FILE NAME LESS THAN SPACES
YES
FATAL-ERROR STOP RUN
NO
WRITE SOURCE RECORD OUT
A
A
IF TYPE NOT EQ 1
YES
P23
NO
IF SUBQUERY-IN = ZERO
YES
P1
NO
IF SUBQUERY # QUERY # FILE NAME CHANGES
YES
P25
PROCESS CHANGE
NO
PØ
INITIALIZE SWITCHES & WORK AREAS
P2

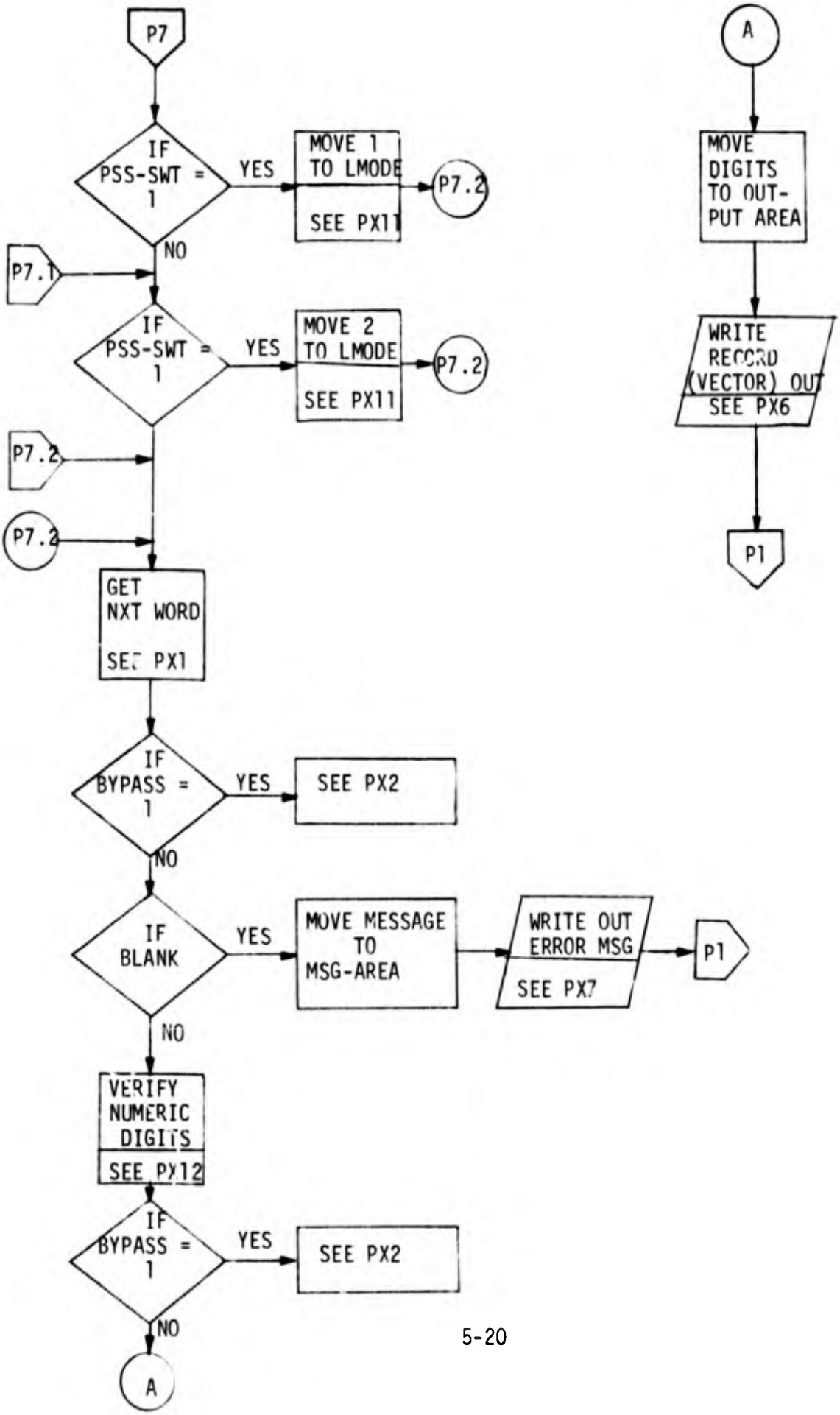

P2
PICK UP FIRST WORD
SEE PX1
IF FIRST WORD EQUALS
LOAD
YES
P3
NO
ANY
YES
P4
NO
ALL
YES
P5
NO
LIB OR LIBRARY
YES
P1
NO
FLAG
YES
P6
NO
QUERY
YES
P1
NO
DEFINE
YES
P8
NO
A
A
MODIFY
YES
P1
NO
SEARCH
YES
P10
NO
RECEIVE
YES
P11
NO
SUBQUERY
YES
P13
NO
IF(
YES
P15.1
NO
IF & AND
YES
P14
NO
OR( OR OR
YES
P16
NO
AND(
YES
P15.1
NO
SORT
YES
P16.1
NO
B
B
MERGE
YES
P18
NO
SELECT
YES
P18.1
NO
STOP
YES
P19
NO
SUMMARY
YES
P20
NO
LIMIT
YES
P21
NO
KEEP
YES
P22
NO
MOVE MESSAGE TO MSG-AREA
WRITE OUT ERROR MSG
SEE PX7
P1

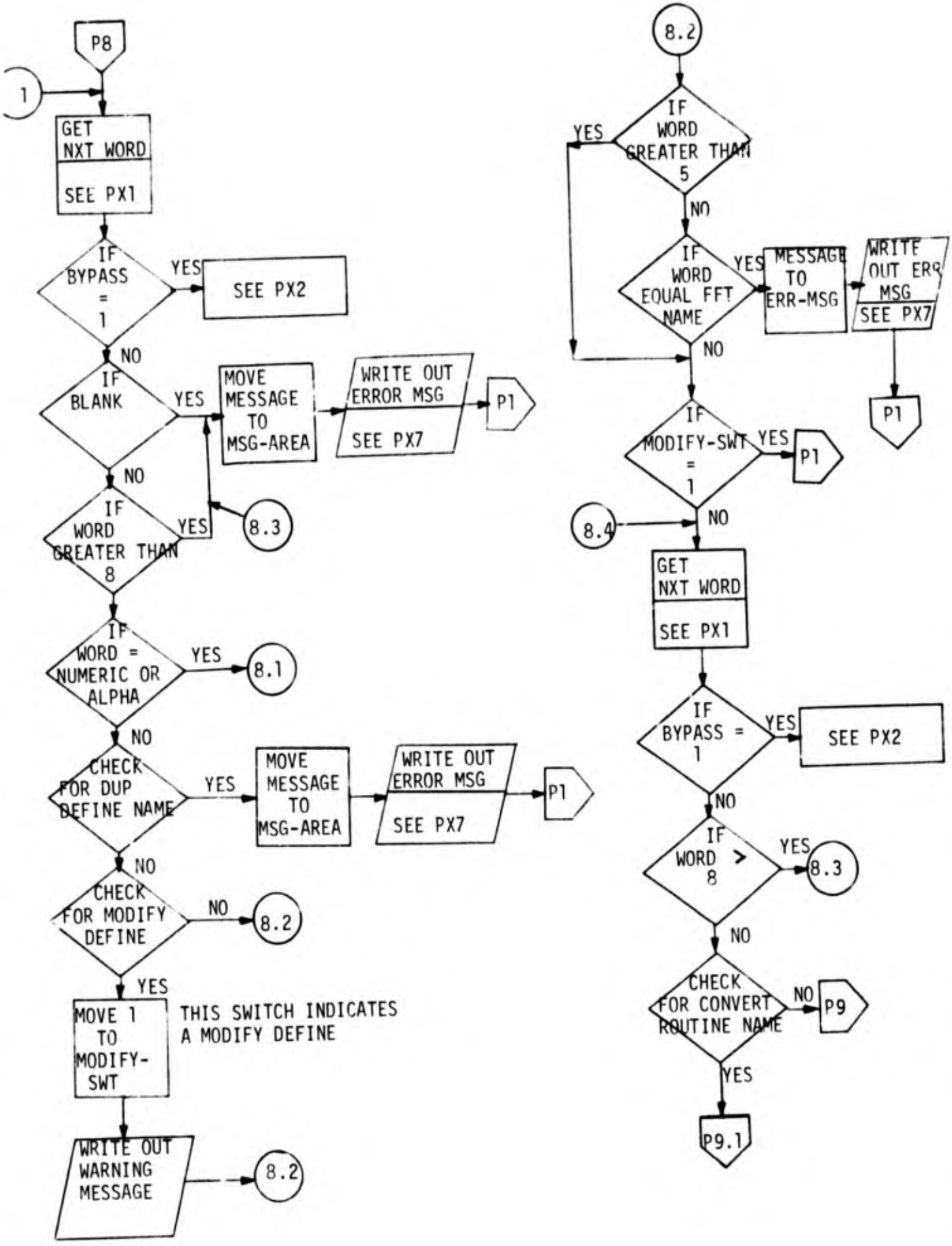

P3
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF NAME GREATER THAN 8
YES
SEE PX3
NO
VERIFY LOAD MODULE NAME
SEE PX13
IF BYPASS = 1
YES
SEE PX2
NO
MOVE LOAD MODULE NAME INTO SAVE AREA
SEE PX4
MOVE 24 TO OP
A
A
MOVE SPECI-FIC INFO. TO OUTPUT AREA
SEE PX5
WRITE RECORD (VECTOR) OUT
SEE PX6
P1

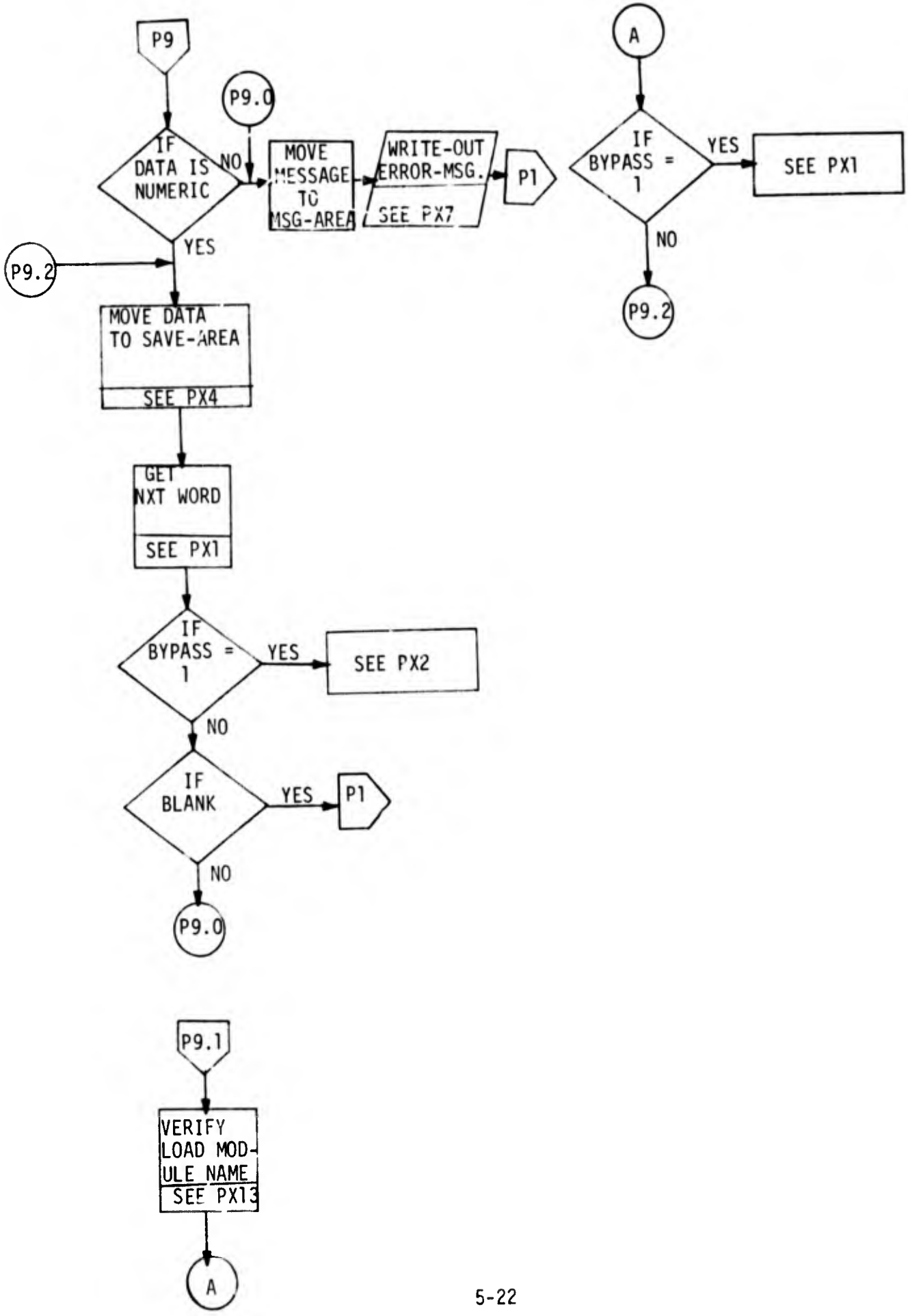

P4
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF BLANKS
YES
MOVE MESSAGE TO MSG-AREA
WRITE OUT ERROR MSG
SEE PX7
P1
NO
IF NXT WORD = "HIT" OR "HITS"
YES
MOVE ZERO TO SWT4
NO
P4.1
DECODE THE A-FIELD
SEE PX8
IF BYPASS = 1
YES
SEE PX2
NO
CHECK FLAGGING MODE "6" OR "7"
NO
P4.2
YES
A
A
CHECK SET-ID GREATER THAN 1
NO
MOVE MESSAGE TO MSG-AREA
WRITE-OUT ERROR-MSG
SEE PX7
P1
YES
P4.2
P4.2
DECODE OPERATION FIELD
SEE PX 9
IF BYPASS = 1
YES
SEE PX2
NO
DECODE B-FIELD
SEE PX10
WRITE RECORD (VECTOR) OUT
P1

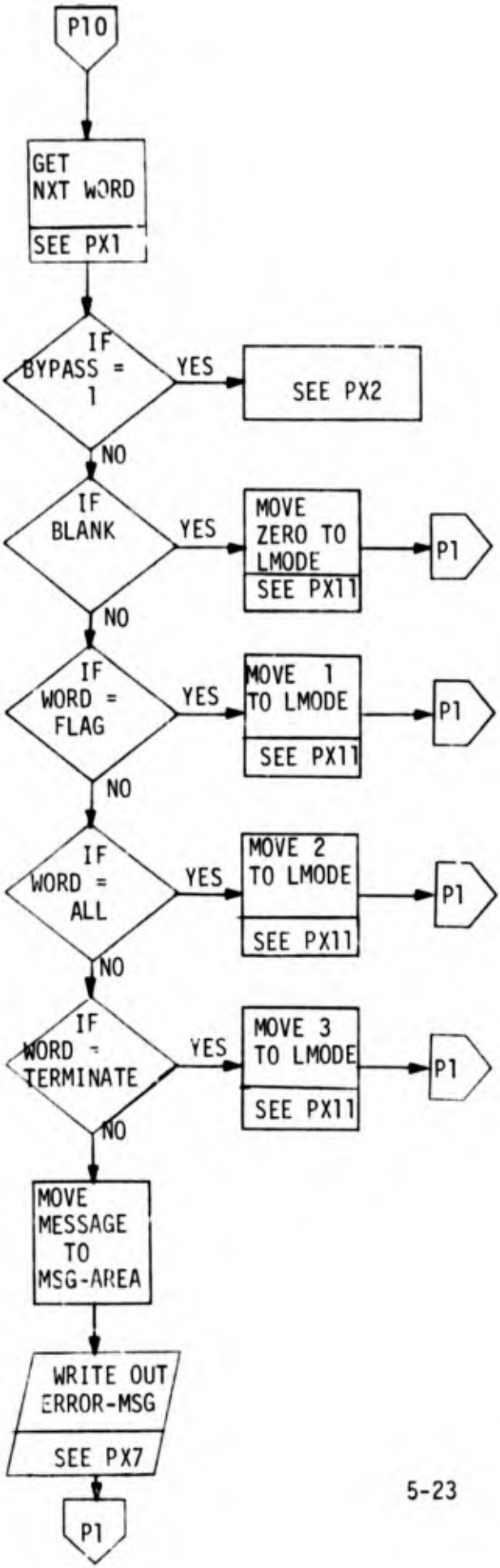

P5
GET
NXT WORD
SEE PX1
IF
BYPASS =
1
YES
SEE PX2
NO
IF
BLANK
YES
MOVE
MESSAGE
TO
MSG-AREA
WRITE OUT
ERROR MSG
SEE PX7
P1
NO
IF
NXT WORD
= "HIT" OR
"HITS"
YES
MOVE
ZERO TO
SWT4
P4.1
NO
P4.1

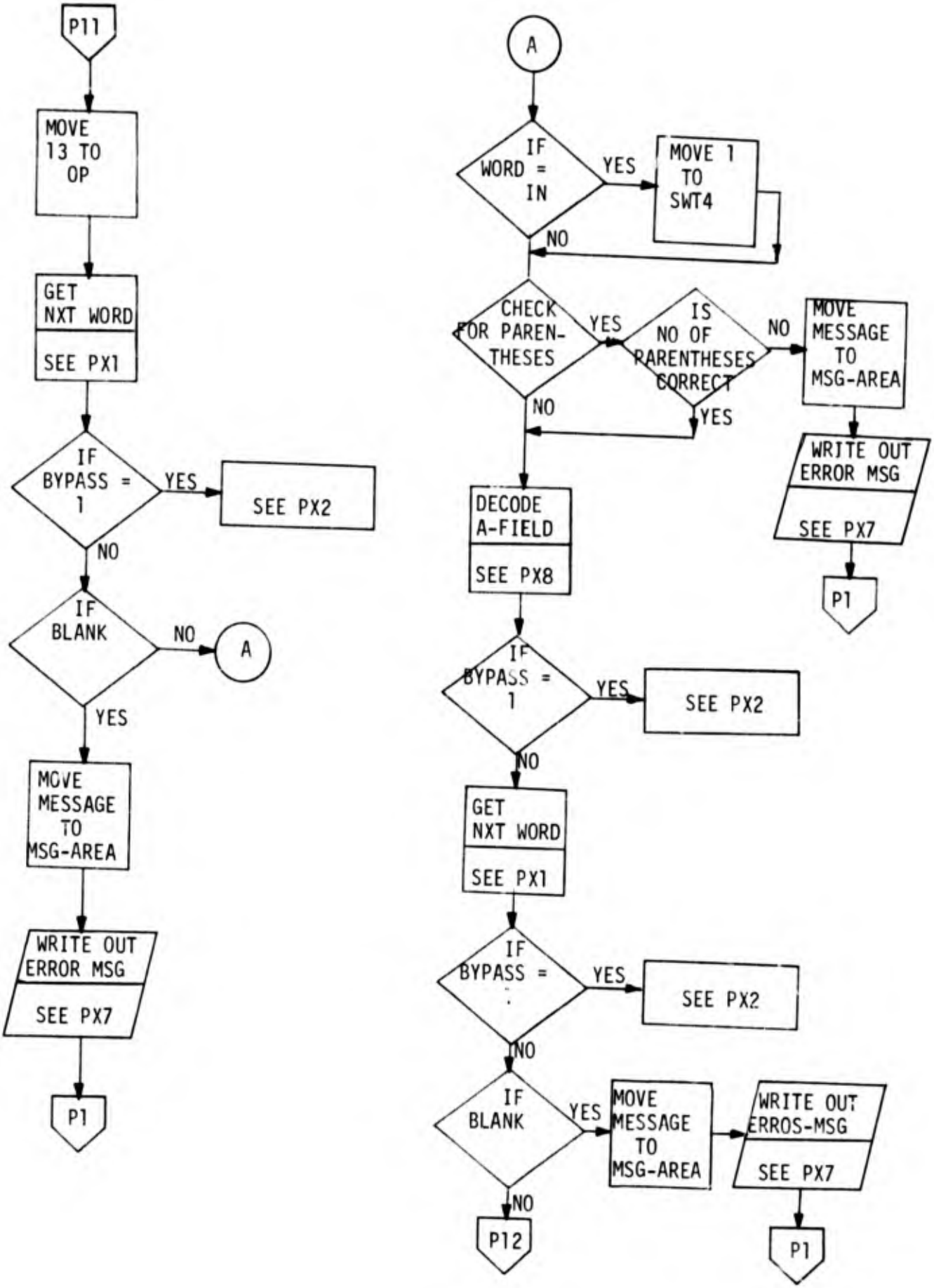

P6
MOVE 15 TO OP
DECODE A-FIELD
SEE PX8
IF BYPASS = 1
YES
SEE PX2
NO
CHECK SET-ID GREATER THAN 1
NO
MOVE MESSAGE TO MSG-AREA
WRITE OUT ERROR MSG.
SEE PX7
P1
YES
CHECK FIRST 3 CHAR = "PSS"
YES
MOVE 1 TO PSS-SWT
A
NO
(PSS-SWT USED TO INDICATE WHEN THE USER IS REFERENCING A PERIODIC CONTROL FIELD)
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF BLANK
YES
MOVE MESSAGE TO MSG-AREA
WRITE OUT ERROR MSG
SEE PX7
P1
NO
A
A
IF NXT WORD = "LOW"
YES
MOVE 3 TO LMODE
SEE PX11
P7
NO
IF NXT WORD = HIGH
YES
MOVE 4 TO LMODE
SEE PX11
P7.1
NO
IF NXT WORD = LAST
YES
MOVE 2 TO LMODE
SEE PX11
P7.2
NO
IF NXT WORD = FIRST
YES
MOVE 1 TO LMODE
SEE PX11
P7.2
NO
MOVE MESSAGE TO MSG-AREA
WRITE-OUT ERROR MSG
SEE PX7
P1

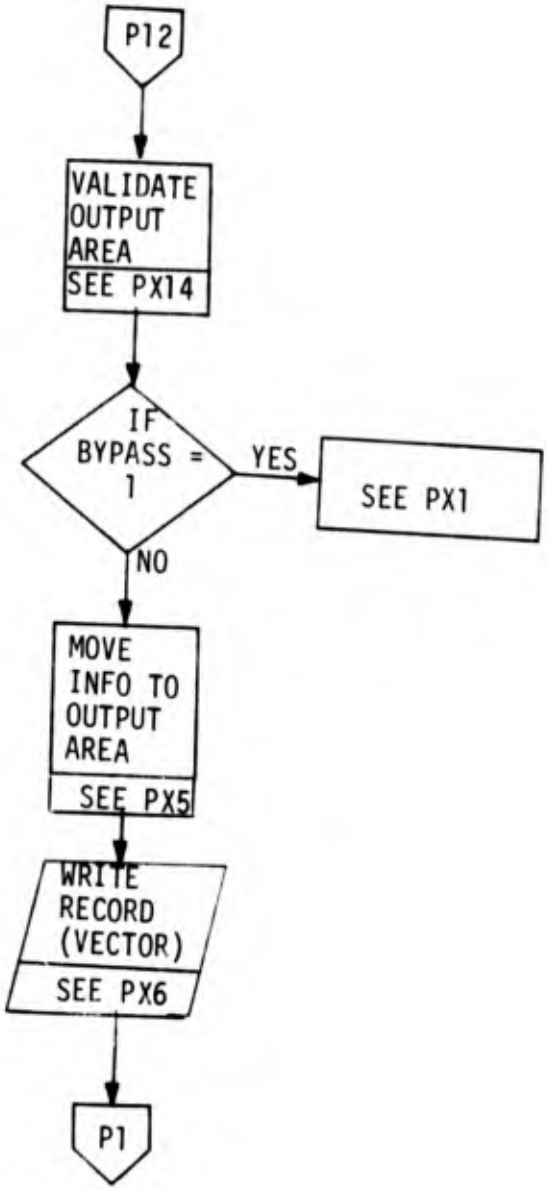

P7
IF PSS-SWT = 1
YES
MOVE 1 TO LMODE
SEE PX11
P7.2
NO
P7.1
IF PSS-SWT = 1
YES
MOVE 2 TO LMODE
SEE PX11
P7.2
P7.2
P7.2
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF BLANK
YES
MOVE MESSAGE TO MSG-AREA
WRITE OUT ERROR MSG
SEE PX7
P1
NO
VERIFY NUMERIC DIGITS
SEE PX12
IF BYPASS = 1
YES
SEE PX2
NO
A
A
MOVE DIGITS TO OUT-PUT AREA
WRITE RECORD (VECTOR) OUT
SEE PX6
P1

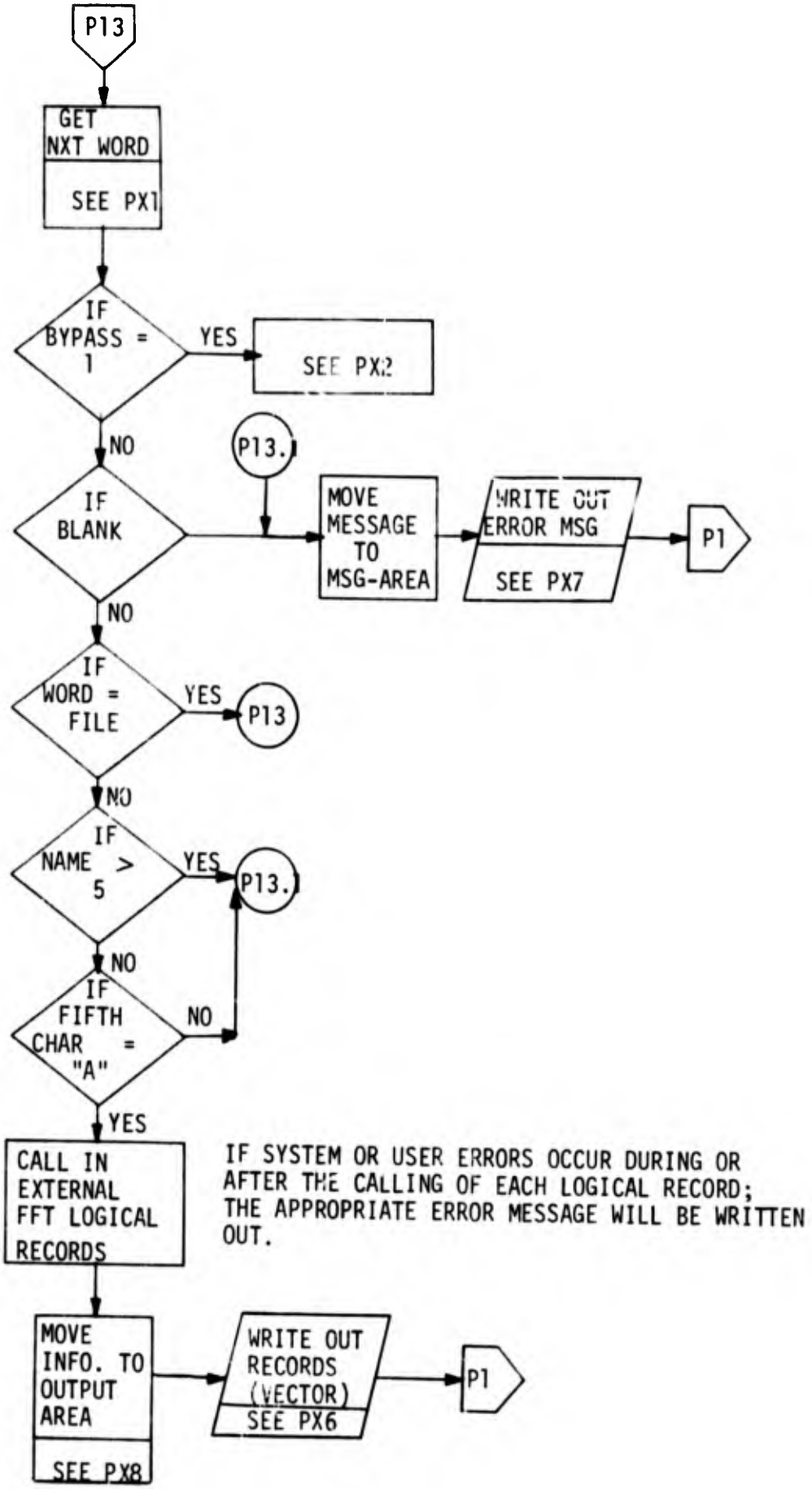

P8
1
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF BLANK
YES
MOVE MESSAGE TO MSG-AREA
WRITE OUT ERROR MSG
SEE PX7
P1
NO
IF WORD GREATER THAN 8
YES
8.3
IF WORD = NUMERIC OR ALPHA
YES
8.1
NO
CHECK FOR DUP DEFINE NAME
YES
MOVE MESSAGE TO MSG-AREA
WRITE OUT ERROR MSG
SEE PX7
P1
NO
CHECK FOR MODIFY DEFINE
NO
8.2
YES
MOVE 1 TO MODIFY-SWT
THIS SWITCH INDICATES A MODIFY DEFINE
WRITE OUT WARNING MESSAGE
8.2
8.2
IF WORD GREATER THAN 5
YES
NO
IF WORD EQUAL FFT NAME
YES
MESSAGE TO ERR-MSG
WRITE OUT ERR MSG
SEE PX7
P1
NO
IF MODIFY-SWT = 1
YES
P1
8.4
NO
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF WORD > 8
YES
8.3
NO
CHECK FOR CONVERT ROUTINE NAME
NO
P9
YES
P9.1

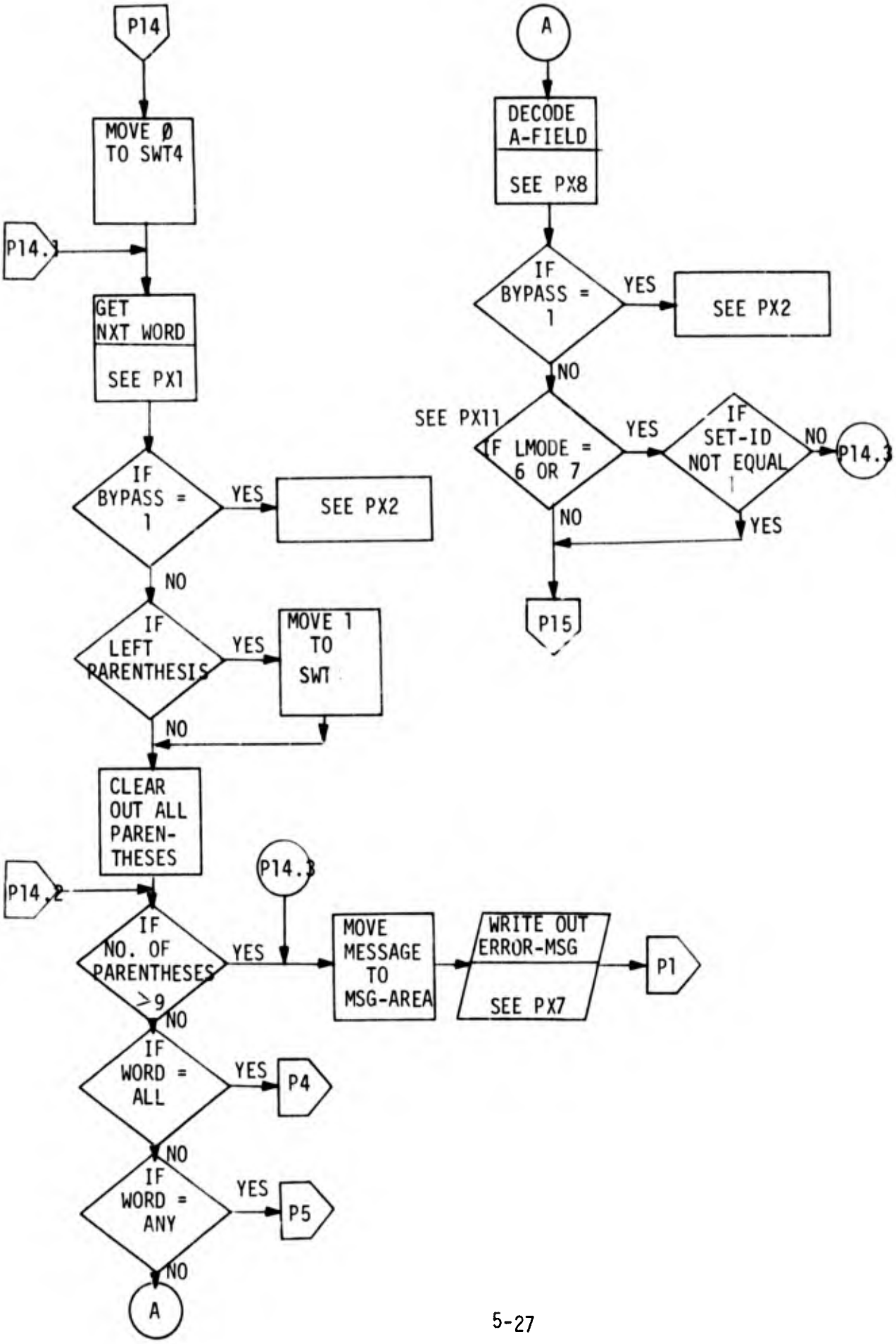

P9
P9.0
IF DATA IS NUMERIC
NO
MOVE MESSAGE TO MSG-AREA
WRITE-OUT ERROR-MSG.
SEE PX7
P1
A
IF BYPASS = 1
YES
SEE PX1
NO
P9.2
YES
P9.2
MOVE DATA TO SAVE-AREA
SEE PX4
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF BLANK
YES
P1
NO
P9.0
P9.1
VERIFY LOAD MOD-ULE NAME
SEE PX13
A

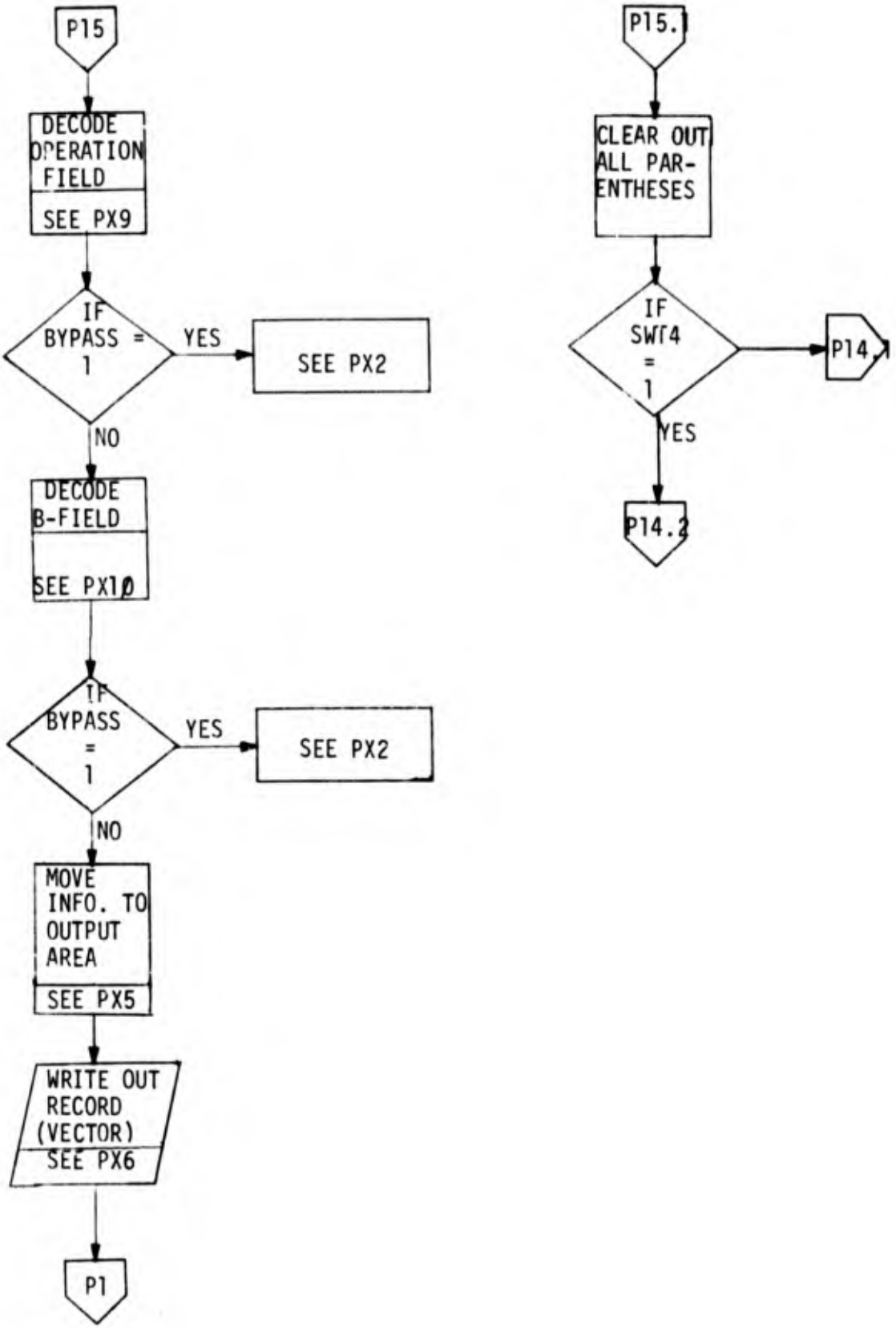

P10
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF BLANK
YES
MOVE ZERO TO LMODE
SEE PX11
P1
NO
IF WORD = FLAG
YES
MOVE 1 TO LMODE
SEE PX11
P1
NO
IF WORD = ALL
YES
MOVE 2 TO LMODE
SEE PX11
P1
NO
IF WORD = TERMINATE
YES
MOVE 3 TO LMODE
SEE PX11
P1
NO
MOVE MESSAGE TO MSG-AREA
WRITE OUT ERROR-MSG
SEE PX7
P1

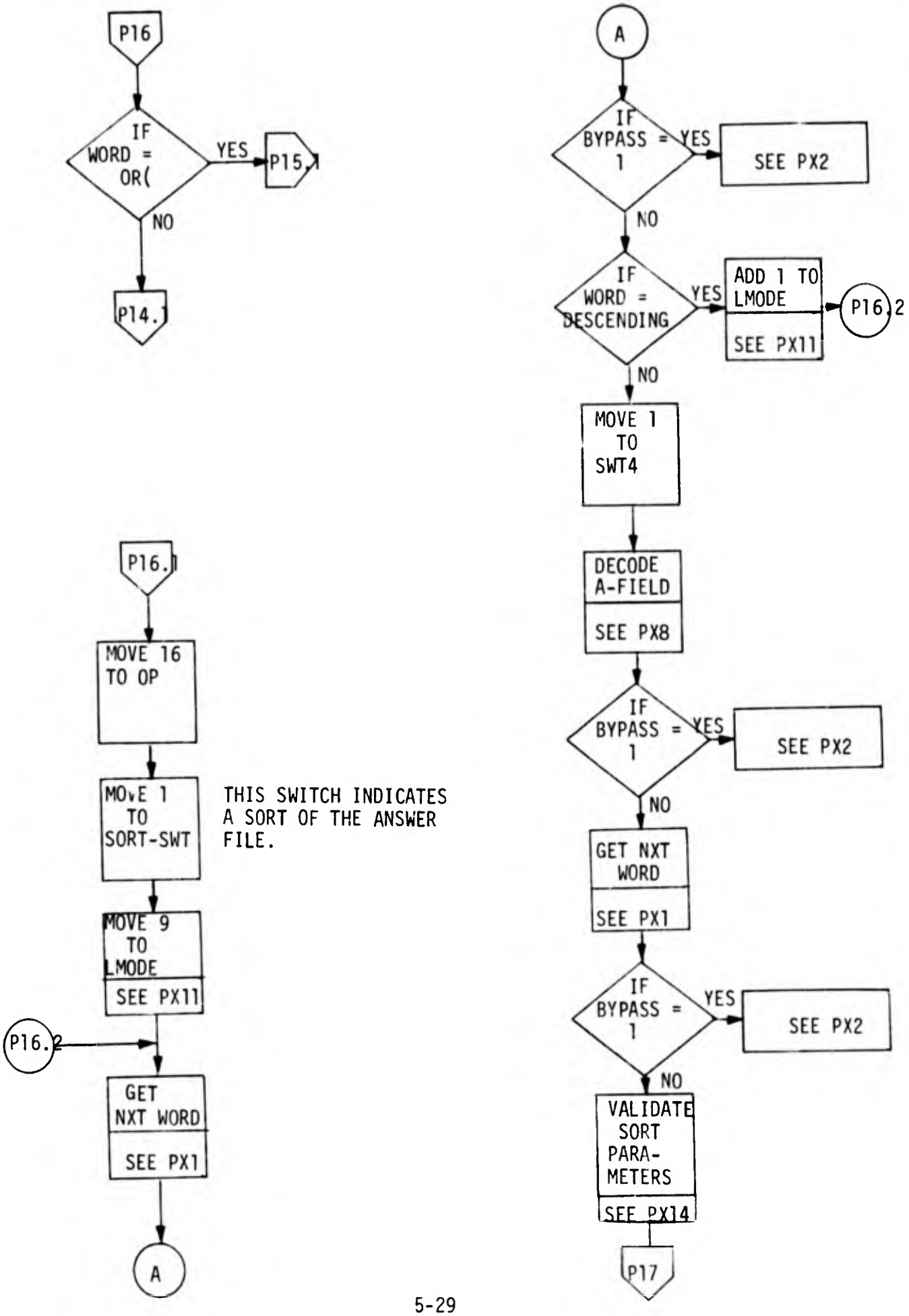

P11
MOVE 13 TO OP
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF BLANK
NO
A
YES
MOVE MESSAGE TO MSG-AREA
WRITE OUT ERROR MSG
SEE PX7
P1
A
IF WORD = IN
YES
MOVE 1 TO SWT4
NO
CHECK FOR PAREN-THESES
YES
IS NO OF PARENTHESES CORRECT
NO
MOVE MESSAGE TO MSG-AREA
YES
NO
WRITE OUT ERROR MSG
SEE PX7
P1
DECODE A-FIELD
SEE PX8
IF BYPASS = 1
YES
SEE PX2
NO
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF BLANK
YES
MOVE MESSAGE TO MSG-AREA
WRITE OUT ERROS-MSG
SEE PX7
P1
NO
P12


P12
VALIDATE OUTPUT AREA
SEE PX14
IF BYPASS = 1
YES
SEE PX1
NO
MOVE INFO TO OUTPUT AREA
SEE PX5
WRITE RECORD (VECTOR)
SEE PX6
P1

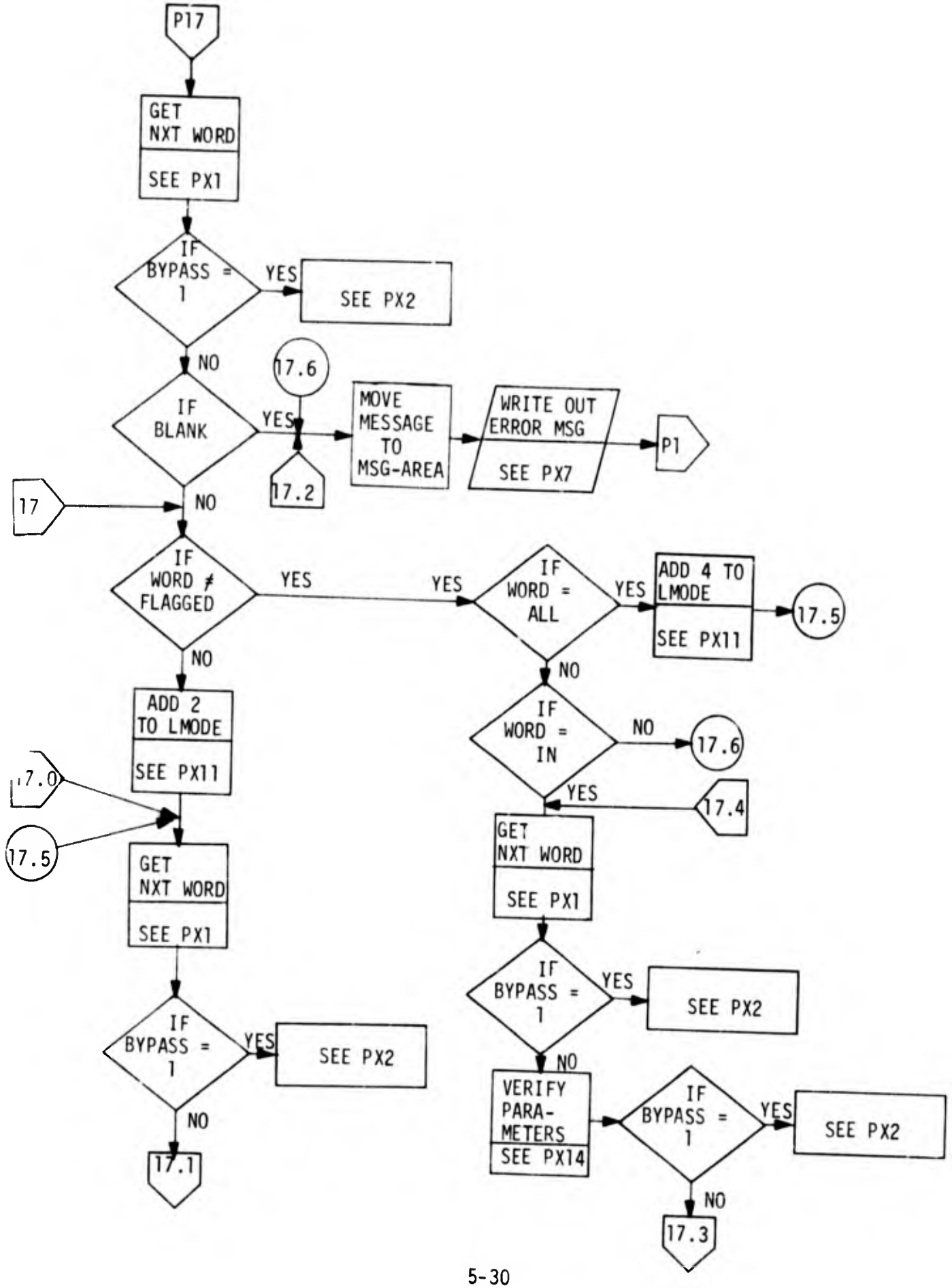

P13
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
P13.1
IF BLANK
MOVE MESSAGE TO MSG-AREA
WRITE OUT ERROR MSG
SEE PX7
P1
NO
IF WORD = FILE
YES
P13
NO
IF NAME > 5
YES
P13.1
NO
IF FIFTH CHAR = "A"
NO
YES
CALL IN EXTERNAL FFT LOGICAL RECORDS
IF SYSTEM OR USER ERRORS OCCUR DURING OR AFTER THE CALLING OF EACH LOGICAL RECORD; THE APPROPRIATE ERROR MESSAGE WILL BE WRITTEN OUT.
MOVE INFO. TO OUTPUT AREA
SEE PX8
WRITE OUT RECORDS (VECTOR)
SEE PX6
P1

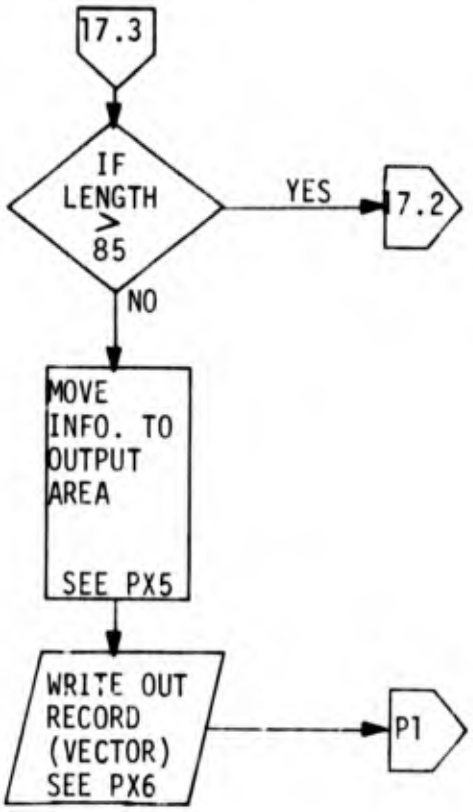

P14
MOVE Ø TO SWT4
P14.1
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF LEFT PARENTHESIS
YES
MOVE 1 TO SWT
NO
CLEAR OUT ALL PAREN-THESES
P14.2
P14.3
IF NO. OF PARENTHESES >9
YES
MOVE MESSAGE TO MSG-AREA
WRITE OUT ERROR-MSG
SEE PX7
P1
NO
IF WORD = ALL
YES
P4
NO
IF WORD = ANY
YES
P5
NO
A
A
DECODE A-FIELD
SEE PX8
IF BYPASS = 1
YES
SEE PX2
NO
SEE PX11
IF LMODE = 6 OR 7
YES
IF SET-ID NOT EQUAL 1
NO
P14.3
YES
NO
P15

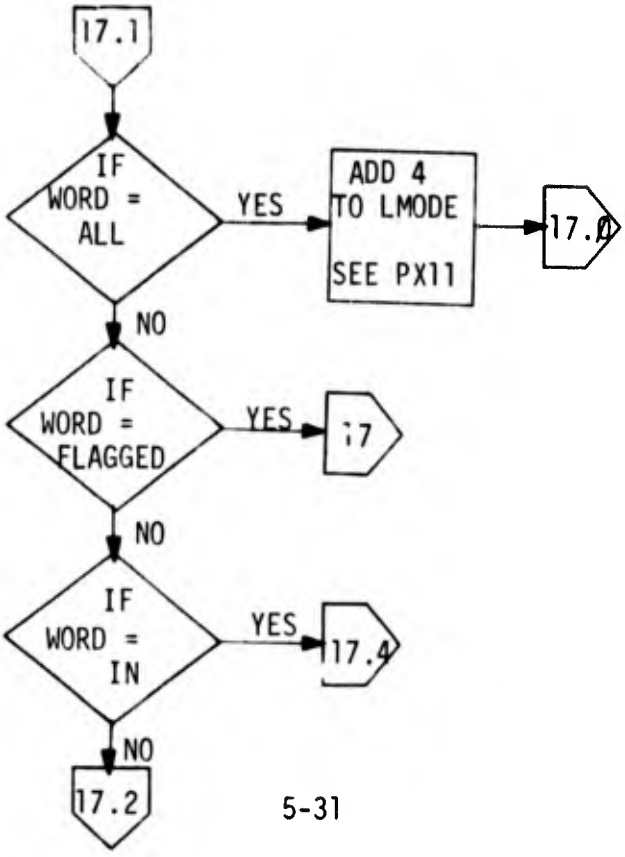

P15
DECODE OPERATION FIELD
SEE PX9
IF BYPASS = 1
YES
SEE PX2
NO
DECODE B-FIELD
SEE PX10
IF BYPASS = 1
YES
SEE PX2
NO
MOVE INFO. TO OUTPUT AREA
SEE PX5
WRITE OUT RECORD (VECTOR)
SEE PX6
P1
P15.1
CLEAR OUT ALL PAR-ENTHESES
IF SW14 = 1
P14.1
YES
P14.2

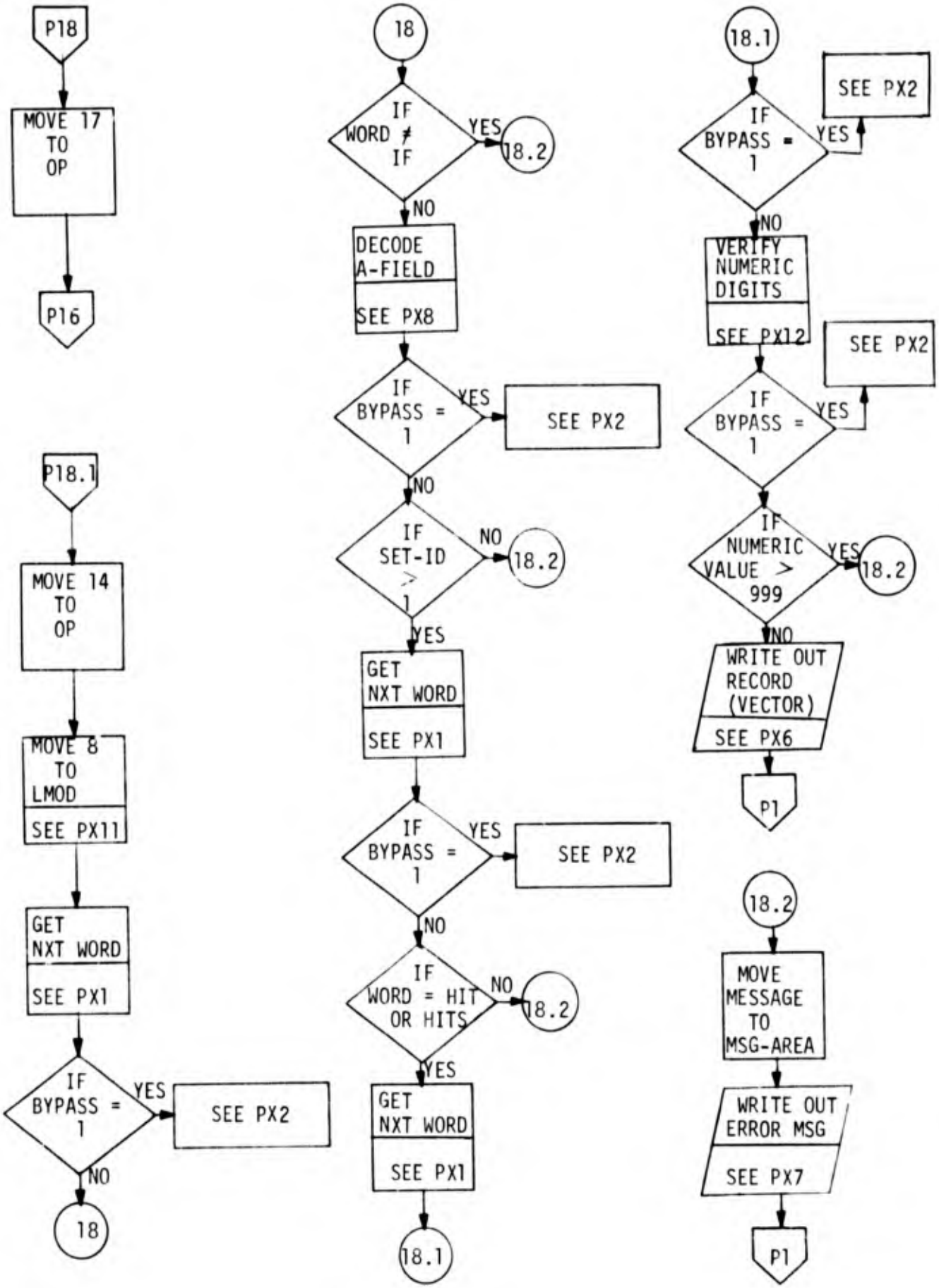


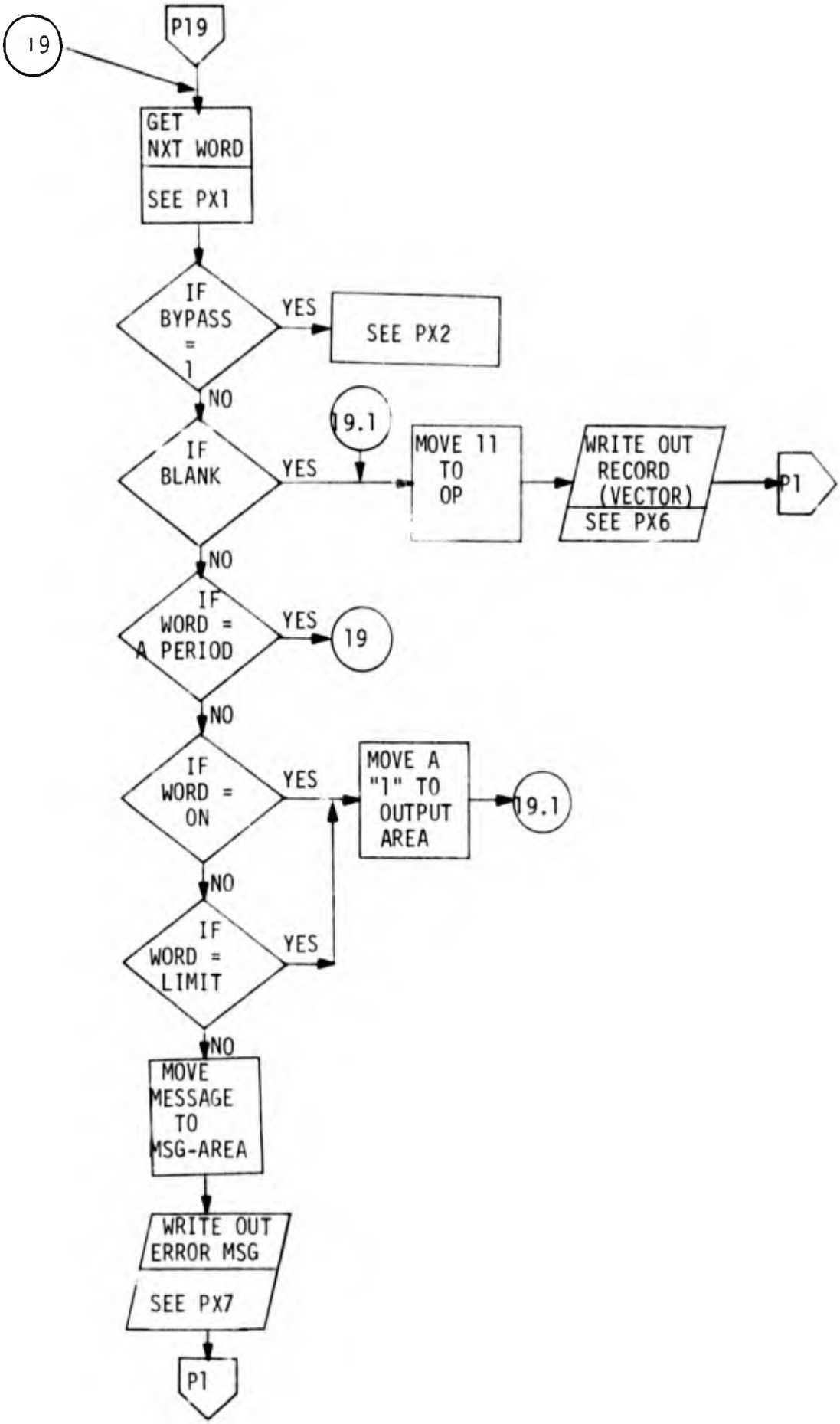

THIS SWITCH INDICATES A SORT OF THE ANSWER FILE.

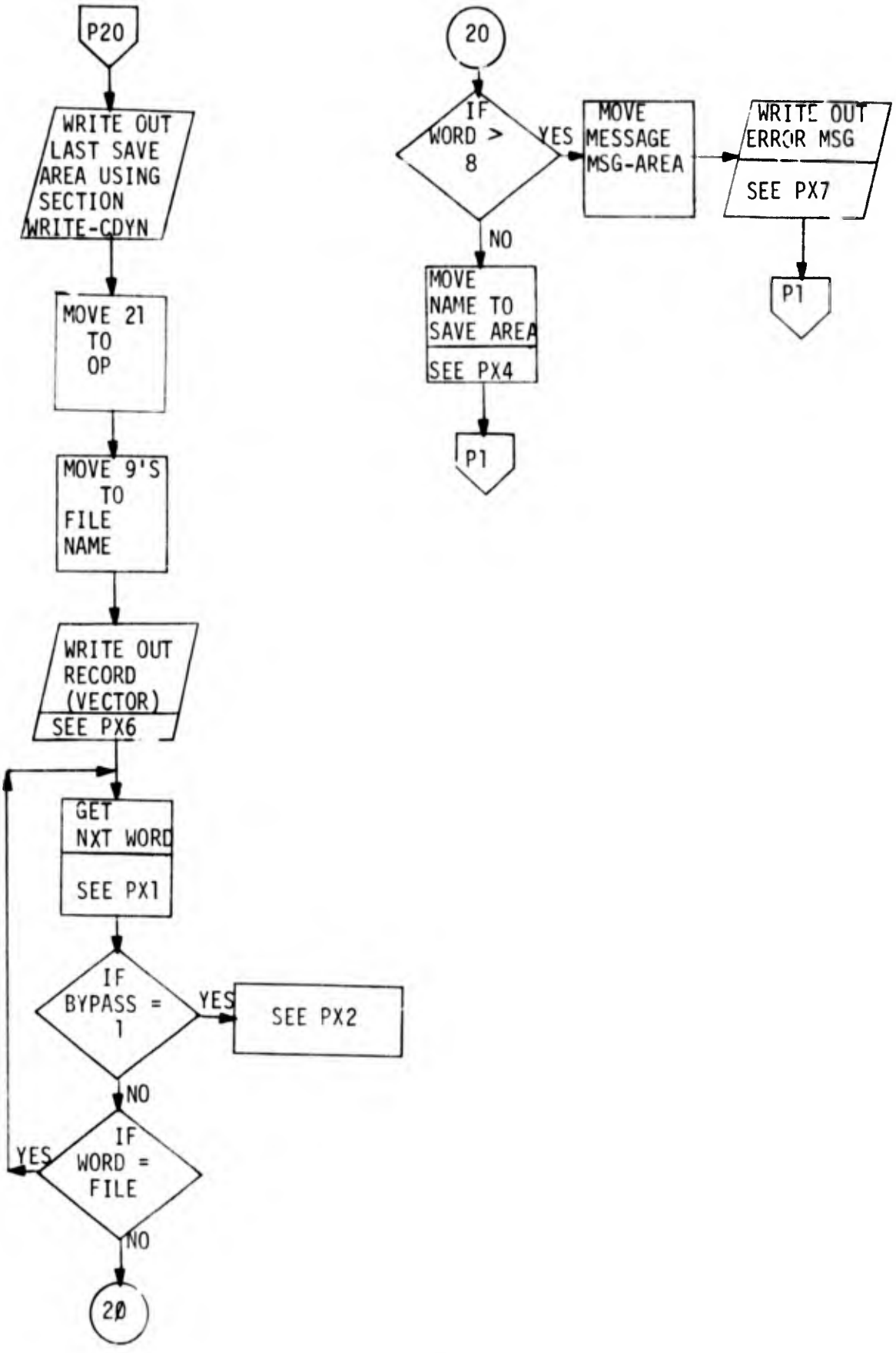

P17
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
17.6
IF BLANK
YES
MOVE MESSAGE TO MSG-AREA
WRITE OUT ERROR MSG
SEE PX7
P1
17.2
17
NO
IF WORD ≠ FLAGGED
YES
YES
IF WORD = ALL
YES
ADD 4 TO LMODE
SEE PX11
17.5
NO
ADD 2 TO LMODE
SEE PX11
IF WORD = IN
NO
17.6
17.0
17.5
YES
17.4
GET NXT WORD
SEE PX1
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF BYPASS = 1
YES
SEE PX2
VERIFY PARA-METERS
SEE PX14
IF BYPASS = 1
YES
SEE PX2
NO
17.1
NO
17.3

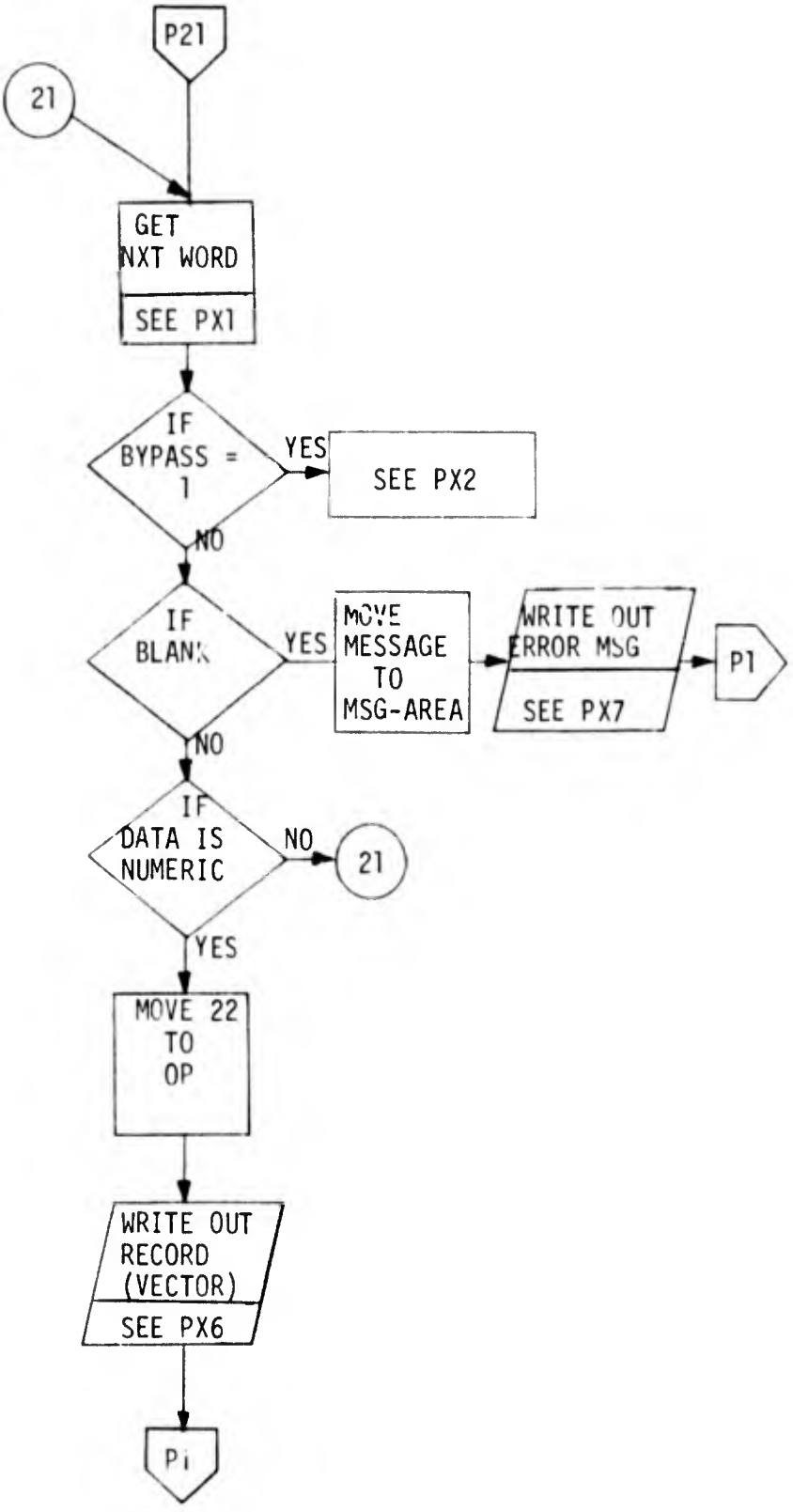

17.3
IF LENGTH > 85
YES
7.2
NO
MOVE INFO. TO OUTPUT AREA
SEE PX5
WRITE OUT RECORD (VECTOR) SEE PX6
P1

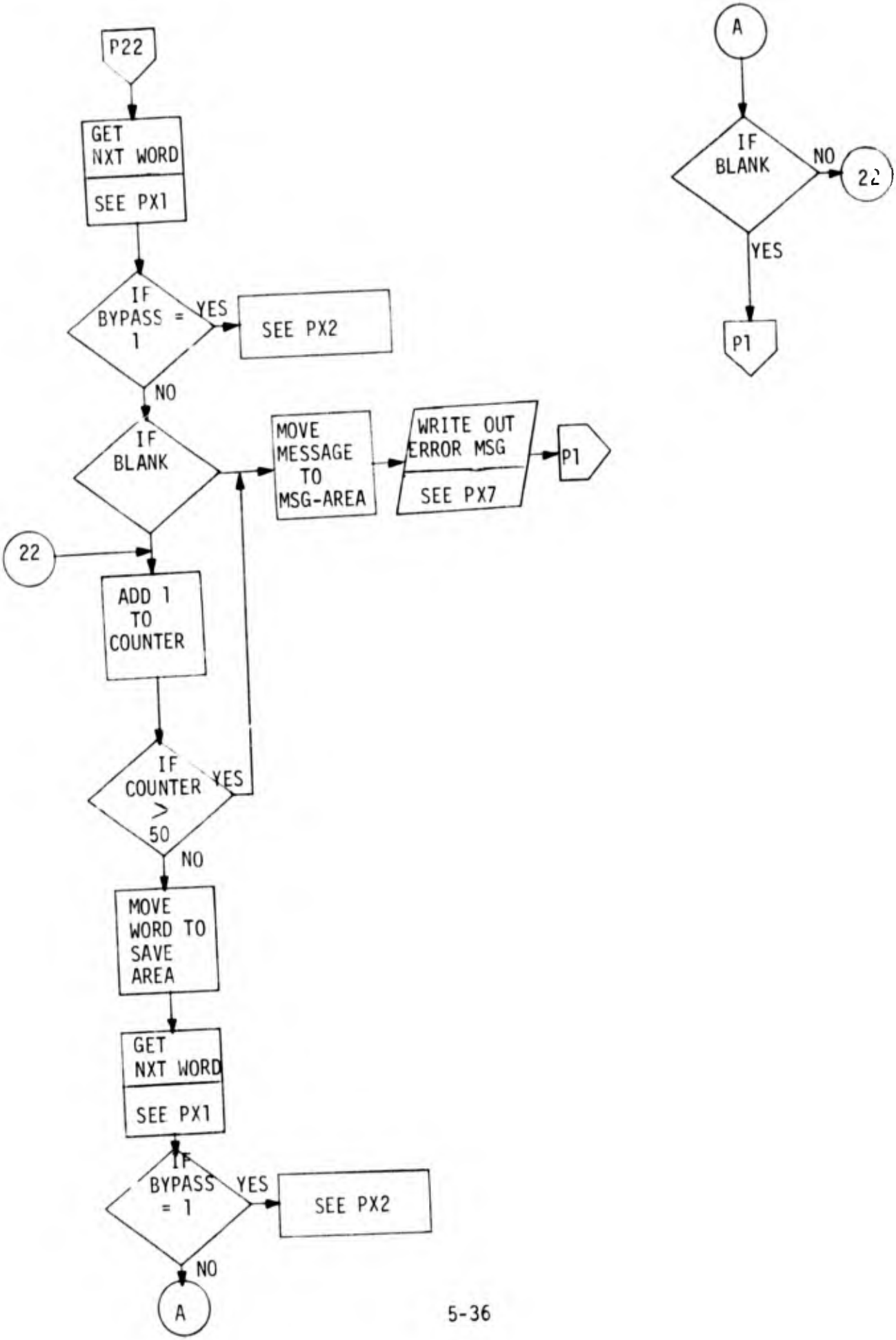

17.1
IF WORD = ALL
YES
ADD 4 TO LMODE
SEE PX11
17.0
NO
IF WORD = FLAGGED
YES
17
NO
IF WORD = IN
YES
17.4
NO
17.2

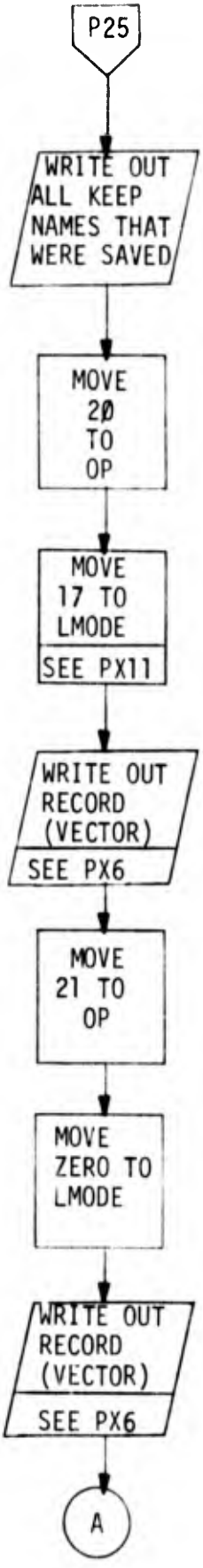

P18
MOVE 17 TO OP
P16
P18.1
MOVE 14 TO OP
MOVE 8 TO LMOD
SEE PX11
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
18
18
IF WORD ≠ IF
YES
18.2
NO
DECODE A-FIELD
SEE PX8
IF BYPASS = 1
YES
SEE PX2
NO
IF SET-ID > 1
NO
18.2
YES
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF WORD = HIT OR HITS
NO
18.2
YES
GET NXT WORD
SEE PX1
18.1
18.1
IF BYPASS = 1
YES
SEE PX2
NO
VERIFY NUMERIC DIGITS
SEE PX12
IF BYPASS = 1
YES
SEE PX2
IF NUMERIC VALUE > 999
YES
18.2
NO
WRITE OUT RECORD (VECTOR)
SEE PX6
P1
18.2
MOVE MESSAGE TO MSG-AREA
WRITE OUT ERROR MSG
SEE PX7
P1

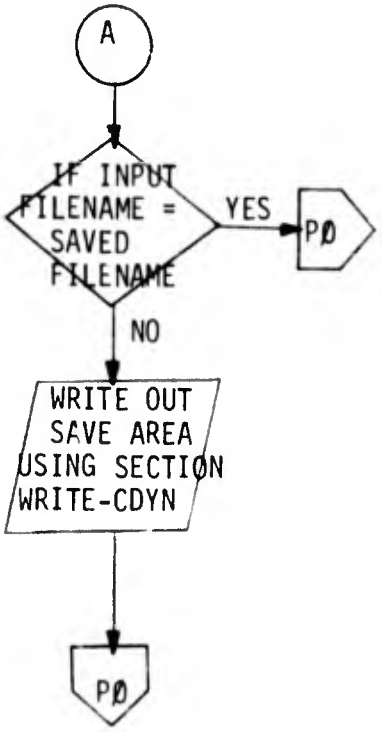

19
P19
GET
NXT WORD
SEE PX1
IF
BYPASS
=
1
YES
SEE PX2
NO
19.1
IF
BLANK
YES
MOVE 11
TO
OP
WRITE OUT
RECORD
(VECTOR)
SEE PX6
P1
NO
IF
WORD =
A PERIOD
YES
19
NO
IF
WORD =
ON
YES
MOVE A
"1" TO
OUTPUT
AREA
19.1
NO
IF
WORD =
LIMIT
YES
NO
MOVE
MESSAGE
TO
MSG-AREA
WRITE OUT
ERROR MSG
SEE PX7
P1

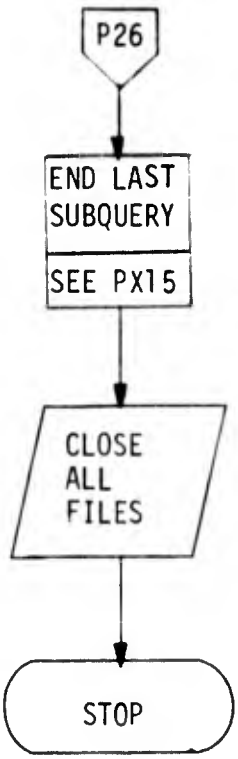

P20
WRITE OUT LAST SAVE AREA USING SECTION WRITE-CDYN
MOVE 21 TO OP
MOVE 9'S TO FILE NAME
WRITE OUT RECORD (VECTOR)
SEE PX6
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF WORD = FILE
YES
NO
20
20
IF WORD > 8
YES
MOVE MESSAGE MSG-AREA
WRITE OUT ERROR MSG
SEE PX7
P1
NO
MOVE NAME TO SAVE AREA
SEE PX4
P1

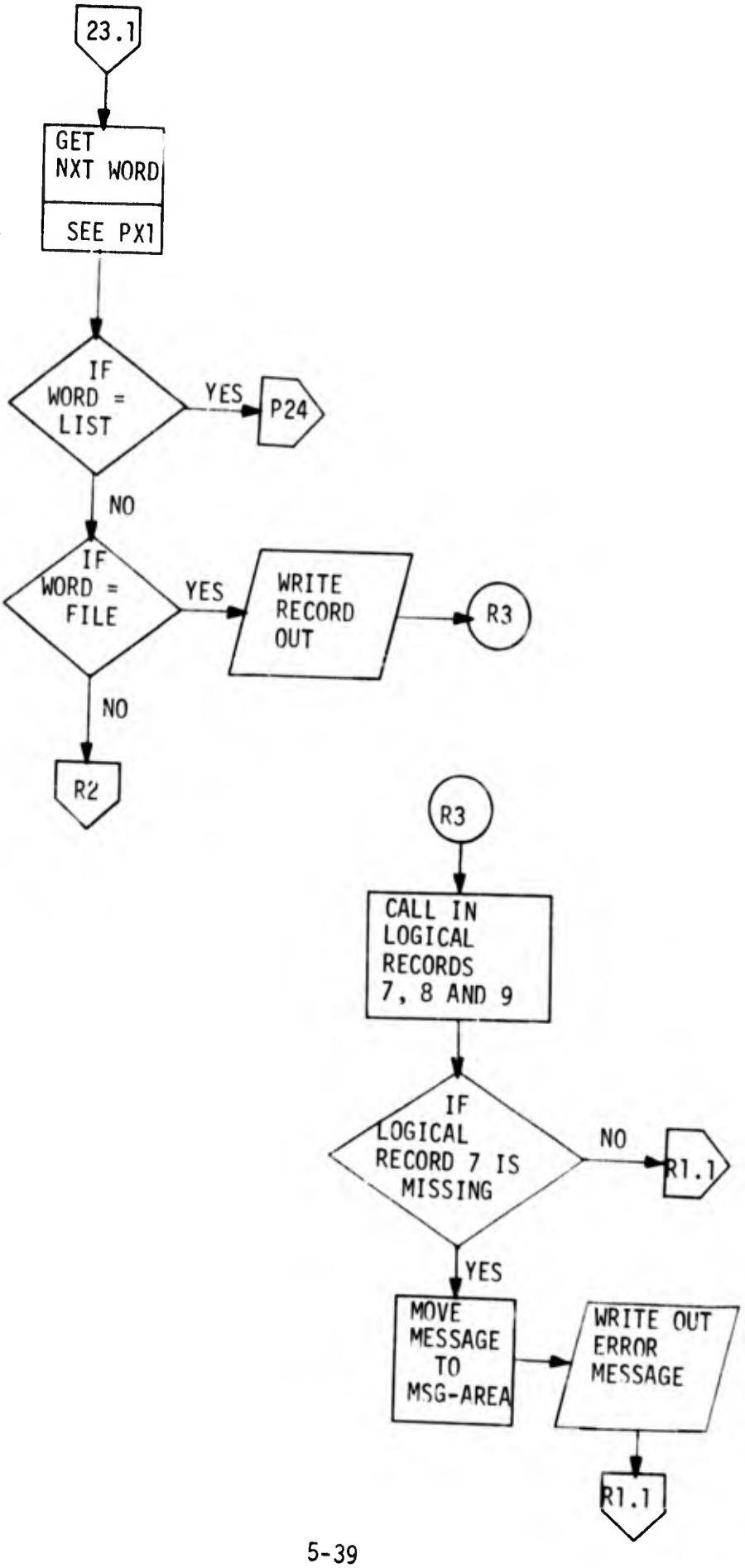

P21
21
GET
NXT WORD
SEE PX1
IF
BYPASS =
1
YES
SEE PX2
NO
IF
BLANK
YES
MOVE
MESSAGE
TO
MSG-AREA
WRITE OUT
ERROR MSG
SEE PX7
P1
NO
IF
DATA IS
NUMERIC
NO
21
YES
MOVE 22
TO
OP
WRITE OUT
RECORD
(VECTOR)
SEE PX6
Pi

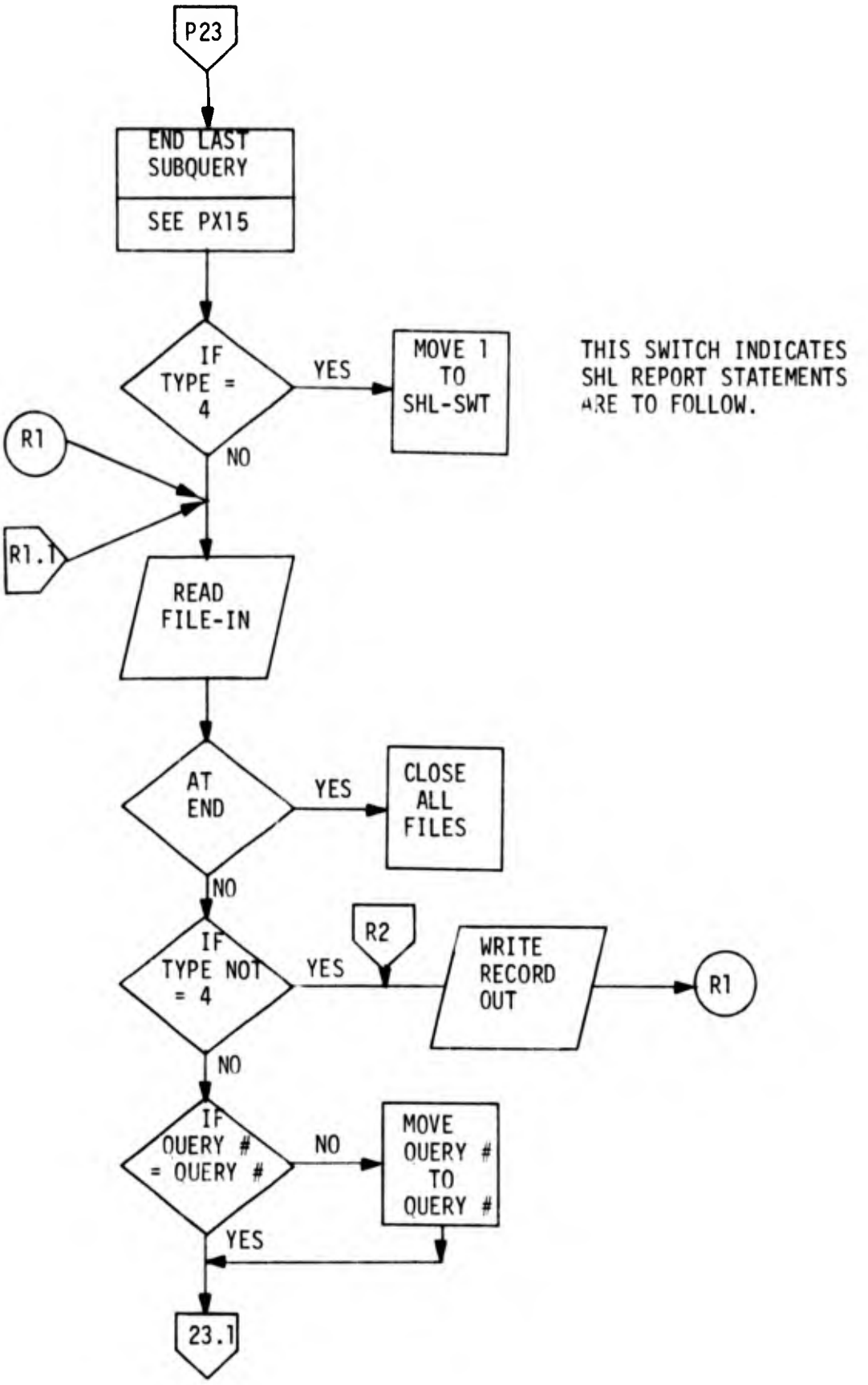

P22
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF BLANK
MOVE MESSAGE TO MSG-AREA
WRITE OUT ERROR MSG
SEE PX7
P1
22
ADD 1 TO COUNTER
IF COUNTER > 50
YES
NO
MOVE WORD TO SAVE AREA
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
A
A
IF BLANK
NO
22
YES
P1


P25
WRITE OUT ALL KEEP NAMES THAT WERE SAVED
MOVE 20 TO OP
MOVE 17 TO LMODE
SEE PX11
WRITE OUT RECORD (VECTOR)
SEE PX6
MOVE 21 TO OP
MOVE ZERO TO LMODE
WRITE OUT RECORD (VECTOR)
SEE PX6
A


A
IF INPUT FILENAME = SAVED FILENAME
YES
PØ
NO
WRITE OUT SAVE AREA USING SECTION WRITE-CDYN
PØ


P26
END LAST SUBQUERY
SEE PX15
CLOSE ALL FILES
STOP


23.1
GET
NXT WORD
SEE PX1
IF
WORD =
LIST
YES
P24
NO
IF
WORD =
FILE
YES
WRITE
RECORD
OUT
R3
NO
R2
R3
CALL IN
LOGICAL
RECORDS
7, 8 AND 9
IF
LOGICAL
RECORD 7 IS
MISSING
NO
R1.1
YES
MOVE
MESSAGE
TO
MSG-AREA
WRITE OUT
ERROR
MESSAGE
R1.1


P23
END LAST SUBQUERY
SEE PX15
IF TYPE = 4
YES
MOVE 1 TO SHL-SWT
THIS SWITCH INDICATES SHL REPORT STATEMENTS ARE TO FOLLOW.
NO
R1
R1.1
READ FILE-IN
AT END
YES
CLOSE ALL FILES
NO
IF TYPE NOT = 4
R2
YES
WRITE RECORD OUT
R1
NO
IF QUERY # = QUERY #
NO
MOVE QUERY # TO QUERY #
YES
23.1

PX1 EXTRACT-WORD Section picks up each word, letter, or number from a user's source statement and moves it to a work area called CARDW. If a continuation is required, another read will be executed using the READ-OP Section. If errors are encountered during the extraction, an appropriate error message is written out.

PX2 The bypass switch is initialized to zero before each record is processed. If an error is encountered, a one (1) is moved to this switch signifying the end of syntax checking and the writing out of the appropriate error message.

PX3 When errors occur within the processing program, an appropriate error message is moved to an error message area and then written out. In addition, a one (1) is moved to bypass switch and, depending upon the severity of the error, a decision is made as to what step the program will execute next (see PX7).

PX4 The LOAD-CONSTANT Section is used to move a character string to a work save area called CONSTANT-POOL. Its length depends upon the value in a numeric area called IL. If the CONSTANT-POOL area exceeds 90 characters, this area is written out and a new one is created using a special write section called WRITE-CDYN.

PX5 For each user's statement submitted, a record (vector) for that operation is written out containing information needed for processing by the next program. See list of vectors.

PX6 WRITE-STATE Section is used to write out records (vectors). In this section certain criteria in the vectors must be checked for each operator or operation before the vector is written out and processed by the next program. If an error is encountered the appropriate message is written out (see PX7).

PX7 WRITE-ERR and WRITE-ERR1 Sections are used to write out all error records. In addition, they move a one (1) to bypass to stop further processing of such records.

PX8 DECODE-A Section is used for decoding and syntaxing of the a-field of the user source statement. Such statements can contain a user defined name, convert routine, special operator, literal, constant, or FFT name - all of which must be decoded into pointers, locations and/or values and moved to a specified area in the record (vector). If the statement requires the extraction of another word, SWT4 will equal zero. If errors are encountered, the appropriate message will be written out.

PX9 DECODE-OP Section extracts the next word. If partial notation is encountered as the next word, Section FLD-PART performs a validity check and, if no errors are found, the proper information is placed into the a-field output area before another word is extracted. If a convert routine name or special operator is found, a verification of the name, using a section called S806-TRAP, is performed before continuing to the next word. The logical operator (e.g., EQUAL, NOT EQUAL, LESS, GREATER, etc.) is next retrieved. Upon comparing this word against a table of operators, a numeric value is placed into the operation field (OP) of the output area. Before leaving this section another word will be retrieved for further processing. All errors found will use the WRITE-ERR or WRITE-ERR1 Sections to write out the appropriate messages.

PX10 DECODE-B Section is used for syntaxing and decoding the b-field of the user source statement. Such statements can contain a FFT name, a user defined name, special operator, output area, multiple b-fields, and/or a numeric value associated with the b-field as required by the statement itself. The DECODE-B Section verifies the number of parentheses given by the user for equality using RIGHT-PARN1 Section. When numeric values are found in the b-field the EXTRACT-NUM Section is used to validate these digits. When a word in the b-field requires numeric or alphanumeric characters to be saved for further processing, the LOAD-CONSTANT Section (see PX4) is performed.

If leading zeros or trailing spaces are necessary, LEAD-ZEROS and TRAIL-SPACE are executed for that particular word. When a multiple b-field is encountered by using EXTRACT-WORD Section (see PX1), the previous record will be written out if it is not a partial notation parameter (see PX6). If partial notation is found, the FLD-PART Section is used to check its validity and, if no errors are found, the appropriate information is moved to the b-field output area. If a special operator is retrieved as the next word, the name is validated by using the S806-TRAP Section before continuing to the next word. All errors found while processing the b-field words will use the WRITE-ERR and WRITE-ERR1 Sections to write out error messages.

PX11 LMODE is a data name in the vector output area. This area contains a numeric value to indicate what type of retrieval is to be applied against the data file when it is processed by the following program.

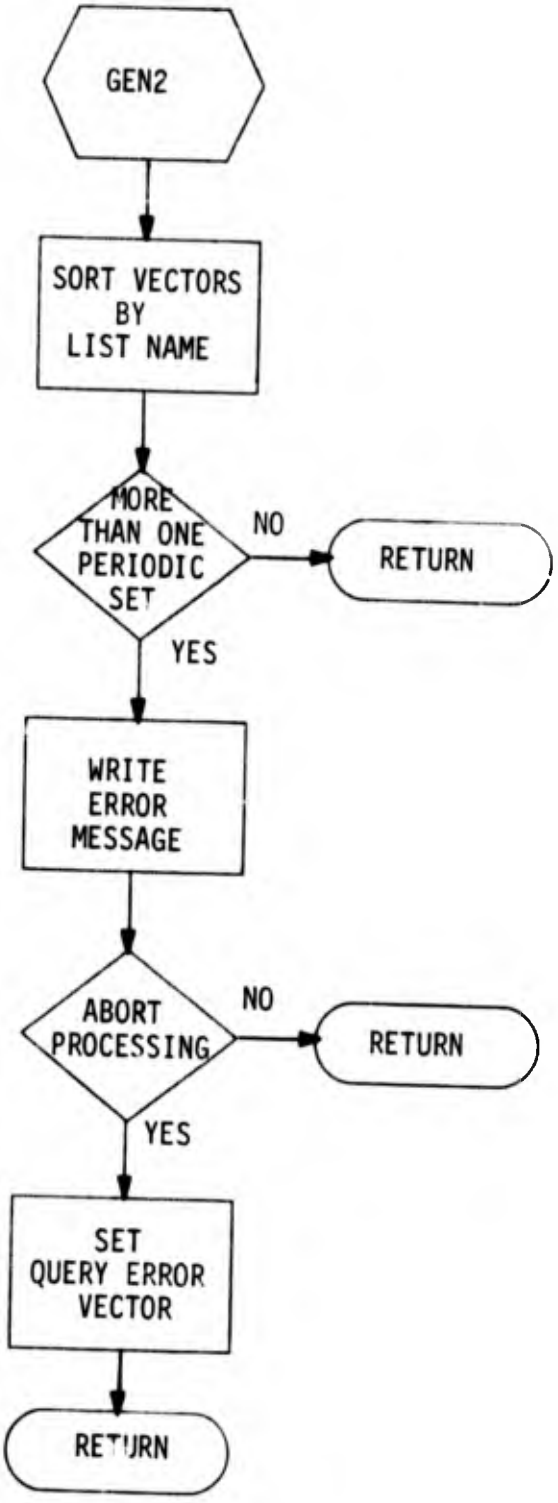
PX12 EXTRACT-NUM Section is used to verify numeric digits found in the user's source statement. If an error is found, the appropriate error message is written out.

PX13 S806-TRAP Section is used to validate that a special operator or convert routine name exists in the system library. If not found, the appropriate error message is written out.

PX14 FLD-PART Section is used to syntax the format provided by the user when validating a partial notation or an output area. If errors are found, the appropriate message is written out.

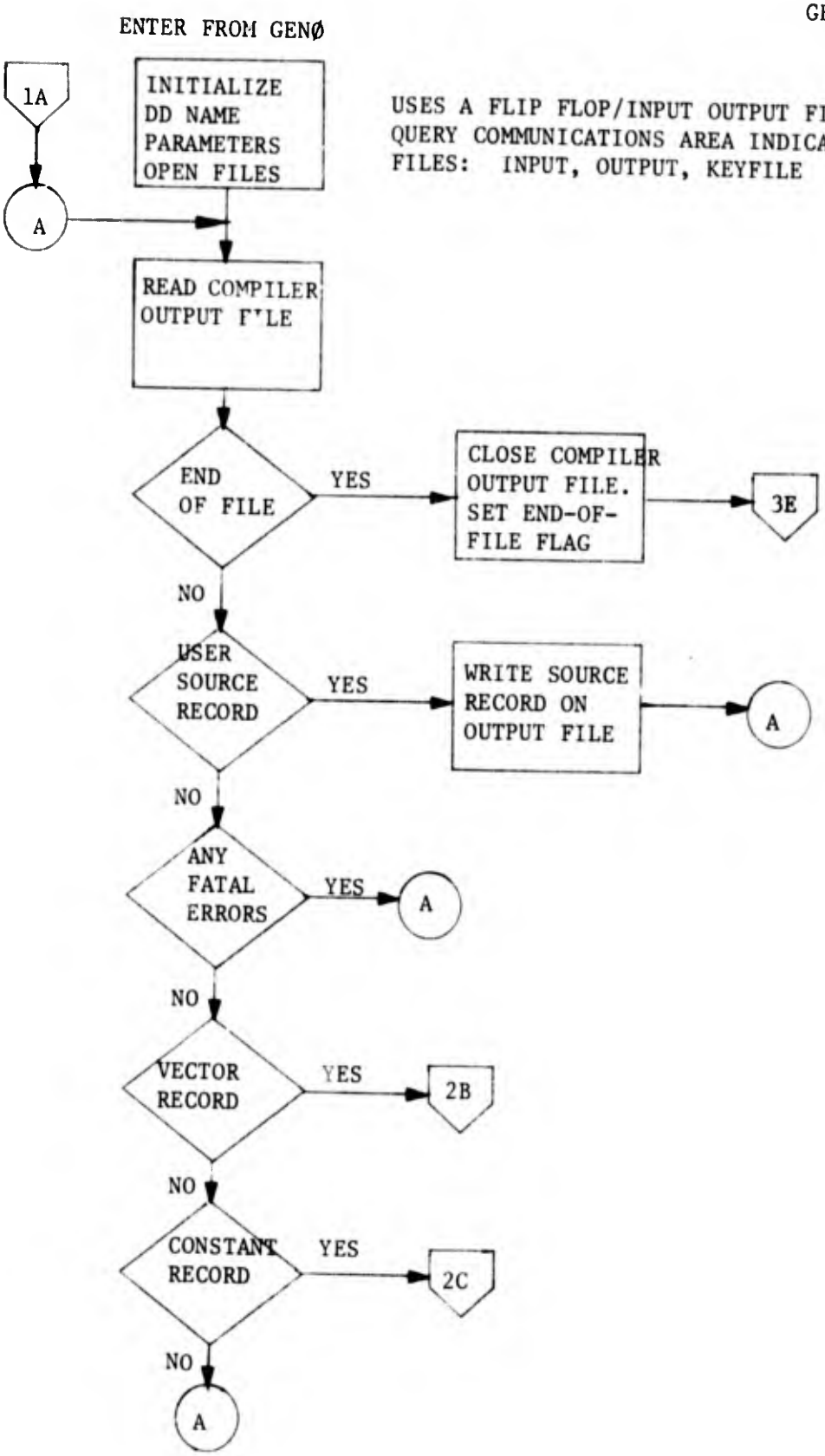
PX15 END-STEP Section is used to complete the processing of a subquery. If the query number is equal to zero, this function is not performed. This section also writes out from a saved area all KEEP names that were previously saved.

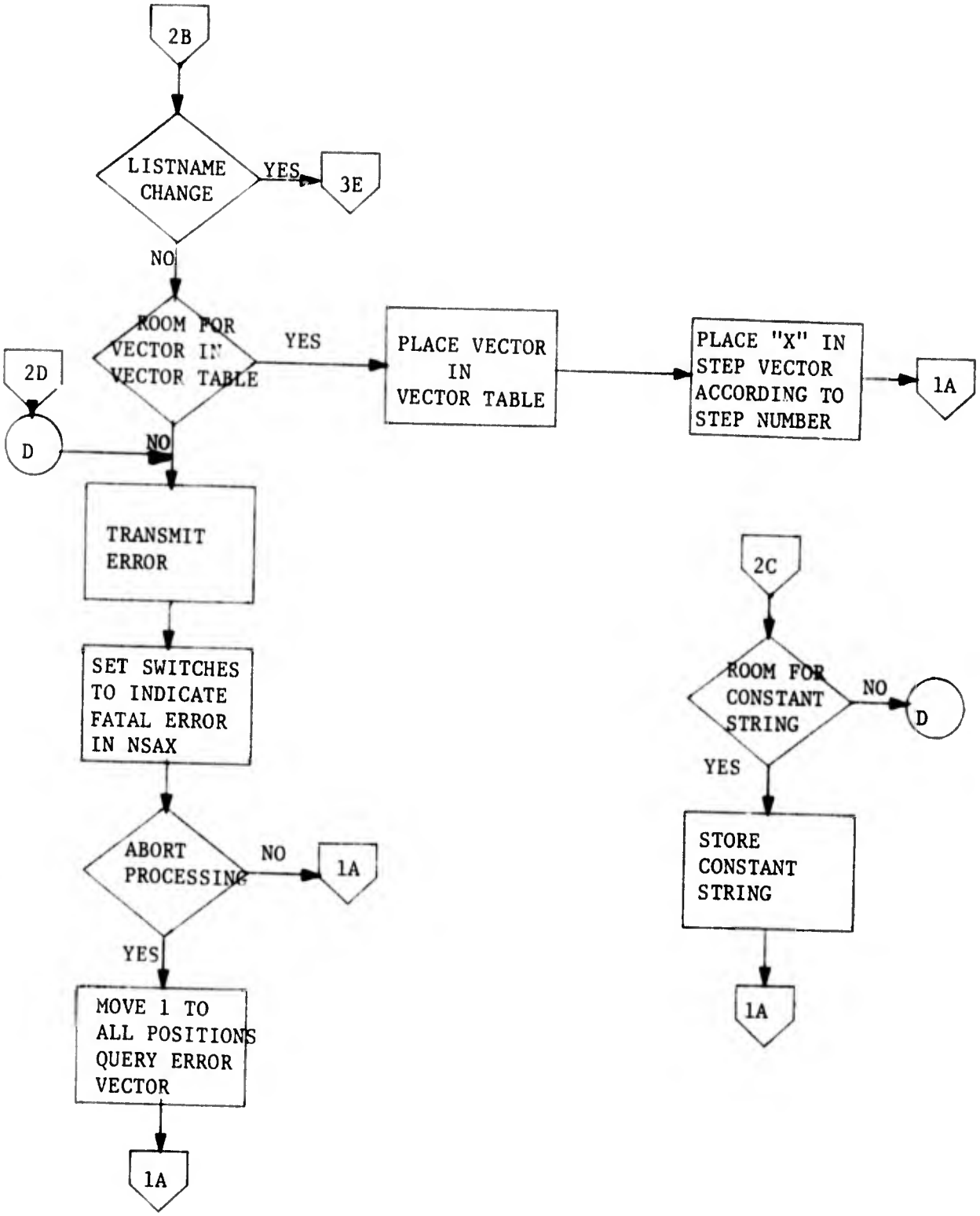
(5) GEN2X.

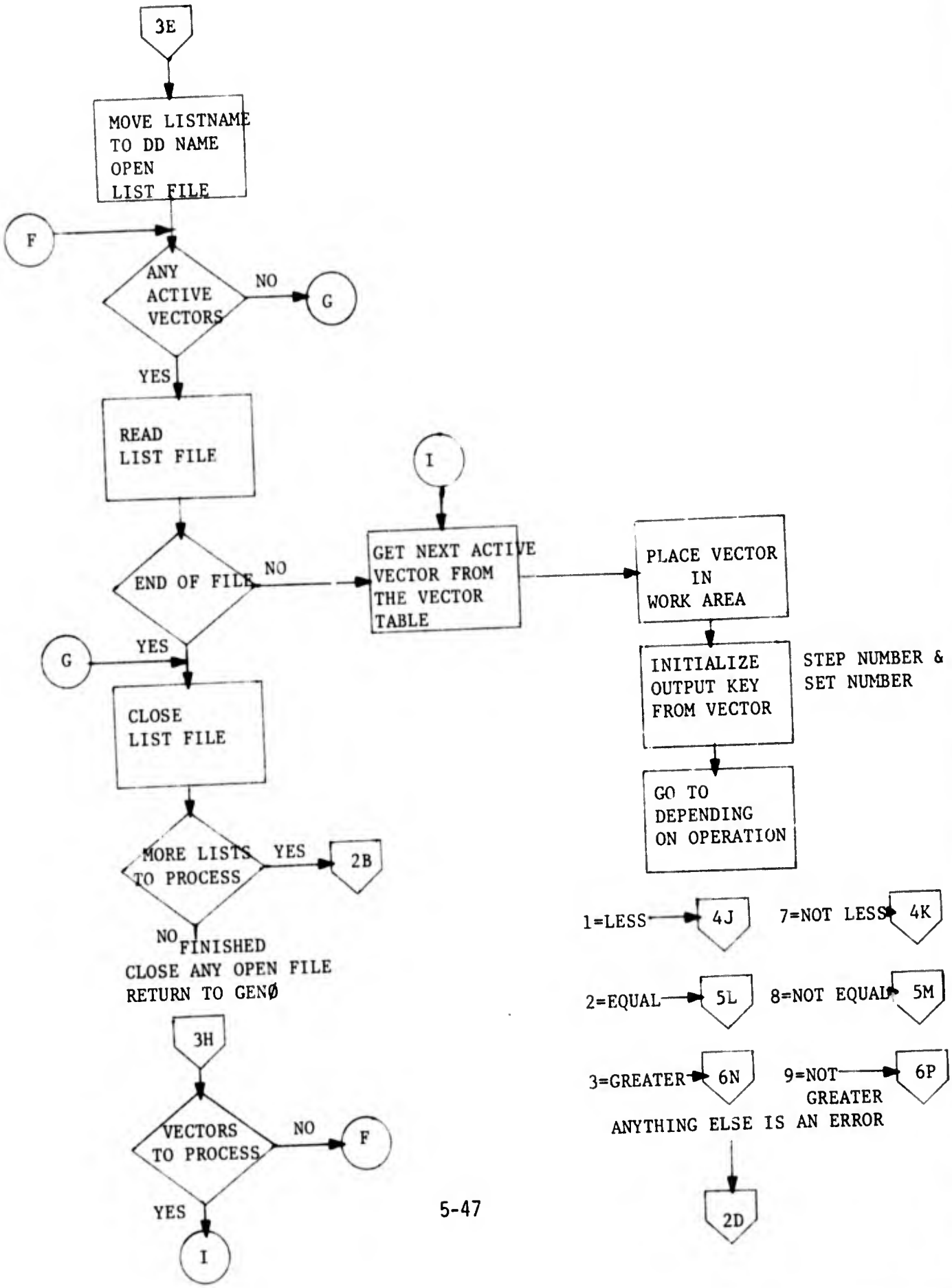

GEN2
SORT VECTORS BY LIST NAME
MORE THAN ONE PERIODIC SET
NO
RETURN
YES
WRITE ERROR MESSAGE
ABORT PROCESSING
NO
RETURN
YES
SET QUERY ERROR VECTOR
RETURN

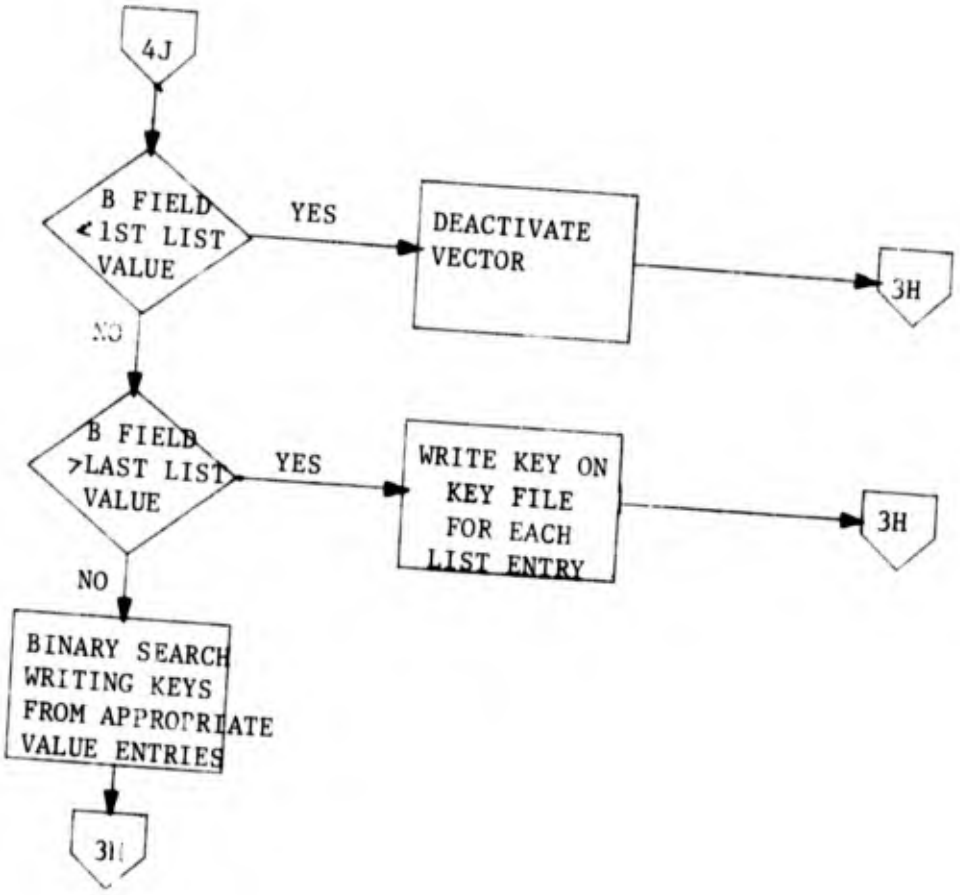
(6) GEN4X1.

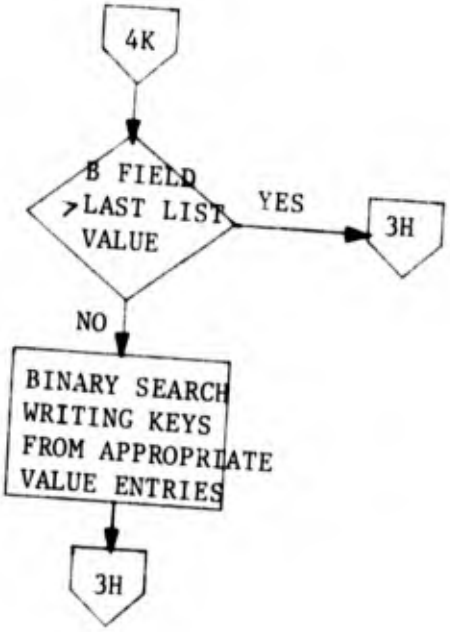
GEN4X1

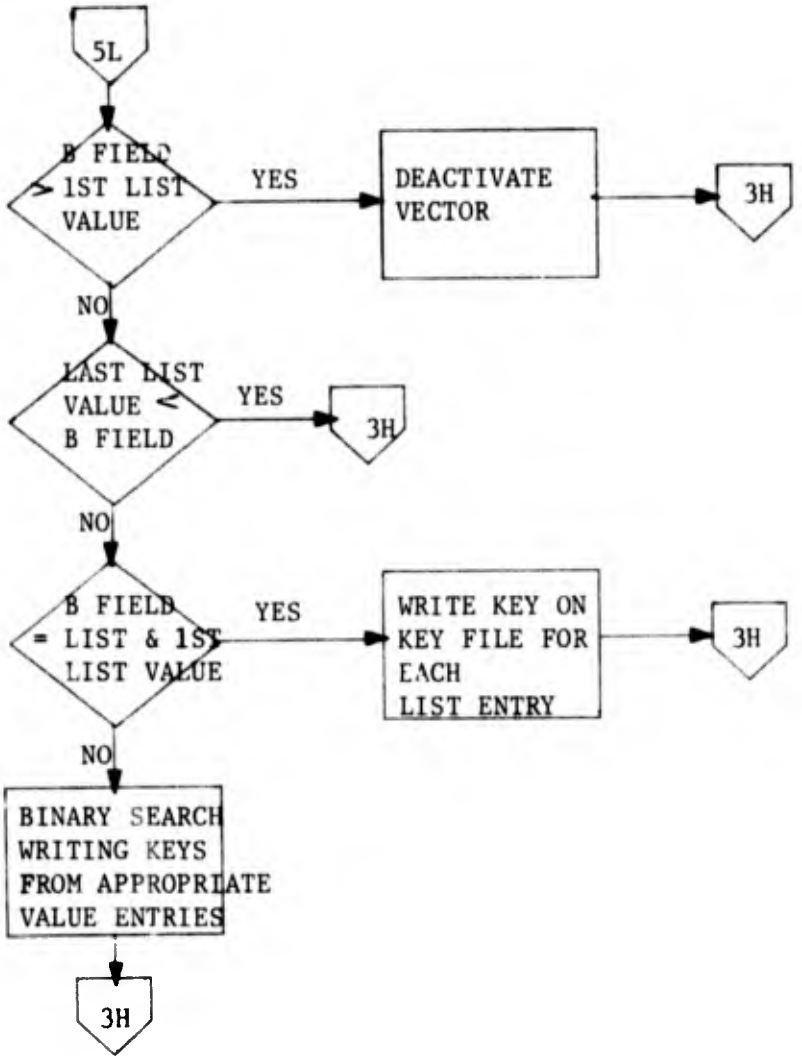


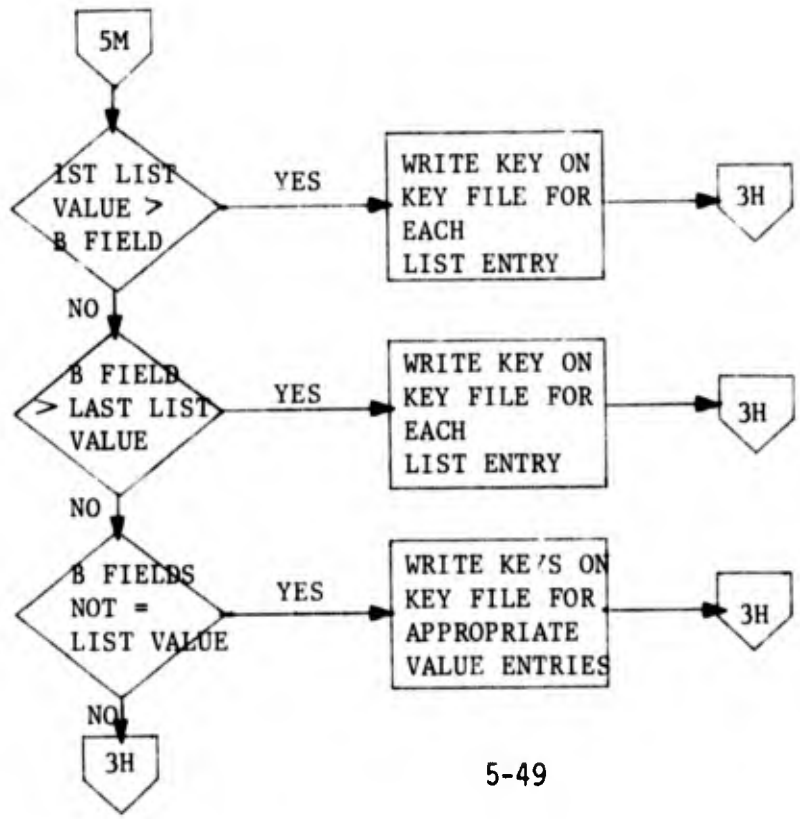

2B
LISTNAME CHANGE
YES
3E
NO
ROOM FOR VECTOR IN VECTOR TABLE
YES
PLACE VECTOR IN VECTOR TABLE
PLACE "X" IN STEP VECTOR ACCORDING TO STEP NUMBER
1A
2D
D
NO
TRANSMIT ERROR
SET SWITCHES TO INDICATE FATAL ERROR IN NSAX
ABORT PROCESSING
NO
1A
YES
MOVE 1 TO ALL POSITIONS QUERY ERROR VECTOR
1A
2C
ROOM FOR CONSTANT STRING
NO
D
YES
STORE CONSTANT STRING
1A

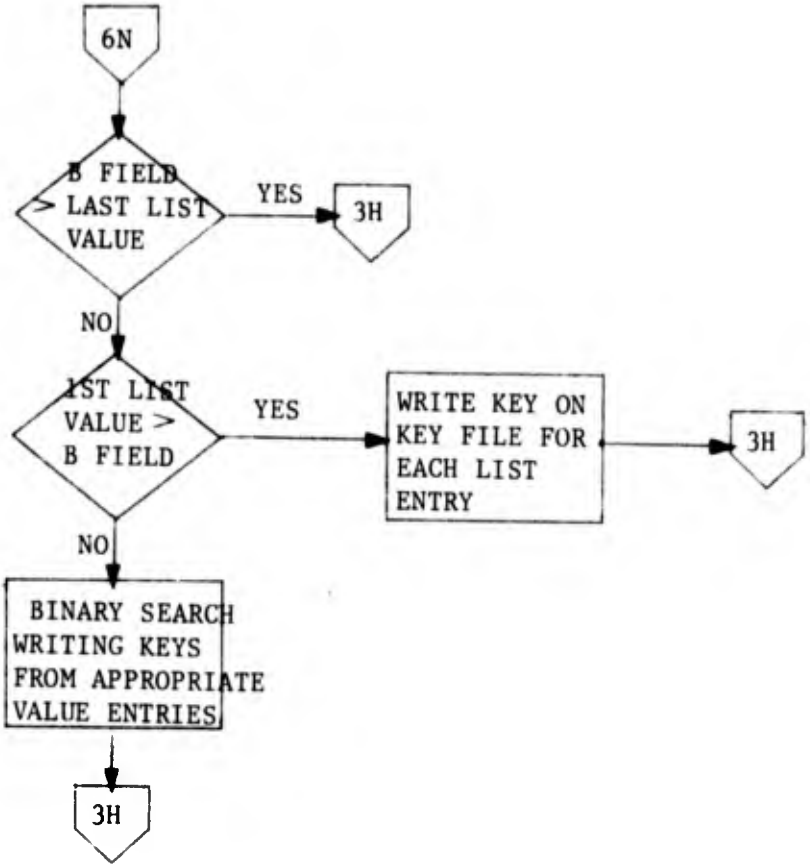

3E
MOVE LISTNAME TO DD NAME OPEN LIST FILE
F
ANY ACTIVE VECTORS
NO
G
YES
READ LIST FILE
I
END OF FILE
NO
GET NEXT ACTIVE VECTOR FROM THE VECTOR TABLE
PLACE VECTOR IN WORK AREA
INITIALIZE OUTPUT KEY FROM VECTOR
STEP NUMBER & SET NUMBER
GO TO DEPENDING ON OPERATION
G
YES
CLOSE LIST FILE
MORE LISTS TO PROCESS
YES
2B
NO
FINISHED
CLOSE ANY OPEN FILE
RETURN TO GENØ
1=LESS
4J
7=NOT LESS
4K
2=EQUAL
5L
8=NOT EQUAL
5M
3=GREATER
6N
9=NOT GREATER
6P
ANYTHING ELSE IS AN ERROR
2D
3H
VECTORS TO PROCESS
NO
F
YES
I

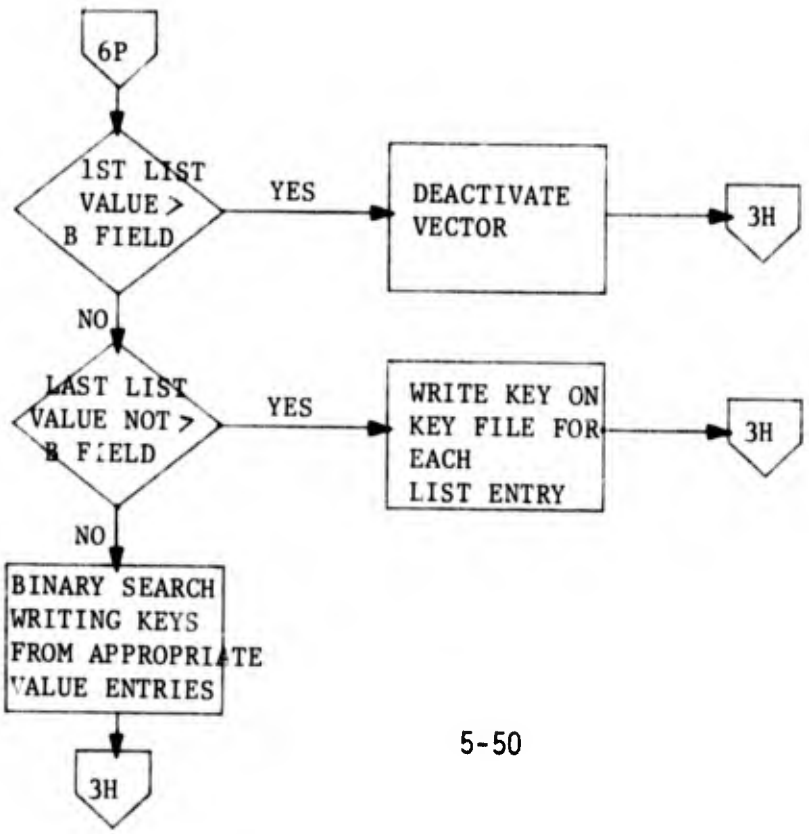

4J
B FIELD < 1ST LIST VALUE
YES
DEACTIVATE VECTOR
3H
NO
B FIELD > LAST LIST VALUE
YES
WRITE KEY ON KEY FILE FOR EACH LIST ENTRY
3H
NO
BINARY SEARCH WRITING KEYS FROM APPROPRIATE VALUE ENTRIES
3H


4K
B FIELD > LAST LIST VALUE
YES
3H
NO
BINARY SEARCH WRITING KEYS FROM APPROPRIATE VALUE ENTRIES
3H


5L
B FIELD > 1ST LIST VALUE
YES
DEACTIVATE VECTOR
3H
NO
LAST LIST VALUE < B FIELD
YES
3H
NO
B FIELD = LIST & 1ST LIST VALUE
YES
WRITE KEY ON KEY FILE FOR EACH LIST ENTRY
3H
NO
BINARY SEARCH WRITING KEYS FROM APPROPRIATE VALUE ENTRIES
3H

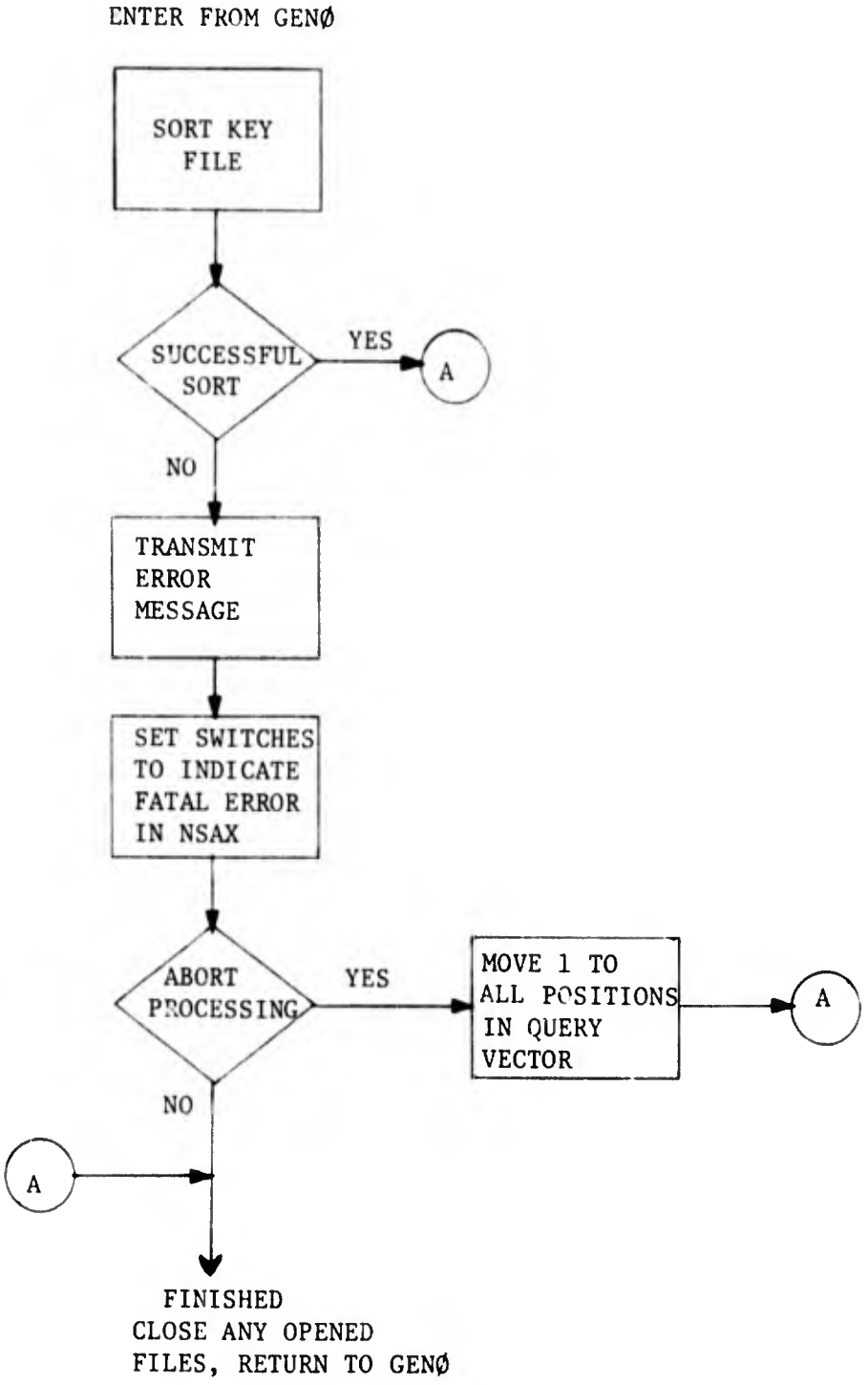

5M
1ST LIST VALUE > B FIELD
YES
WRITE KEY ON KEY FILE FOR EACH LIST ENTRY
3H
NO
B FIELD > LAST LIST VALUE
YES
WRITE KEY ON KEY FILE FOR EACH LIST ENTRY
3H
NO
B FIELDS NOT = LIST VALUE
YES
WRITE KEYS ON KEY FILE FOR APPROPRIATE VALUE ENTRIES
3H
NO
3H


6N
B FIELD > LAST LIST VALUE
YES
3H
NO
1ST LIST VALUE > B FIELD
YES
WRITE KEY ON KEY FILE FOR EACH LIST ENTRY
3H
NO
BINARY SEARCH WRITING KEYS FROM APPROPRIATE VALUE ENTRIES
3H

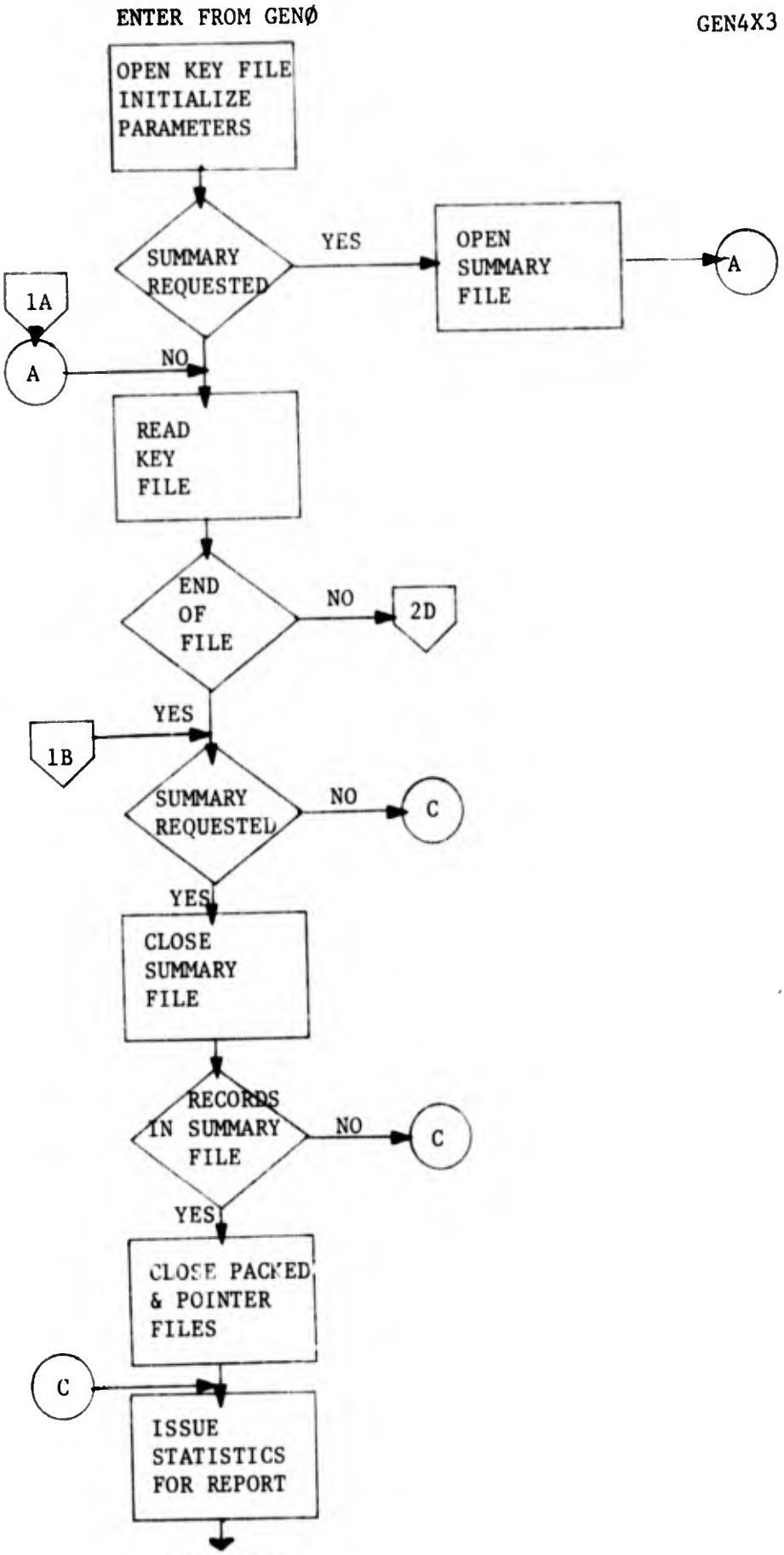

6P
1ST LIST VALUE > B FIELD
YES
DEACTIVATE VECTOR
3H
NO
LAST LIST VALUE NOT > B FIELD
YES
WRITE KEY ON KEY FILE FOR EACH LIST ENTRY
3H
NO
BINARY SEARCH WRITING KEYS FROM APPROPRIATE VALUE ENTRIES
3H

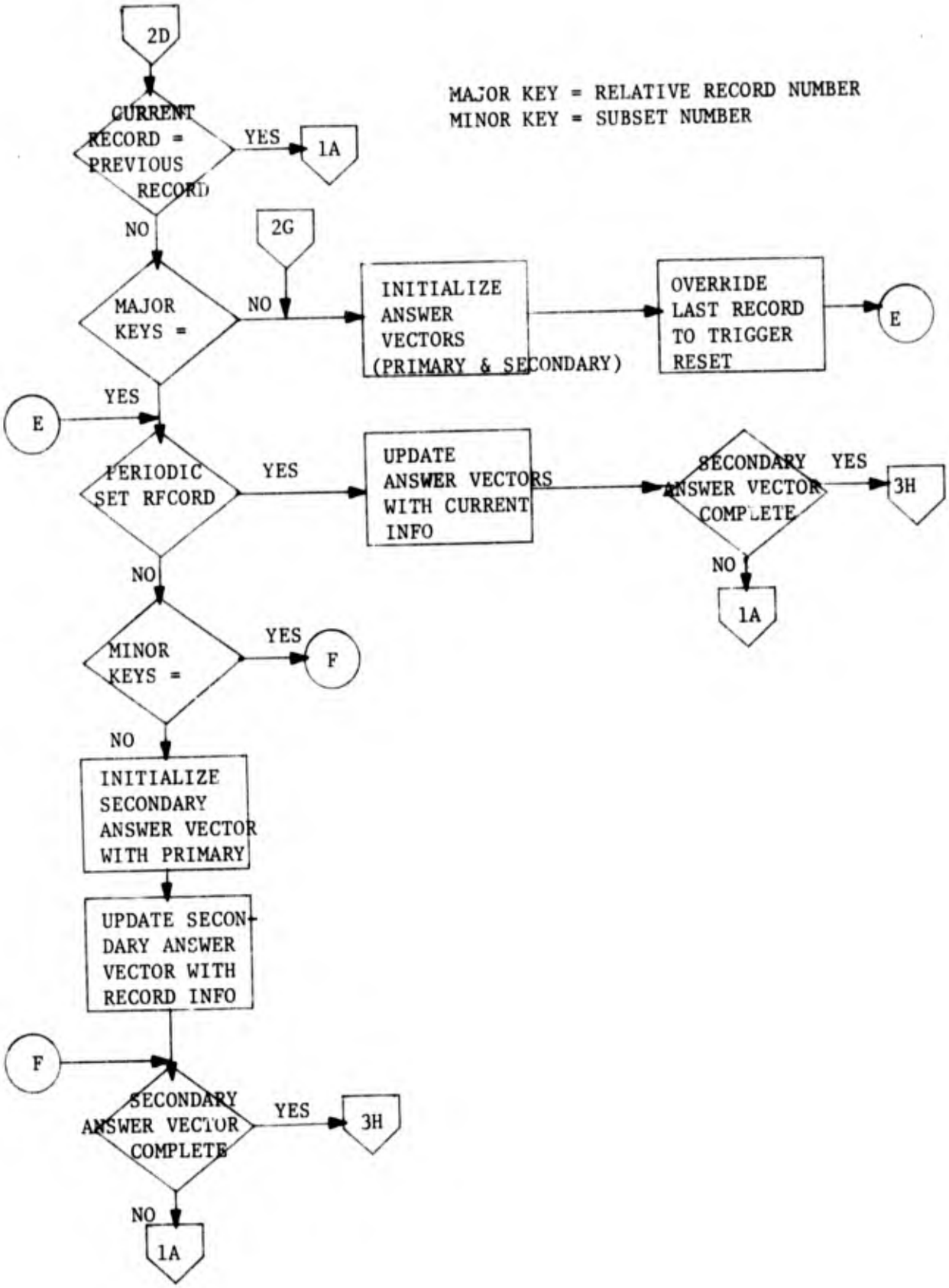
(7) GEN4X2.

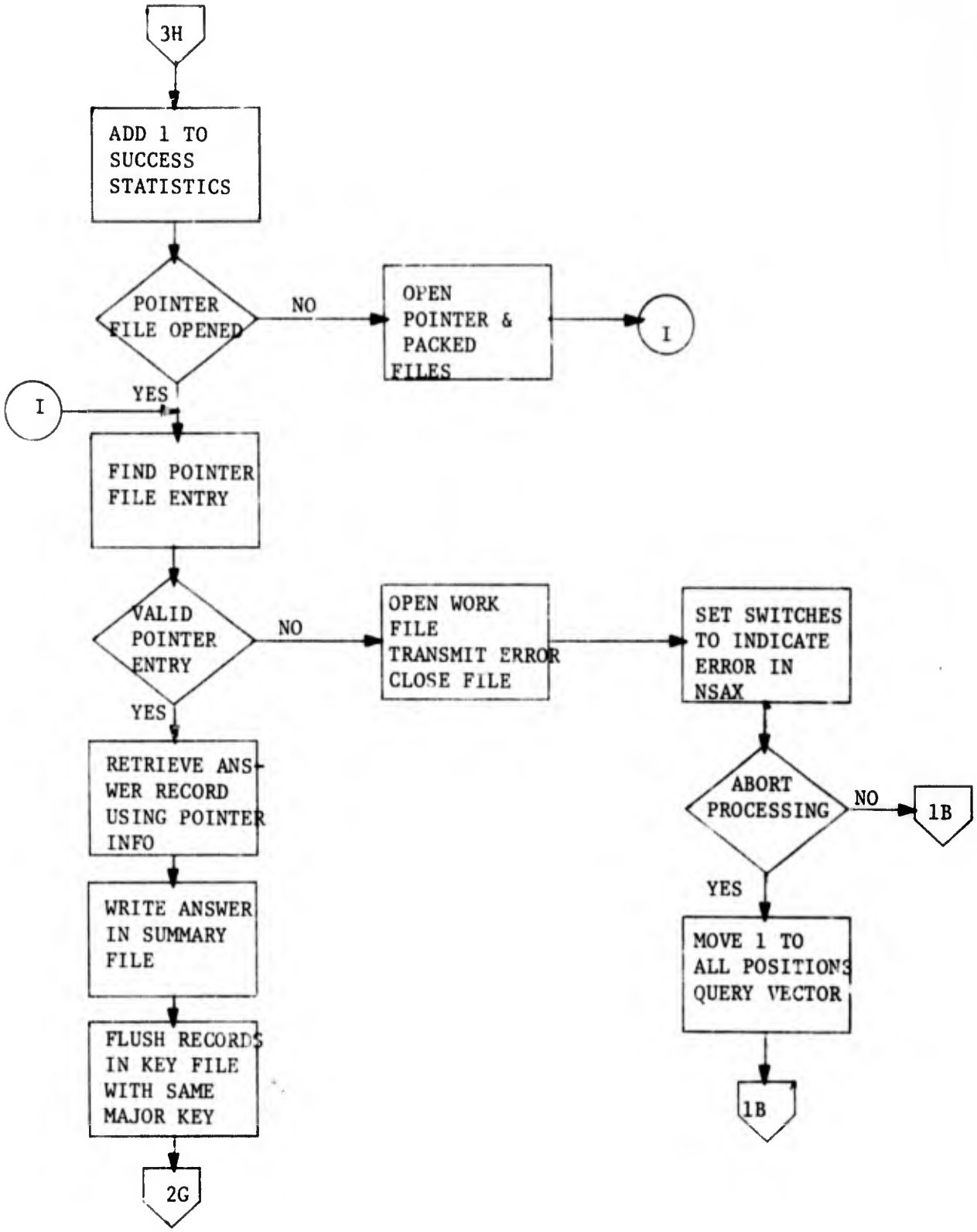



ENTER FROM GENØ
SORT KEY FILE
SUCCESSFUL SORT
YES
A
NO
TRANSMIT ERROR MESSAGE
SET SWITCHES TO INDICATE FATAL ERROR IN NSAX
ABORT PROCESSING
YES
MOVE 1 TO ALL POSITIONS IN QUERY VECTOR
A
NO
A
FINISHED
CLOSE ANY OPENED FILES, RETURN TO GENØ

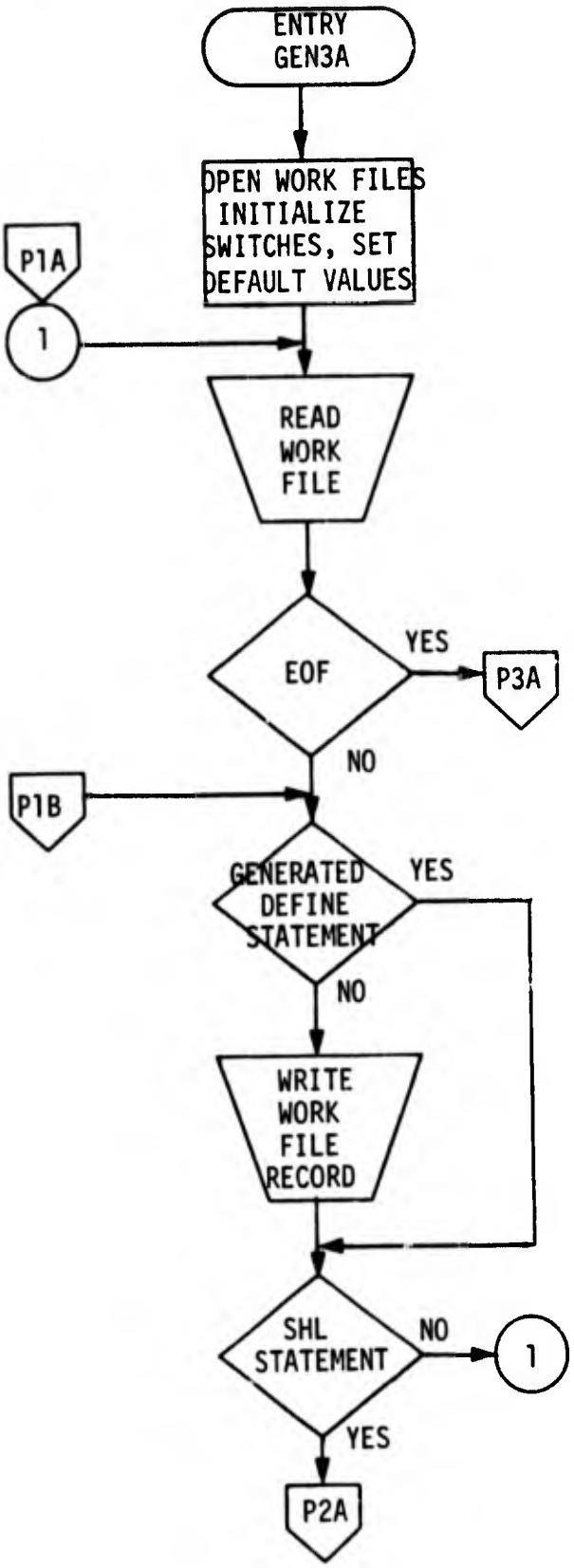
(8) GEN4X3.

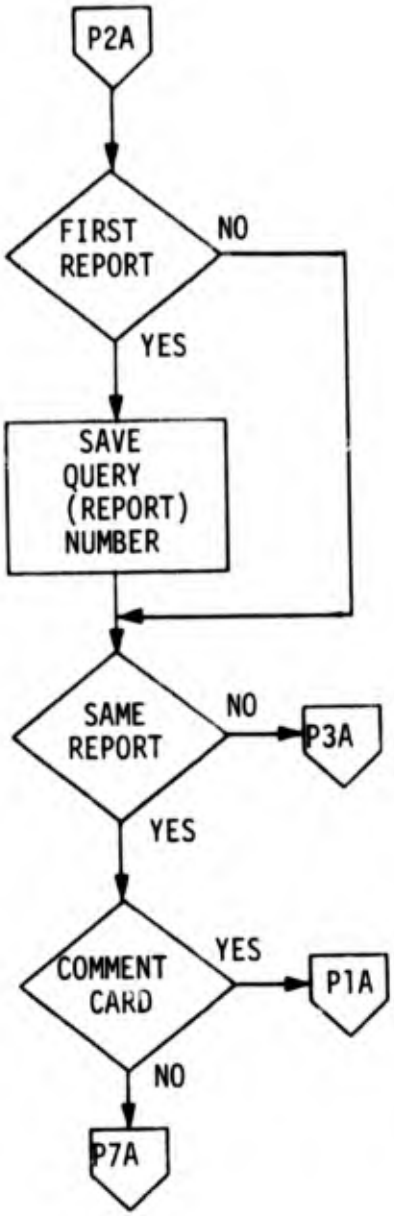

ENTER FROM GENØ
GEN4X3
OPEN KEY FILE
INITIALIZE
PARAMETERS
SUMMARY
REQUESTED
YES
OPEN
SUMMARY
FILE
A
1A
A
NO
READ
KEY
FILE
END
OF
FILE
NO
2D
YES
1B
SUMMARY
REQUESTED
NO
C
YES
CLOSE
SUMMARY
FILE
RECORDS
IN SUMMARY
FILE
NO
C
YES
CLOSE PACKED
& POINTER
FILES
C
ISSUE
STATISTICS
FOR REPORT
FINISHED
RETURN TO GENØ

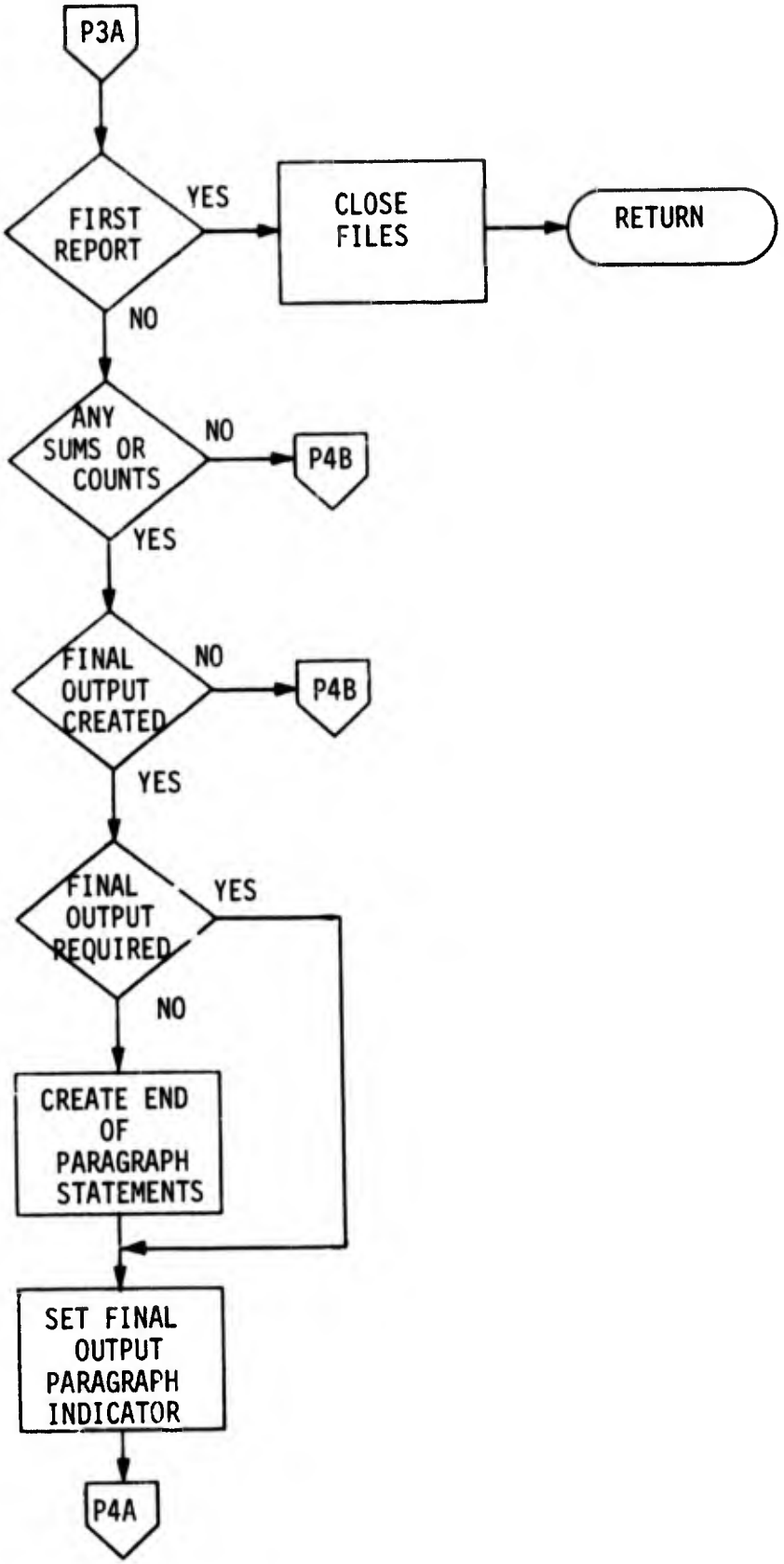

2D
CURRENT RECORD = PREVIOUS RECORD
YES
1A
NO
2G
MAJOR KEYS =
NO
INITIALIZE ANSWER VECTORS (PRIMARY & SECONDARY)
OVERRIDE LAST RECORD TO TRIGGER RESET
E
E
YES
PERIODIC SET RECORD
YES
UPDATE ANSWER VECTORS WITH CURRENT INFO
SECONDARY ANSWER VECTOR COMPLETE
YES
3H
NO
1A
NO
MINOR KEYS =
YES
F
NO
INITIALIZE SECONDARY ANSWER VECTOR WITH PRIMARY
UPDATE SECONDARY ANSWER VECTOR WITH RECORD INFO
F
SECONDARY ANSWER VECTOR COMPLETE
YES
3H
NO
1A
MAJOR KEY = RELATIVE RECORD NUMBER
MINOR KEY = SUBSET NUMBER

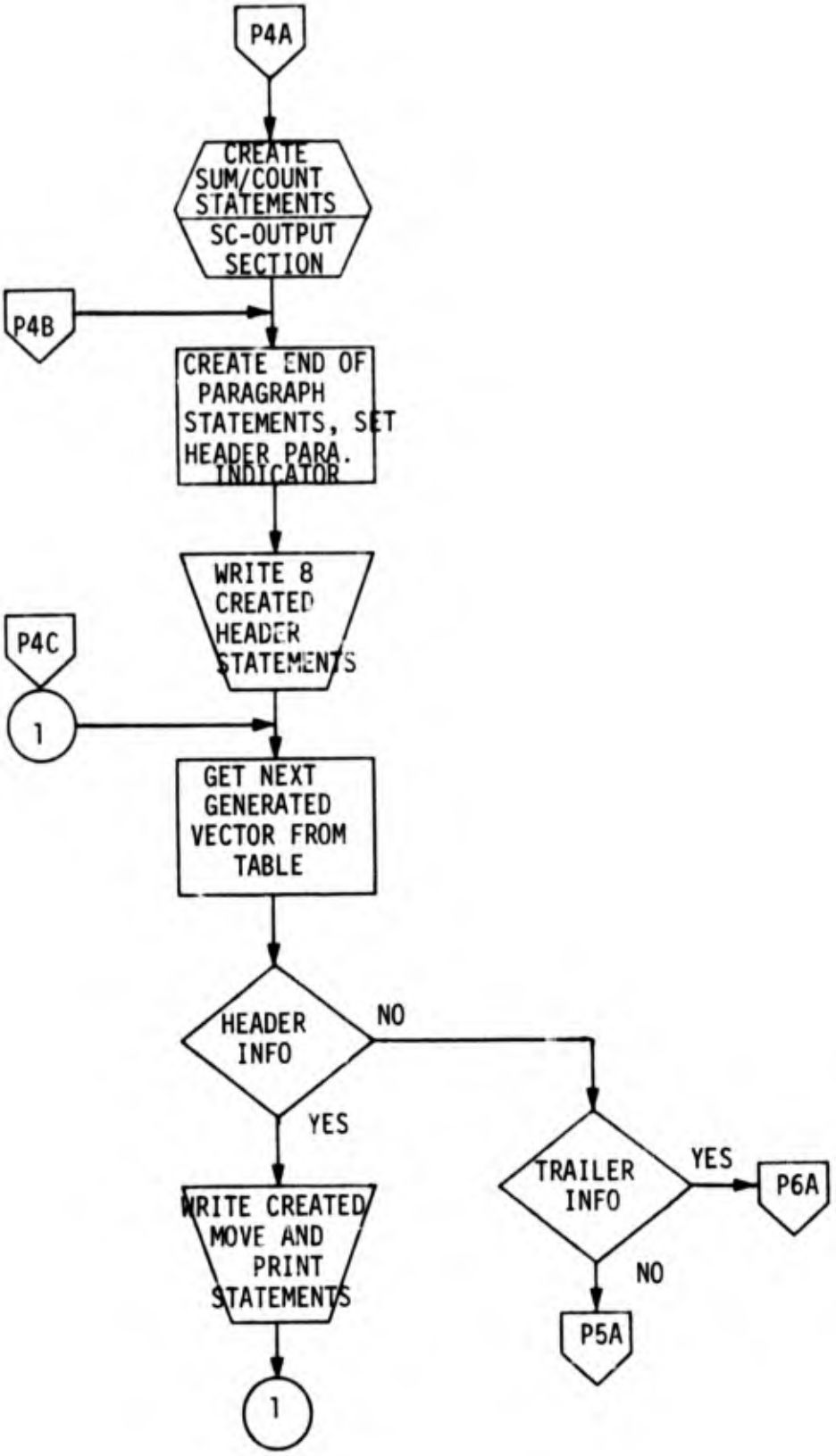

3H
ADD 1 TO SUCCESS STATISTICS
POINTER FILE OPENED
NO
OPEN POINTER & PACKED FILES
I
I
YES
FIND POINTER FILE ENTRY
VALID POINTER ENTRY
NO
OPEN WORK FILE TRANSMIT ERROR CLOSE FILE
SET SWITCHES TO INDICATE ERROR IN NSAX
YES
RETRIEVE ANSWER RECORD USING POINTER INFO
ABORT PROCESSING
NO
1B
YES
WRITE ANSWER IN SUMMARY FILE
MOVE 1 TO ALL POSITIONS QUERY VECTOR
FLUSH RECORDS IN KEY FILE WITH SAME MAJOR KEY
1B
2G

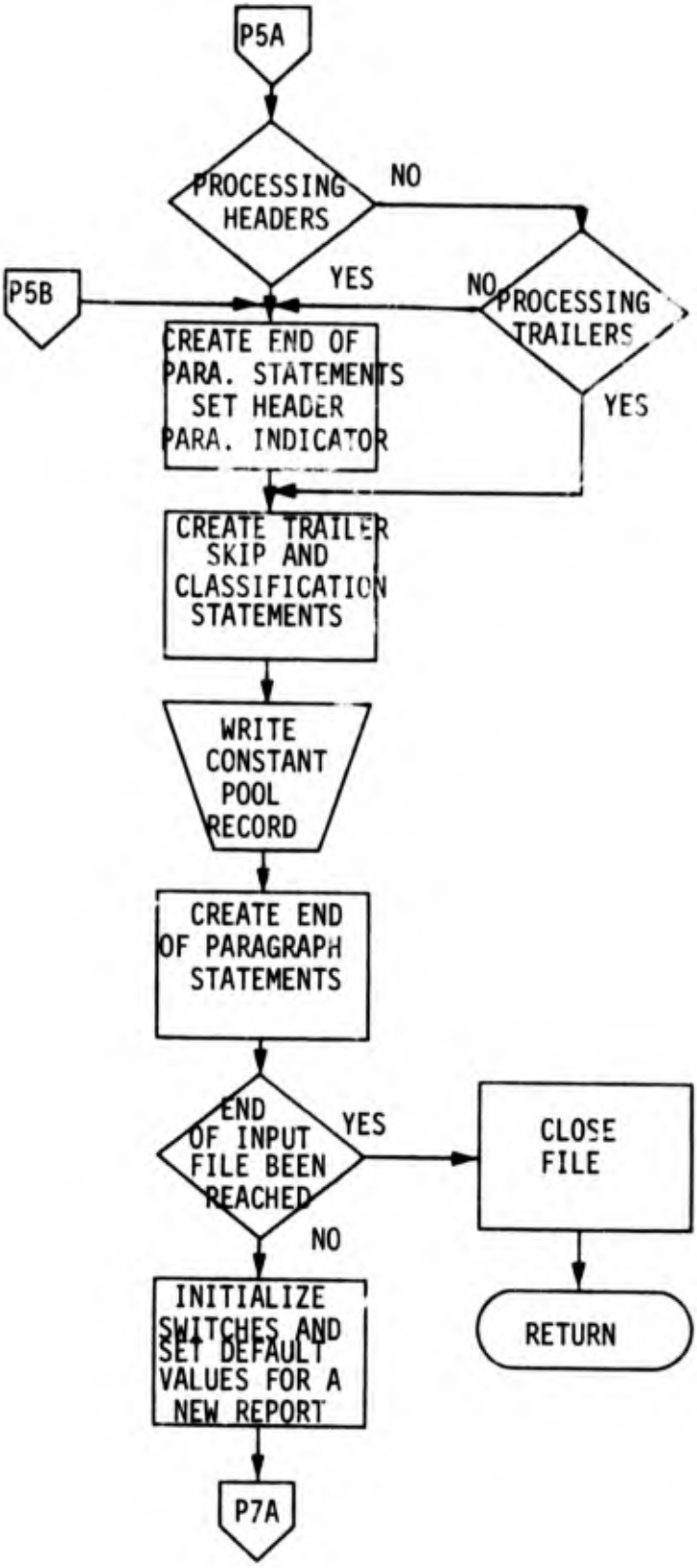
(9) GEN3A.

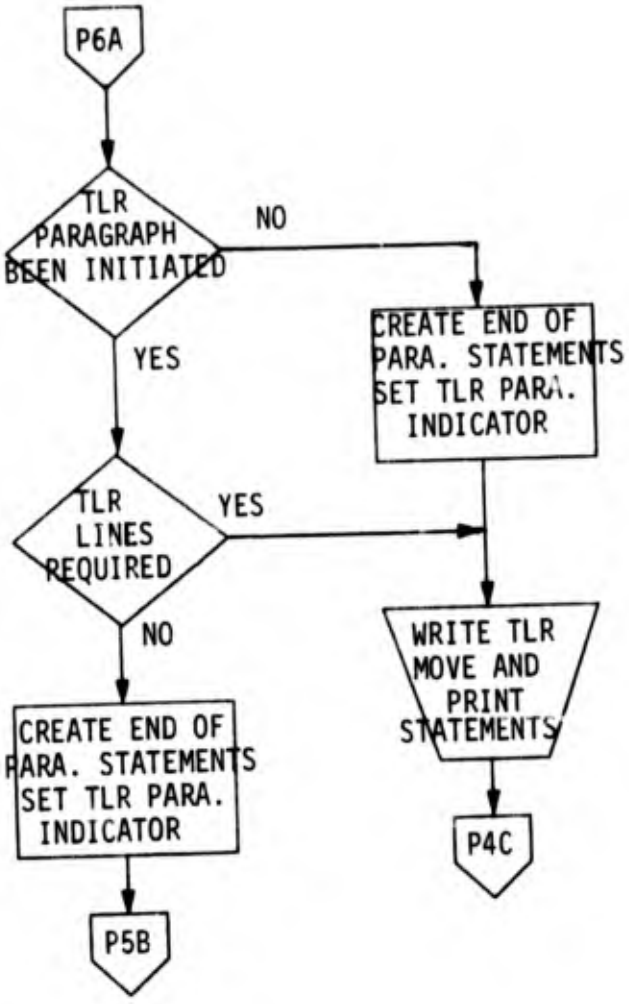

ENTRY
GEN3A
OPEN WORK FILES
INITIALIZE
SWITCHES, SET
DEFAULT VALUES
P1A
1
READ
WORK
FILE
EOF
YES
P3A
NO
P1B
GENERATED
DEFINE
STATEMENT
YES
NO
WRITE
WORK
FILE
RECORD
SHL
STATEMENT
NO
1
YES
P2A

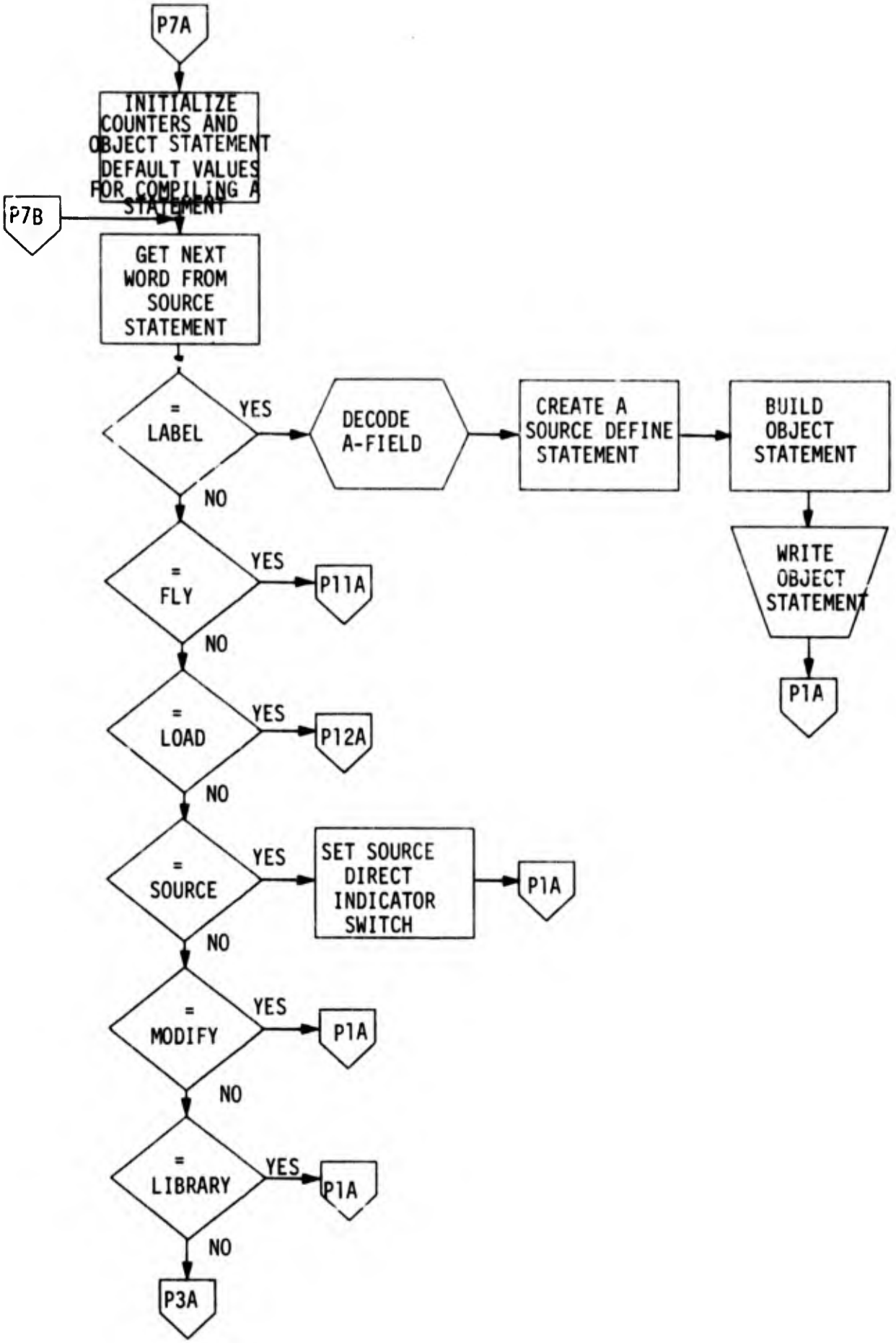

P2A
FIRST REPORT
NO
YES
SAVE QUERY (REPORT) NUMBER
SAME REPORT
NO
P3A
YES
COMMENT CARD
YES
P1A
NO
P7A

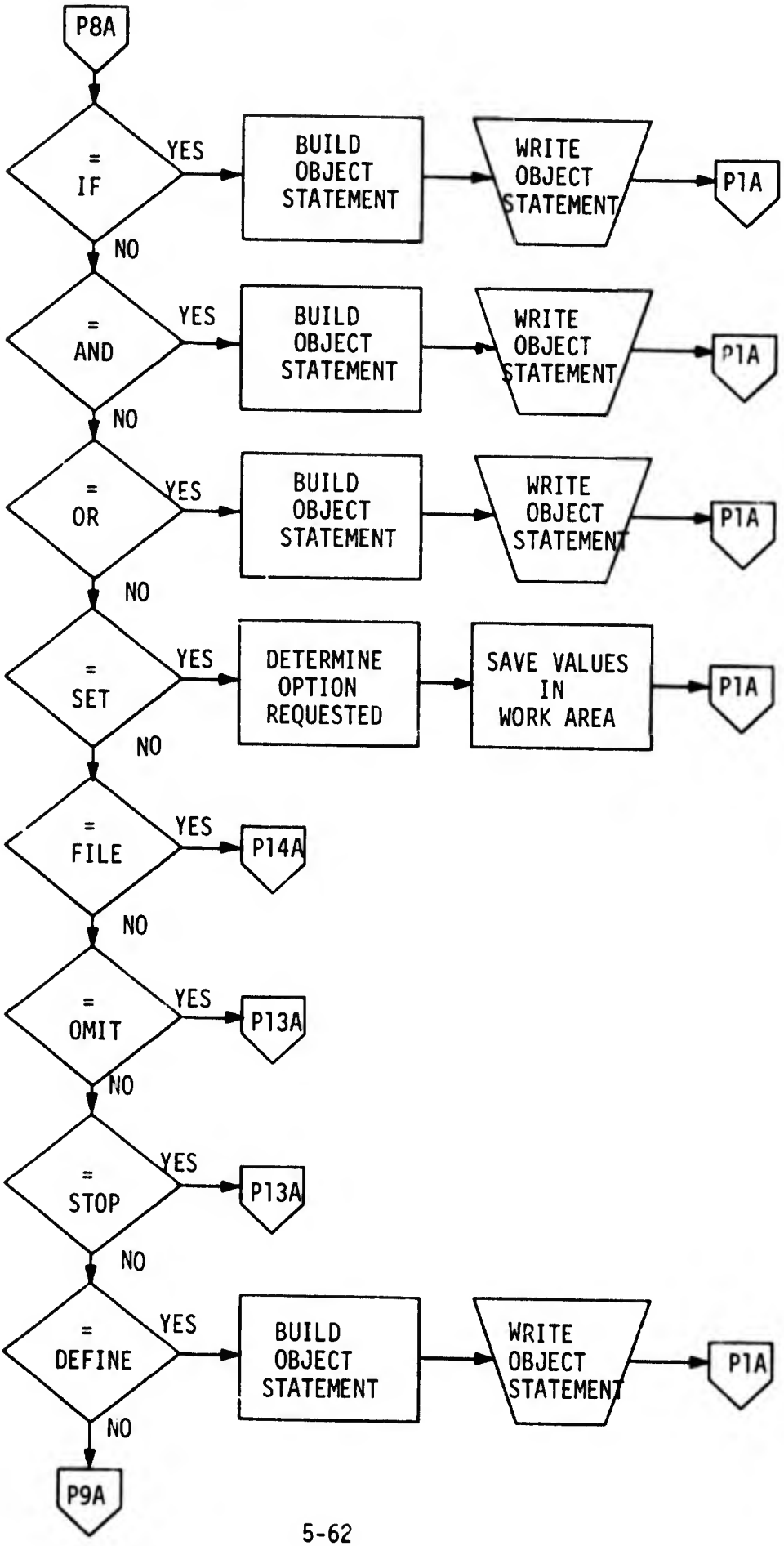

P3A
FIRST REPORT
YES
CLOSE FILES
RETURN
NO
ANY SUMS OR COUNTS
NO
P4B
YES
FINAL OUTPUT CREATED
NO
P4B
YES
FINAL OUTPUT REQUIRED
YES
NO
CREATE END OF PARAGRAPH STATEMENTS
SET FINAL OUTPUT PARAGRAPH INDICATOR
P4A

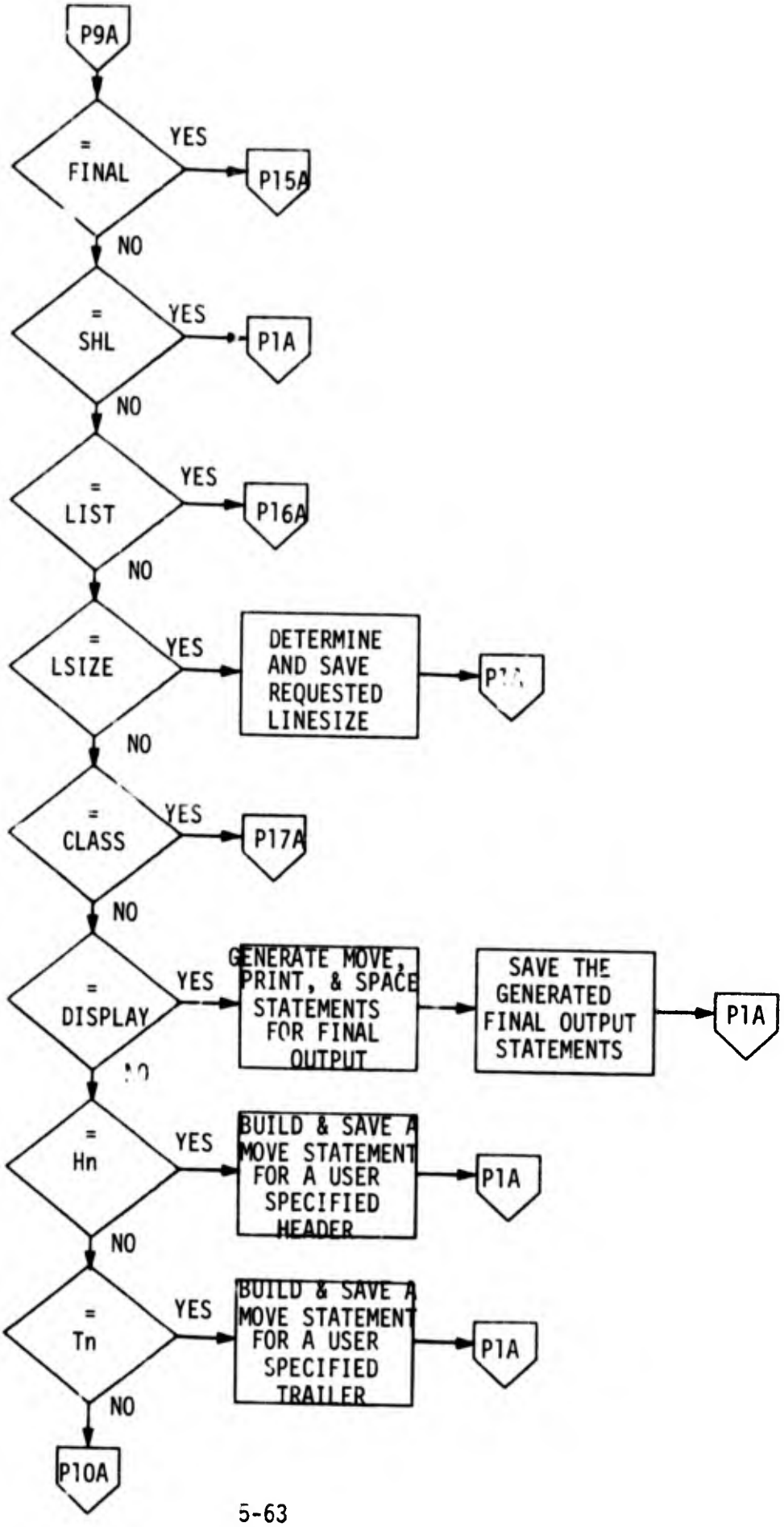

P4A
CREATE SUM/COUNT STATEMENTS
SC-OUTPUT SECTION
P4B
CREATE END OF PARAGRAPH STATEMENTS, SET HEADER PARA. INDICATOR
WRITE 8 CREATED HEADER STATEMENTS
P4C
1
GET NEXT GENERATED VECTOR FROM TABLE
HEADER INFO
NO
YES
TRAILER INFO
YES
P6A
NO
P5A
WRITE CREATED MOVE AND PRINT STATEMENTS
1

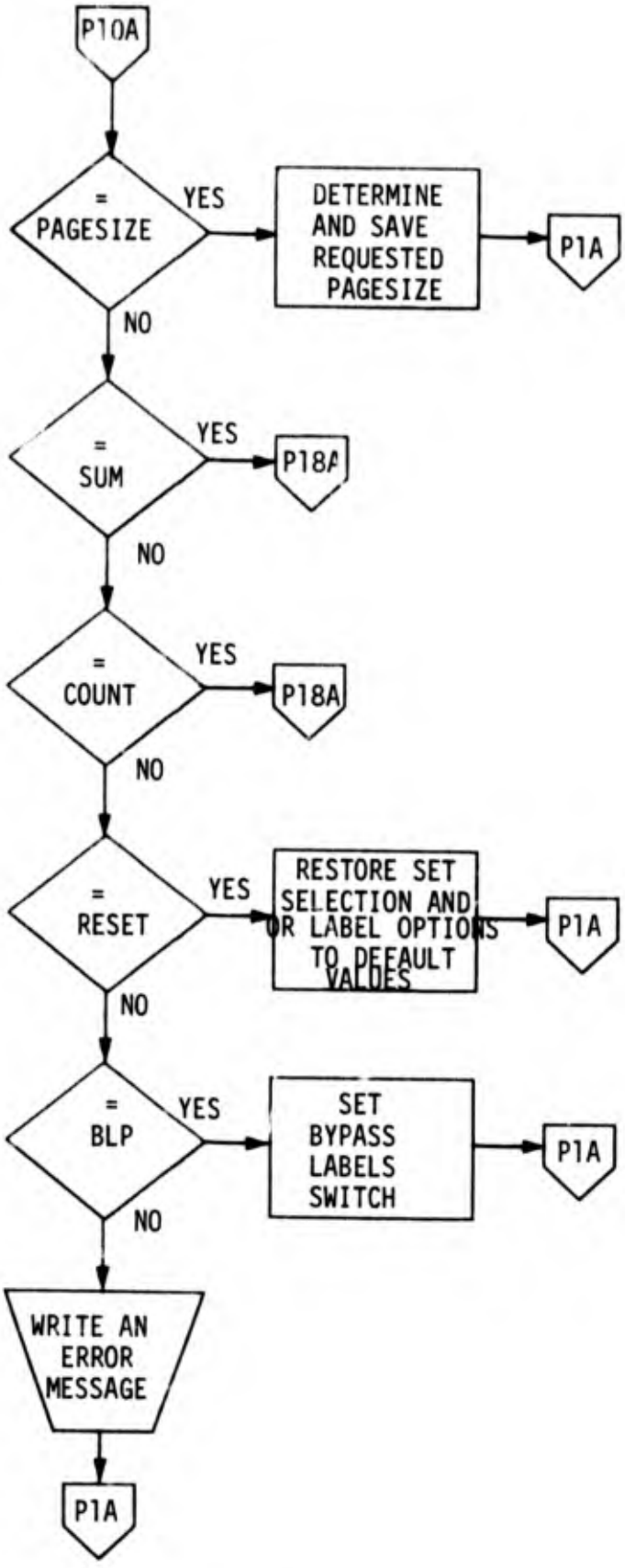

P5A
PROCESSING HEADERS
NO
YES
P5B
NO
PROCESSING TRAILERS
YES
CREATE END OF PARA. STATEMENTS SET HEADER PARA. INDICATOR
CREATE TRAILER SKIP AND CLASSIFICATION STATEMENTS
WRITE CONSTANT POOL RECORD
CREATE END OF PARAGRAPH STATEMENTS
END OF INPUT FILE BEEN REACHED
YES
CLOSE FILE
NO
RETURN
INITIALIZE SWITCHES AND SET DEFAULT VALUES FOR A NEW REPORT
P7A

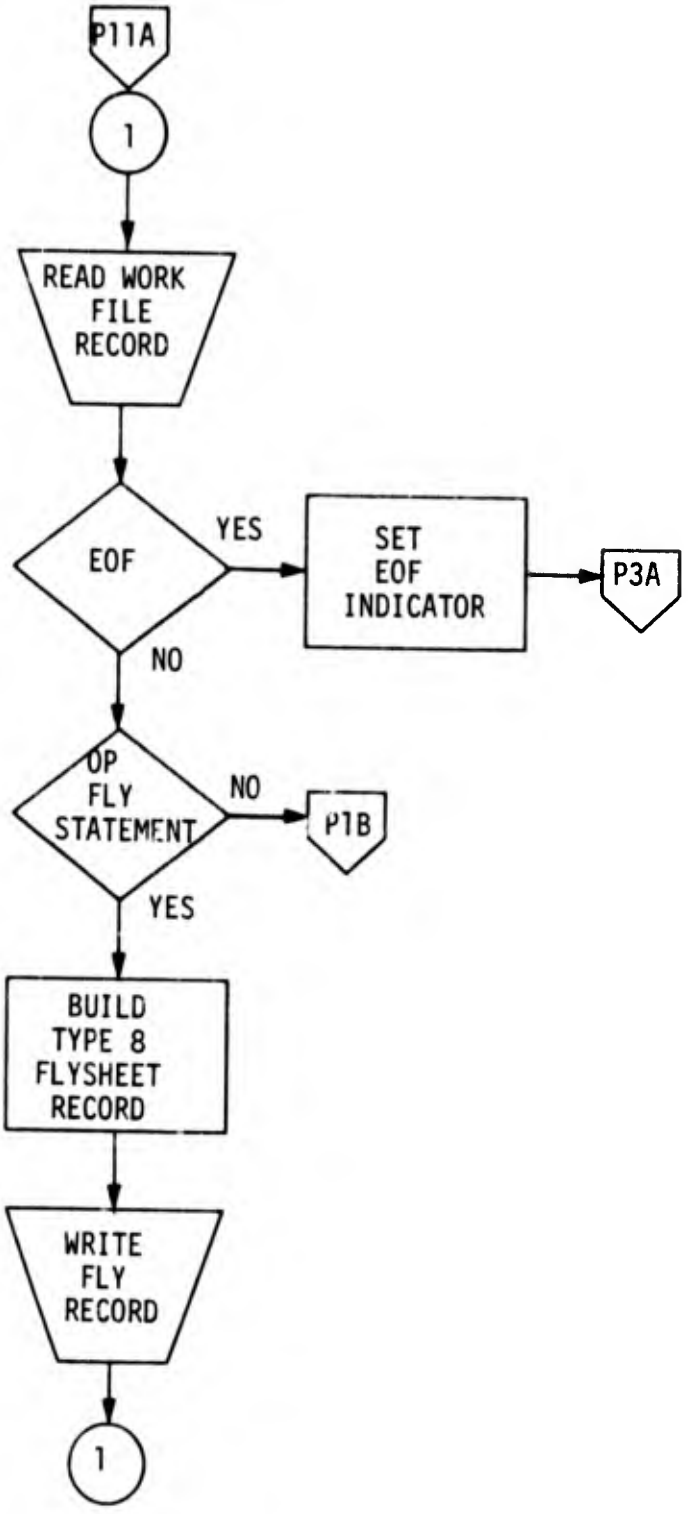

P6A
TLR PARAGRAPH BEEN INITIATED
NO
YES
CREATE END OF PARA. STATEMENTS SET TLR PARA. INDICATOR
TLR LINES REQUIRED
YES
NO
WRITE TLR MOVE AND PRINT STATEMENTS
P4C
CREATE END OF PARA. STATEMENTS SET TLR PARA. INDICATOR
P5B

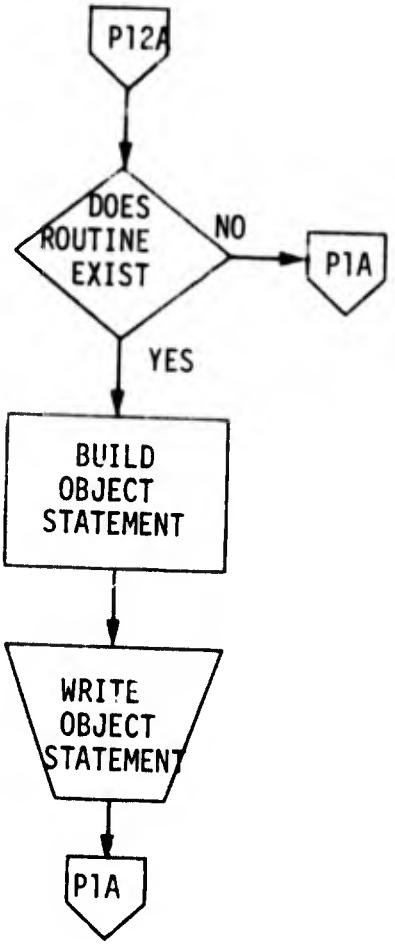

P7A
INITIALIZE COUNTERS AND OBJECT STATEMENT DEFAULT VALUES FOR COMPILING A STATEMENT
P7B
GET NEXT WORD FROM SOURCE STATEMENT
= LABEL
YES
DECODE A-FIELD
CREATE A SOURCE DEFINE STATEMENT
BUILD OBJECT STATEMENT
WRITE OBJECT STATEMENT
P1A
NO
= FLY
YES
P11A
NO
= LOAD
YES
P12A
NO
= SOURCE
YES
SET SOURCE DIRECT INDICATOR SWITCH
P1A
NO
= MODIFY
YES
P1A
NO
= LIBRARY
YES
P1A
NO
P3A

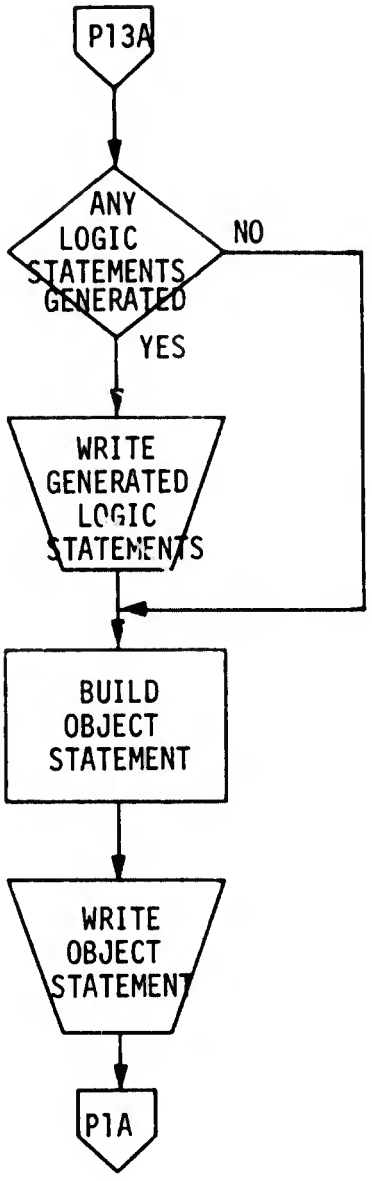

P8A
= IF
YES
BUILD OBJECT STATEMENT
WRITE OBJECT STATEMENT
P1A
NO
= AND
YES
BUILD OBJECT STATEMENT
WRITE OBJECT STATEMENT
P1A
NO
= OR
YES
BUILD OBJECT STATEMENT
WRITE OBJECT STATEMENT
P1A
NO
= SET
YES
DETERMINE OPTION REQUESTED
SAVE VALUES IN WORK AREA
P1A
NO
= FILE
YES
P14A
NO
= OMIT
YES
P13A
NO
= STOP
YES
P13A
NO
= DEFINE
YES
BUILD OBJECT STATEMENT
WRITE OBJECT STATEMENT
P1A
NO
P9A

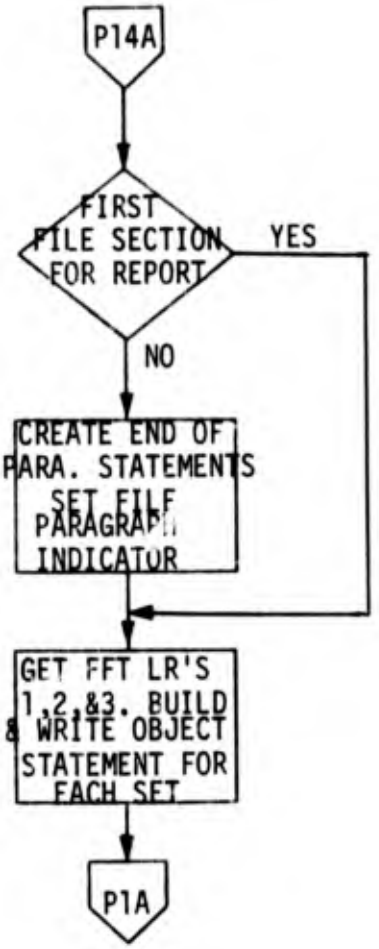

P9A
= FINAL
YES
P15A
NO
= SHL
YES
P1A
NO
= LIST
YES
P16A
NO
= LSIZE
YES
DETERMINE AND SAVE REQUESTED LINESIZE
NO
= CLASS
YES
P17A
NO
= DISPLAY
YES
GENERATE MOVE, PRINT, & SPACE STATEMENTS FOR FINAL OUTPUT
SAVE THE GENERATED FINAL OUTPUT STATEMENTS
P1A
NO
= Hn
YES
BUILD & SAVE A MOVE STATEMENT FOR A USER SPECIFIED HEADER
P1A
NO
= Tn
YES
BUILD & SAVE A MOVE STATEMENT FOR A USER SPECIFIED TRAILER
P1A
NO
P10A

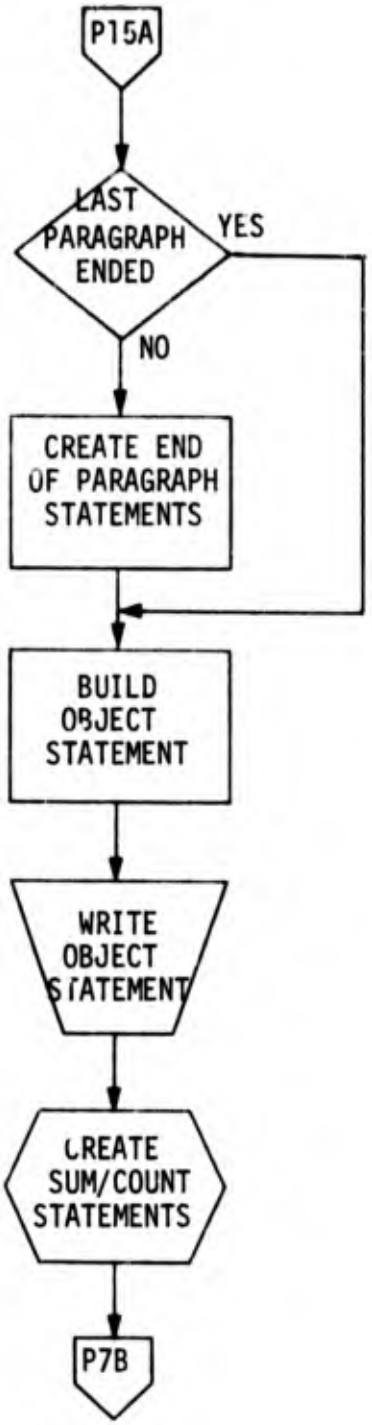

P10A
= PAGESIZE
YES
DETERMINE AND SAVE REQUESTED PAGESIZE
P1A
NO
= SUM
YES
P18A
NO
= COUNT
YES
P18A
NO
= RESET
YES
RESTORE SET SELECTION AND OR LABEL OPTIONS TO DEFAULT VALUES
P1A
NO
= BLP
YES
SET BYPASS LABELS SWITCH
P1A
NO
WRITE AN ERROR MESSAGE
P1A

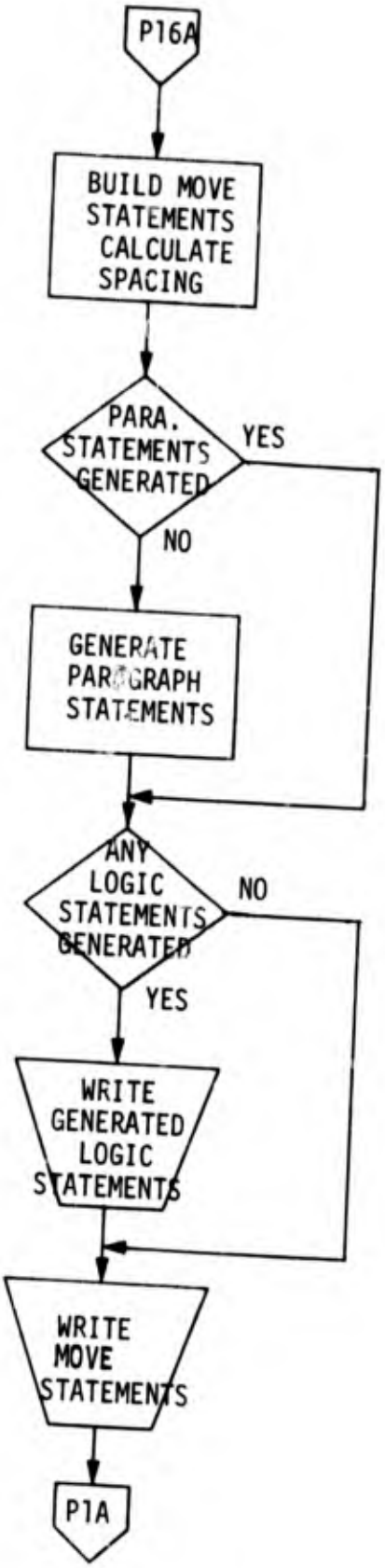

P11A
1
READ WORK FILE RECORD
EOF
YES
SET EOF INDICATOR
P3A
NO
OP FLY STATEMENT
NO
P1B
YES
BUILD TYPE 8 FLYSHEET RECORD
WRITE FLY RECORD
1

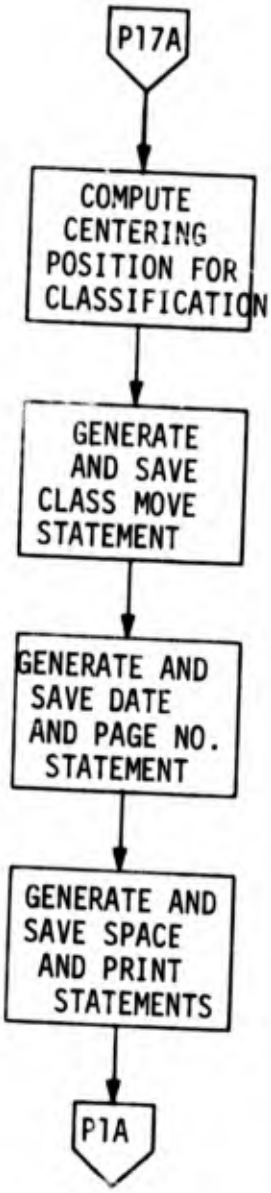

P12A
DOES ROUTINE EXIST
NO
P1A
YES
BUILD OBJECT STATEMENT
WRITE OBJECT STATEMENT
P1A

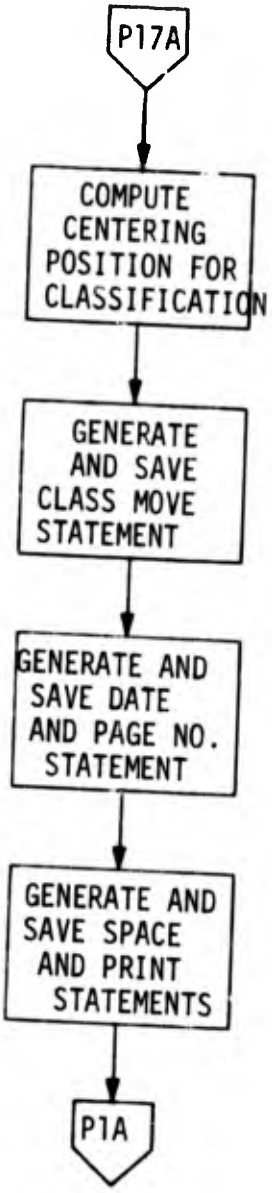

P13A
ANY LOGIC STATEMENTS GENERATED
NO
YES
WRITE GENERATED LOGIC STATEMENTS
BUILD OBJECT STATEMENT
WRITE OBJECT STATEMENT
P1A

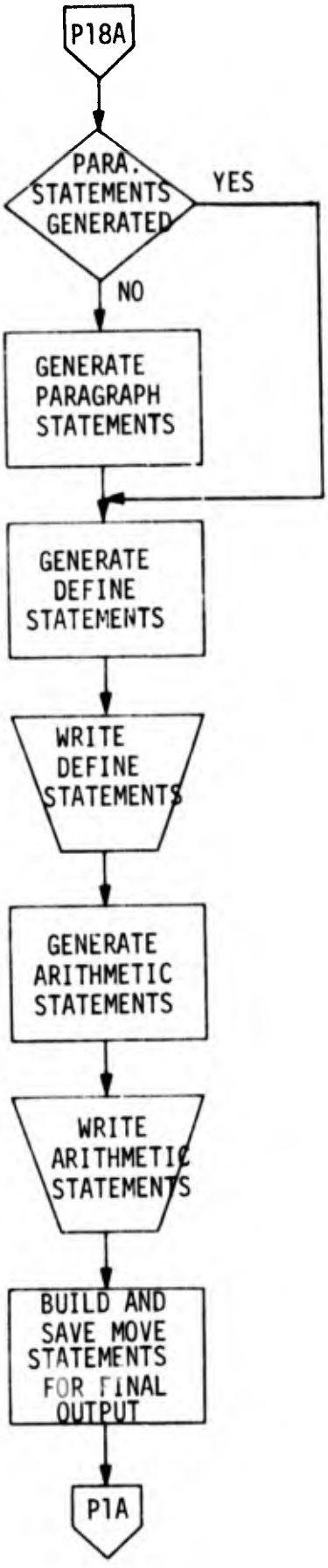

P14A
FIRST FILE SECTION FOR REPORT
YES
NO
CREATE END OF PARA. STATEMENTS SET FILE PARAGRAPH INDICATOR
GET FFT LR'S 1,2,&3. BUILD & WRITE OBJECT STATEMENT FOR EACH SET
P1A


P15A
LAST PARAGRAPH ENDED
YES
NO
CREATE END OF PARAGRAPH STATEMENTS
BUILD OBJECT STATEMENT
WRITE OBJECT STATEMENT
CREATE SUM/COUNT STATEMENTS
P7B

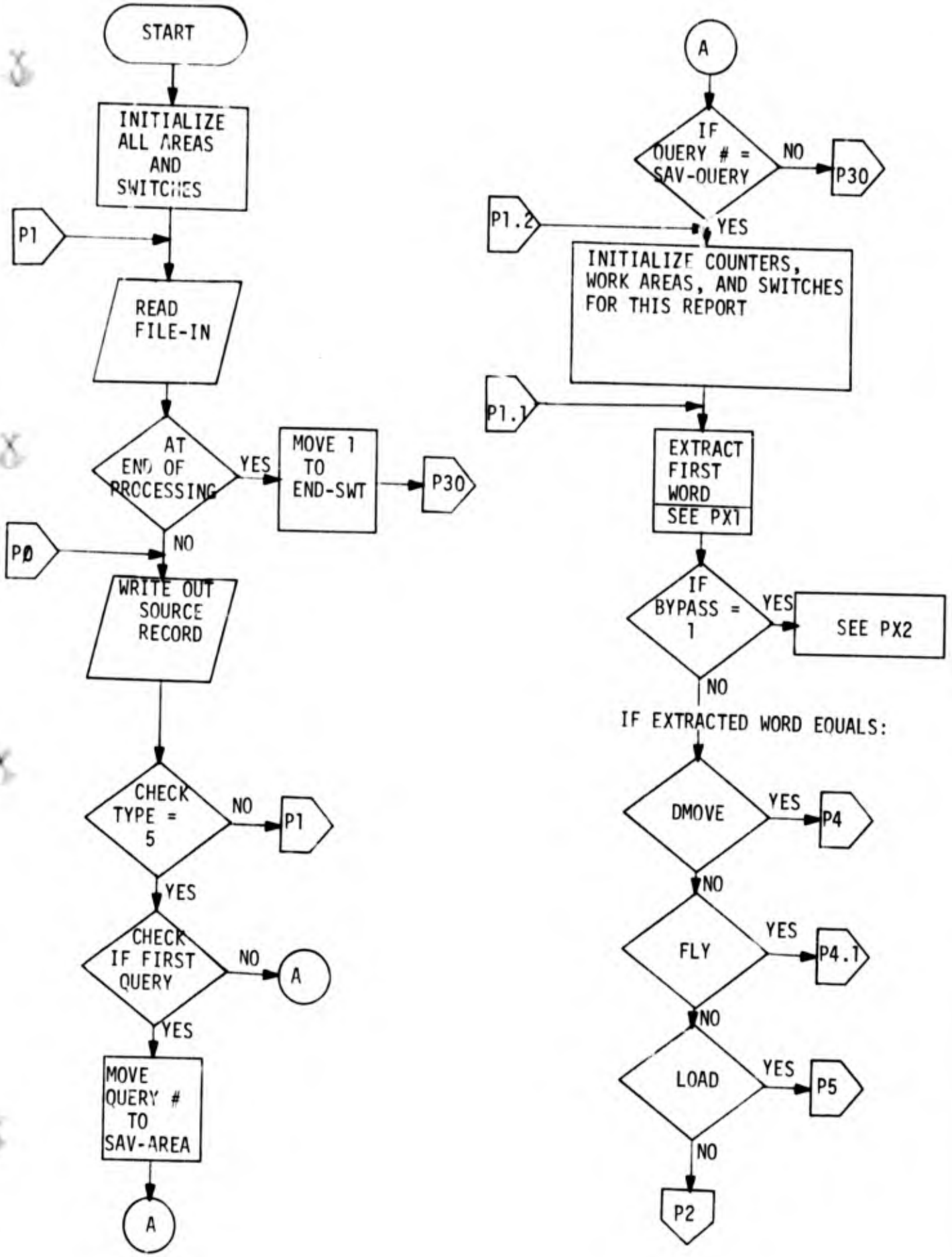

P16A
BUILD MOVE STATEMENTS CALCULATE SPACING
PARA. STATEMENTS GENERATED
YES
NO
GENERATE PARAGRAPH STATEMENTS
ANY LOGIC STATEMENTS GENERATED
NO
YES
WRITE GENERATED LOGIC STATEMENTS
WRITE MOVE STATEMENTS
P1A

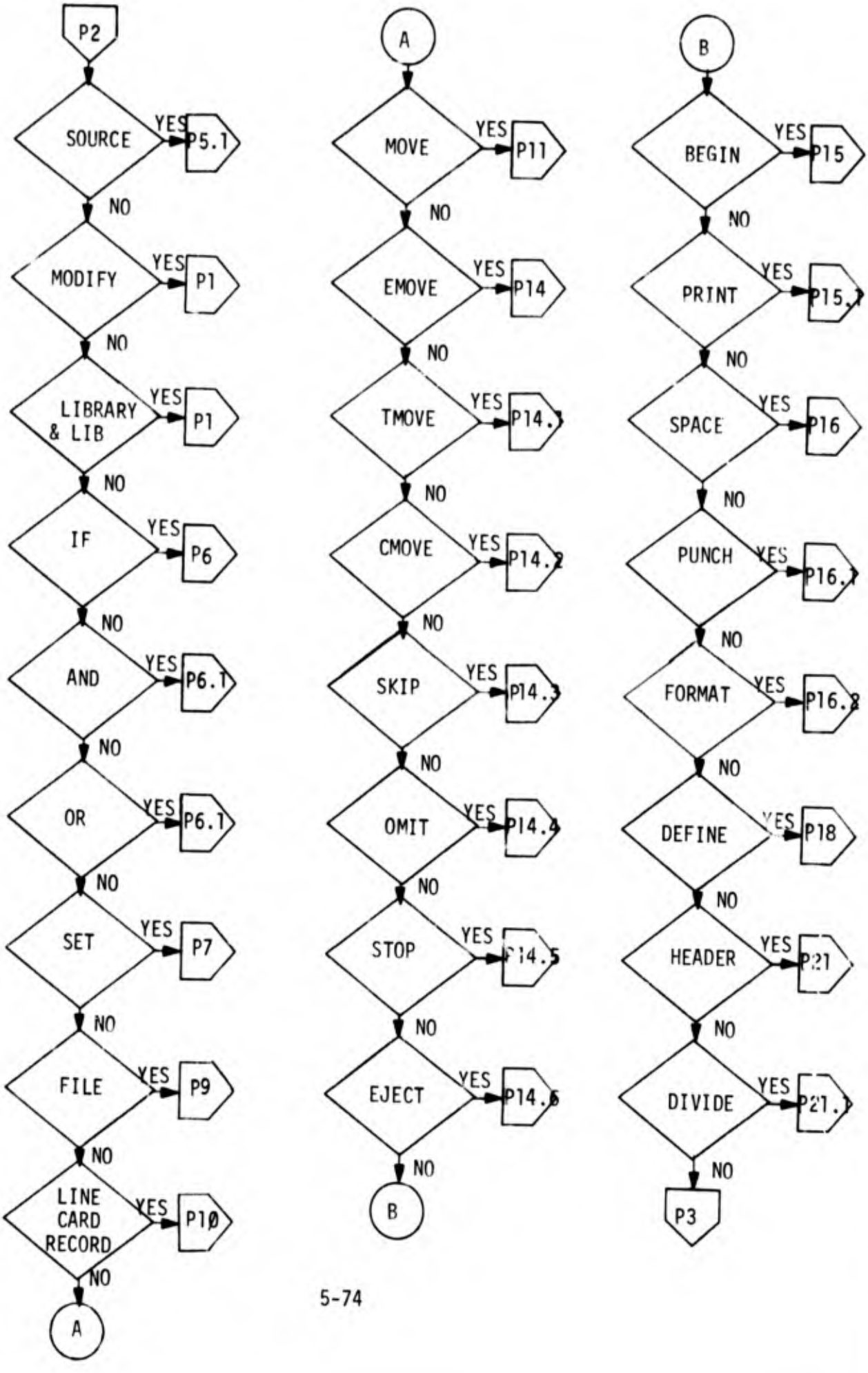

P17A
COMPUTE CENTERING POSITION FOR CLASSIFICATION
GENERATE AND SAVE CLASS MOVE STATEMENT
GENERATE AND SAVE DATE AND PAGE NO. STATEMENT
GENERATE AND SAVE SPACE AND PRINT STATEMENTS
P1A

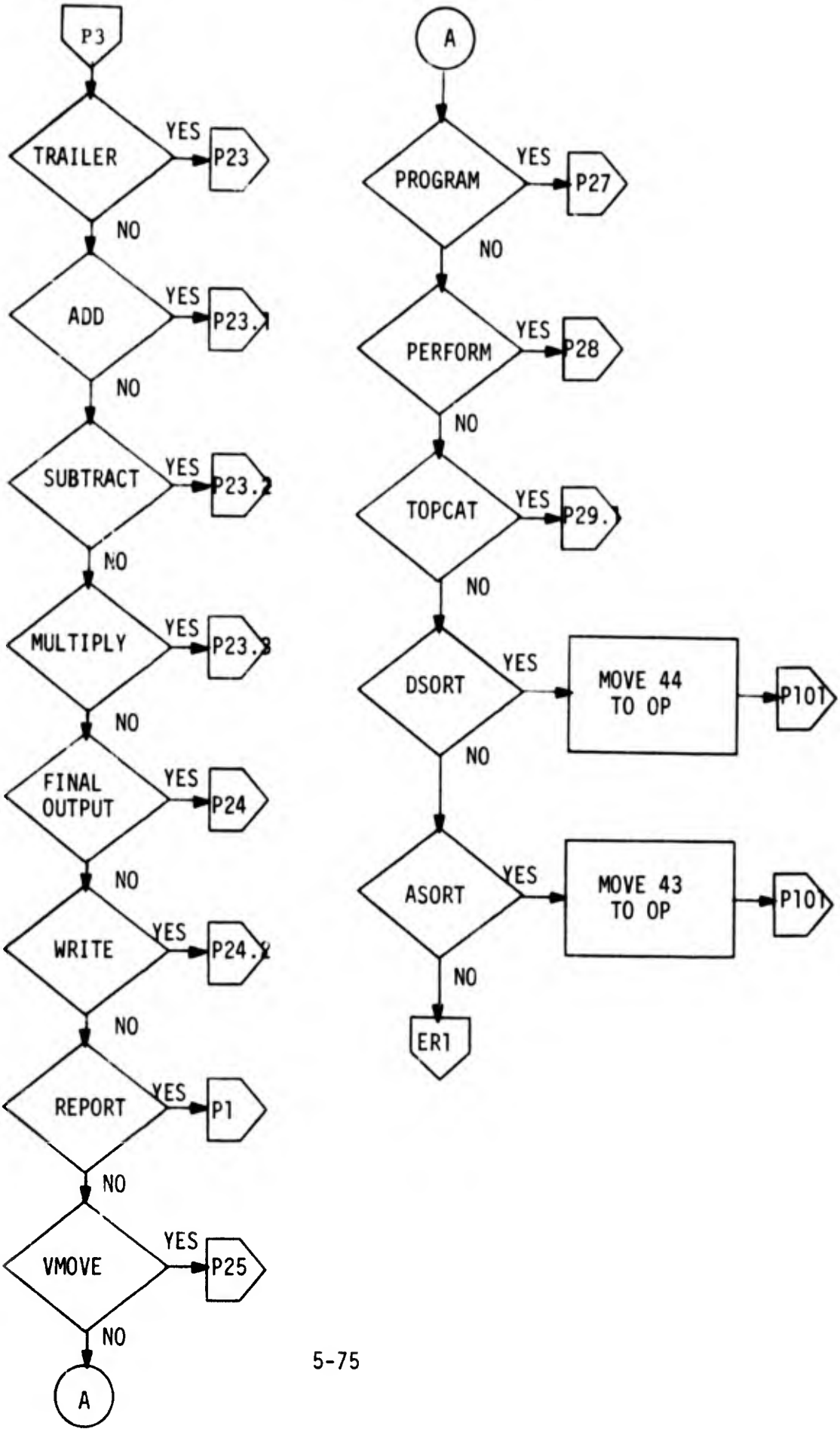

P17A
COMPUTE CENTERING POSITION FOR CLASSIFICATION
GENERATE AND SAVE CLASS MOVE STATEMENT
GENERATE AND SAVE DATE AND PAGE NO. STATEMENT
GENERATE AND SAVE SPACE AND PRINT STATEMENTS
P1A

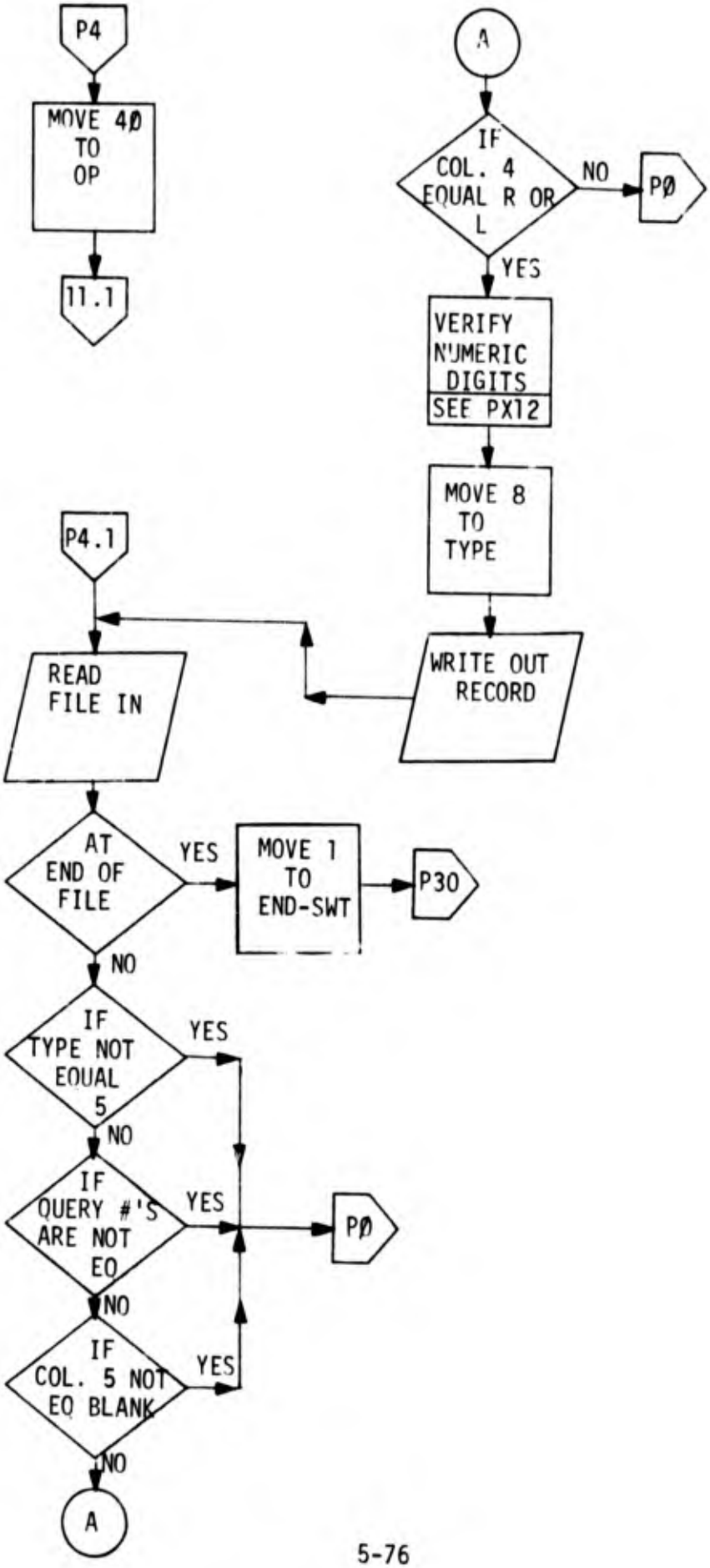

P18A
PARA. STATEMENTS GENERATED
YES
NO
GENERATE PARAGRAPH STATEMENTS
GENERATE DEFINE STATEMENTS
WRITE DEFINE STATEMENTS
GENERATE ARITHMETIC STATEMENTS
WRITE ARITHMETIC STATEMENTS
BUILD AND SAVE MOVE STATEMENTS FOR FINAL OUTPUT
P1A

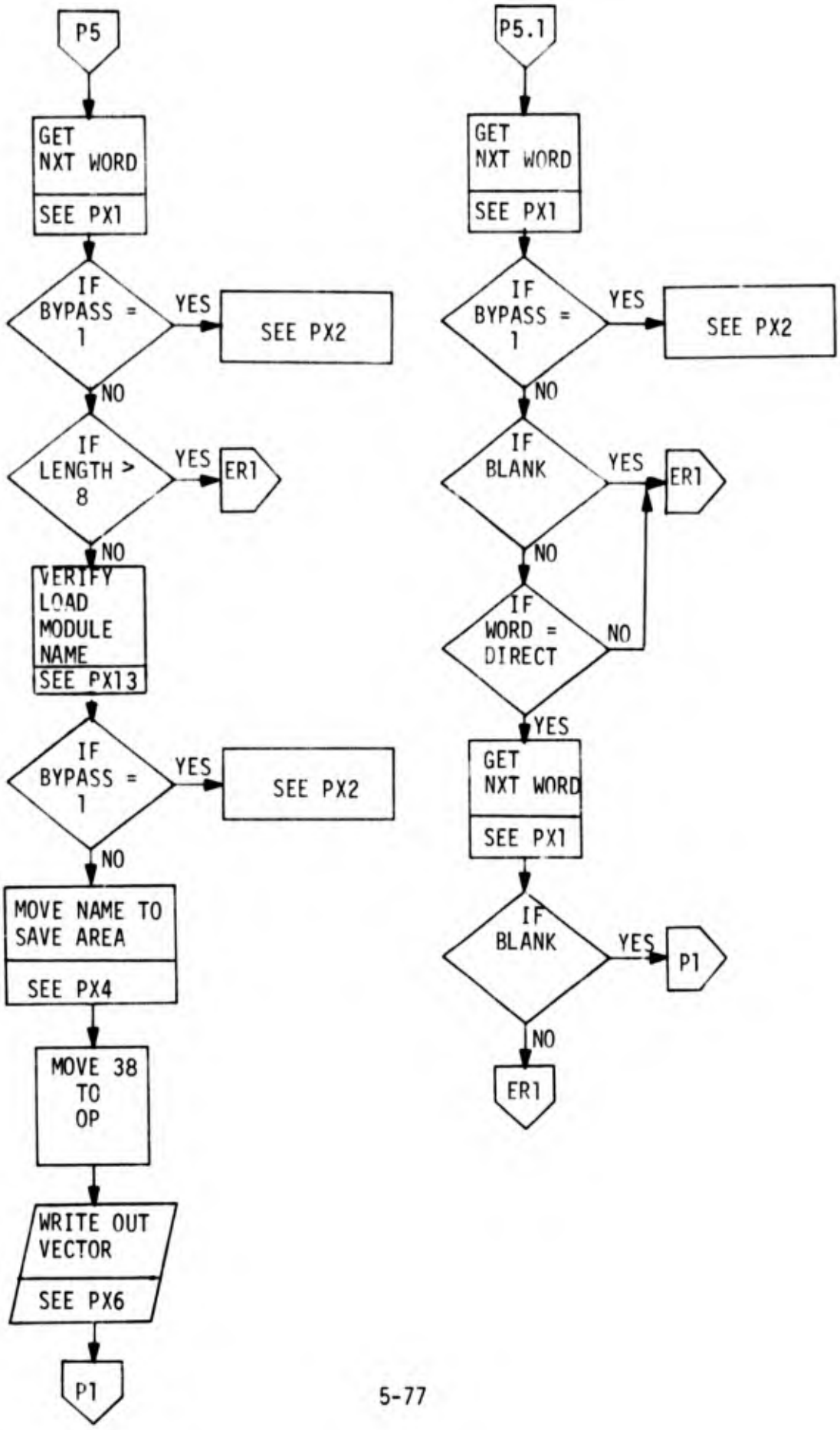
(10) GEN3.

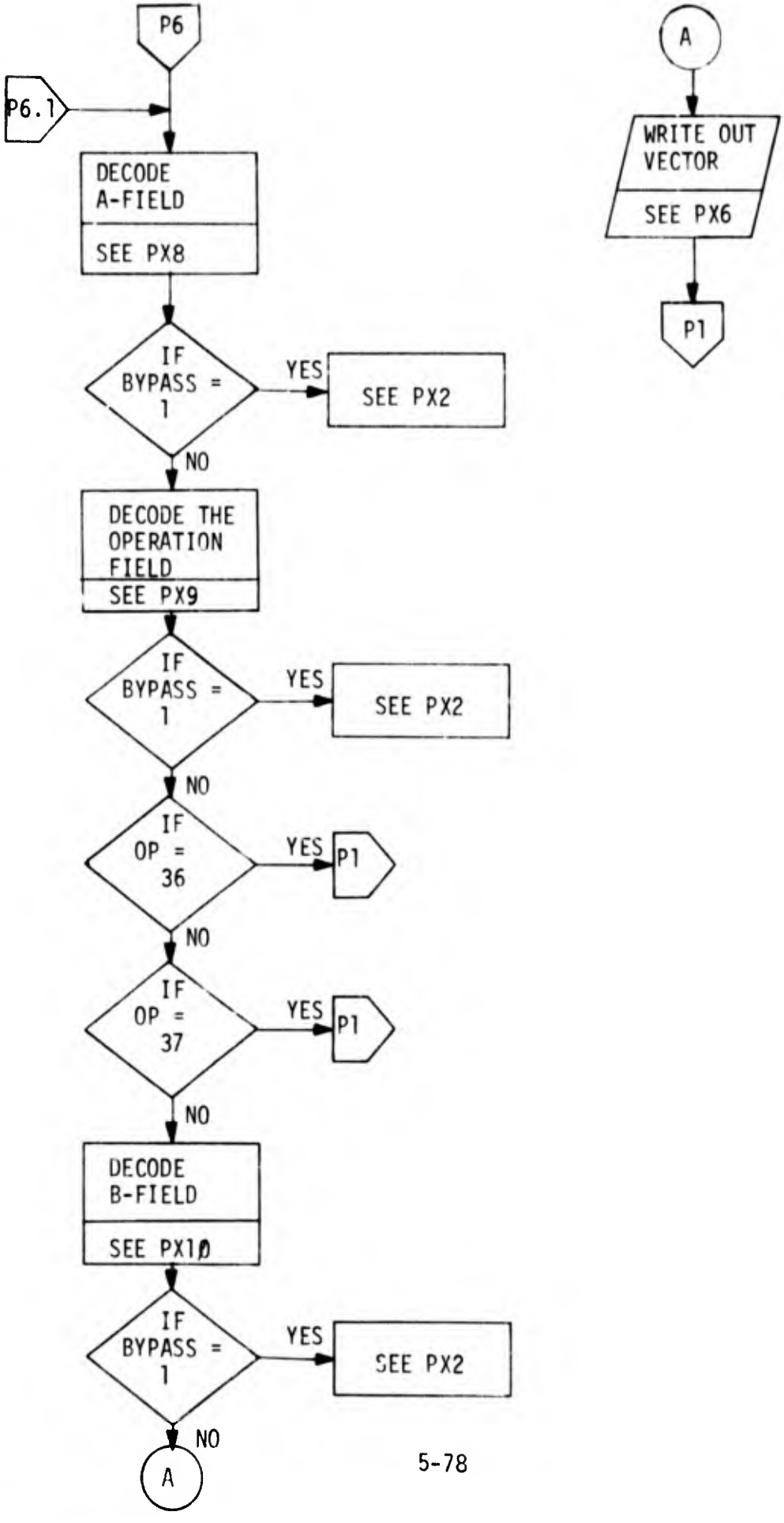

START
INITIALIZE ALL AREAS AND SWITCHES
P1
READ FILE-IN
AT END OF PROCESSING
YES
MOVE 1 TO END-SWT
P30
P0
NO
WRITE OUT SOURCE RECORD
CHECK TYPE = 5
NO
P1
YES
CHECK IF FIRST QUERY
NO
A
YES
MOVE QUERY # TO SAV-AREA
A
A
IF QUERY # = SAV-QUERY
NO
P30
P1.2
YES
INITIALIZE COUNTERS, WORK AREAS, AND SWITCHES FOR THIS REPORT
P1.1
EXTRACT FIRST WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF EXTRACTED WORD EQUALS:
DMOVE
YES
P4
NO
FLY
YES
P4.1
NO
LOAD
YES
P5
NO
P2

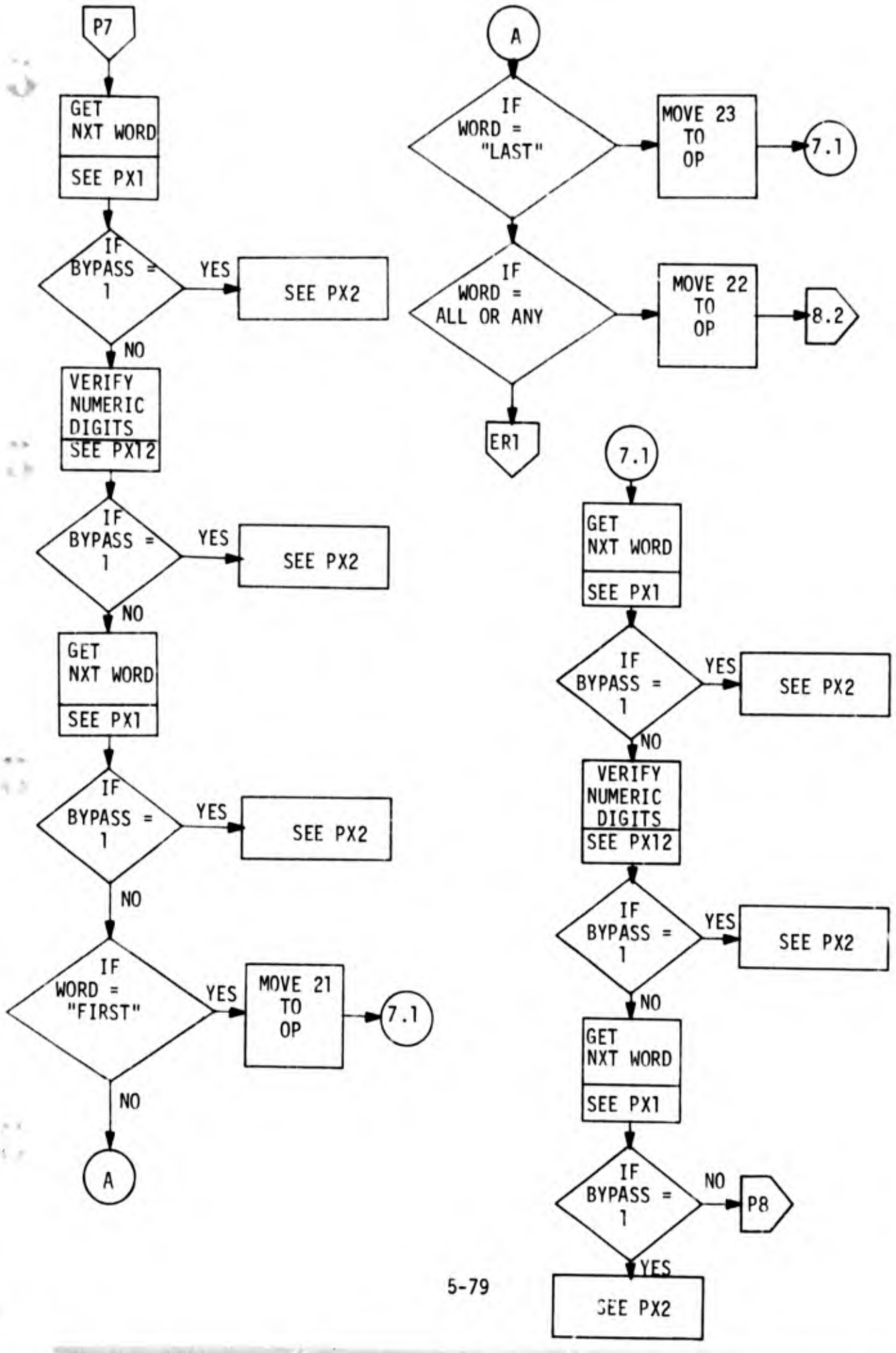

P2
SOURCE
YES
P5.1
NO
MODIFY
YES
P1
NO
LIBRARY & LIB
YES
P1
NO
IF
YES
P6
NO
AND
YES
P6.1
NO
OR
YES
P6.1
NO
SET
YES
P7
NO
FILE
YES
P9
NO
LINE CARD RECORD
YES
P10
NO
A
A
MOVE
YES
P11
NO
EMOVE
YES
P14
NO
TMOVE
YES
P14.1
NO
CMOVE
YES
P14.2
NO
SKIP
YES
P14.3
NO
OMIT
YES
P14.4
NO
STOP
YES
P14.5
NO
EJECT
YES
P14.6
NO
B
B
BEGIN
YES
P15
NO
PRINT
YES
P15.1
NO
SPACE
YES
P16
NO
PUNCH
YES
P16.1
NO
FORMAT
YES
P16.2
NO
DEFINE
YES
P18
NO
HEADER
YES
P21
NO
DIVIDE
YES
P21.1
NO
P3

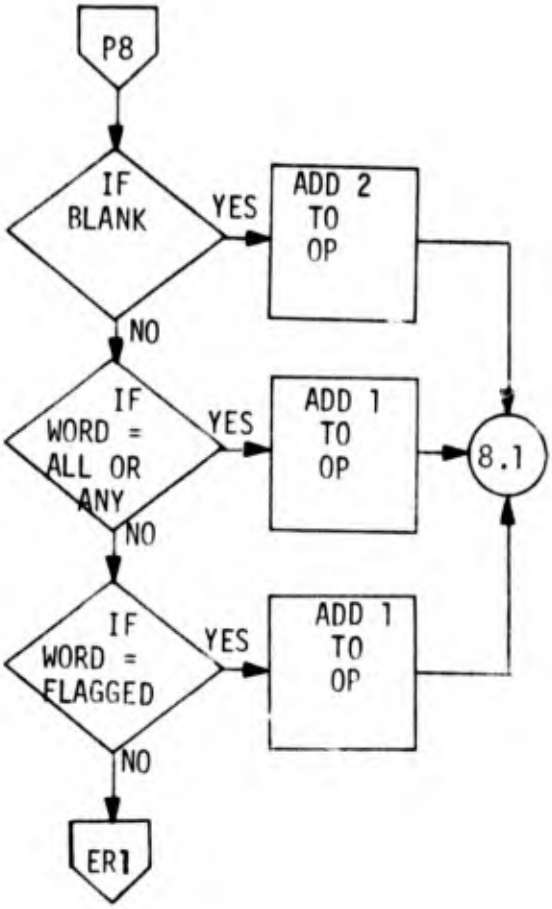

P3
TRAILER
YES
P23
NO
ADD
YES
P23.1
NO
SUBTRACT
YES
P23.2
NO
MULTIPLY
YES
P23.3
NO
FINAL OUTPUT
YES
P24
NO
WRITE
YES
P24.1
NO
REPORT
YES
P1
NO
VMOVE
YES
P25
NO
A
A
PROGRAM
YES
P27
NO
PERFORM
YES
P28
NO
TOPCAT
YES
P29.1
NO
DSORT
YES
MOVE 44 TO OP
P101
NO
ASORT
YES
MOVE 43 TO OP
P101
NO
ER1

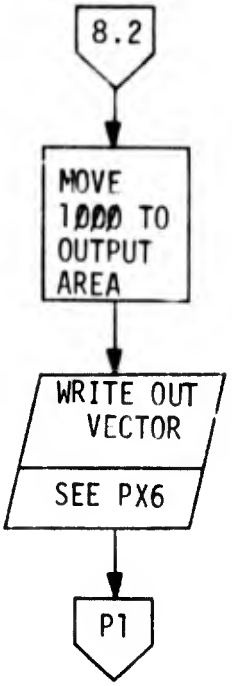

P4
MOVE 40
TO
OP
11.1
A
IF
COL. 4
EQUAL R OR
L
NO
PØ
YES
VERIFY
NUMERIC
DIGITS
SEE PX12
MOVE 8
TO
TYPE
P4.1
READ
FILE IN
WRITE OUT
RECORD
AT
END OF
FILE
YES
MOVE 1
TO
END-SWT
P30
NO
IF
TYPE NOT
EQUAL
5
YES
NO
IF
QUERY #'S
ARE NOT
EQ
YES
PØ
NO
IF
COL. 5 NOT
EQ BLANK
YES
NO
A

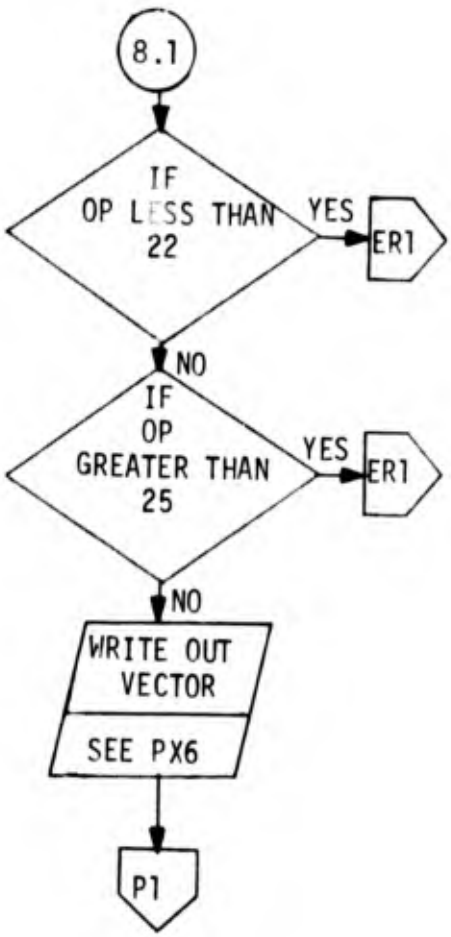

P5
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF LENGTH > 8
YES
ER1
NO
VERIFY LOAD MODULE NAME
SEE PX13
IF BYPASS = 1
YES
SEE PX2
NO
MOVE NAME TO SAVE AREA
SEE PX4
MOVE 38 TO OP
WRITE OUT VECTOR
SEE PX6
P1
P5.1
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF BLANK
YES
ER1
NO
IF WORD = DIRECT
NO
YES
GET NXT WORD
SEE PX1
IF BLANK
YES
P1
NO
ER1

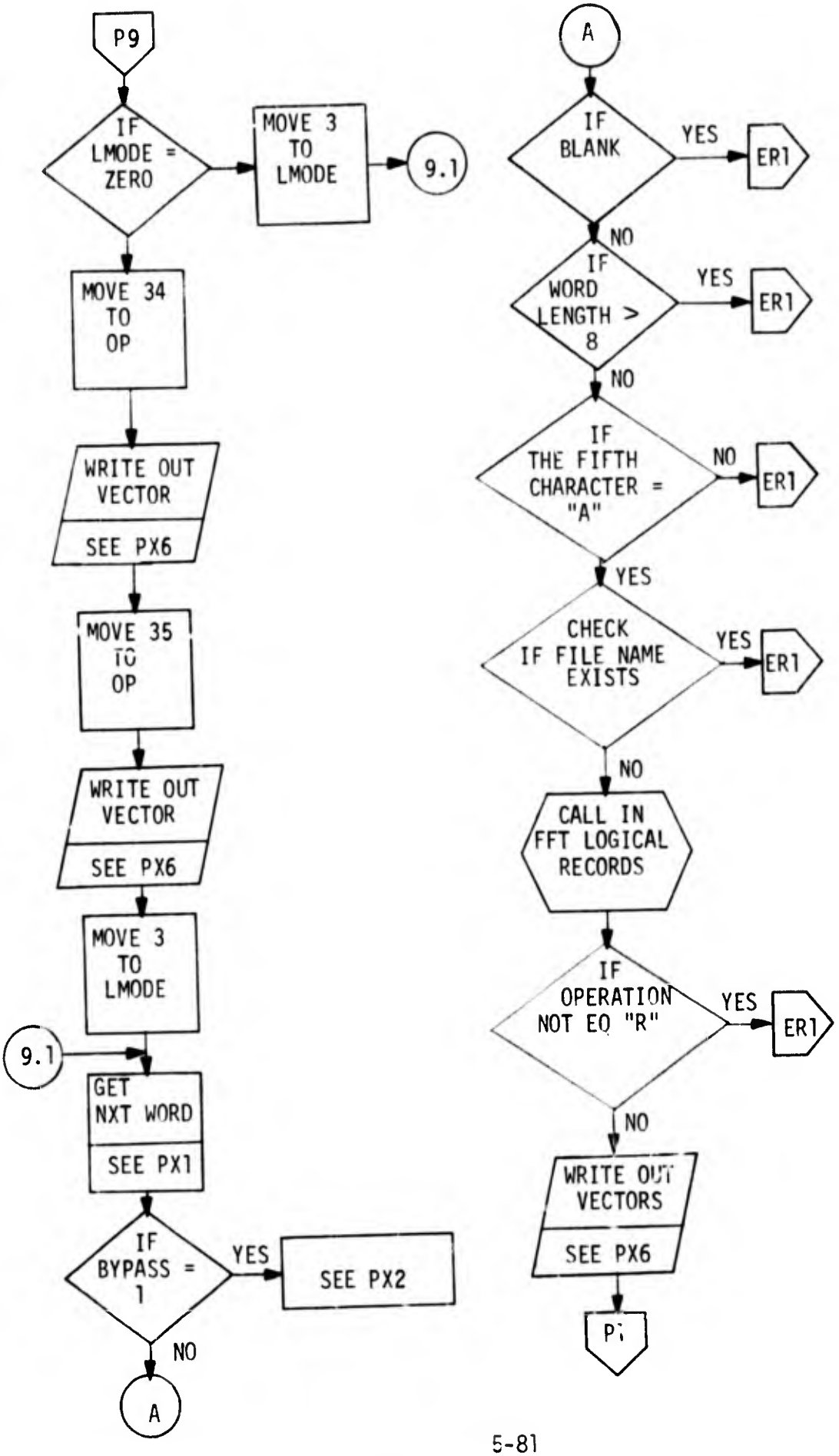

P6
P6.1
DECODE A-FIELD
SEE PX8
IF BYPASS = 1
YES
SEE PX2
NO
DECODE THE OPERATION FIELD
SEE PX9
IF BYPASS = 1
YES
SEE PX2
NO
IF OP = 36
YES
P1
NO
IF OP = 37
YES
P1
NO
DECODE B-FIELD
SEE PX10
IF BYPASS = 1
YES
SEE PX2
NO
A
A
WRITE OUT VECTOR
SEE PX6
P1


P7
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
VERIFY NUMERIC DIGITS
SEE PX12
IF BYPASS = 1
YES
SEE PX2
NO
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF WORD = "FIRST"
YES
MOVE 21 TO OP
7.1
NO
A
A
IF WORD = "LAST"
MOVE 23 TO OP
7.1
IF WORD = ALL OR ANY
MOVE 22 TO OP
8.2
ER1
7.1
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
VERIFY NUMERIC DIGITS
SEE PX12
IF BYPASS = 1
YES
SEE PX2
NO
GET NXT WORD
SEE PX1
IF BYPASS = 1
NO
P8
YES
SEE PX2

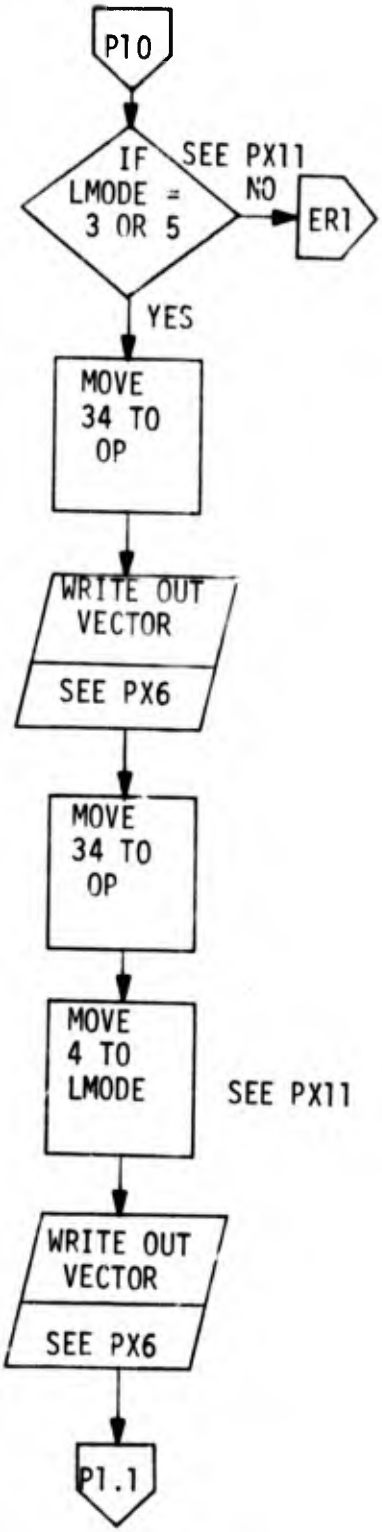

P8
IF BLANK
YES
ADD 2 TO OP
NO
IF WORD = ALL OR ANY
YES
ADD 1 TO OP
8.1
NO
IF WORD = FLAGGED
YES
ADD 1 TO OP
NO
ER1

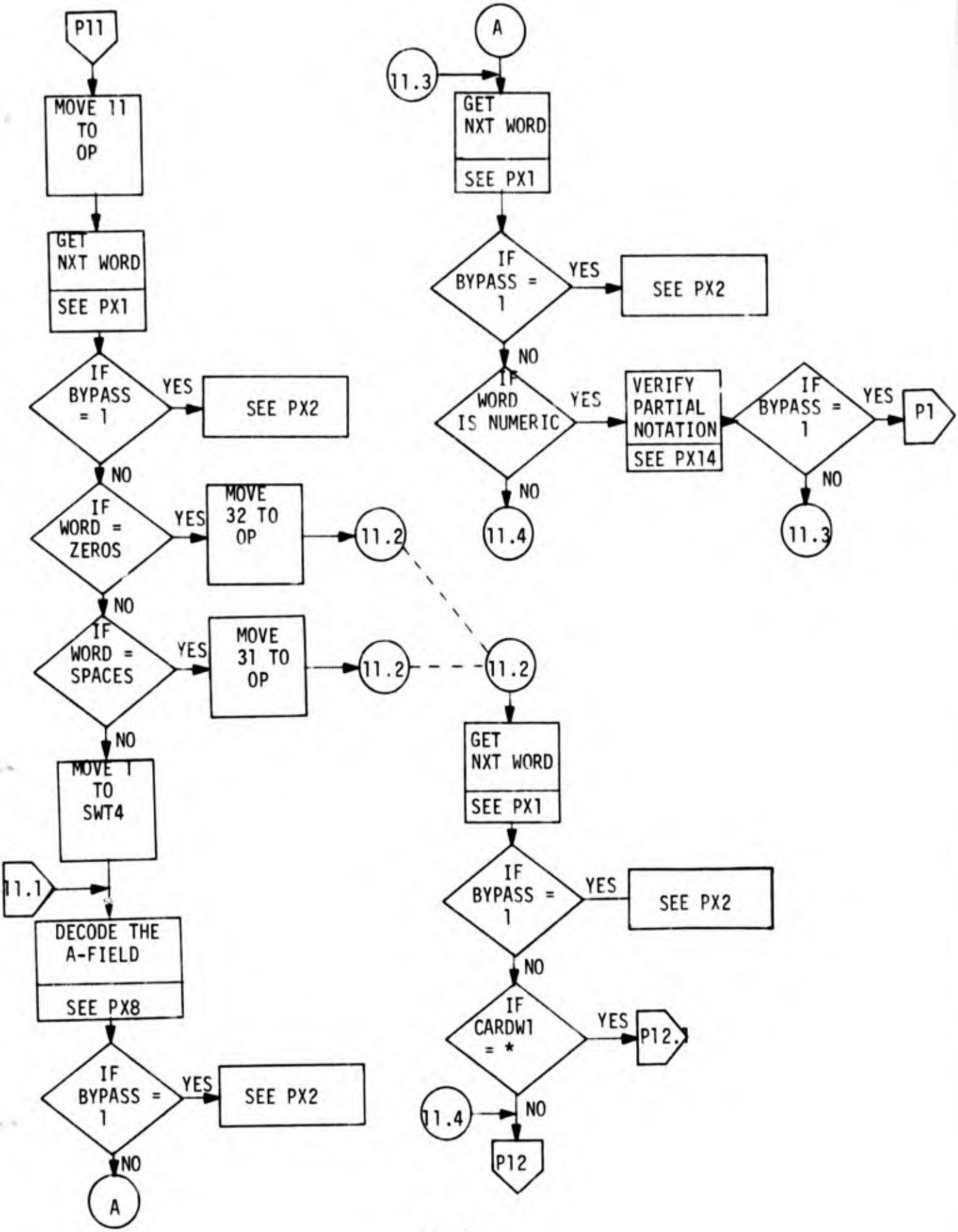

8.2
MOVE 1000 TO OUTPUT AREA
WRITE OUT VECTOR
SEE PX6
P1

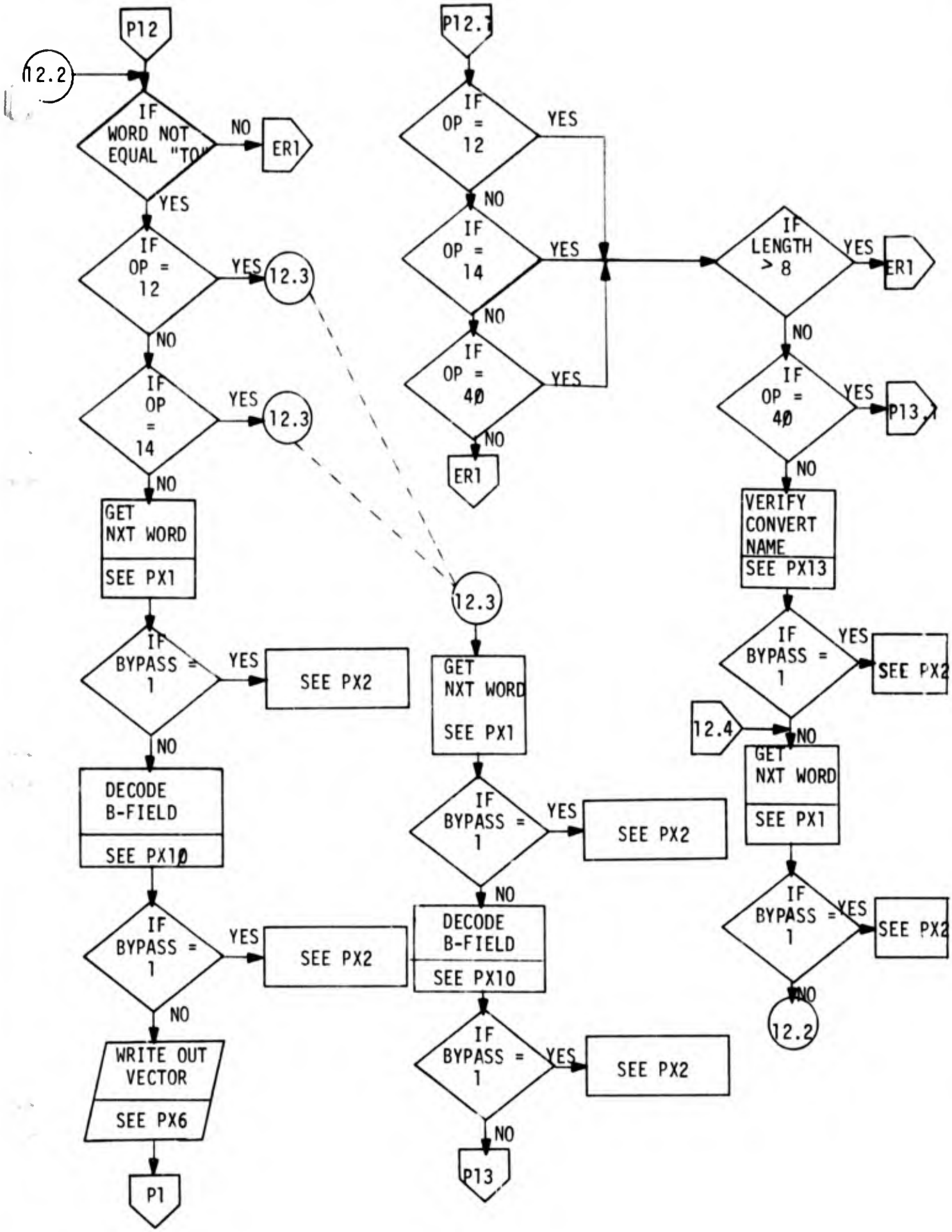

8.1
IF OP LESS THAN 22
YES
ER1
NO
IF OP GREATER THAN 25
YES
ER1
NO
WRITE OUT VECTOR
SEE PX6
P1

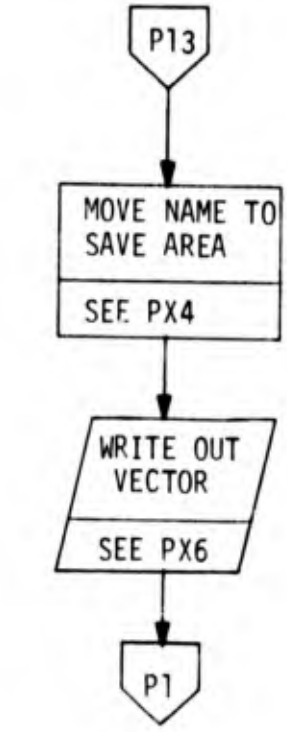


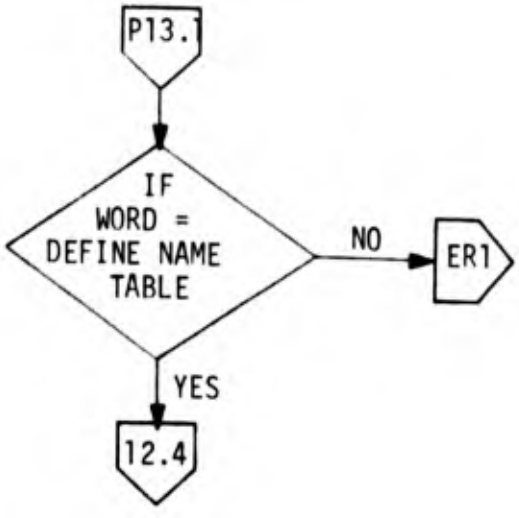

IF THE RETURN FIELD, "OPERATION", FROM THE CALLED PROGRAM CONTAINS ANYTHING OTHER THAN AN "R", AN ERROR EXISTS.

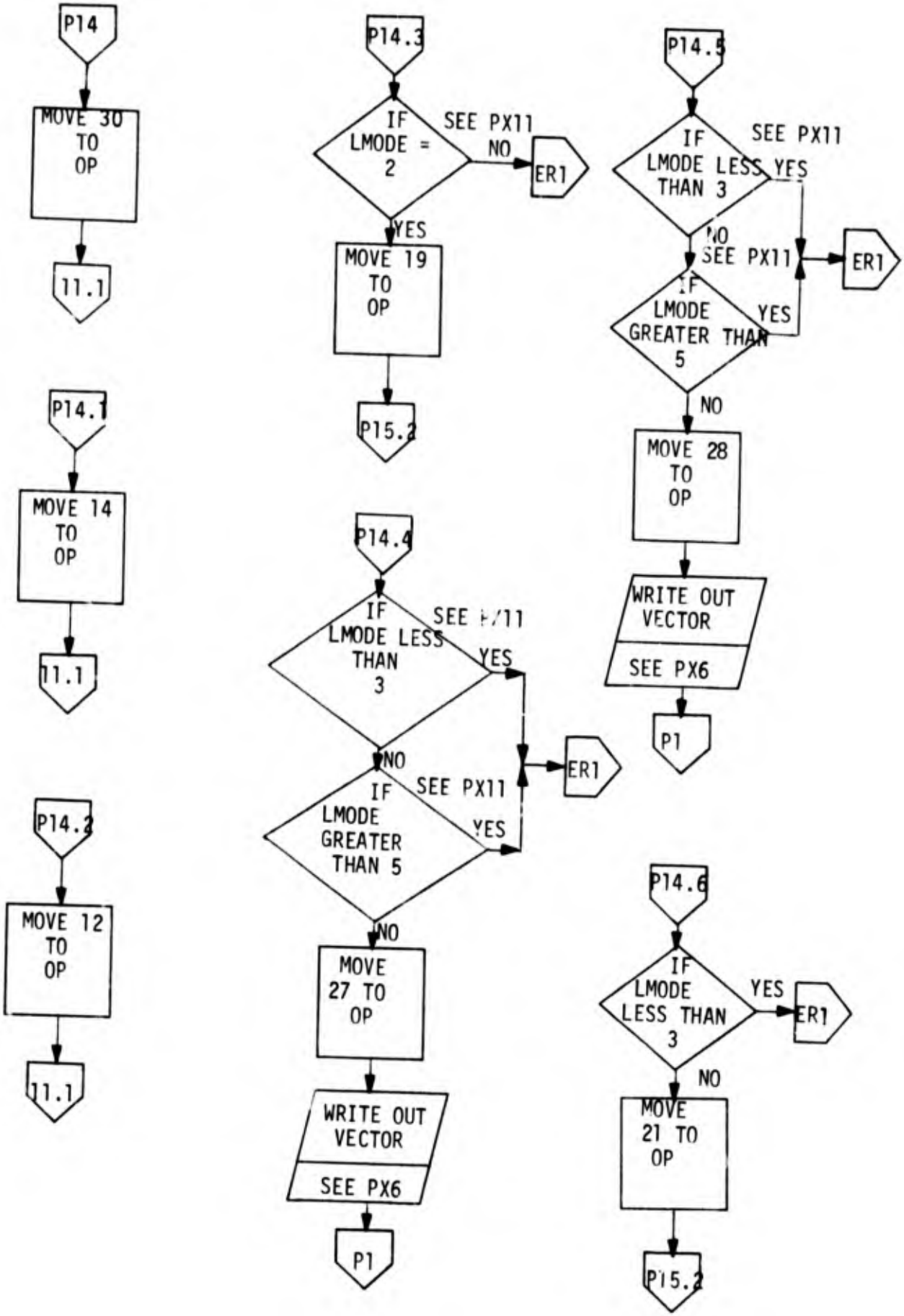

P10
IF LMODE = 3 OR 5
SEE PX11
NO
ER1
YES
MOVE 34 TO OP
WRITE OUT VECTOR
SEE PX6
MOVE 34 TO OP
MOVE 4 TO LMODE
SEE PX11
WRITE OUT VECTOR
SEE PX6
P1.1

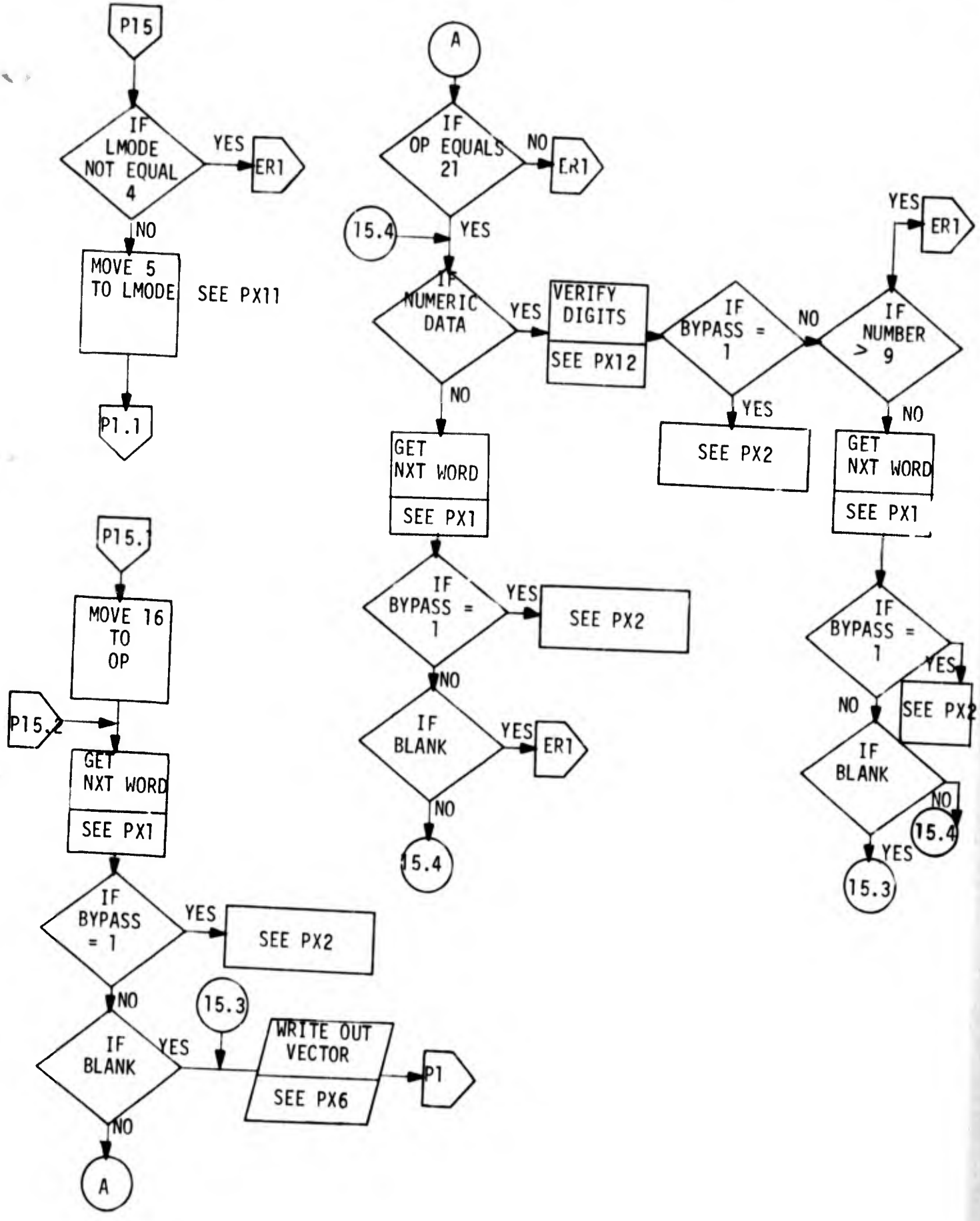

P11
MOVE 11 TO OP
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF WORD = ZEROS
YES
MOVE 32 TO OP
11.2
NO
IF WORD = SPACES
YES
MOVE 31 TO OP
11.2
NO
MOVE 1 TO SWT4
11.1
DECODE THE A-FIELD
SEE PX8
IF BYPASS = 1
YES
SEE PX2
NO
A
A
11.3
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF WORD IS NUMERIC
YES
VERIFY PARTIAL NOTATION
SEE PX14
IF BYPASS = 1
YES
P1
NO
11.3
NO
11.4
11.2
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF CARDW1 = *
YES
P12.1
11.4
NO
P12

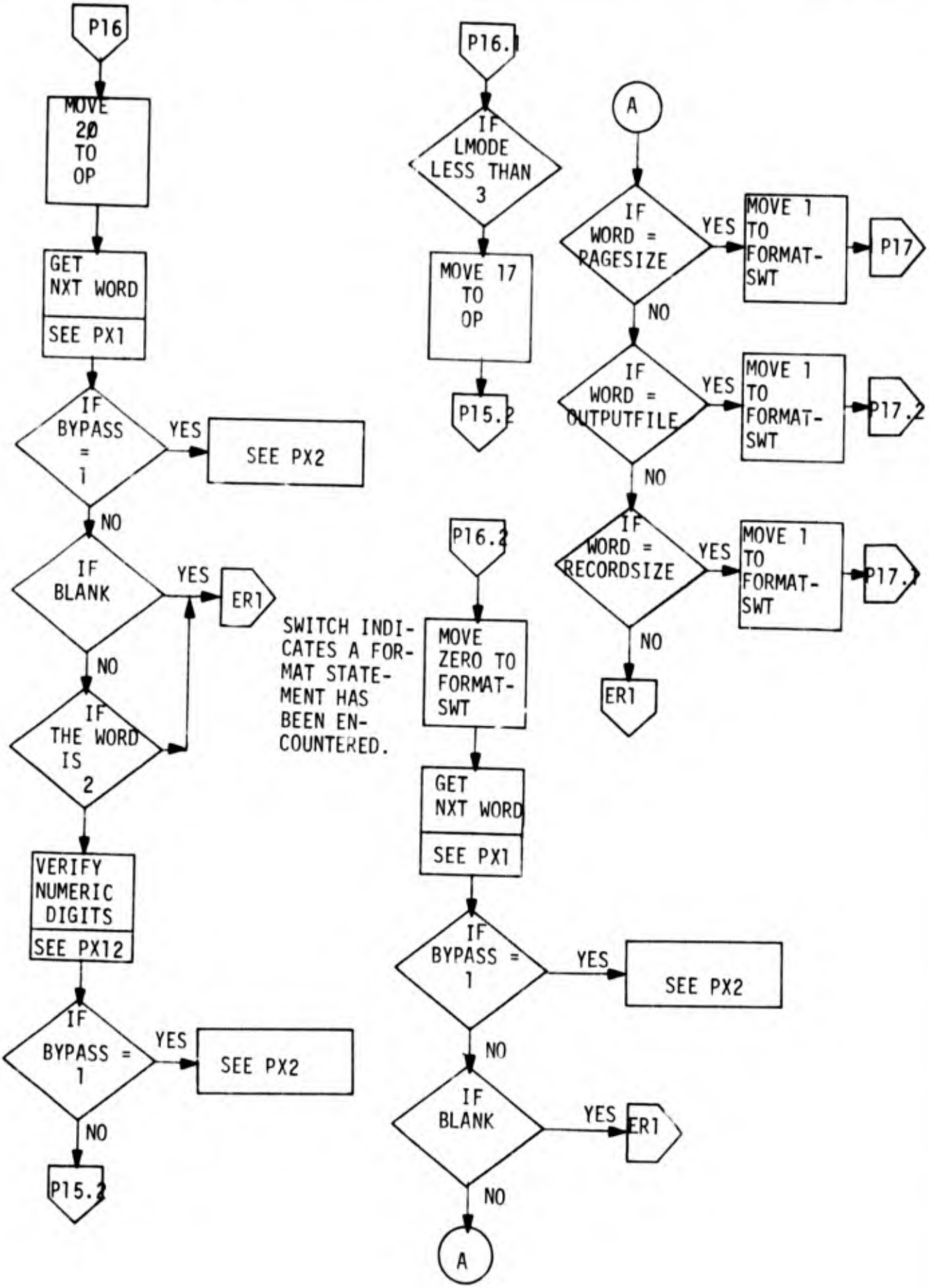

P12
12.2
IF WORD NOT EQUAL "TO"
NO
ER1
YES
IF OP = 12
YES
12.3
NO
IF OP = 14
YES
12.3
NO
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
DECODE B-FIELD
SEE PX10
IF BYPASS = 1
YES
SEE PX2
NO
WRITE OUT VECTOR
SEE PX6
P1
P12.3
IF OP = 12
YES
NO
IF OP = 14
YES
NO
IF OP = 40
YES
NO
ER1
12.3
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
DECODE B-FIELD
SEE PX10
IF BYPASS = 1
YES
SEE PX2
NO
P13
IF LENGTH > 8
YES
ER1
NO
IF OP = 40
YES
P13.1
NO
VERIFY CONVERT NAME
SEE PX13
IF BYPASS = 1
YES
SEE PX2
12.4
NO
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
12.2

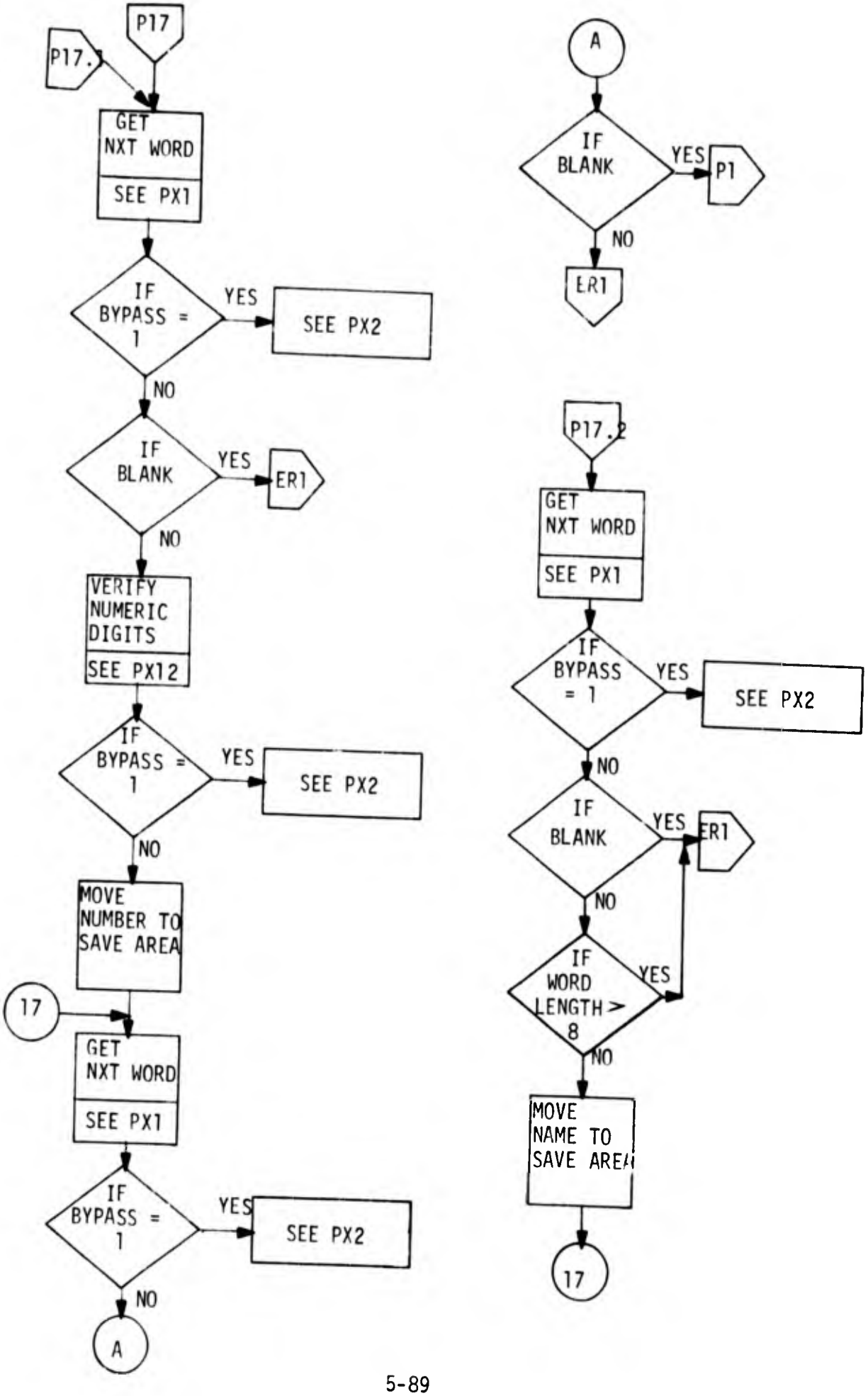

P13
MOVE NAME TO SAVE AREA
SEE PX4
WRITE OUT VECTOR
SEE PX6
P1

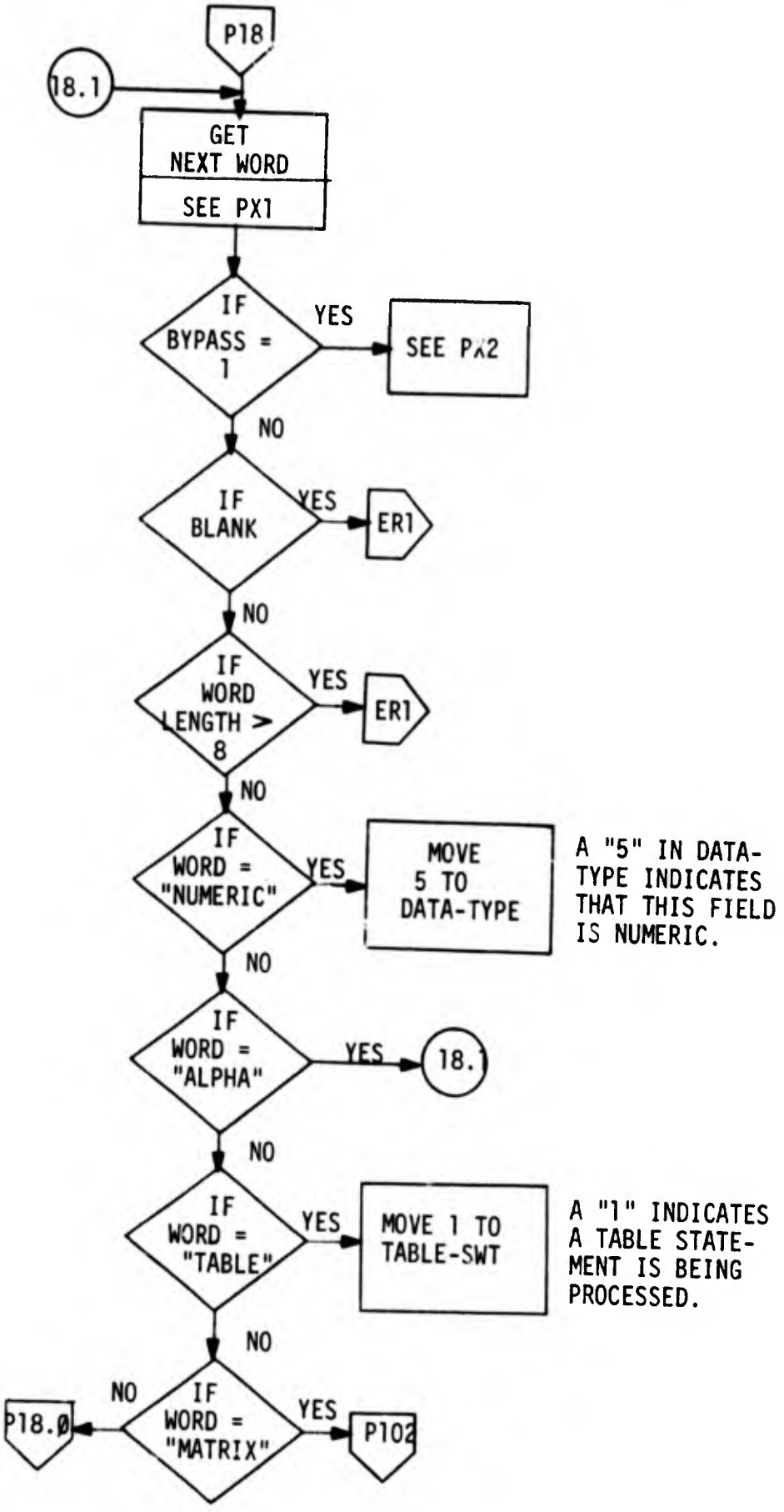

P13.1
IF WORD = DEFINE NAME TABLE
NO
ER1
YES
12.4

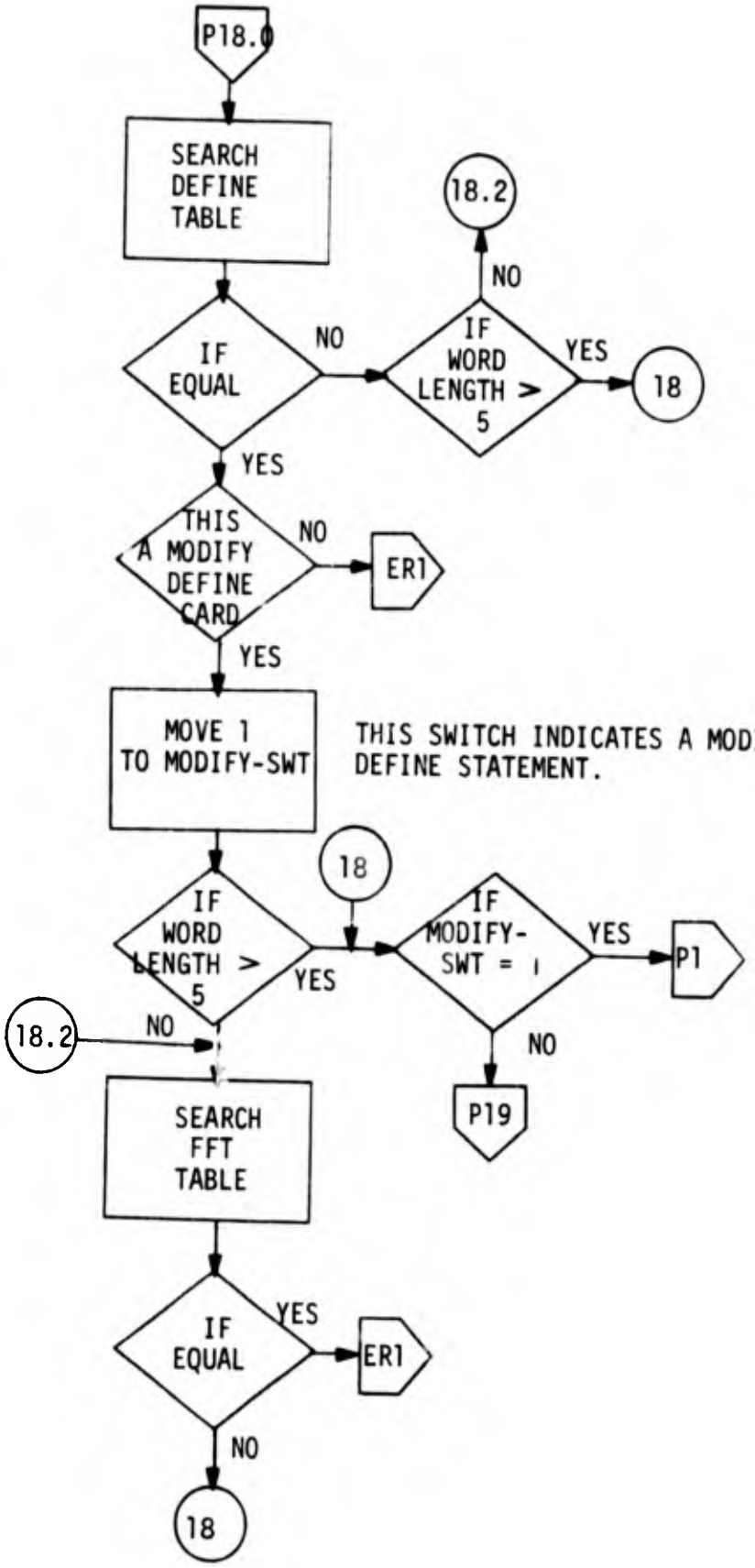

P14
MOVE 30 TO OP
11.1
P14.1
MOVE 14 TO OP
11.1
P14.2
MOVE 12 TO OP
11.1
P14.3
IF LMODE = 2
SEE PX11
NO
ER1
YES
MOVE 19 TO OP
P15.2
P14.4
IF LMODE LESS THAN 3
SEE PX11
YES
NO
ER1
IF LMODE GREATER THAN 5
SEE PX11
YES
NO
MOVE 27 TO OP
WRITE OUT VECTOR
SEE PX6
P1
P14.5
IF LMODE LESS THAN 3
SEE PX11
YES
NO
SEE PX11
ER1
IF LMODE GREATER THAN 5
YES
NO
MOVE 28 TO OP
WRITE OUT VECTOR
SEE PX6
P1
P14.6
IF LMODE LESS THAN 3
YES
ER1
NO
MOVE 21 TO OP
P15.2

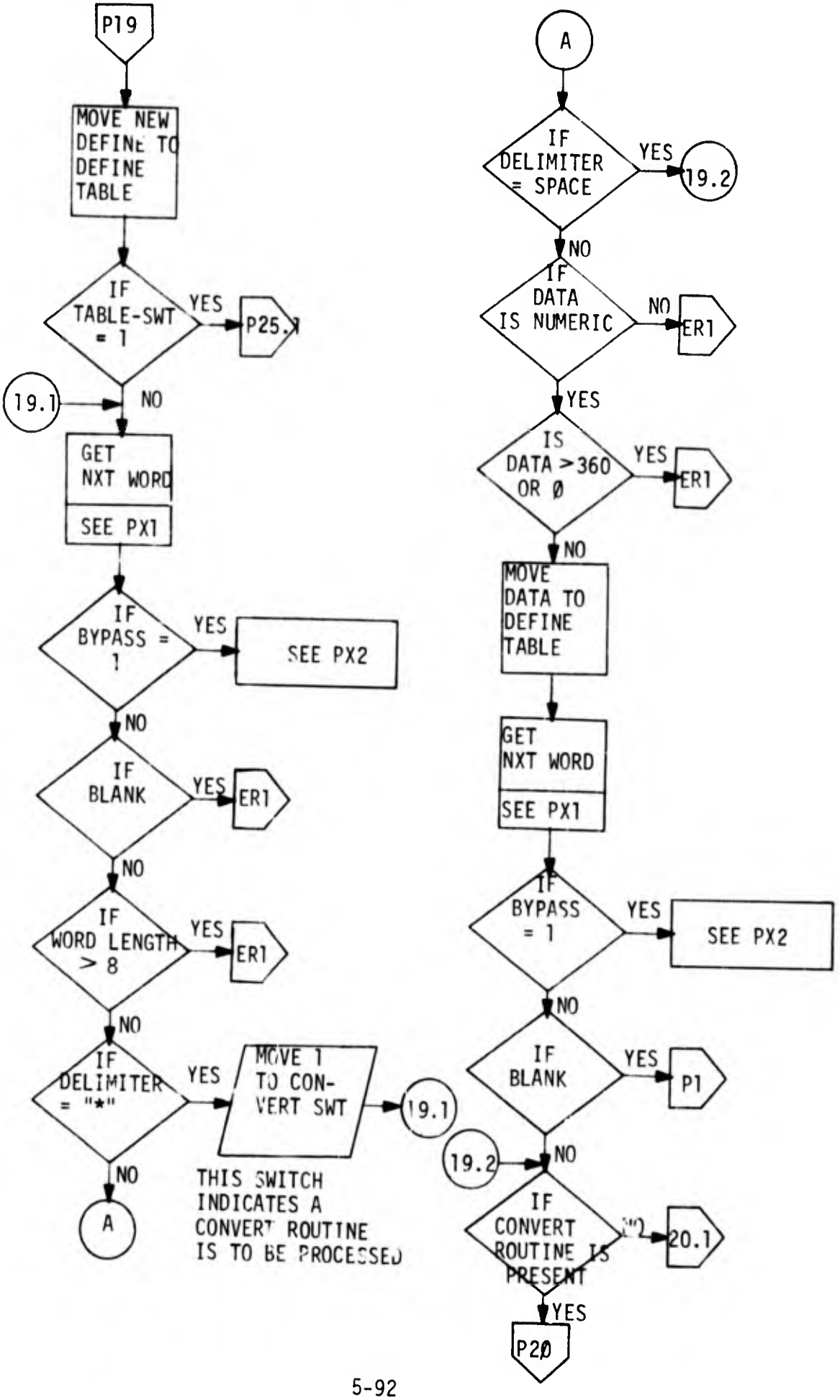

P15
IF LMODE NOT EQUAL 4
YES
ER1
NO
MOVE 5 TO LMODE
SEE PX11
P1.1
A
IF OP EQUALS 21
NO
ER1
15.4
YES
IF NUMERIC DATA
YES
VERIFY DIGITS
SEE PX12
IF BYPASS = 1
NO
YES
SEE PX2
IF NUMBER > 9
YES
ER1
NO
GET NXT WORD
SEE PX1
NO
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF BLANK
YES
ER1
NO
15.4
IF BYPASS = 1
YES
SEE PX2
NO
IF BLANK
NO
15.4
YES
15.3
P15.1
MOVE 16 TO OP
P15.2
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
15.3
IF BLANK
YES
WRITE OUT VECTOR
SEE PX6
P1
NO
A

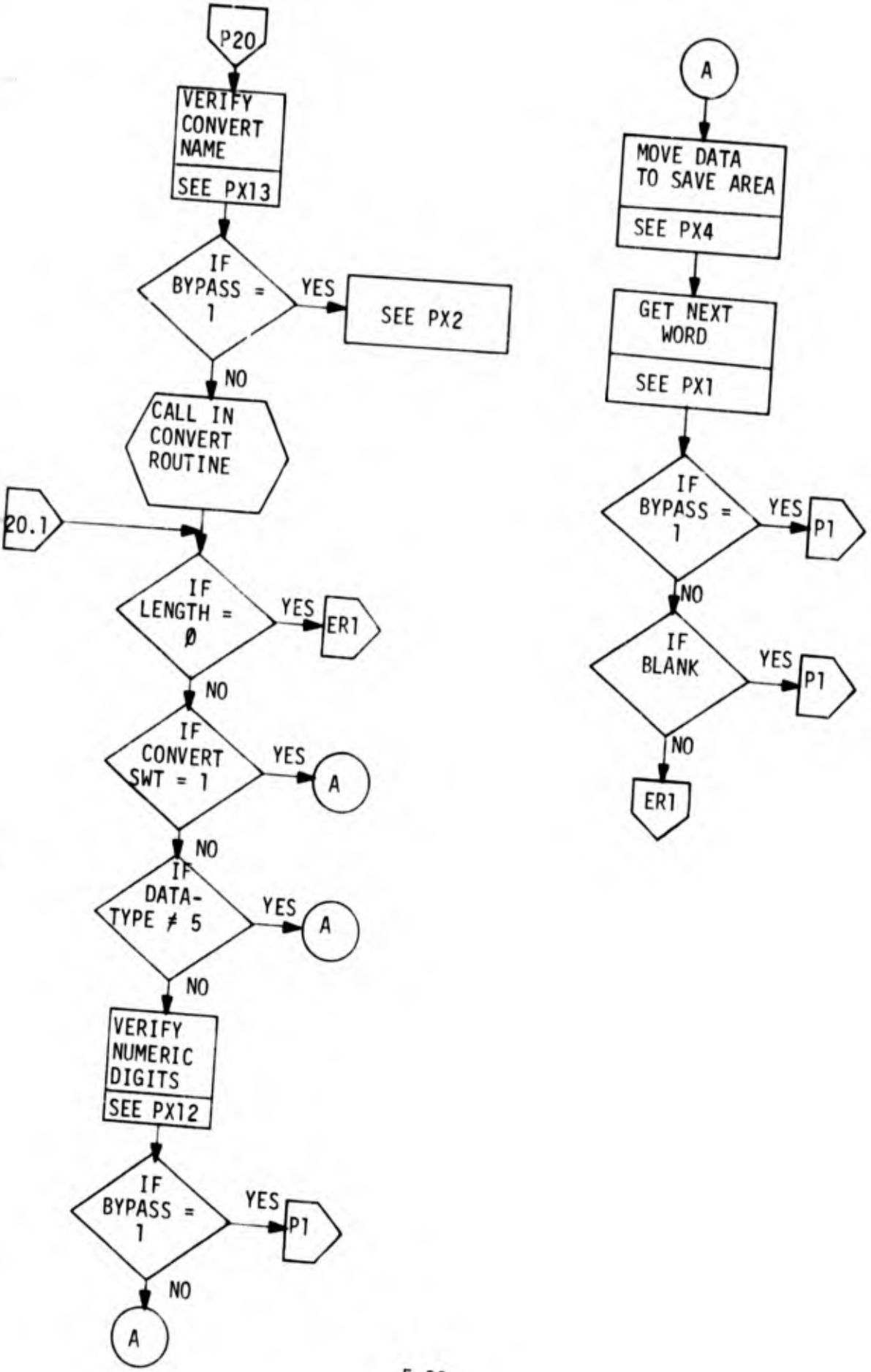

P16
MOVE 20 TO OP
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF BLANK
YES
ER1
NO
IF THE WORD IS 2
VERIFY NUMERIC DIGITS
SEE PX12
IF BYPASS = 1
YES
SEE PX2
NO
P15.2
P16.1
IF LMODE LESS THAN 3
MOVE 17 TO OP
P15.2
A
IF WORD = PAGESIZE
YES
MOVE 1 TO FORMAT-SWT
P17
NO
IF WORD = OUTPUTFILE
YES
MOVE 1 TO FORMAT-SWT
P17.2
NO
IF WORD = RECORDSIZE
YES
MOVE 1 TO FORMAT-SWT
P17.1
NO
ER1
P16.2
SWITCH INDICATES A FORMAT STATEMENT HAS BEEN ENCOUNTERED.
MOVE ZERO TO FORMAT-SWT
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF BLANK
YES
ER1
NO
A

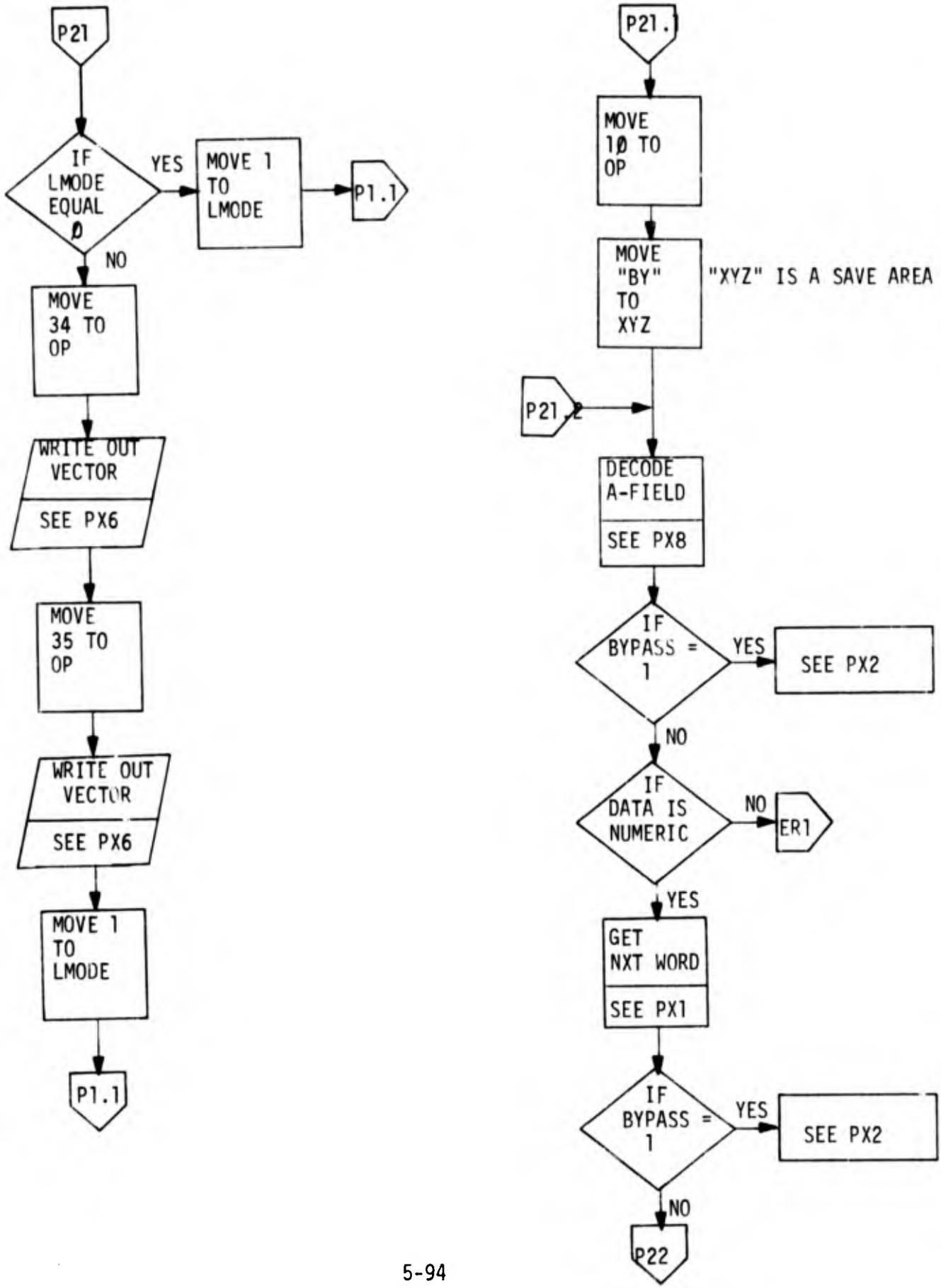

P17
P17.1
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF BLANK
YES
ER1
NO
VERIFY NUMERIC DIGITS
SEE PX12
IF BYPASS = 1
YES
SEE PX2
NO
MOVE NUMBER TO SAVE AREA
17
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
A
A
IF BLANK
YES
P1
NO
ER1
P17.2
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF BLANK
YES
ER1
NO
IF WORD LENGTH > 8
YES
NO
MOVE NAME TO SAVE AREA
17

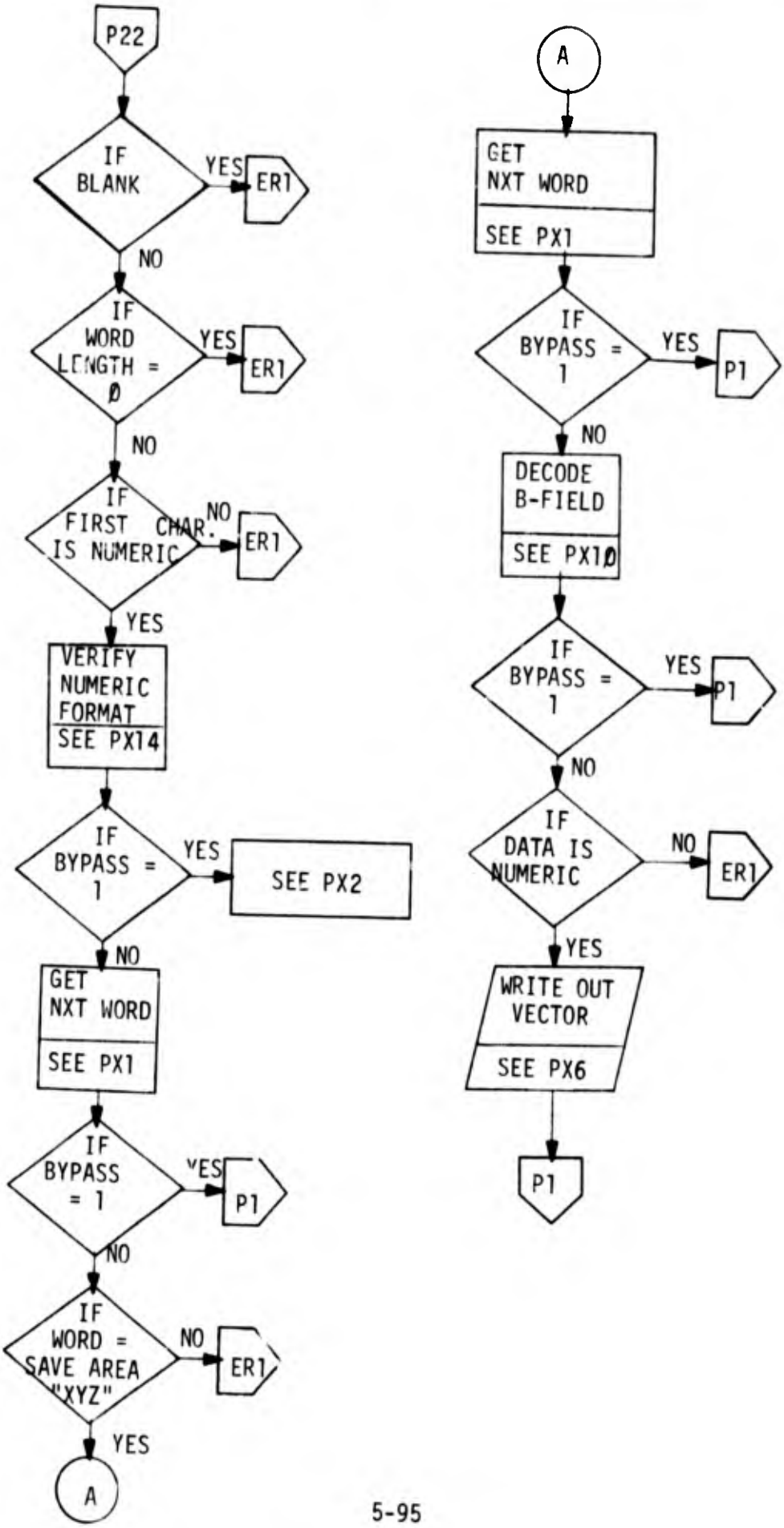

P18
18.1
GET
NEXT WORD
SEE PX1
IF
BYPASS =
1
YES
SEE PX2
NO
IF
BLANK
YES
ER1
NO
IF
WORD
LENGTH >
8
YES
ER1
NO
IF
WORD =
"NUMERIC"
YES
MOVE
5 TO
DATA-TYPE
A "5" IN DATA-TYPE INDICATES THAT THIS FIELD IS NUMERIC.
NO
IF
WORD =
"ALPHA"
YES
18.1
NO
IF
WORD =
"TABLE"
YES
MOVE 1 TO
TABLE-SWT
A "1" INDICATES A TABLE STATEMENT IS BEING PROCESSED.
NO
IF
WORD =
"MATRIX"
NO
P18.Ø
YES
P102

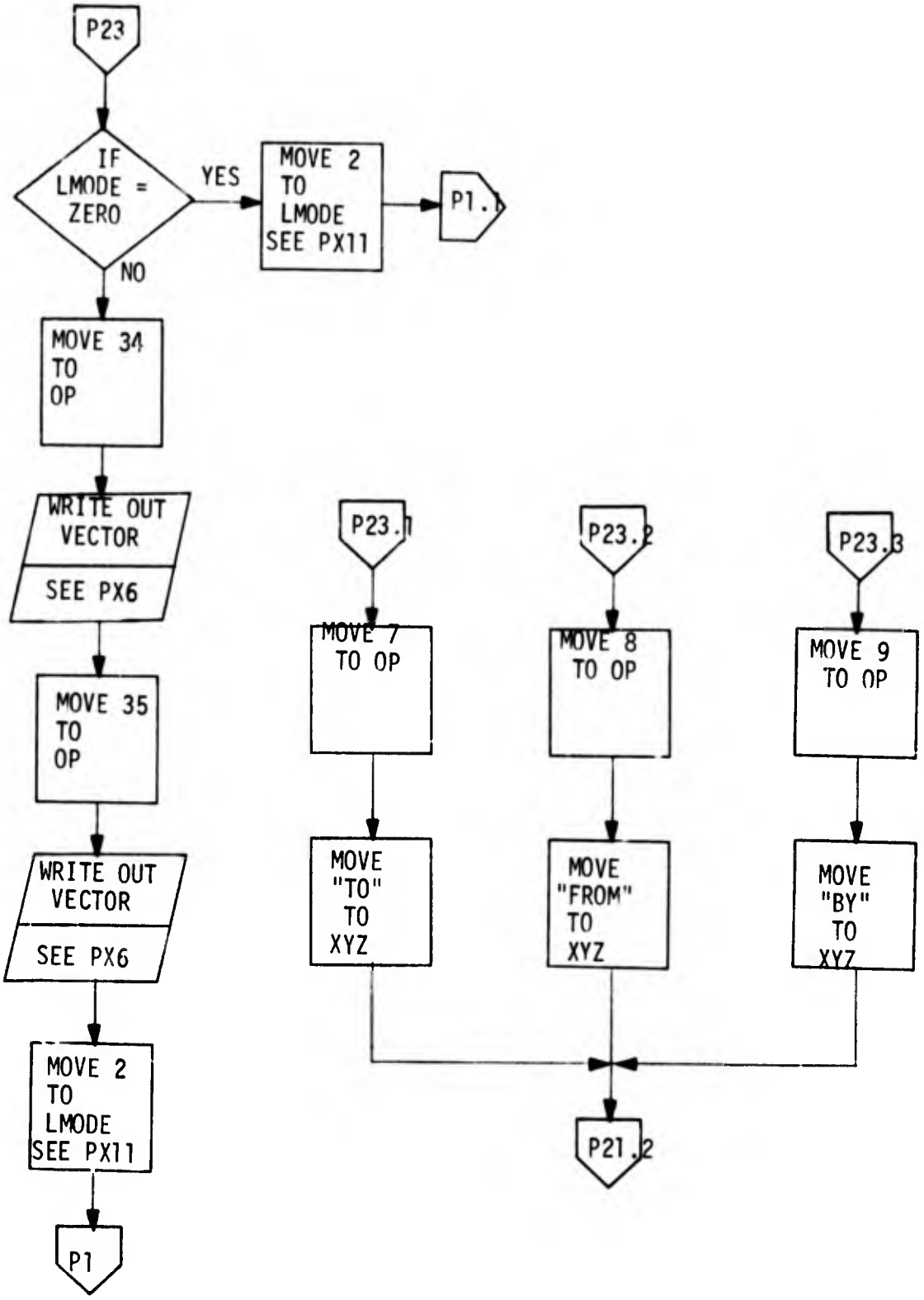

P18.0
SEARCH DEFINE TABLE
IF EQUAL
NO
IF WORD LENGTH > 5
YES
18
18.2
NO
YES
THIS A MODIFY DEFINE CARD
NO
ER1
YES
MOVE 1 TO MODIFY-SWT
THIS SWITCH INDICATES A MODIFY DEFINE STATEMENT.
18
IF WORD LENGTH > 5
YES
IF MODIFY-SWT = 1
YES
P1
NO
P19
18.2
NO
SEARCH FFT TABLE
IF EQUAL
YES
ER1
NO
18

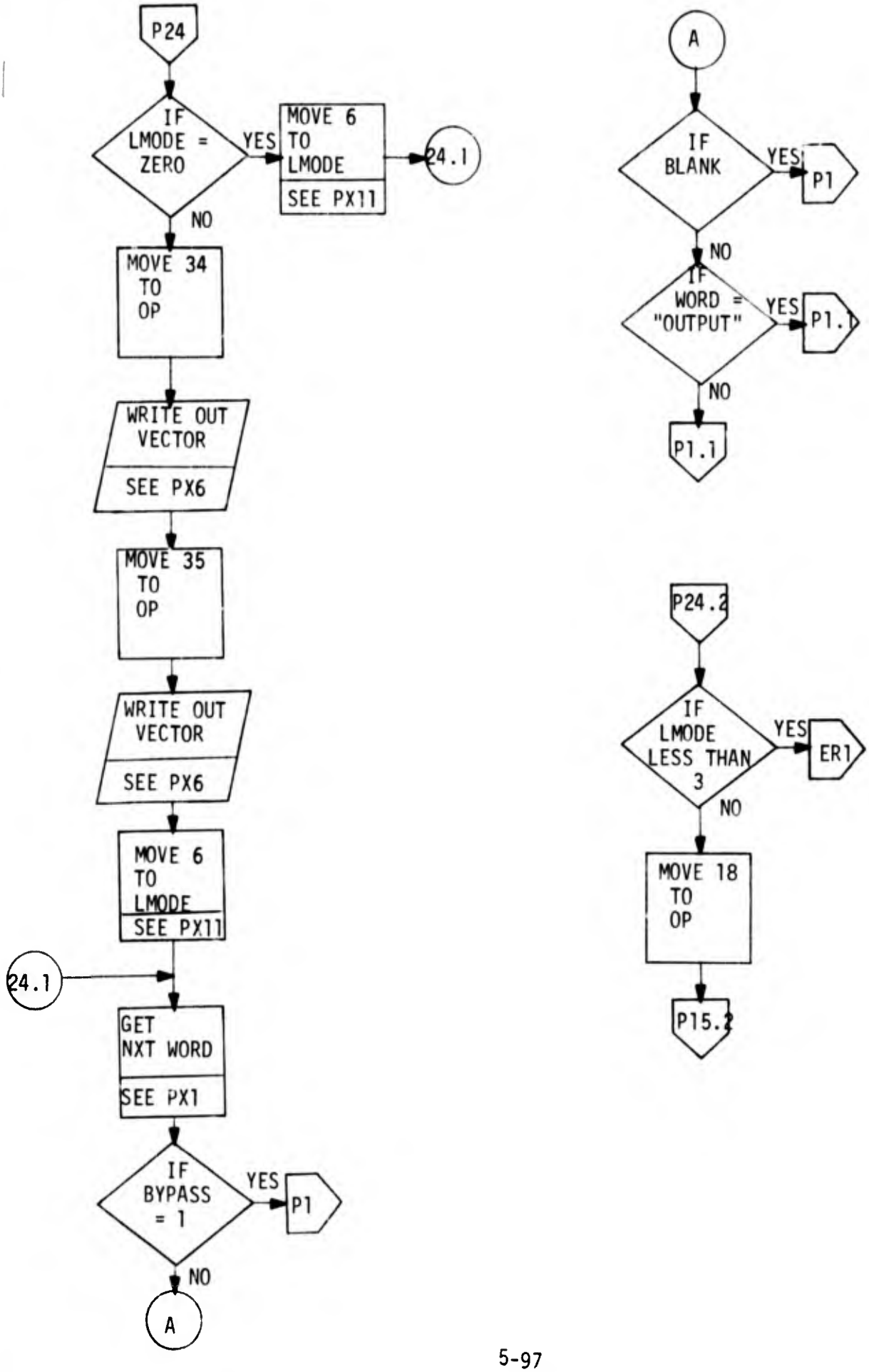

P19
MOVE NEW DEFINE TO DEFINE TABLE
IF TABLE-SWT = 1
YES
P25.1
19.1
NO
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF BLANK
YES
ER1
NO
IF WORD LENGTH > 8
YES
ER1
NO
IF DELIMITER = "*"
YES
MOVE 1 TO CONVERT SWT
19.1
NO
A
THIS SWITCH INDICATES A CONVERT ROUTINE IS TO BE PROCESSED
A
IF DELIMITER = SPACE
YES
19.2
NO
IF DATA IS NUMERIC
NO
ER1
YES
IS DATA >360 OR Ø
YES
ER1
NO
MOVE DATA TO DEFINE TABLE
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF BLANK
YES
P1
19.2
NO
IF CONVERT ROUTINE IS PRESENT
20.1
YES
P20

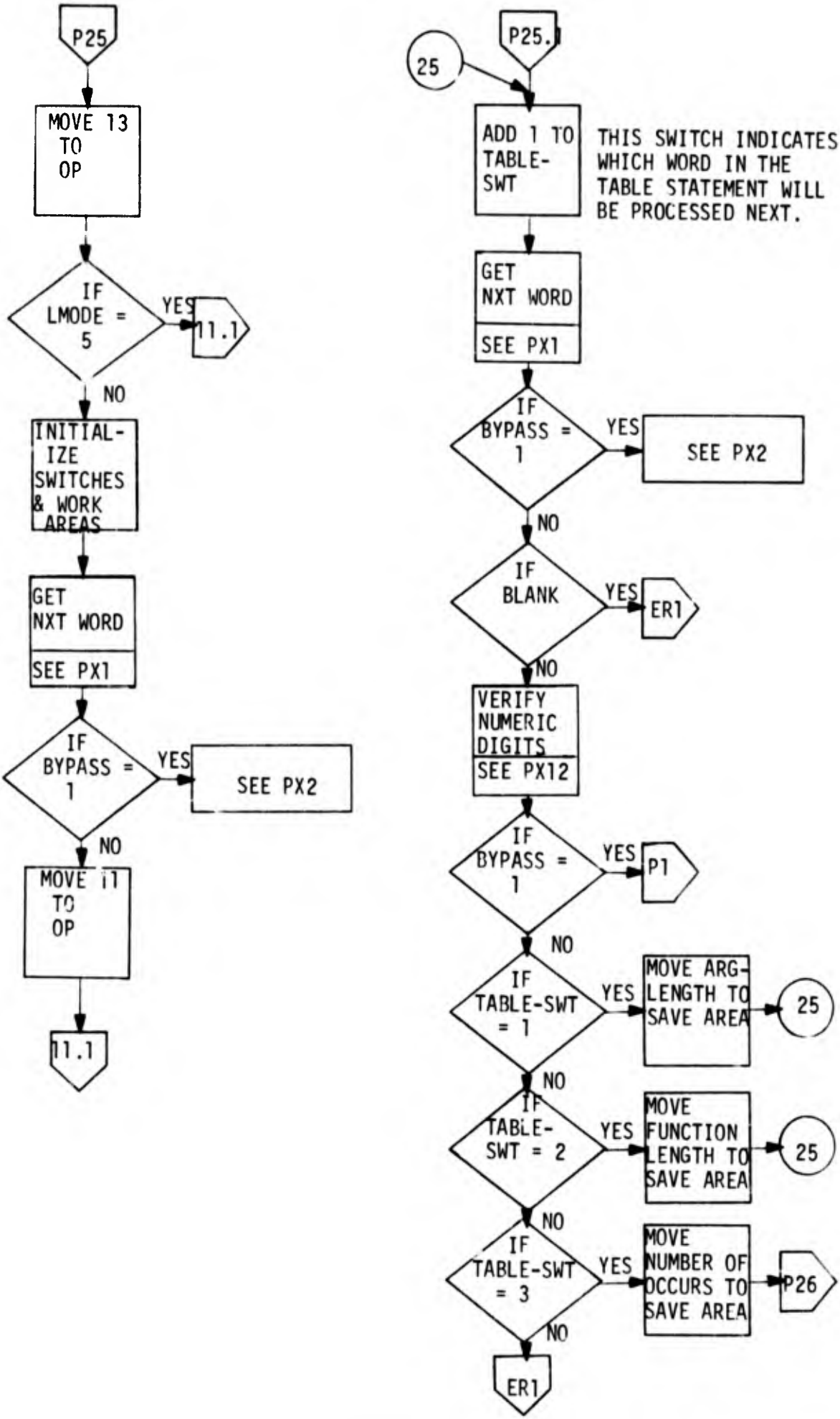

P20
VERIFY CONVERT NAME
SEE PX13
IF BYPASS = 1
YES
SEE PX2
NO
CALL IN CONVERT ROUTINE
20.1
IF LENGTH = Ø
YES
ER1
NO
IF CONVERT SWT = 1
YES
A
NO
IF DATA-TYPE ≠ 5
YES
A
NO
VERIFY NUMERIC DIGITS
SEE PX12
IF BYPASS = 1
YES
P1
NO
A
A
MOVE DATA TO SAVE AREA
SEE PX4
GET NEXT WORD
SEE PX1
IF BYPASS = 1
YES
P1
NO
IF BLANK
YES
P1
NO
ER1

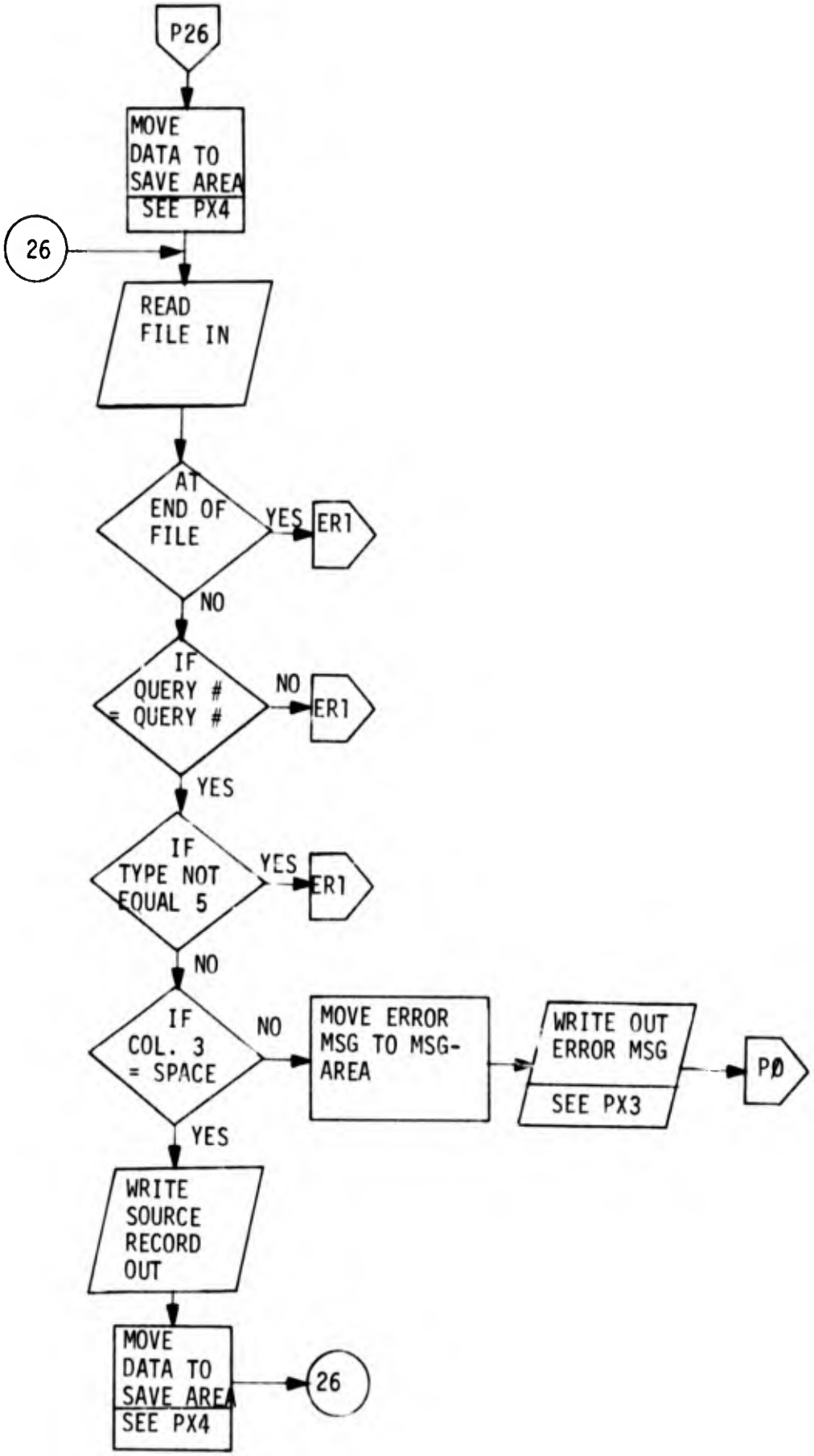

P21
IF LMODE EQUAL Ø
YES
MOVE 1 TO LMODE
P1.1
NO
MOVE 34 TO OP
WRITE OUT VECTOR
SEE PX6
MOVE 35 TO OP
WRITE OUT VECTOR
SEE PX6
MOVE 1 TO LMODE
P1.1
P21.1
MOVE 1Ø TO OP
MOVE "BY" TO XYZ
"XYZ" IS A SAVE AREA
P21.2
DECODE A-FIELD
SEE PX8
IF BYPASS = 1
YES
SEE PX2
NO
IF DATA IS NUMERIC
NO
ER1
YES
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
P22

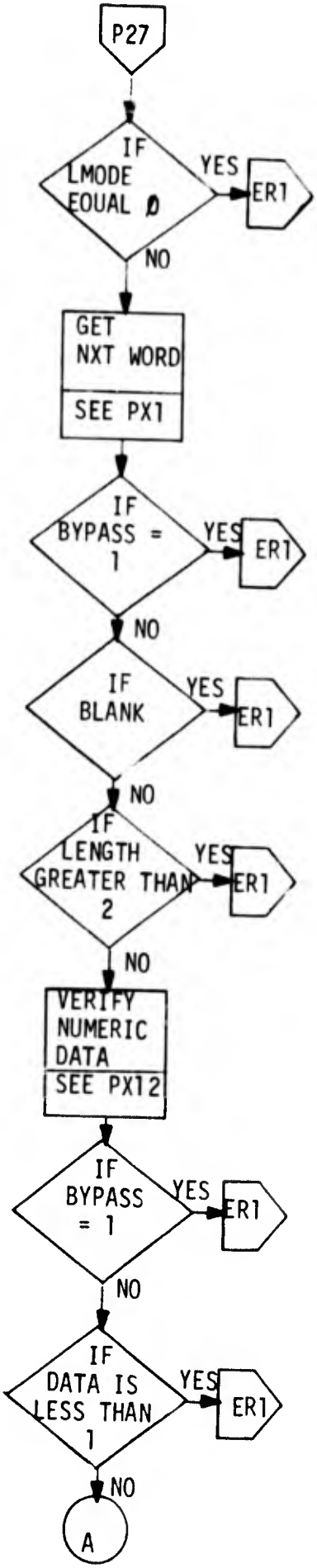

P22
IF BLANK
YES
ER1
NO
IF WORD LENGTH = 0
YES
ER1
NO
IF FIRST CHAR. IS NUMERIC
NO
ER1
YES
VERIFY NUMERIC FORMAT
SEE PX14
IF BYPASS = 1
YES
SEE PX2
NO
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
P1
NO
IF WORD = SAVE AREA "XYZ"
NO
ER1
YES
A
A
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
P1
NO
DECODE B-FIELD
SEE PX10
IF BYPASS = 1
YES
P1
NO
IF DATA IS NUMERIC
NO
ER1
YES
WRITE OUT VECTOR
SEE PX6
P1

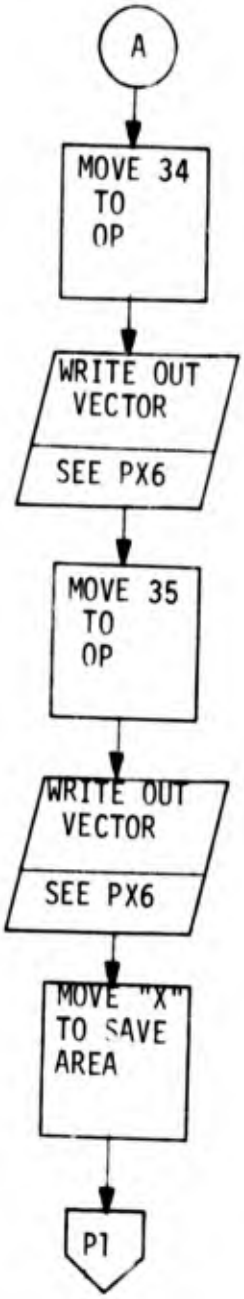

P23
IF LMODE = ZERO
YES
MOVE 2 TO LMODE SEE PX11
P1.1
NO
MOVE 34 TO OP
WRITE OUT VECTOR
SEE PX6
MOVE 35 TO OP
WRITE OUT VECTOR
SEE PX6
MOVE 2 TO LMODE SEE PX11
P1
P23.1
MOVE 7 TO OP
MOVE "TO" TO XYZ
P23.2
MOVE 8 TO OP
MOVE "FROM" TO XYZ
P23.3
MOVE 9 TO OP
MOVE "BY" TO XYZ
P21.2


P24
IF LMODE = ZERO
YES
MOVE 6 TO LMODE
SEE PX11
24.1
NO
MOVE 34 TO OP
WRITE OUT VECTOR
SEE PX6
MOVE 35 TO OP
WRITE OUT VECTOR
SEE PX6
MOVE 6 TO LMODE
SEE PX11
24.1
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
P1
NO
A
A
IF BLANK
YES
P1
NO
IF WORD = "OUTPUT"
YES
P1.1
NO
P1.1
P24.2
IF LMODE LESS THAN 3
YES
ER1
NO
MOVE 18 TO OP
P15.2

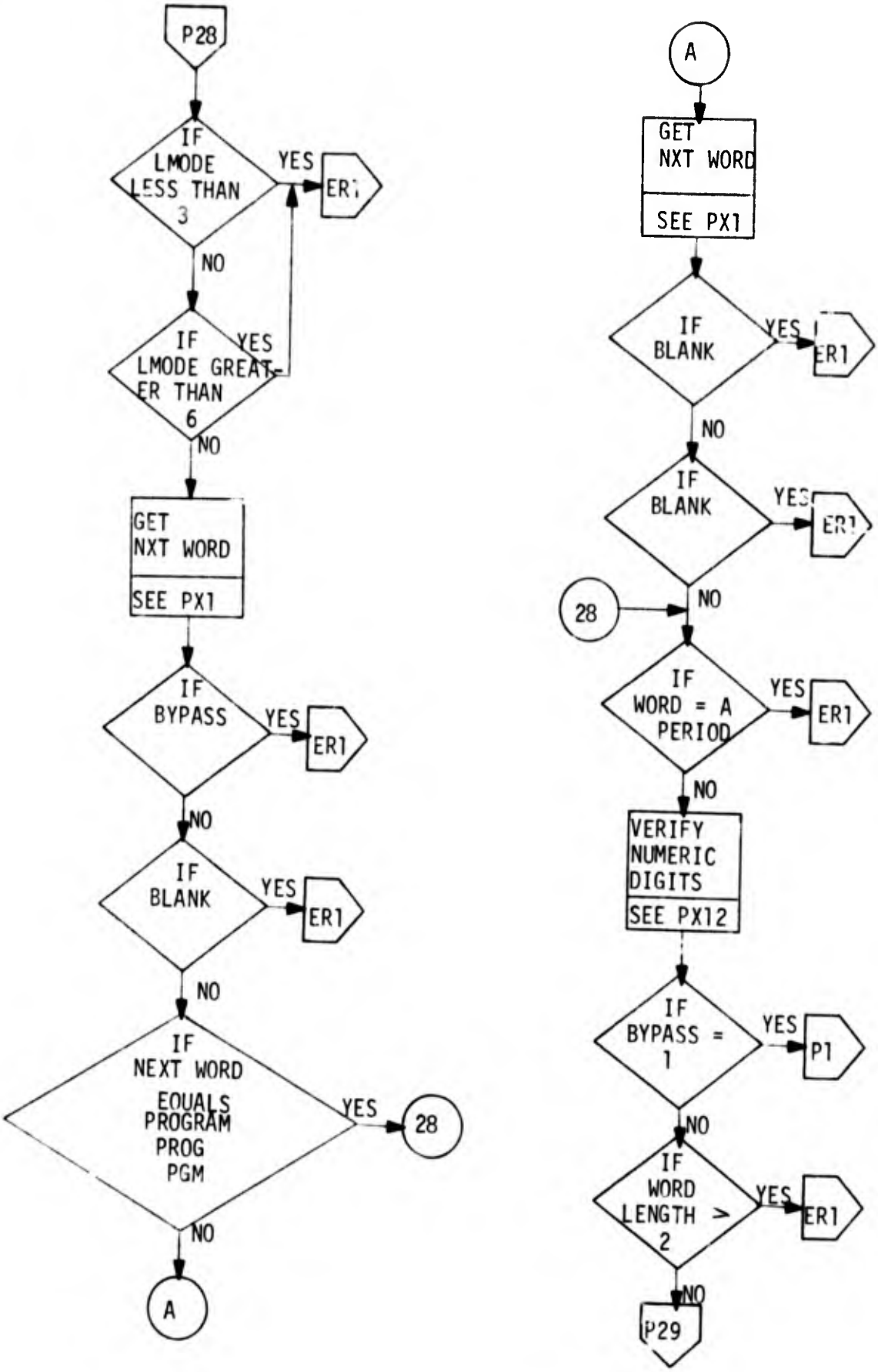

P25
MOVE 13 TO OP
IF LMODE = 5
YES
11.1
NO
INITIAL-IZE SWITCHES & WORK AREAS
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
MOVE 11 TO OP
11.1
25
P25.1
ADD 1 TO TABLE-SWT
THIS SWITCH INDICATES WHICH WORD IN THE TABLE STATEMENT WILL BE PROCESSED NEXT.
GET NXT WORD
SEE PX1
IF BYPASS = 1
YES
SEE PX2
NO
IF BLANK
YES
ER1
NO
VERIFY NUMERIC DIGITS
SEE PX12
IF BYPASS = 1
YES
P1
NO
IF TABLE-SWT = 1
YES
MOVE ARG-LENGTH TO SAVE AREA
25
NO
IF TABLE-SWT = 2
YES
MOVE FUNCTION LENGTH TO SAVE AREA
25
NO
IF TABLE-SWT = 3
YES
MOVE NUMBER OF OCCURS TO SAVE AREA
P26
NO
ER1

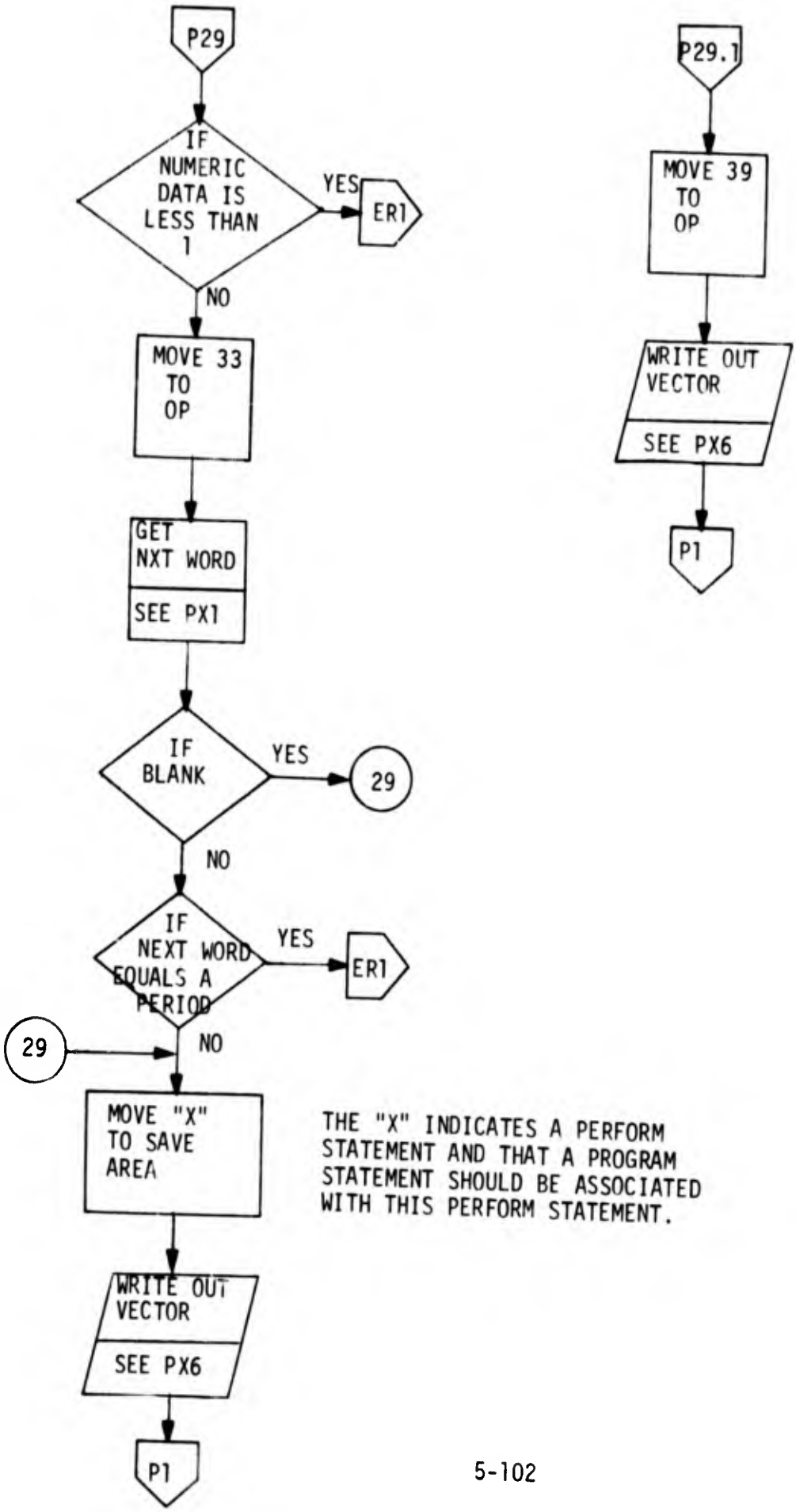

P26
MOVE DATA TO SAVE AREA
SEE PX4
26
READ FILE IN
AT END OF FILE
YES
ER1
NO
IF QUERY # = QUERY #
NO
ER1
YES
IF TYPE NOT EQUAL 5
YES
ER1
NO
IF COL. 3 = SPACE
NO
MOVE ERROR MSG TO MSG-AREA
WRITE OUT ERROR MSG
SEE PX3
PØ
YES
WRITE SOURCE RECORD OUT
MOVE DATA TO SAVE AREA
SEE PX4
26



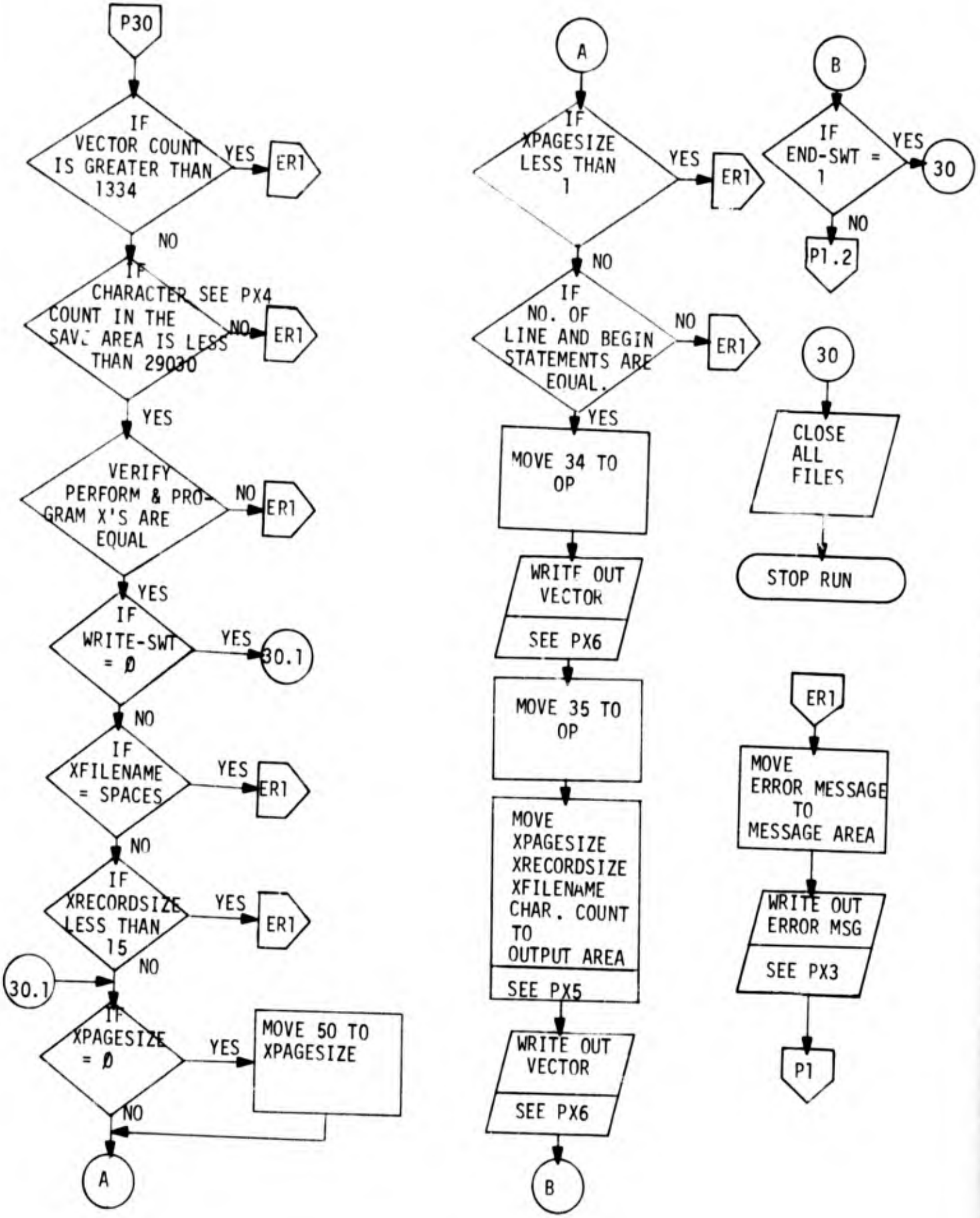



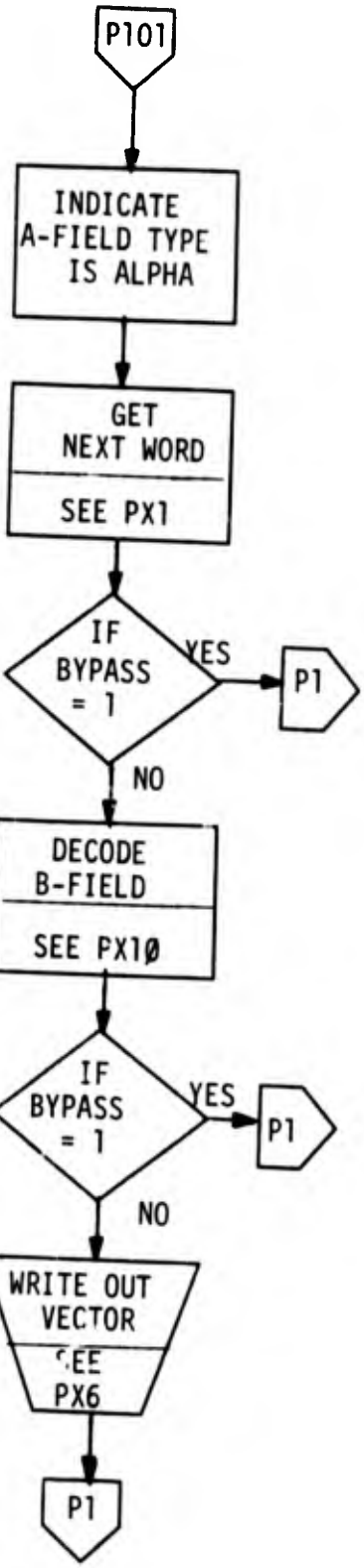

THE "X"INDICATES A PROGRAM STATEMENT HAS BEEN SUBMITTED BY THE USER AND THAT A PERFORM STATEMENT SHOULD BE FOUND ASSOCIATED WITH THE PROGRAM STATEMENT.

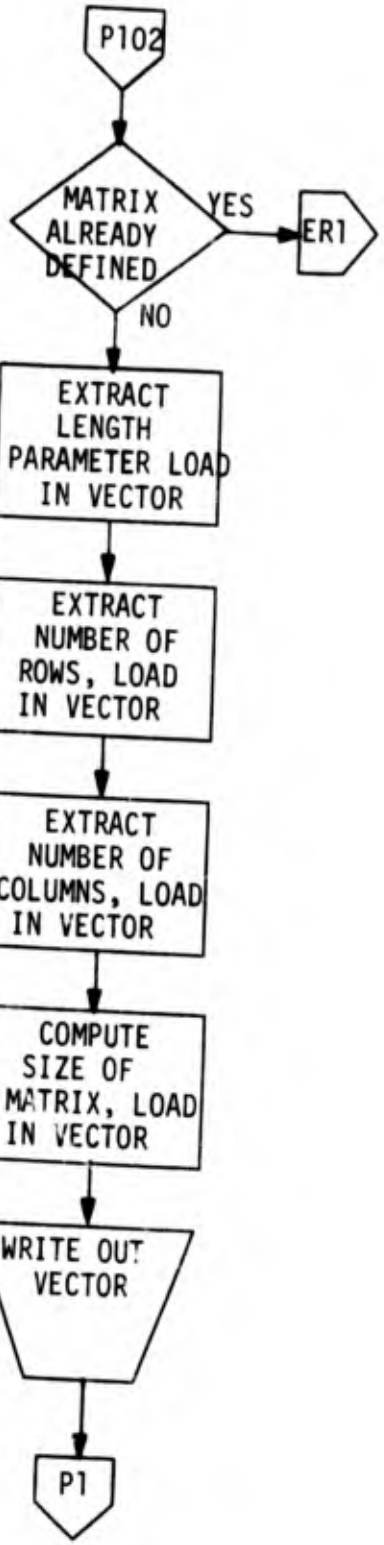

P28
IF LMODE LESS THAN 3
YES
ER1
NO
IF LMODE GREATER THAN 6
YES
NO
GET NXT WORD
SEE PX1
IF BYPASS
YES
ER1
NO
IF BLANK
YES
ER1
NO
IF NEXT WORD EQUALS PROGRAM PROG PGM
YES
28
NO
A
A
GET NXT WORD
SEE PX1
IF BLANK
YES
ER1
NO
IF BLANK
YES
ER1
28
NO
IF WORD = A PERIOD
YES
ER1
NO
VERIFY NUMERIC DIGITS
SEE PX12
IF BYPASS = 1
YES
P1
NO
IF WORD LENGTH > 2
YES
ER1
NO
P29




THE "X" INDICATES A PERFORM STATEMENT AND THAT A PROGRAM STATEMENT SHOULD BE ASSOCIATED WITH THIS PERFORM STATEMENT.


P30
IF VECTOR COUNT IS GREATER THAN 1334
YES
ER1
NO
IF CHARACTER SEE PX4 COUNT IN THE SAVE AREA IS LESS THAN 29030
NO
ER1
YES
VERIFY PERFORM & PROGRAM X'S ARE EQUAL
NO
ER1
YES
IF WRITE-SWT = Ø
YES
30.1
NO
IF XFILENAME = SPACES
YES
ER1
NO
IF XRECORDSIZE LESS THAN 15
YES
ER1
NO
30.1
IF XPAGESIZE = Ø
YES
MOVE 50 TO XPAGESIZE
NO
A
A
IF XPAGESIZE LESS THAN 1
YES
ER1
NO
IF NO. OF LINE AND BEGIN STATEMENTS ARE EQUAL.
NO
ER1
YES
MOVE 34 TO OP
WRITE OUT VECTOR
SEE PX6
MOVE 35 TO OP
MOVE XPAGESIZE XRECORDSIZE XFILENAME CHAR. COUNT TO OUTPUT AREA
SEE PX5
WRITE OUT VECTOR
SEE PX6
B
B
IF END-SWT = 1
YES
30
NO
P1.2
30
CLOSE ALL FILES
STOP RUN
ER1
MOVE ERROR MESSAGE TO MESSAGE AREA
WRITE OUT ERROR MSG
SEE PX3
P1


P101
INDICATE A-FIELD TYPE IS ALPHA
GET NEXT WORD
SEE PX1
IF BYPASS = 1
YES
P1
NO
DECODE B-FIELD
SEE PX10
IF BYPASS = 1
YES
P1
NO
WRITE OUT VECTOR
SEE PX6
P1


P102
MATRIX ALREADY DEFINED
YES
ER1
NO
EXTRACT LENGTH PARAMETER LOAD IN VECTOR
EXTRACT NUMBER OF ROWS, LOAD IN VECTOR
EXTRACT NUMBER OF COLUMNS, LOAD IN VECTOR
COMPUTE SIZE OF MATRIX, LOAD IN VECTOR
WRITE OUT VECTOR
P1

PX1 EXTRACT-WORD Section picks up each word, letter, or number from a user's source statement and moves it to a work area called CARDW. If a continuation is required, another read will be executed using the READ-OP Section. If errors are encountered during the extraction, an appropriate error message is written out.

PX2 The bypass switch is initialized to zero before each record is processed. If an error is encountered, a one (1) is moved to this switch signifying the end of syntax checking and the writing out of the appropriate error message.

PX3 When errors occur within the processing program, an appropriate error message is moved to an error message area and then written out. In addition, a one (1) is moved to bypass switch and, depending upon the severity of the error, a decision is made as to what step the program will execute next (see PX7).

PX4 The LOAD-CONSTANT Section is used to move a character string to a work save area called CONSTANT-POOL. Its length depends upon the value in a numeric area called IL. If the CONSTANT-POOL area exceeds 90 characters, this area is written out and a new one is created using a special write section called WRITE-CDYN.

PX5 For each user's statement submitted, a record (vector) for that operation is written out containing information needed for processing by the next program. See list of vectors.

PX6 WRITE-STATE Section is used to write out records (vectors). In this section certain criteria in the vectors must be checked for each operator or operation before the vector is written out and processed by the next program. If an error is encountered, the appropriate message is written out (see PX7).

PX7 WRITE-ERR and WRITE-ERR1 Sections are used to write out all error records. In addition, they move a one (1) to bypass to stop further processing of such records.

PX8 DECODE-A Section is used for the decoding and syntaxing of the a-field of the user source statement. Such statements can contain a user defined name, convert routine, special operator, literal, constant, or FFT name, all of which must be decoded into pointers, locations and/or values and moved to a specified area in the record (vector). If the statement requires the extraction of another word, SWT4 will equal zero. If errors are encountered, the appropriate message will be written out.

PX9 DECODE-OP Section extracts the next word. If partial notation is encountered as the next word, Section FLD-PART performs a validity check and, if no errors are found, the proper information is placed into the a-field output area before another word is extracted. If a convert routine name or special operator is found, a verification of the name, using a section called S806-TRAP, is performed before continuing to the next word. The logical operator (e.g., EQUAL, NOT EQUAL, LESS, GREATER, etc.) is next retrieved. Upon comparing this word against a table of operators, a numeric value is placed into the operation field (OP) of the output area. Before leaving this section another word will be retrieved for further processing. All errors found will use the WRITE-ERR or WRITE-ERR1 sections to write out the appropriate messages.

PX10 DECODE-B Section is used for syntaxing and decoding the b-field of the user source statement. Such statements can contain a FFT name, a user defined name, special operator, output area, multiple b-fields, and/or a numeric value associated with the b-field as required by the statement itself. The DECODE-B Section verifies the number of parentheses given by the user for equality using RIGHT-PARN1 Section. When numeric values are found in the b-field the EXTRACT-NUM Section is used to validate these digits. When a word in the b-field requires numeric or alphanumeric characters to be saved for further processing,

the LOAD-CONSTANT Section is performed (see PX4). If leading zeros or trailing spaces are necessary, LEAD-ZEROS and TRAIL-SPACE are executed for that particular word. When a multiple b-field is encountered by using EXTRACT-WORD Section (see PX1), the previous record will be written out (see PX6) if it is not a partial notation parameter. If partial notation is found, the FLD-PART Section is used to check its validity and, if no errors are found, the appropriate information is moved to the b-field output area. If a special operator is retrieved as the next word the name is validated by using the S806-TRAP Section before continuing to the next word. All errors found while processing the b-field words will use the WRITE-ERR and WRITE-ERR1 Sections to write out error messages.

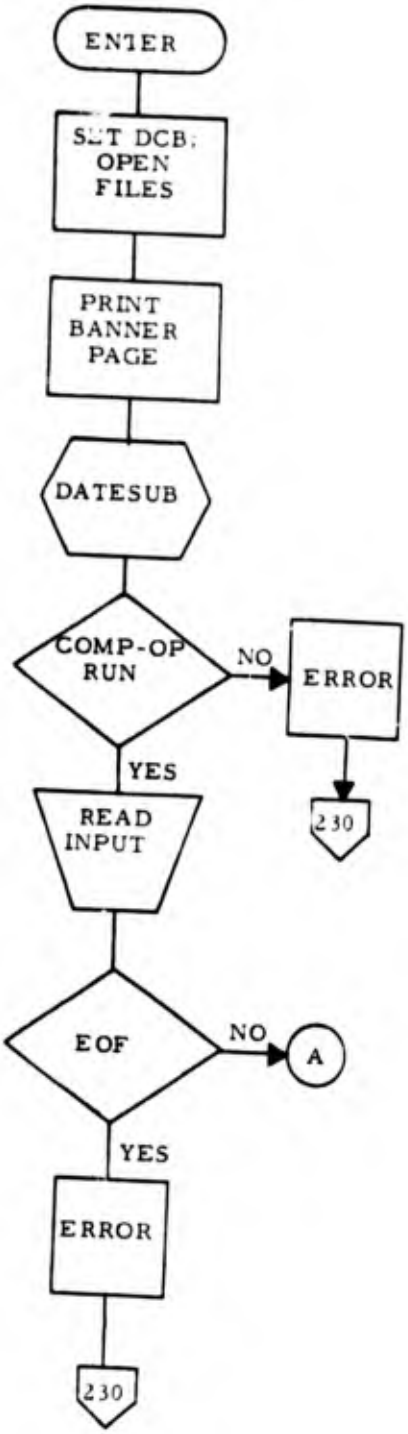
PX11 The word LMODE is a data name in the vector output area. This area contains a numeric value to indicate the section and/or paragraph in which the user's source statement is located.

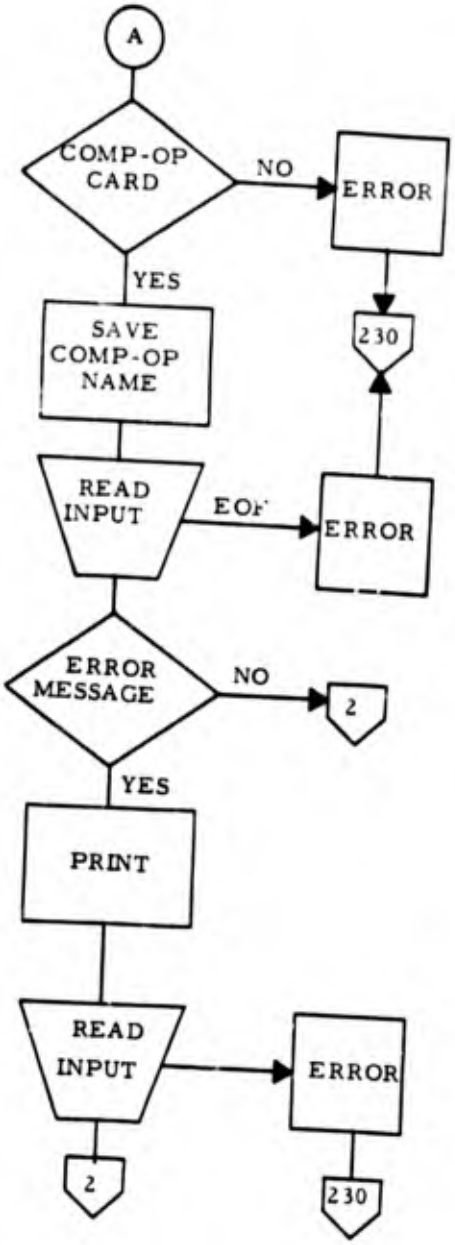
PX12 EXTRACT-NUM Section is used to verify numeric digits found in the user's source statement. If an error is found, the appropriate error message is written out.

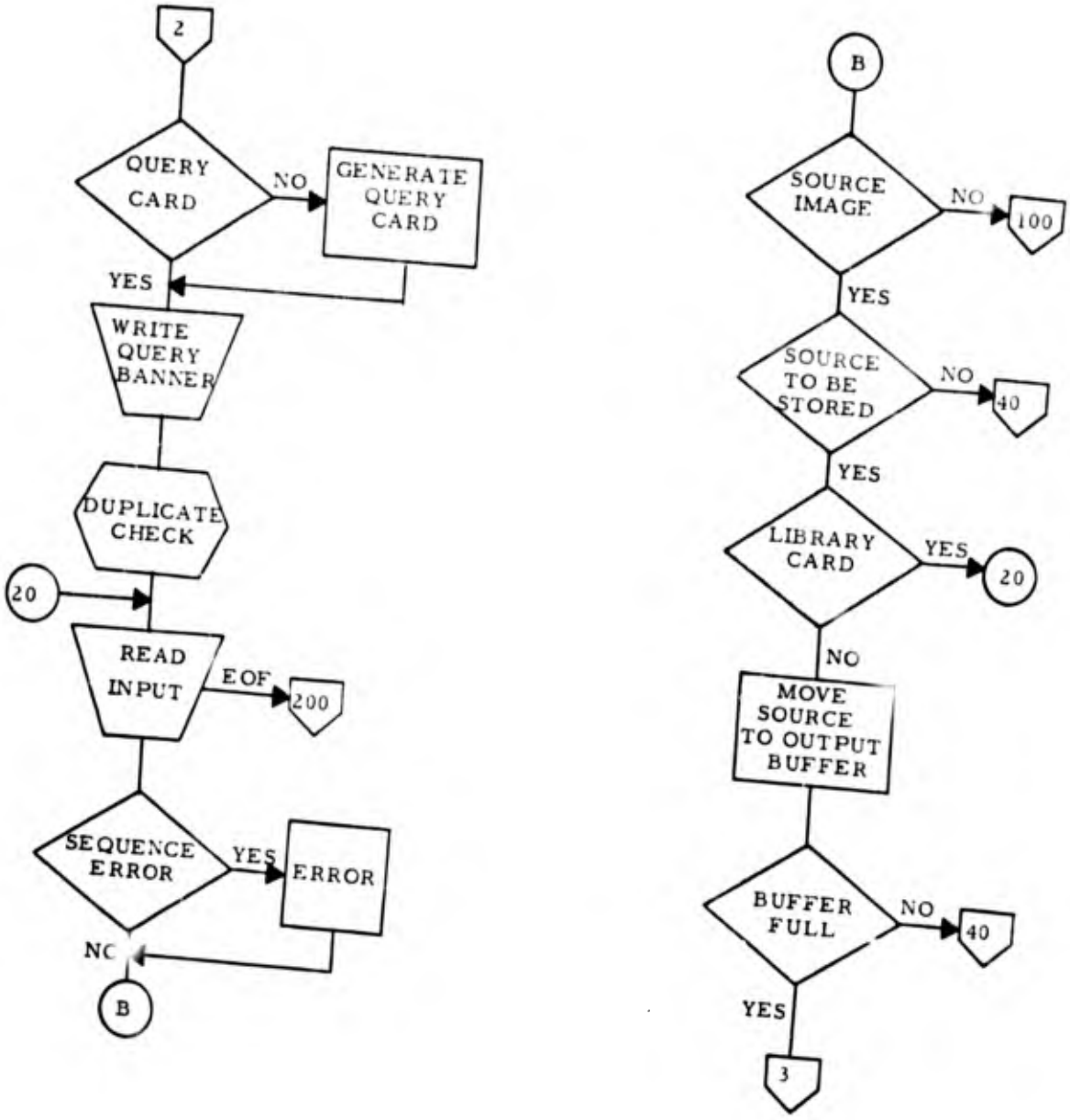
PX13 S806-TRAP Section is used to validate that a special operator or convert routine name exists in the system library. If not found, the appropriate error message is written out.

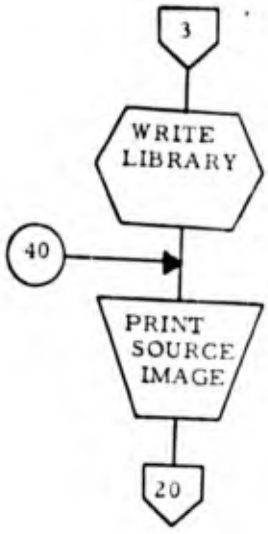
PX14 FLD-PART Section is used to syntax the format provided by the user when validating partial notation or an output area. If errors are found the appropriate message is written out.

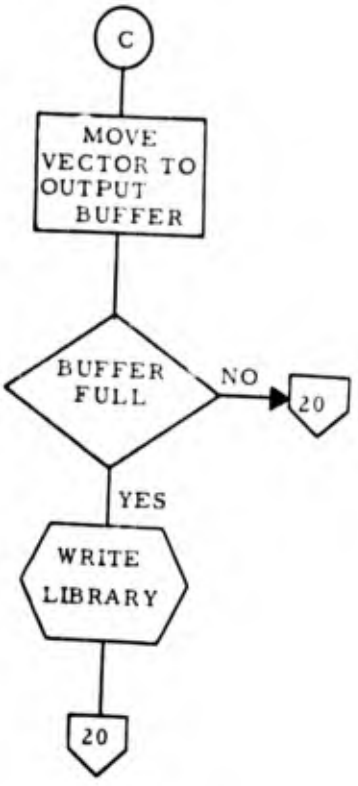
(10.1) GEN3B

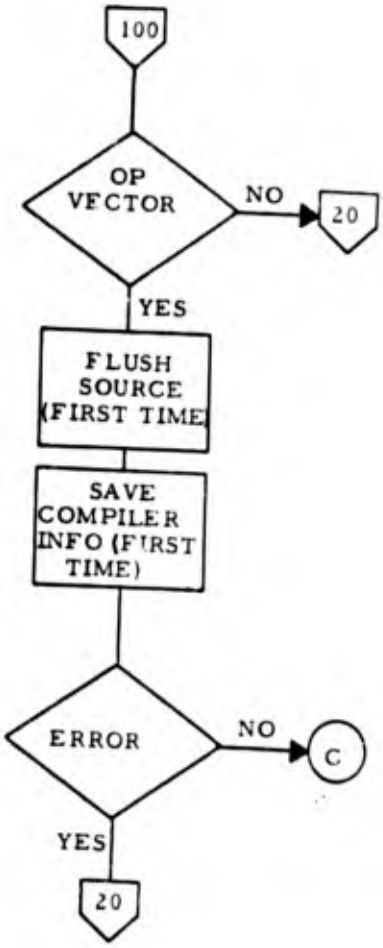

ENTER
SET DCB; OPEN FILES
PRINT BANNER PAGE
DATESUB
COMP-OP RUN
NO
ERROR
YES
READ INPUT
230
EOF
NO
A
YES
ERROR
230

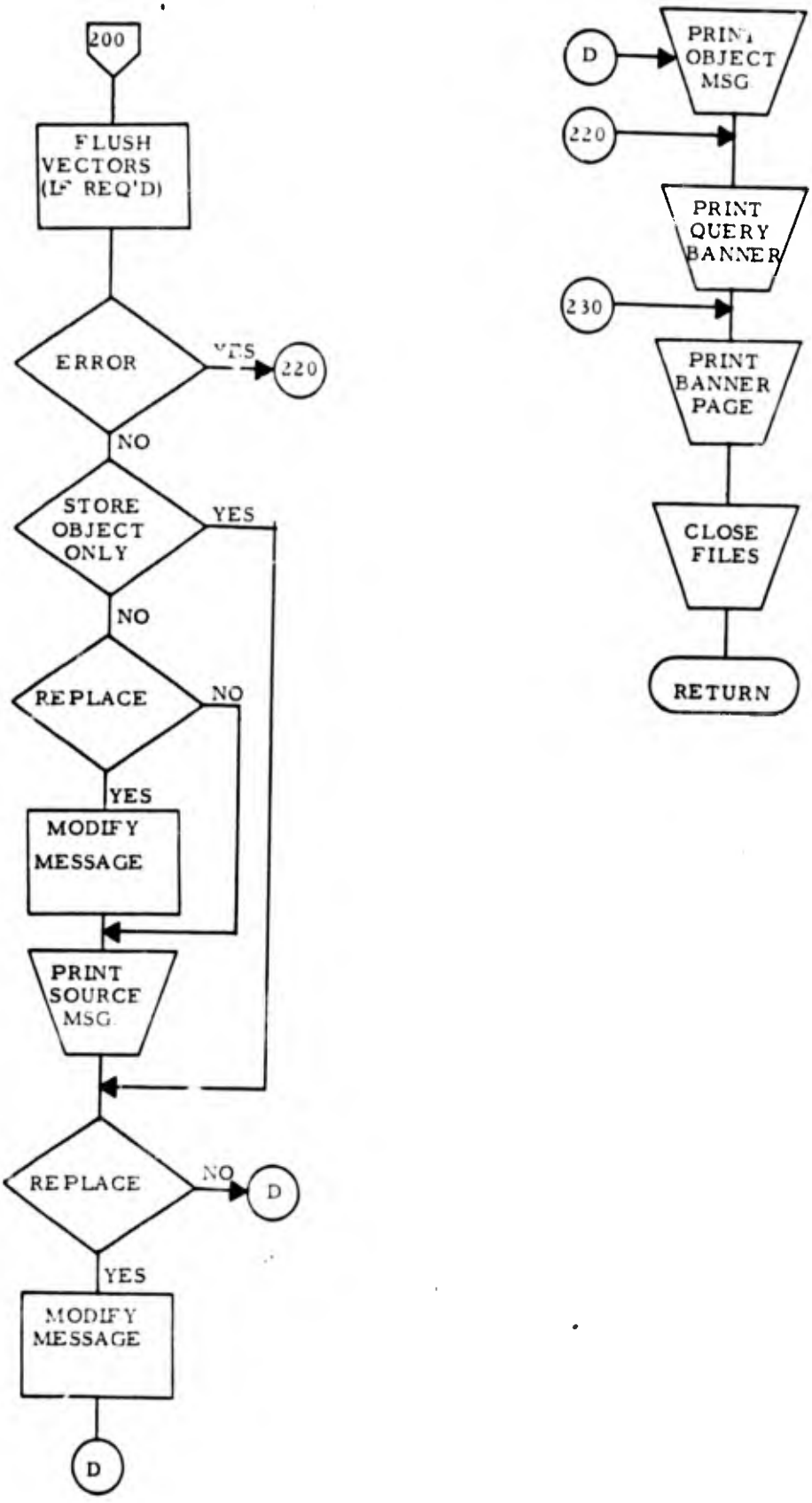

A
COMP-OP CARD
NO
ERROR
YES
230
SAVE COMP-OP NAME
READ INPUT
EOF
ERROR
ERROR MESSAGE
NO
2
YES
PRINT
READ INPUT
ERROR
2
230


2
QUERY CARD
NO
GENERATE QUERY CARD
YES
WRITE QUERY BANNER
DUPLICATE CHECK
20
READ INPUT
EOF
200
SEQUENCE ERROR
YES
ERROR
NO
B
B
SOURCE IMAGE
NO
100
YES
SOURCE TO BE STORED
NO
40
YES
LIBRARY CARD
YES
20
NO
MOVE SOURCE TO OUTPUT BUFFER
BUFFER FULL
NO
40
YES
3

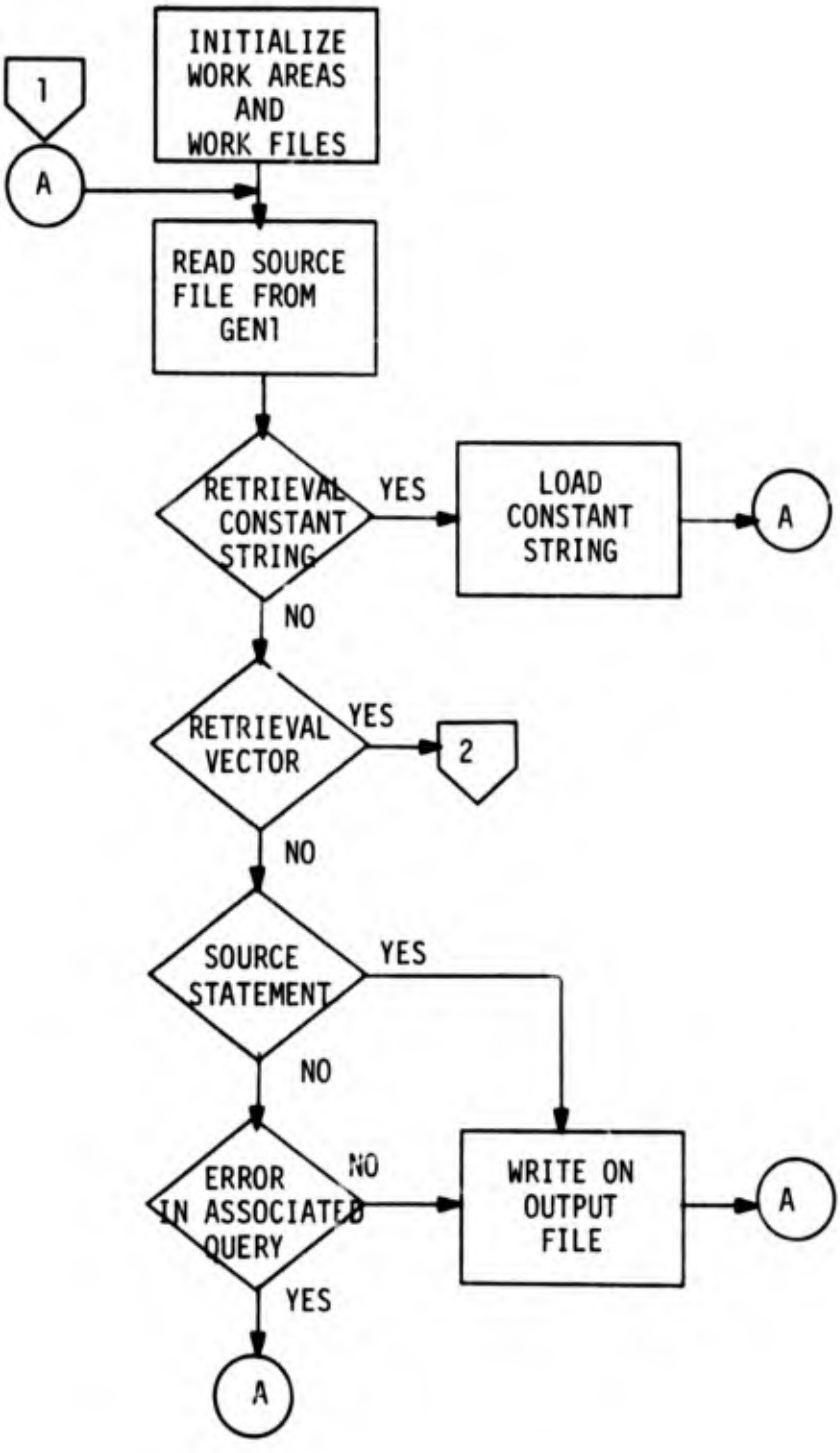

3
WRITE LIBRARY
40
PRINT SOURCE IMAGE
20

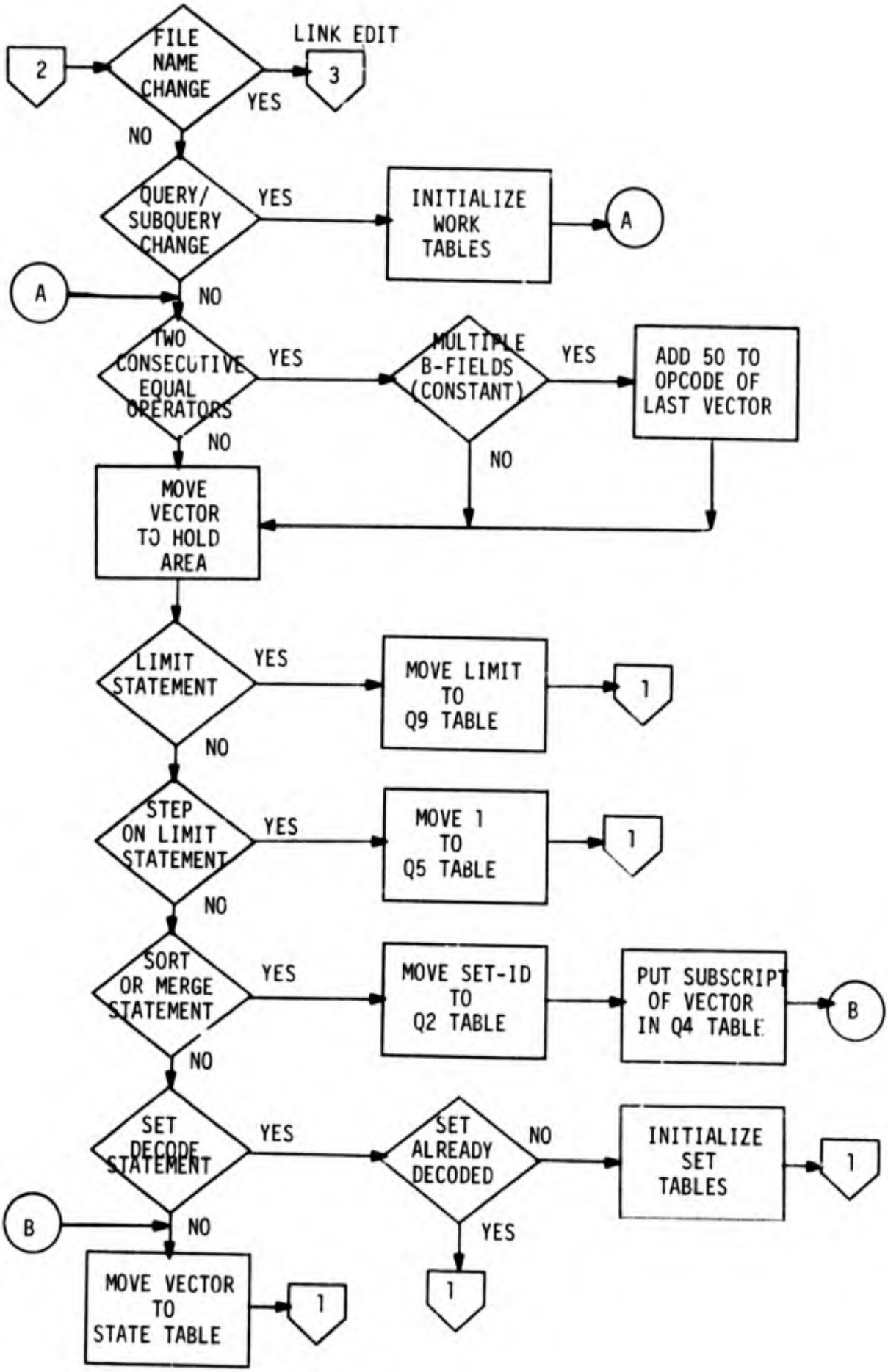

C
MOVE VECTOR TO OUTPUT BUFFER
BUFFER FULL
NO
20
YES
WRITE LIBRARY
20

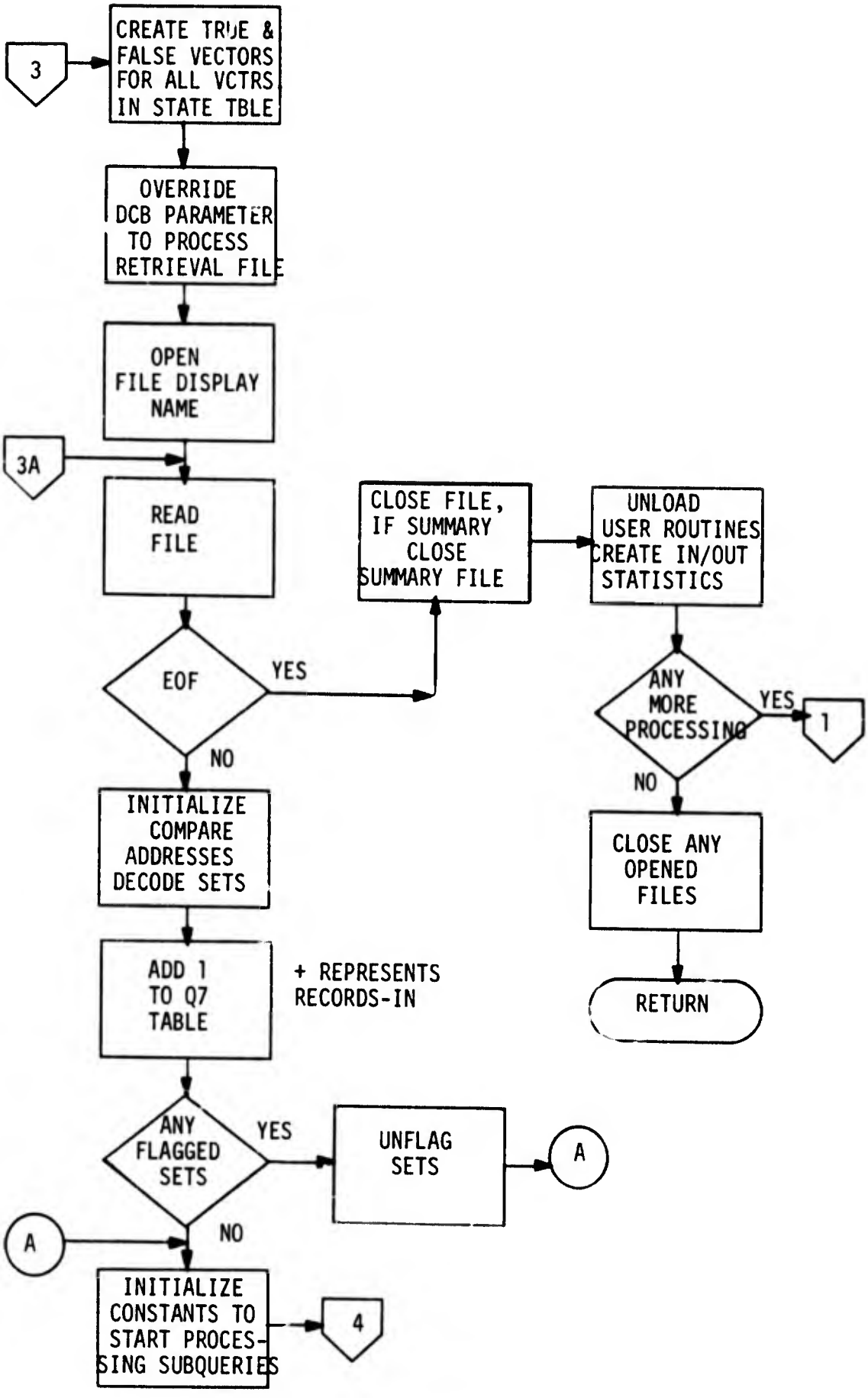

100
OP VECTOR
NO
20
YES
FLUSH SOURCE (FIRST TIME)
SAVE COMPILER INFO (FIRST TIME)
ERROR
NO
C
YES
20

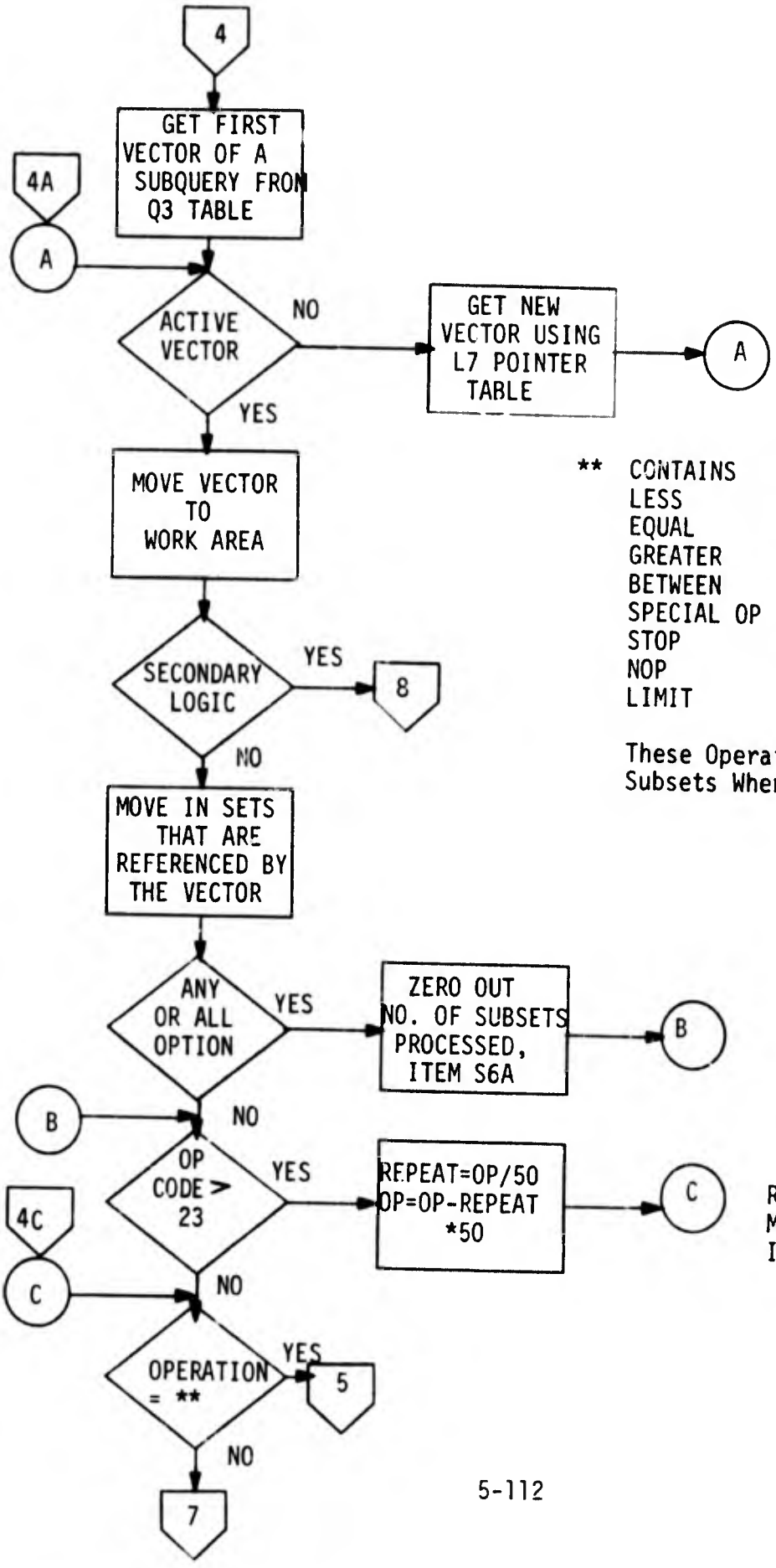

200
FLUSH VECTORS (IF REQ'D)
ERROR
YES
220
NO
STORE OBJECT ONLY
YES
NO
REPLACE
NO
YES
MODIFY MESSAGE
PRINT SOURCE MSG
REPLACE
NO
D
YES
MODIFY MESSAGE
D
D
PRINT OBJECT MSG
220
PRINT QUERY BANNER
230
PRINT BANNER PAGE
CLOSE FILES
RETURN

(11) GEN4 and GEN4A.




2
FILE NAME CHANGE
YES
LINK EDIT
3
NO
QUERY/ SUBQUERY CHANGE
YES
INITIALIZE WORK TABLES
A
A
NO
TWO CONSECUTIVE EQUAL OPERATORS
YES
MULTIPLE B-FIELDS (CONSTANT)
YES
ADD 50 TO OPCODE OF LAST VECTOR
NO
NO
MOVE VECTOR TO HOLD AREA
LIMIT STATEMENT
YES
MOVE LIMIT TO Q9 TABLE
1
NO
STEP ON LIMIT STATEMENT
YES
MOVE 1 TO Q5 TABLE
1
NO
SORT OR MERGE STATEMENT
YES
MOVE SET-ID TO Q2 TABLE
PUT SUBSCRIPT OF VECTOR IN Q4 TABLE
B
NO
SET DECODE STATEMENT
YES
SET ALREADY DECODED
NO
INITIALIZE SET TABLES
1
B
NO
YES
MOVE VECTOR TO STATE TABLE
1
1

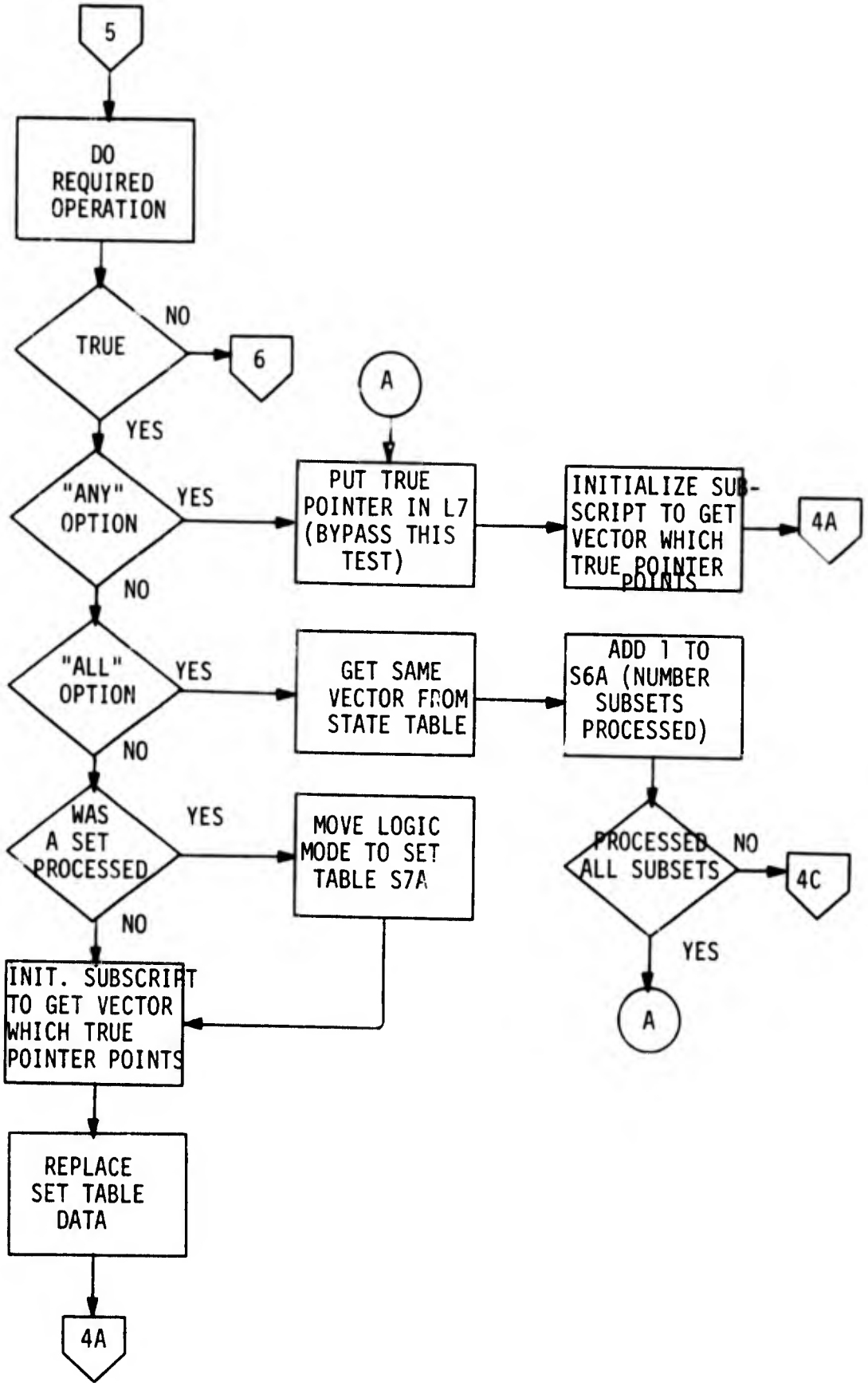

3
CREATE TRUE & FALSE VECTORS FOR ALL VCTRS IN STATE TBLE
OVERRIDE DCB PARAMETER TO PROCESS RETRIEVAL FILE
OPEN FILE DISPLAY NAME
3A
READ FILE
EOF
YES
NO
CLOSE FILE, IF SUMMARY CLOSE SUMMARY FILE
UNLOAD USER ROUTINES CREATE IN/OUT STATISTICS
ANY MORE PROCESSING
YES
1
NO
CLOSE ANY OPENED FILES
RETURN
INITIALIZE COMPARE ADDRESSES DECODE SETS
ADD 1 TO Q7 TABLE
+ REPRESENTS RECORDS-IN
ANY FLAGGED SETS
YES
UNFLAG SETS
A
A
NO
INITIALIZE CONSTANTS TO START PROCES-SING SUBQUERIES
4

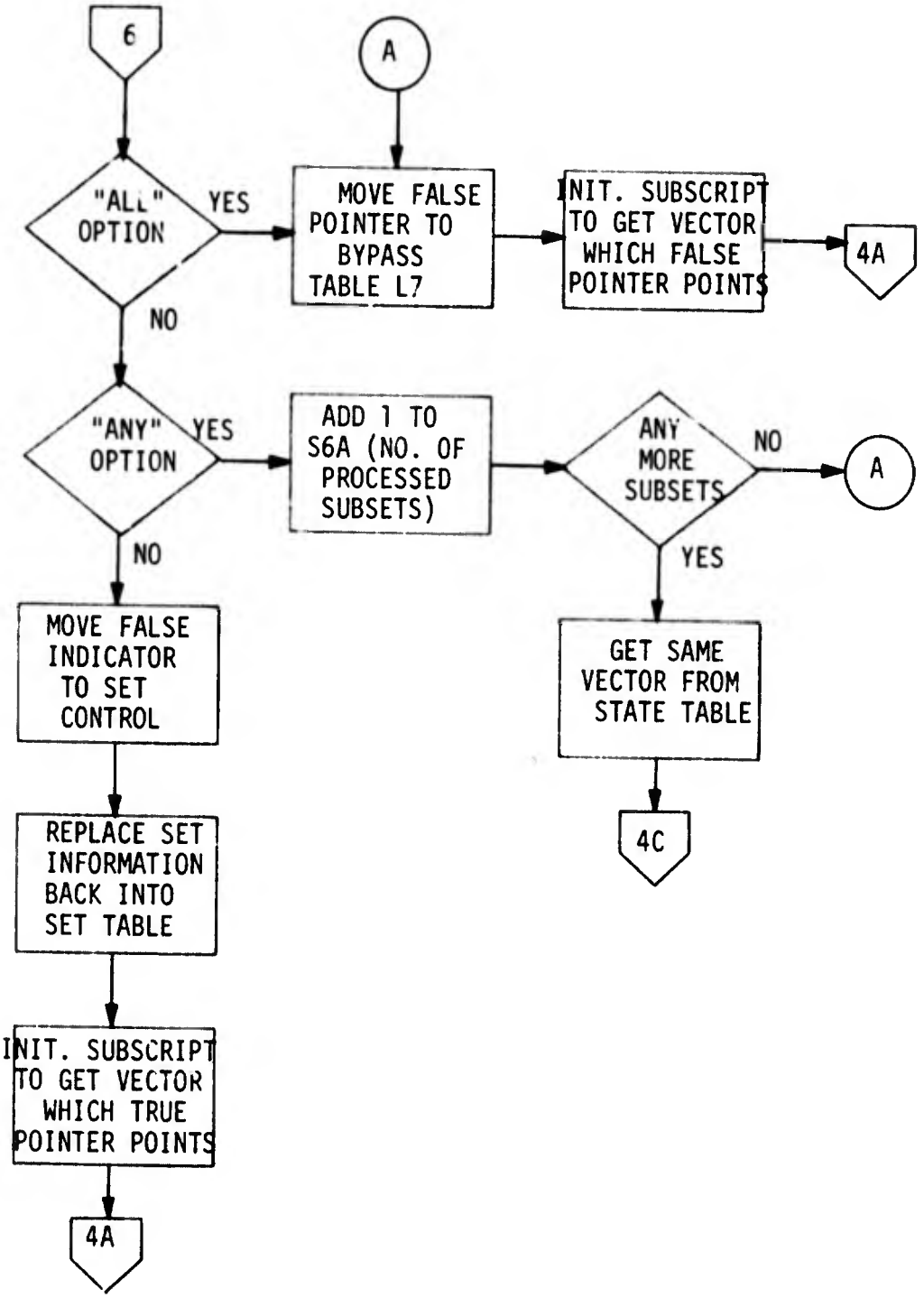


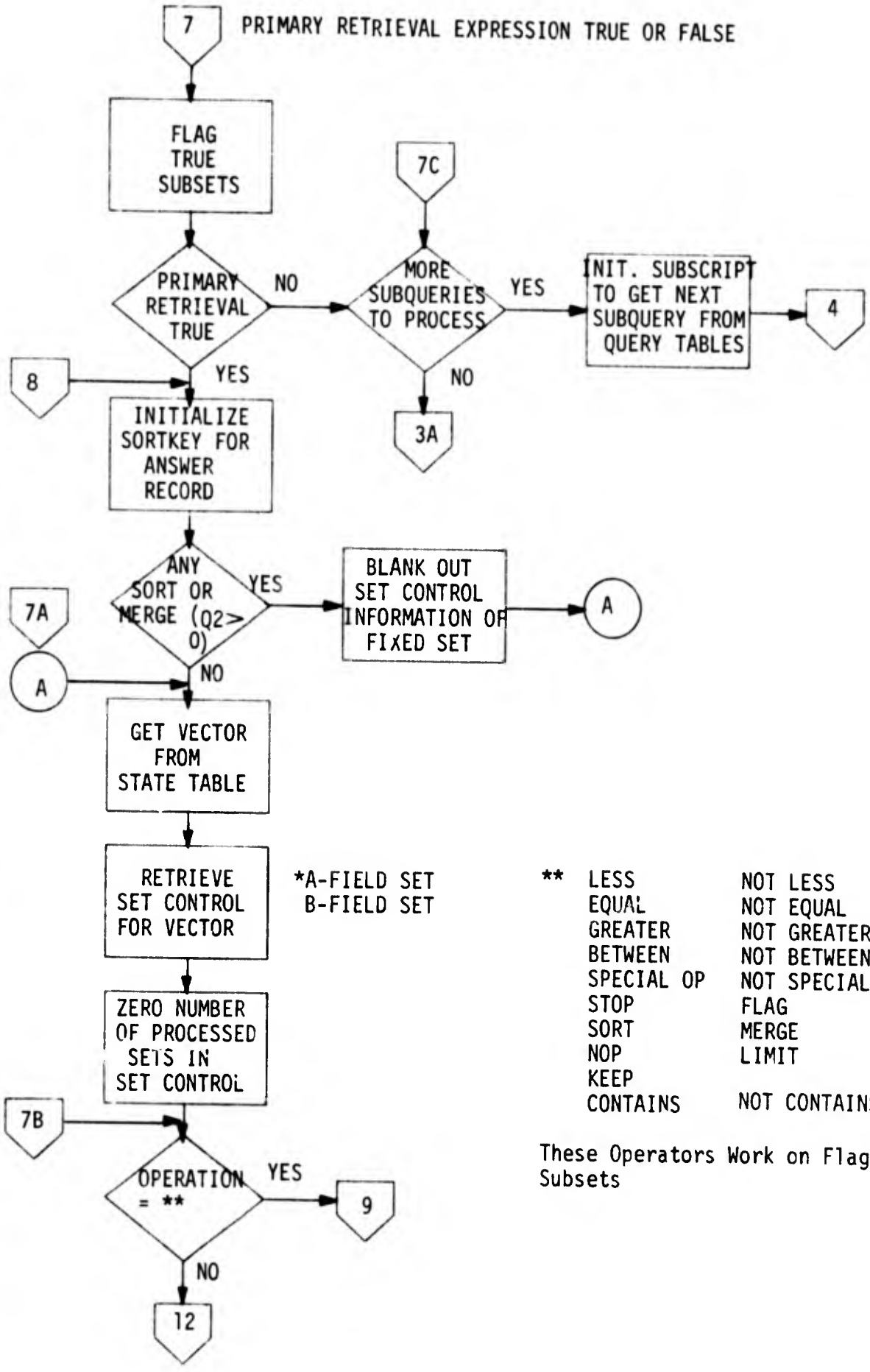

| | | |
|---|---|---|
| ** | CONTAINS | NOT CONTAINS |
| | LESS | NOT LESS |
| | EQUAL | NOT EQUAL |
| | GREATER | NOT GREATER |
| | BETWEEN | NOT BETWEEN |
| | SPECIAL OP | NOT SPECIAL OP |
| | STOP | FLAG |
| | NOP | SET DECODE |
| | LIMIT | KEEP |

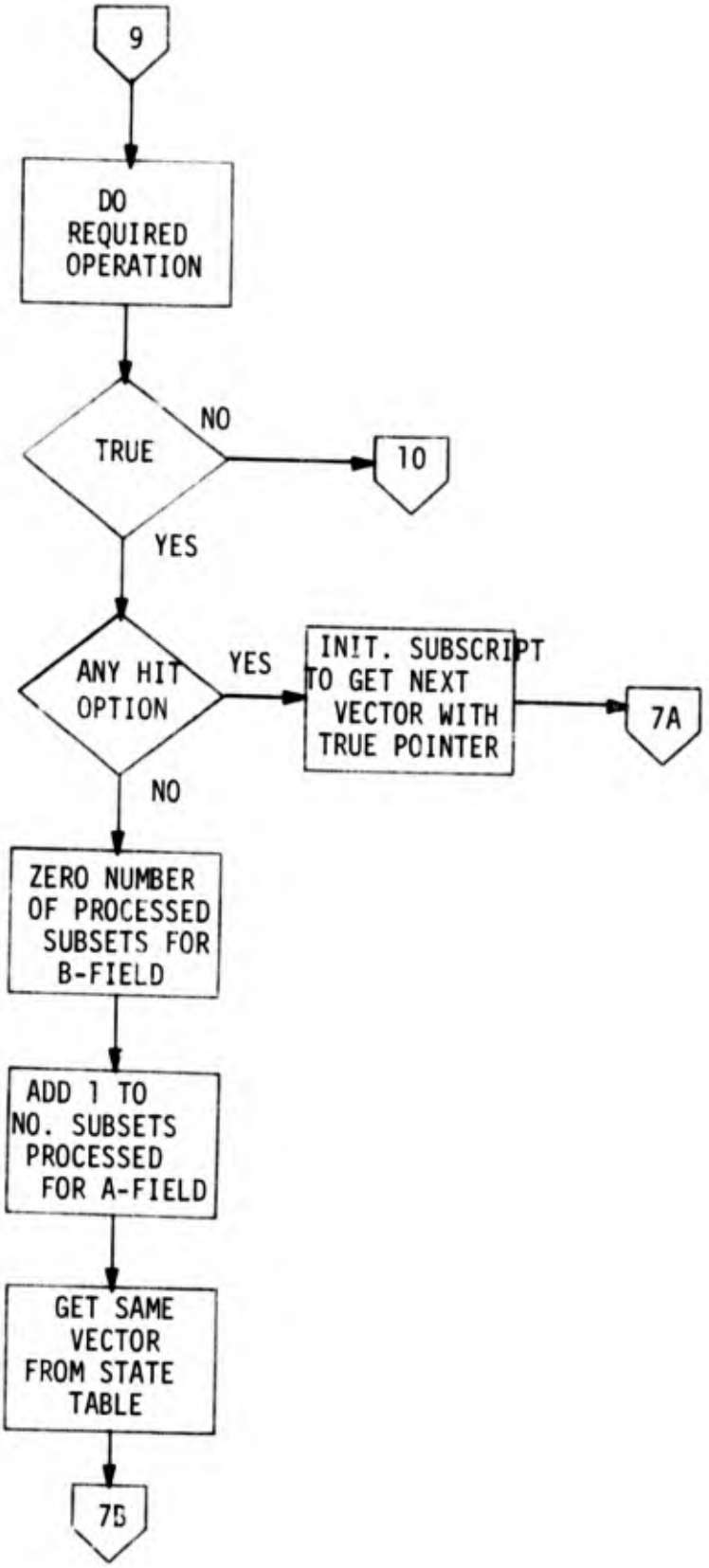
These Operations Will Flag Subsets When appropriate.

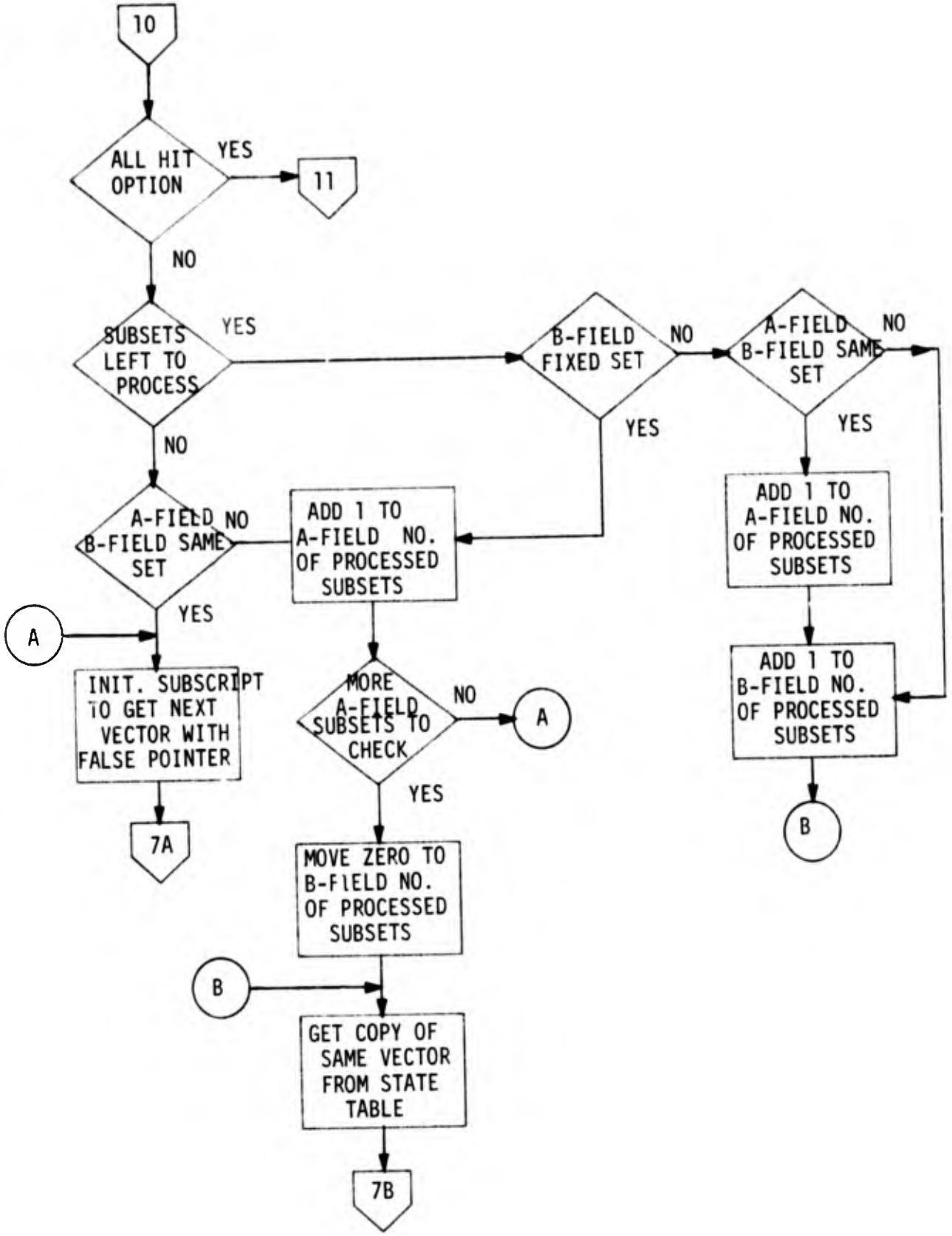
REPEAT IS USED FOR MULTIPLE B-FIELDS IN CONSTANT POOL

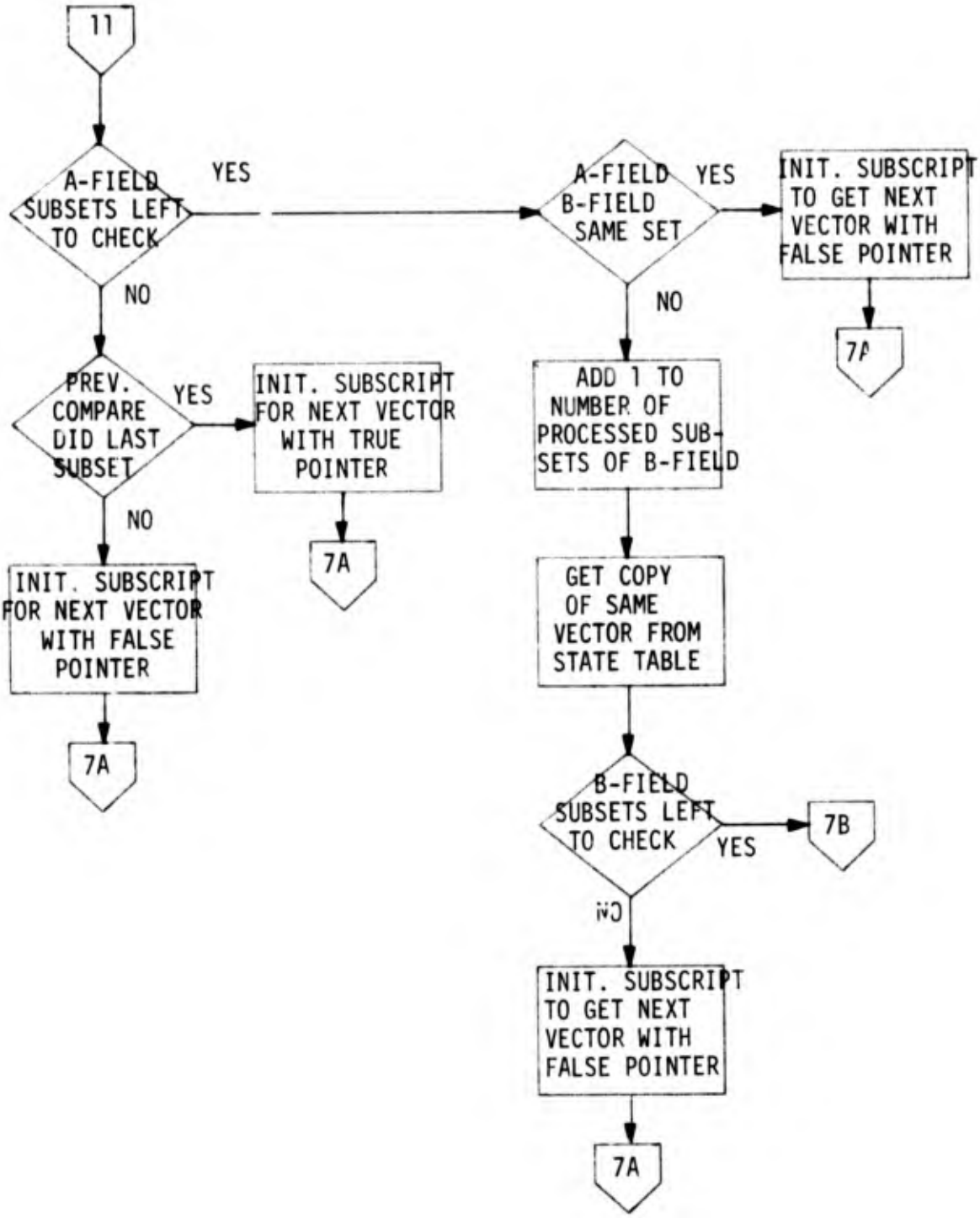

5
DO REQUIRED OPERATION
TRUE
NO
6
YES
A
"ANY" OPTION
YES
PUT TRUE POINTER IN L7 (BYPASS THIS TEST)
INITIALIZE SUB-SCRIPT TO GET VECTOR WHICH TRUE POINTER POINTS
4A
NO
"ALL" OPTION
YES
GET SAME VECTOR FROM STATE TABLE
ADD 1 TO S6A (NUMBER SUBSETS PROCESSED)
NO
WAS A SET PROCESSED
YES
MOVE LOGIC MODE TO SET TABLE S7A
PROCESSED ALL SUBSETS
NO
4C
YES
A
NO
INIT. SUBSCRIPT TO GET VECTOR WHICH TRUE POINTER POINTS
REPLACE SET TABLE DATA
4A

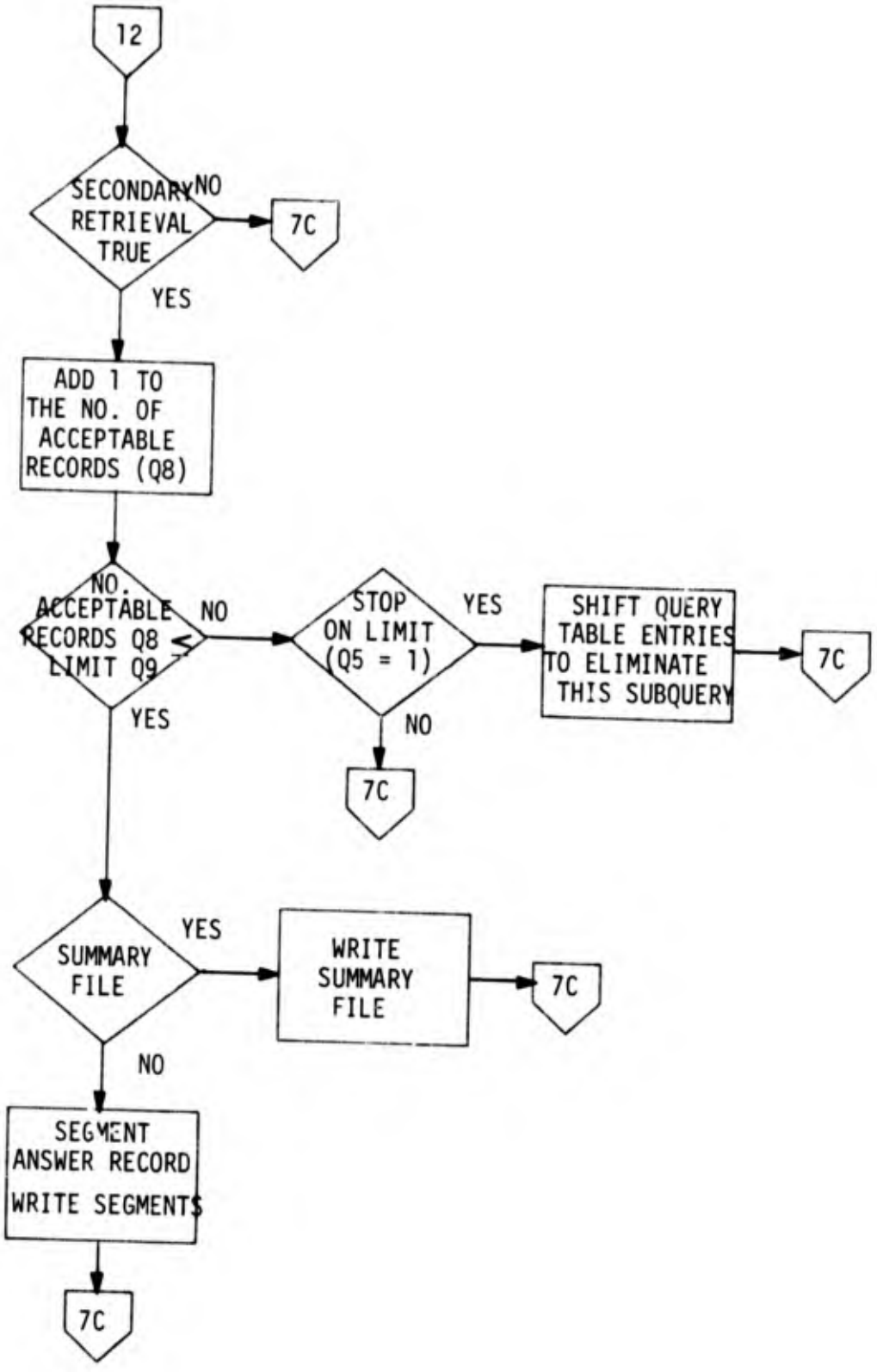

6
A
"ALL" OPTION
YES
MOVE FALSE POINTER TO BYPASS TABLE L7
INIT. SUBSCRIPT TO GET VECTOR WHICH FALSE POINTER POINTS
4A
NO
"ANY" OPTION
YES
ADD 1 TO S6A (NO. OF PROCESSED SUBSETS)
ANY MORE SUBSETS
NO
A
YES
NO
MOVE FALSE INDICATOR TO SET CONTROL
GET SAME VECTOR FROM STATE TABLE
4C
REPLACE SET INFORMATION BACK INTO SET TABLE
INIT. SUBSCRIPT TO GET VECTOR WHICH TRUE POINTER POINTS
4A


7
PRIMARY RETRIEVAL EXPRESSION TRUE OR FALSE
FLAG TRUE SUBSETS
7C
PRIMARY RETRIEVAL TRUE
NO
MORE SUBQUERIES TO PROCESS
YES
INIT. SUBSCRIPT TO GET NEXT SUBQUERY FROM QUERY TABLES
4
8
YES
NO
3A
INITIALIZE SORTKEY FOR ANSWER RECORD
ANY SORT OR MERGE (Q2> 0)
YES
BLANK OUT SET CONTROL INFORMATION OF FIXED SET
A
7A
A
NO
GET VECTOR FROM STATE TABLE
RETRIEVE SET CONTROL FOR VECTOR
*A-FIELD SET
B-FIELD SET
ZERO NUMBER OF PROCESSED SETS IN SET CONTROL
7B
OPERATION = **
YES
9
NO
12
** LESS NOT LESS
EQUAL NOT EQUAL
GREATER NOT GREATER
BETWEEN NOT BETWEEN
SPECIAL OP NOT SPECIAL OP
STOP FLAG
SORT MERGE
NOP LIMIT
KEEP
CONTAINS NOT CONTAINS
These Operators Work on Flagged Subsets

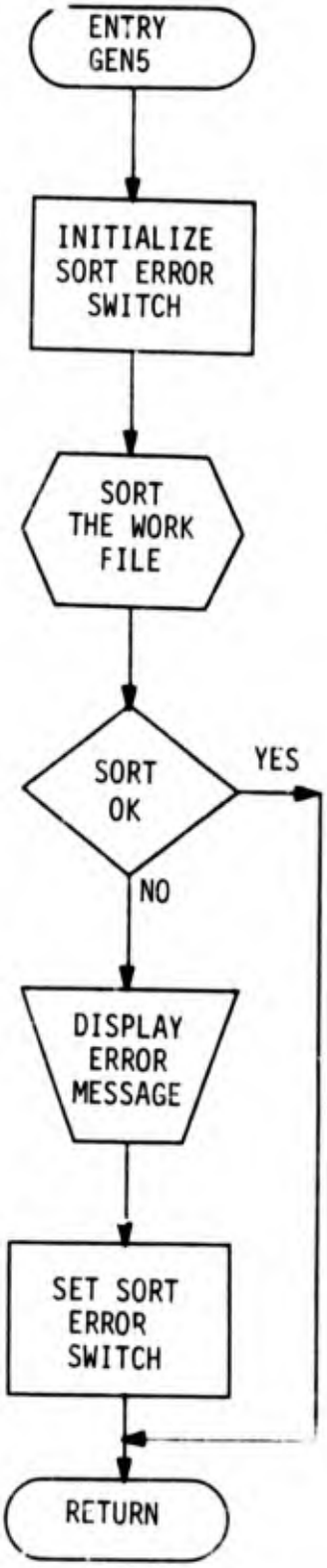

9
DO REQUIRED OPERATION
TRUE
NO
10
YES
ANY HIT OPTION
YES
INIT. SUBSCRIPT TO GET NEXT VECTOR WITH TRUE POINTER
7A
NO
ZERO NUMBER OF PROCESSED SUBSETS FOR B-FIELD
ADD 1 TO NO. SUBSETS PROCESSED FOR A-FIELD
GET SAME VECTOR FROM STATE TABLE
7B


10
ALL HIT OPTION
YES
11
NO
SUBSETS LEFT TO PROCESS
YES
NO
B-FIELD FIXED SET
NO
A-FIELD B-FIELD SAME SET
NO
YES
YES
A-FIELD B-FIELD SAME SET
NO
YES
ADD 1 TO A-FIELD NO. OF PROCESSED SUBSETS
ADD 1 TO A-FIELD NO. OF PROCESSED SUBSETS
A
INIT. SUBSCRIPT TO GET NEXT VECTOR WITH FALSE POINTER
MORE A-FIELD SUBSETS TO CHECK
NO
A
ADD 1 TO B-FIELD NO. OF PROCESSED SUBSETS
7A
YES
B
MOVE ZERO TO B-FIELD NO. OF PROCESSED SUBSETS
B
GET COPY OF SAME VECTOR FROM STATE TABLE
7B

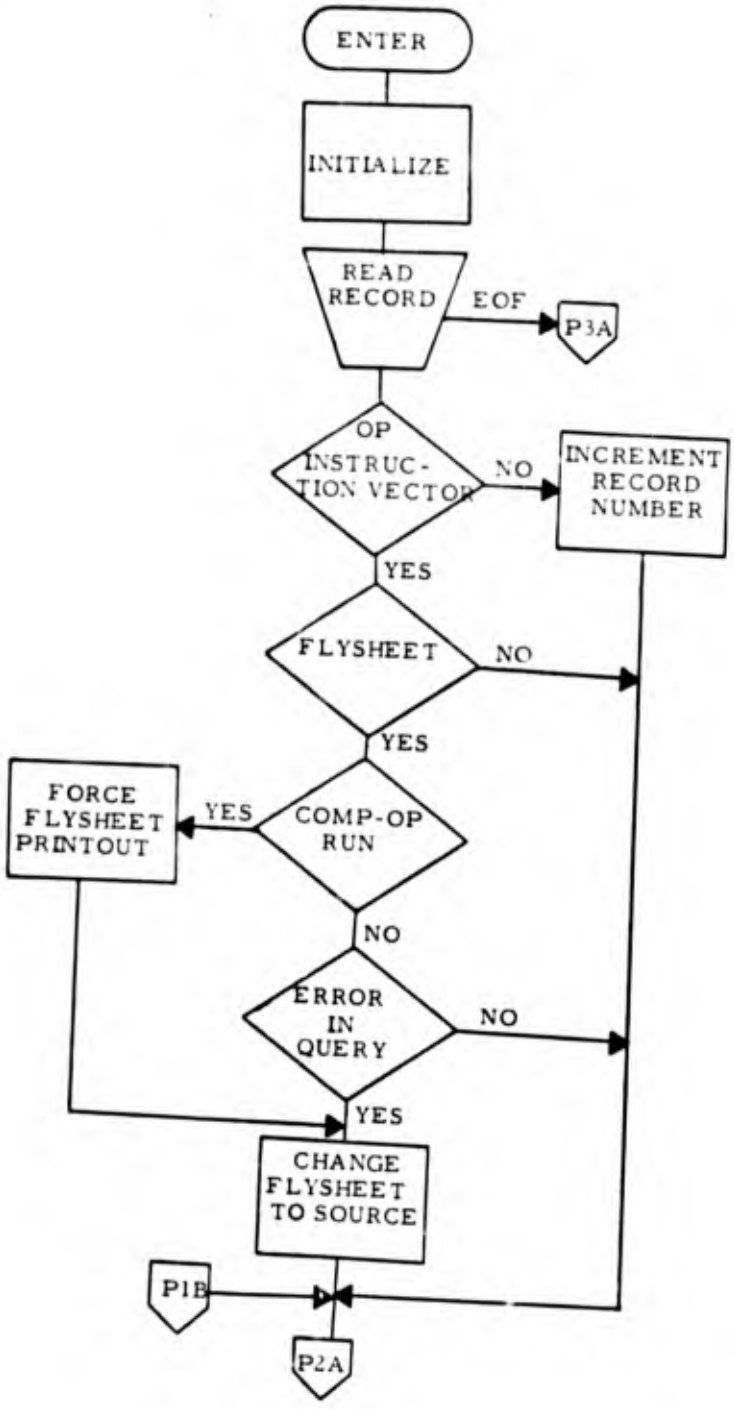

11
A-FIELD SUBSETS LEFT TO CHECK
YES
NO
A-FIELD B-FIELD SAME SET
YES
NO
INIT. SUBSCRIPT TO GET NEXT VECTOR WITH FALSE POINTER
7A
PREV. COMPARE DID LAST SUBSET
YES
NO
INIT. SUBSCRIPT FOR NEXT VECTOR WITH TRUE POINTER
7A
INIT. SUBSCRIPT FOR NEXT VECTOR WITH FALSE POINTER
7A
ADD 1 TO NUMBER OF PROCESSED SUB-SETS OF B-FIELD
GET COPY OF SAME VECTOR FROM STATE TABLE
B-FIELD SUBSETS LEFT TO CHECK
YES
7B
NO
INIT. SUBSCRIPT TO GET NEXT VECTOR WITH FALSE POINTER
7A

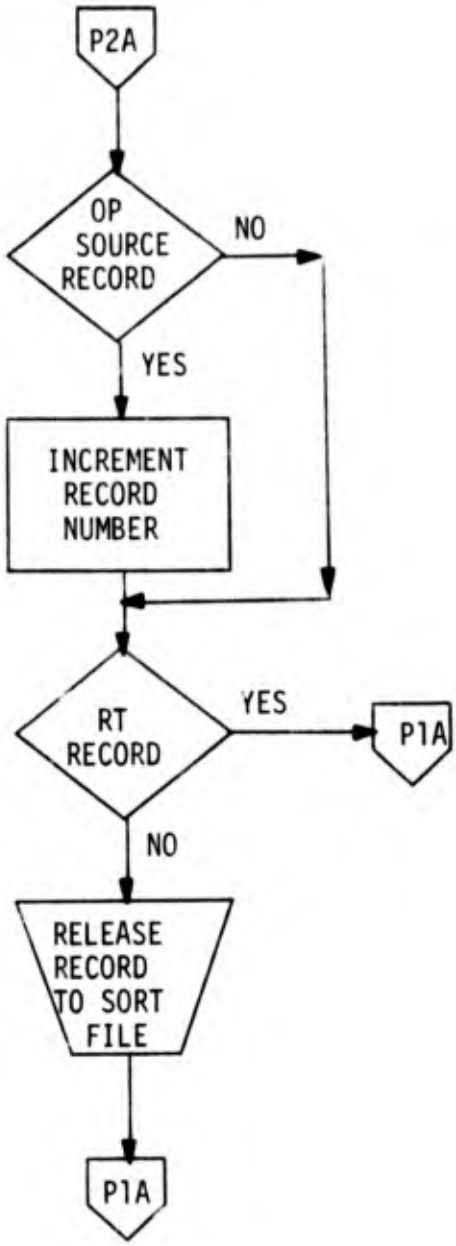

12
SECONDARY RETRIEVAL TRUE
NO
7C
YES
ADD 1 TO THE NO. OF ACCEPTABLE RECORDS (Q8)
NO. ACCEPTABLE RECORDS Q8 ≤ LIMIT Q9
NO
STOP ON LIMIT (Q5 = 1)
YES
SHIFT QUERY TABLE ENTRIES TO ELIMINATE THIS SUBQUERY
7C
NO
7C
YES
SUMMARY FILE
YES
WRITE SUMMARY FILE
7C
NO
SEGMENT ANSWER RECORD
WRITE SEGMENTS
7C

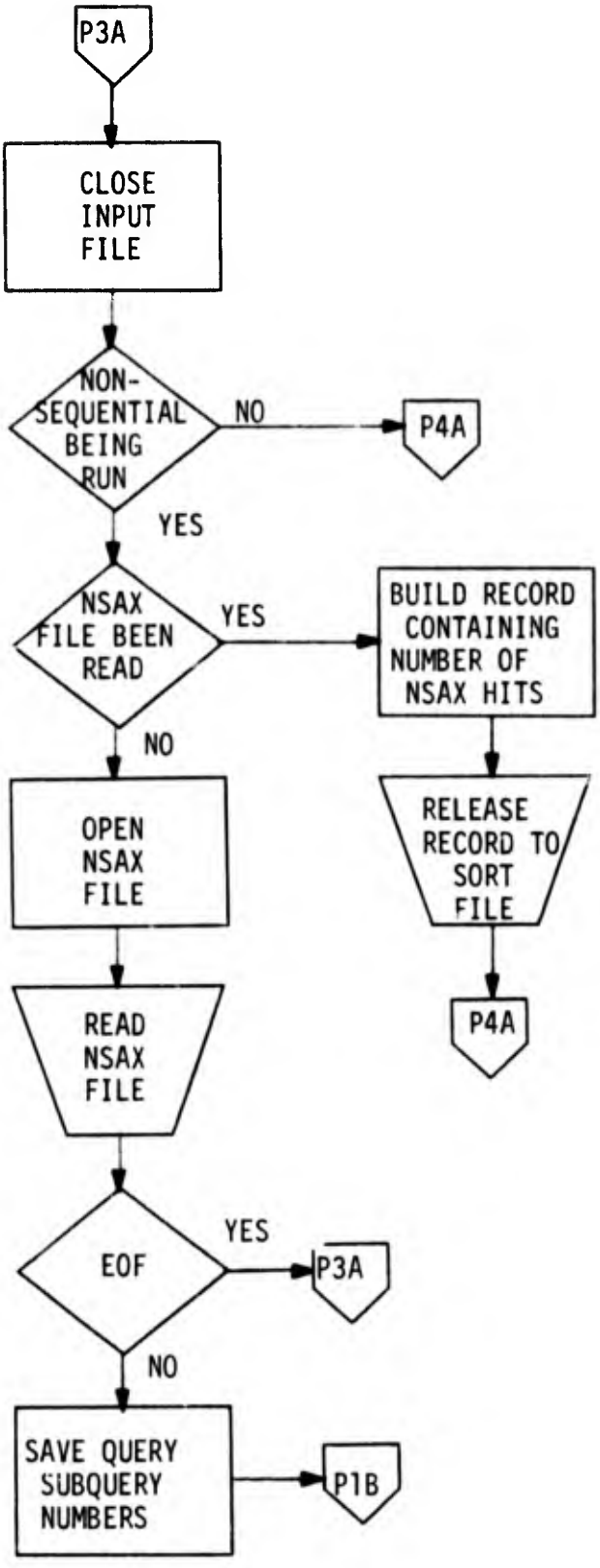
(12) GEN5.

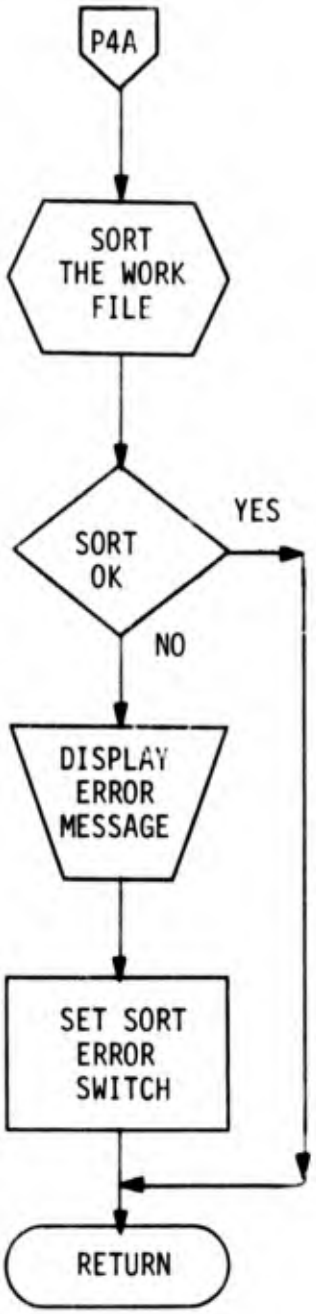

ENTRY
GEN5
INITIALIZE
SORT ERROR
SWITCH
SORT
THE WORK
FILE
SORT
OK
YES
NO
DISPLAY
ERROR
MESSAGE
SET SORT
ERROR
SWITCH
RETURN

(13) GEN5A

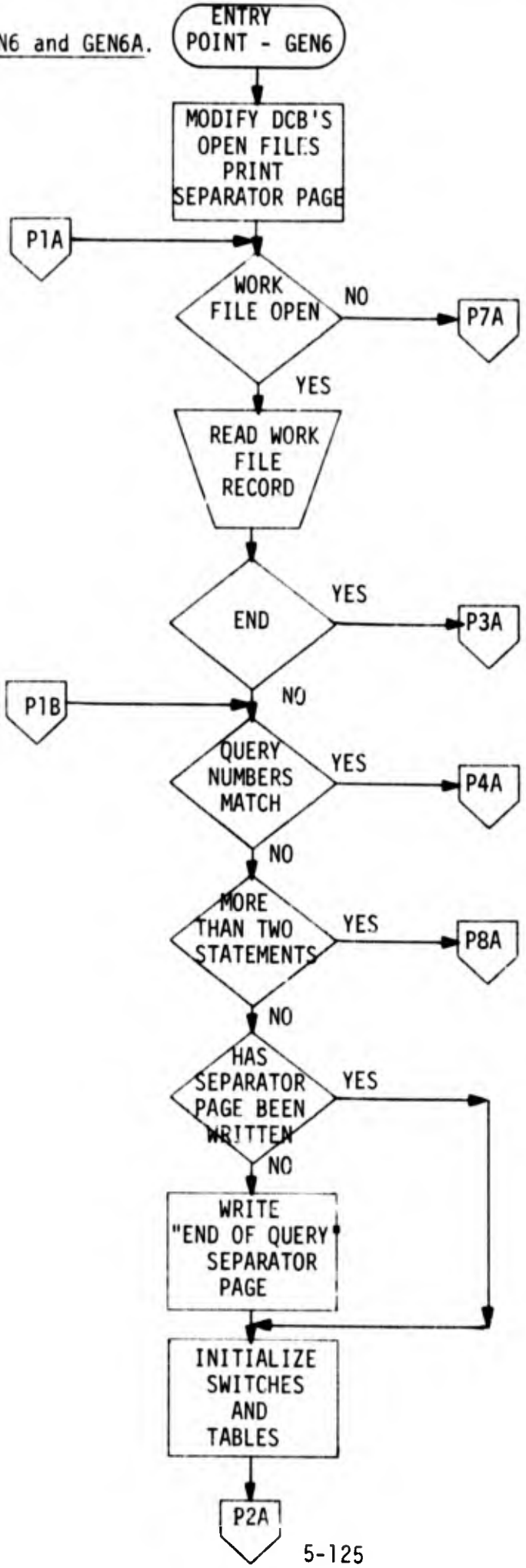

ENTER
INITIALIZE
READ RECORD
EOF
P3A
OP INSTRUC-TION VECTOR
NO
INCREMENT RECORD NUMBER
YES
FLYSHEET
NO
YES
FORCE FLYSHEET PRINTOUT
YES
COMP-OP RUN
NO
ERROR IN QUERY
NO
YES
CHANGE FLYSHEET TO SOURCE
P1B
P2A

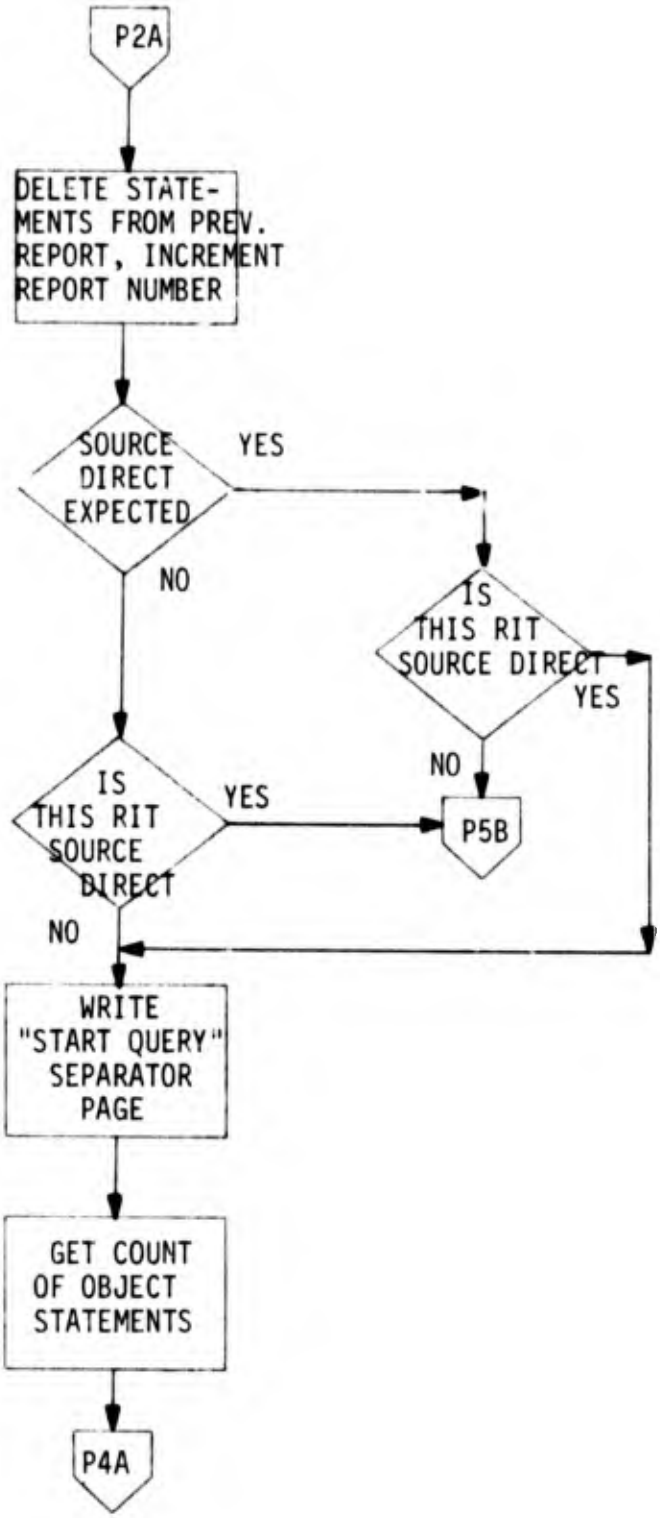

P2A
OP SOURCE RECORD
NO
YES
INCREMENT RECORD NUMBER
RT RECORD
YES
P1A
NO
RELEASE RECORD TO SORT FILE
P1A

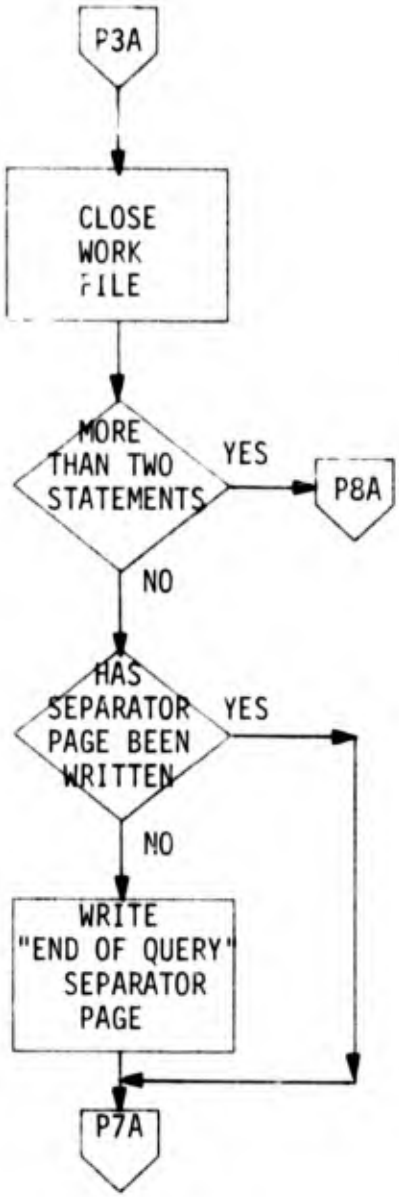

P3A
CLOSE INPUT FILE
NON-SEQUENTIAL BEING RUN
NO
P4A
YES
NSAX FILE BEEN READ
YES
BUILD RECORD CONTAINING NUMBER OF NSAX HITS
NO
OPEN NSAX FILE
RELEASE RECORD TO SORT FILE
P4A
READ NSAX FILE
EOF
YES
P3A
NO
SAVE QUERY SUBQUERY NUMBERS
P1B

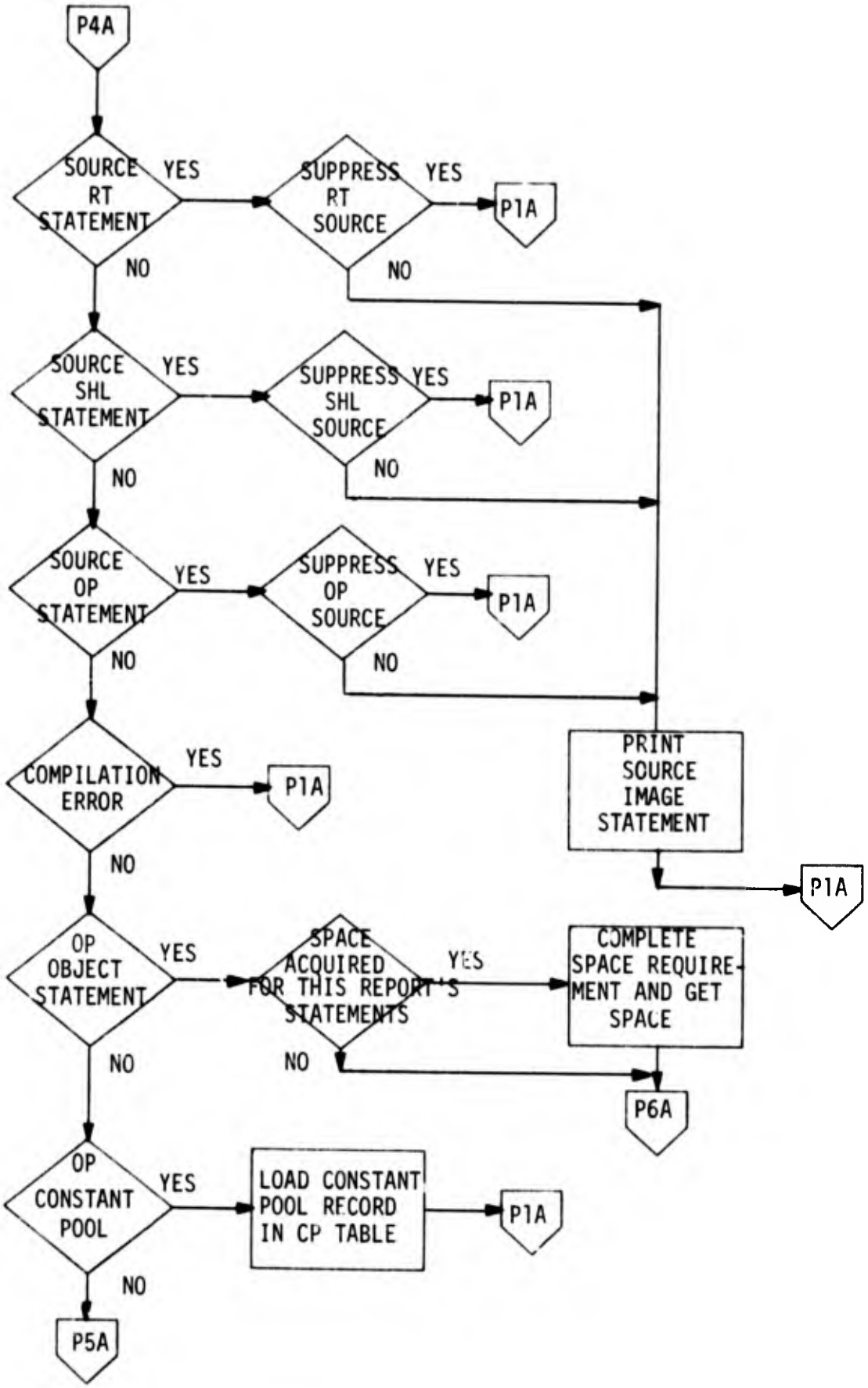

P4A
SORT THE WORK FILE
SORT OK
YES
NO
DISPLAY ERROR MESSAGE
SET SORT ERROR SWITCH
RETURN

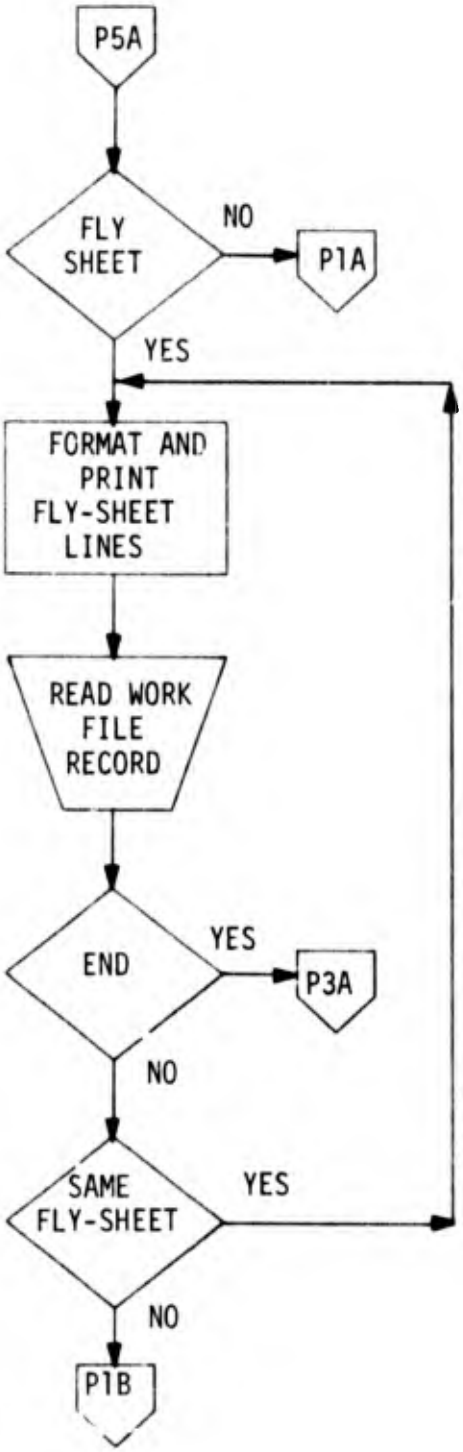
(14) GEN6 and GEN6A.

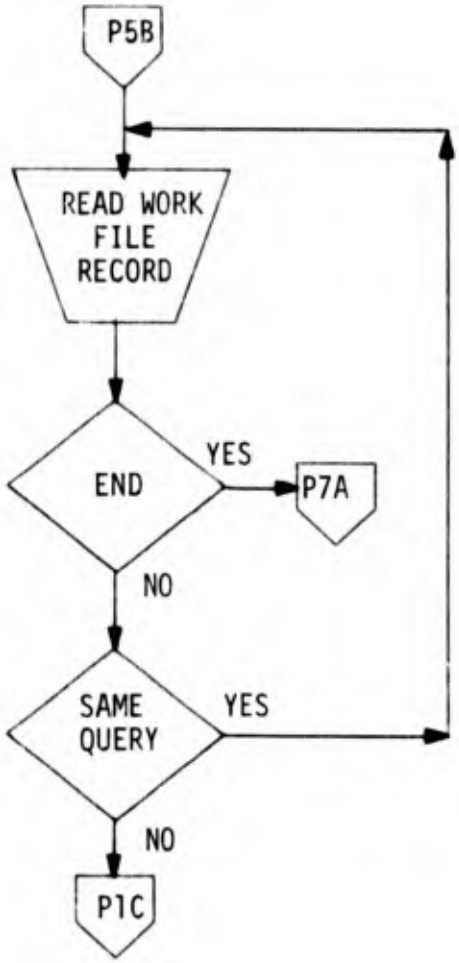


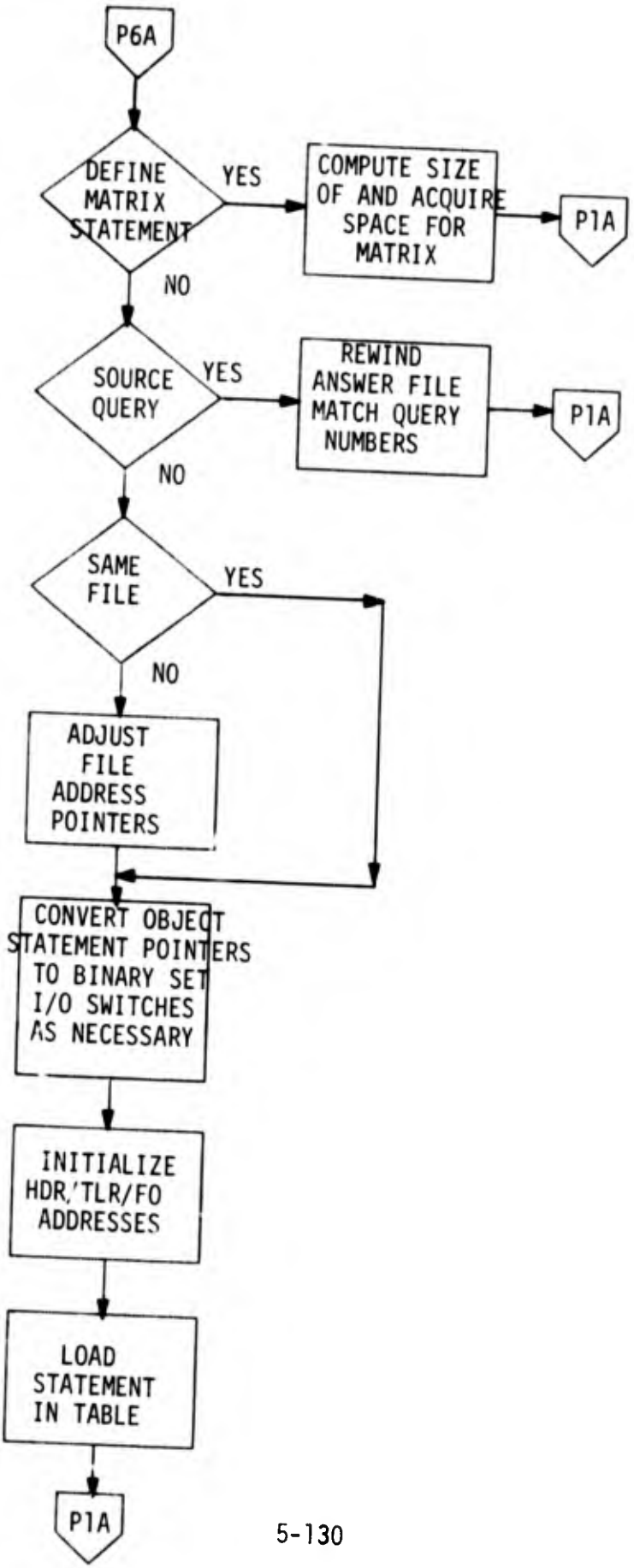

P2A
DELETE STATE-MENTS FROM PREV. REPORT, INCREMENT REPORT NUMBER
SOURCE DIRECT EXPECTED
YES
NO
IS THIS RIT SOURCE DIRECT
YES
NO
IS THIS RIT SOURCE DIRECT
YES
P5B
NO
WRITE "START QUERY" SEPARATOR PAGE
GET COUNT OF OBJECT STATEMENTS
P4A

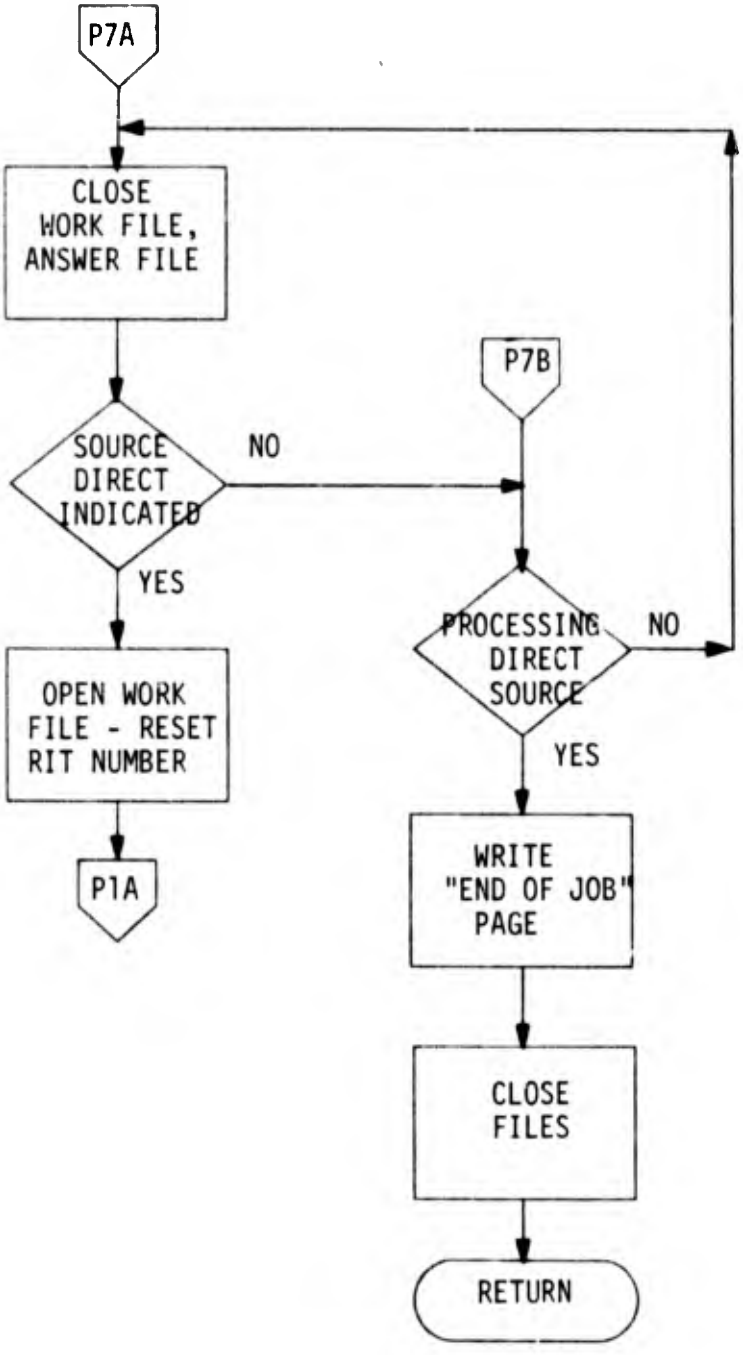

P3A
CLOSE WORK FILE
MORE THAN TWO STATEMENTS
YES
P8A
NO
HAS SEPARATOR PAGE BEEN WRITTEN
YES
NO
WRITE "END OF QUERY" SEPARATOR PAGE
P7A

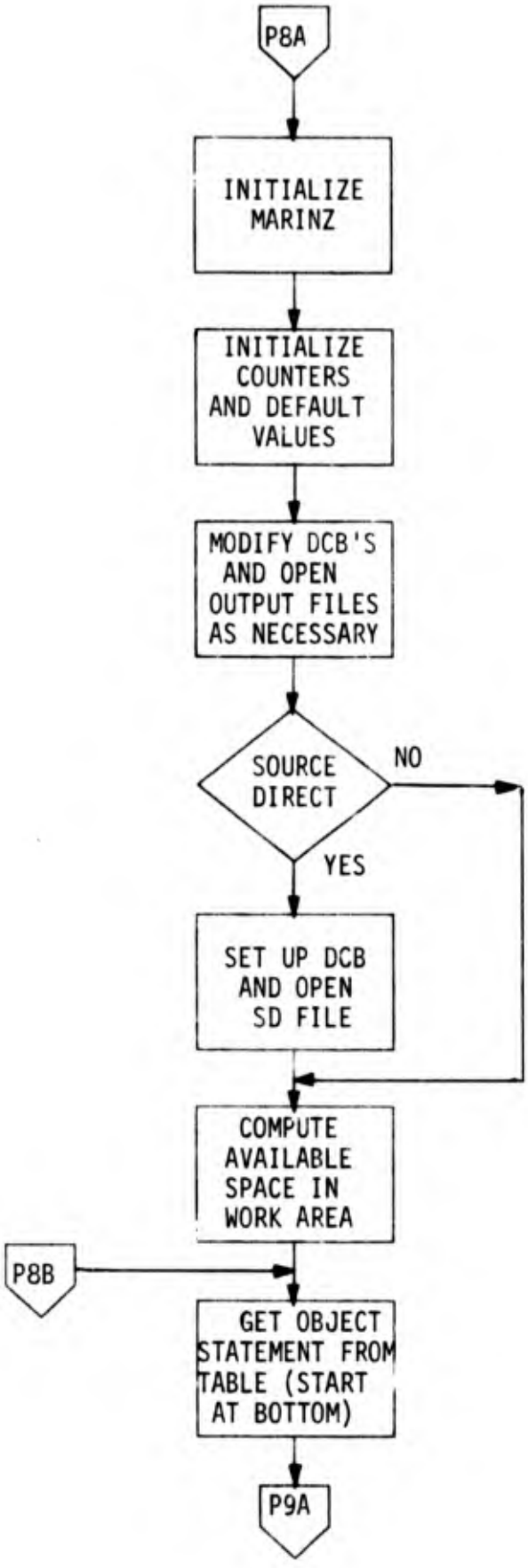

P4A
SOURCE RT STATEMENT
YES
SUPPRESS RT SOURCE
YES
P1A
NO
NO
SOURCE SHL STATEMENT
YES
SUPPRESS SHL SOURCE
YES
P1A
NO
NO
SOURCE OP STATEMENT
YES
SUPPRESS OP SOURCE
YES
P1A
NO
NO
COMPILATION ERROR
YES
P1A
PRINT SOURCE IMAGE STATEMENT
NO
P1A
OP OBJECT STATEMENT
YES
SPACE ACQUIRED FOR THIS REPORT'S STATEMENTS
YES
COMPLETE SPACE REQUIREMENT AND GET SPACE
NO
NO
P6A
OP CONSTANT POOL
YES
LOAD CONSTANT POOL RECORD IN CP TABLE
P1A
NO
P5A

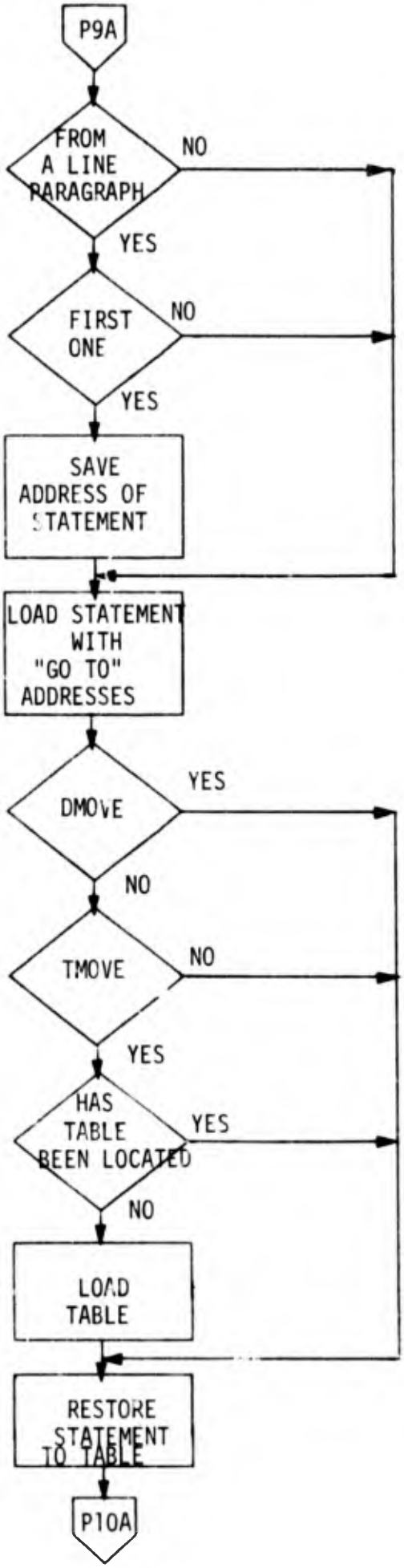

P5A
FLY SHEET
NO
P1A
YES
FORMAT AND PRINT FLY-SHEET LINES
READ WORK FILE RECORD
END
YES
P3A
NO
SAME FLY-SHEET
YES
NO
P1B

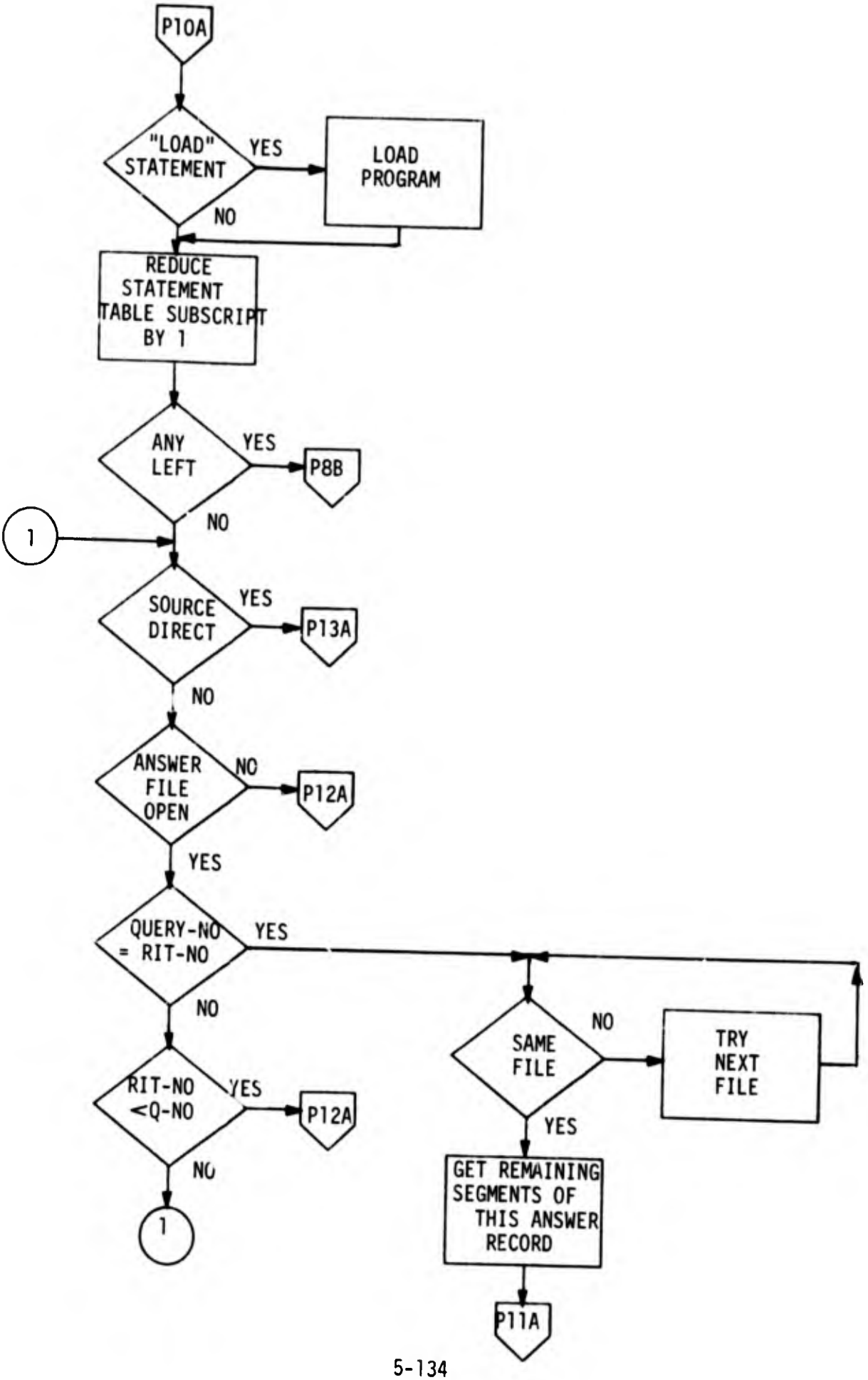

P5B
READ WORK FILE RECORD
END
YES
P7A
NO
SAME QUERY
YES
NO
P1C

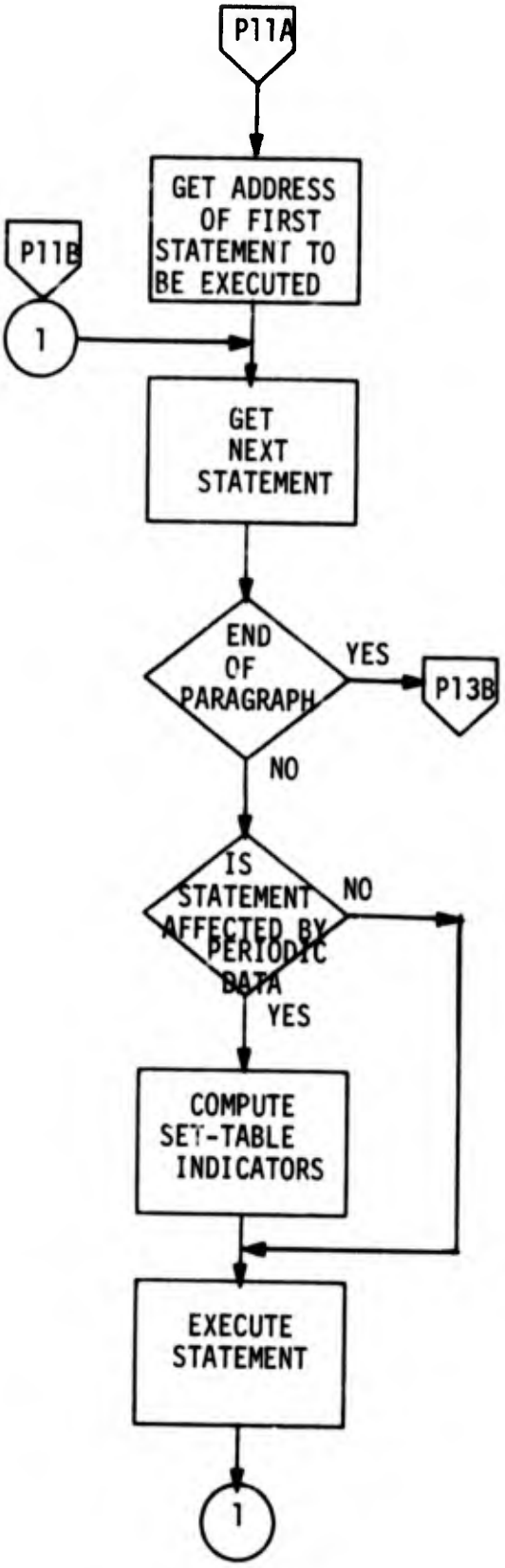

P6A
DEFINE MATRIX STATEMENT
YES
COMPUTE SIZE OF AND ACQUIRE SPACE FOR MATRIX
P1A
NO
SOURCE QUERY
YES
REWIND ANSWER FILE MATCH QUERY NUMBERS
P1A
NO
SAME FILE
YES
NO
ADJUST FILE ADDRESS POINTERS
CONVERT OBJECT STATEMENT POINTERS TO BINARY SET I/O SWITCHES AS NECESSARY
INITIALIZE HDR,'TLR/FO ADDRESSES
LOAD STATEMENT IN TABLE
P1A

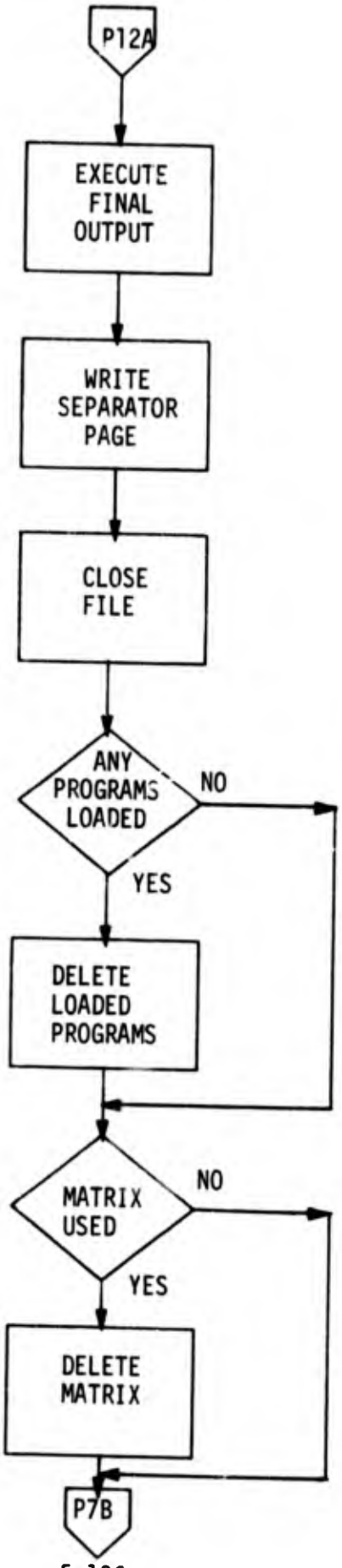

P7A
CLOSE WORK FILE, ANSWER FILE
P7B
SOURCE DIRECT INDICATED
NO
YES
PROCESSING DIRECT SOURCE
NO
YES
OPEN WORK FILE - RESET RIT NUMBER
P1A
WRITE "END OF JOB" PAGE
CLOSE FILES
RETURN

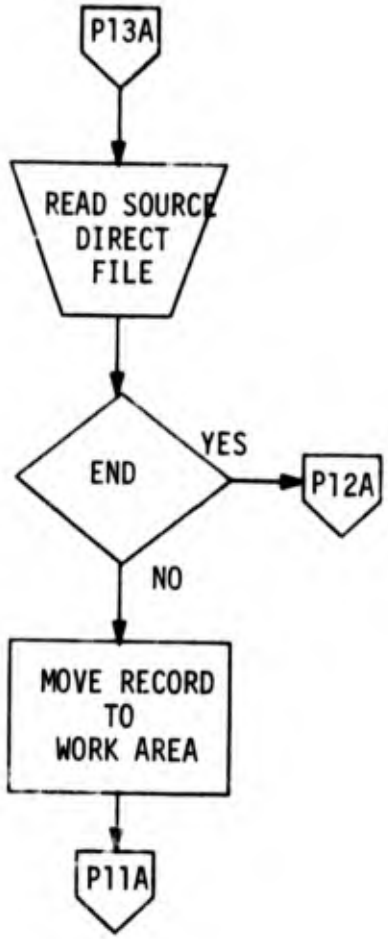

P8A
INITIALIZE MARINZ
INITIALIZE COUNTERS AND DEFAULT VALUES
MODIFY DCB'S AND OPEN OUTPUT FILES AS NECESSARY
SOURCE DIRECT
NO
YES
SET UP DCB AND OPEN SD FILE
COMPUTE AVAILABLE SPACE IN WORK AREA
P8B
GET OBJECT STATEMENT FROM TABLE (START AT BOTTOM)
P9A

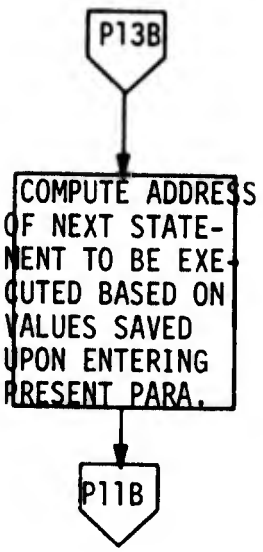

P9A
FROM A LINE PARAGRAPH
NO
YES
FIRST ONE
NO
YES
SAVE ADDRESS OF STATEMENT
LOAD STATEMENT WITH "GO TO" ADDRESSES
DMOVE
YES
NO
TMOVE
NO
YES
HAS TABLE BEEN LOCATED
YES
NO
LOAD TABLE
RESTORE STATEMENT TO TABLE
P10A


P10A
"LOAD" STATEMENT
YES
LOAD PROGRAM
NO
REDUCE STATEMENT TABLE SUBSCRIPT BY 1
ANY LEFT
YES
P8B
NO
1
SOURCE DIRECT
YES
P13A
NO
ANSWER FILE OPEN
NO
P12A
YES
QUERY-NO = RIT-NO
YES
NO
SAME FILE
NO
TRY NEXT FILE
YES
GET REMAINING SEGMENTS OF THIS ANSWER RECORD
P11A
RIT-NO <Q-NO
YES
P12A
NO
1


P11A
GET ADDRESS OF FIRST STATEMENT TO BE EXECUTED
P11B
1
GET NEXT STATEMENT
END OF PARAGRAPH
YES
P13B
NO
IS STATEMENT AFFECTED BY PERIODIC DATA
NO
YES
COMPUTE SET-TABLE INDICATORS
EXECUTE STATEMENT
1


P12A
EXECUTE FINAL OUTPUT
WRITE SEPARATOR PAGE
CLOSE FILE
ANY PROGRAMS LOADED
NO
YES
DELETE LOADED PROGRAMS
MATRIX USED
NO
YES
DELETE MATRIX
P7B


P13A
READ SOURCE DIRECT FILE
END
YES
P12A
NO
MOVE RECORD TO WORK AREA
P11A


P13B
COMPUTE ADDRESS OF NEXT STATE-MENT TO BE EXE-CUTED BASED ON VALUES SAVED UPON ENTERING PRESENT PARA.
P11B

b. Program Narrative.

(1) GEN0.

PROCEDURE DIVISION

Initialize

STEP1

Offset OP-CNT.
Execute GEN1.
Perform COMP-OP processing if TYPE-S (2) is not equal nine (i.e. TYPE-S (2) = 7 or 8).
If multiple subqueries, execute GEN1A.
Q-SWT (108) equal to five indicates bad sort; terminate.

STEP1A

Flip-Flop work files. If non-sequential is present, flip-flop for NSAX.

STEP2

Check for subqueries and shorthand language.

STEP2A

Call GEN2 and flip-flop work files.

STEP3A

If SHL is indicated by TYPE-S (4), call GEN3A and flip-flop work files.

STEP3

If OP is indicated by TYPE-S (5), call GEN3 and flip-flop work files.

STEP4

If NSAX is required, call NSAX compiler and processors; flip-flop work files.

STEP4-1

If no subqueries, go to work file sort. If TYPE-S (11) equal to zero, no periodic fields were referenced by retrieval, otherwise call GEN4.

STEP4A

Only called when no periodic fields are referenced in any subquery.

STEP5

Check for ABORT condition or user requested sort.

STEP55

Call answer record sort, check for invalid sort.

STEP5A

Call work file sort, check for invalid sort.

STEP 6

If periodics are referenced by OP, TYPE-S (12) not equal to zero, call GEN6 and stop run.

STEP6A

GEN6A is only called if no periodic fields are referenced in any report.

STEP7

Compile a Shorthand Language or Standard Language report. (GEN1 sets TYPE-S (4) = 1 if an SHL-RPT card is encountered)

STEP7B

Sort the source input and object vectors (GEN5A) and save them on the MIDMS library (GEN3B).

FLIP

This paragraph is PERFORMed to alternate input and output files for intermodule communication

(2) GEN1

DATA DIVISION

(GEN1 Data Division Additions for COMP-OP)

| Name | Function |
|---|---|
| COMP-OUT | Working storage area of 108 bytes for output of object vectors. |
| COMP-JCL | Working storage area for COMP-OP and Library card scans. |
| COMP-NAM | Working storage area for storage and testing of library and COMP-OP names. |
| LIB-CARD-SAVE | Working storage area to insure the proper saving of LINKAGE SECTION data. |
| CURR-NAM | One byte with value of space for testing and loading in card scan. |
| CARD-ERR | Element = 1 byte Purpose |
| | (1) Indicates COMP-OP Card |
| | (2) Not used |
| | (3) Report or SHL Card |
| | (4) Library name ends in "R" |
| | (5) Library name ends in "F" |
| | (6) . . . (10) Not Used |

GEN1 COMP-OP SWITCH TABLE

| Switch-Name | Variable (V) or Fixed (F) | Values | Purpose |
|---|---|---|---|
| H1 | V | Depends on name | Counter used in COMP-OP Scan |
| H2 | V | Depends on LIB Card | Counter used in LIB Card Scan |
| H3 | V | Depends on LIB Name | Counter used in LIB Name Scan |
| J-SWT | V | 0<br>1 | Initial Value<br>Indicates execution of an object report |
| X | V | Depends on COMP-OP | Counter used in COMP-OP Scan |
| CP | V | Dependent on number of LIB cards | Indicates number of LIB Cards |
| I | V | Depends on Name | Counter used in Card Scan |
| J | V | Depends on Name | Counter used in Card Scan |
| K | V | Depends on Name | Counter used in Card Scan |
| ERR | V | 0<br>1 | Initial Value<br>Indicates error vector has been written |

PROCEDURE DIVISION

The first paragraph is the entry point of the load module GEN1 called by GEMIDMS, the driver. DD-INPUT and DD-OUTPUT are Linkage Section labels containing the names of the input and output work files, respectively. The two calls to MARINM move the specified DD names to the DCB's in the program. The files are opened and housekeeping is performed. The first record is read to determine whether the ABORT option is to be exercised. This option is only permitted on the first card in the MIDMS job and must precede all QUERY cards. If the ABORT is found, the QUERY error vector will have a one moved into position one hundred. Each preceding position in QUERY-SWT corresponds to the query number of the jobs in the batch. A one in any position indicates an error in that query and will prevent that query from being executed.

G1.

The first statement initializes ERR, which is set to one whenever an error condition is encountered processing a COMP-OP or COMP-OP and LIBRARY card combination. A source card is then read, and if this card is a COMP-OP Card, CARD-ERR(1) is checked to see that it is the first COMP-OP card encountered. If it is not, an error routine is performed and an exit made from the program.

G2.

If the first card is a COMP-OP Card, CARD-ERR(1) is loaded to so indicate and 8 is moved to TYP-ST(2) to indicate a COMP-OP run and the input image (CD-IN) is moved to the COMP-JCL Work area. The Switch CP is set to one to indicate a COMP-OP run and the COMP-OP card scans are performed to load the COMP-OP name. In the case where CP is equal to one (COMP-OP card processed) and the card in question is a LIBRARY Card, 8 is moved to TYP-ST (2) to indicate a COMP-OP run, the input image is moved to the working storage area, and the routines which extract the library name are performed. The "ERR" Switch is then checked to see if an error occurred in the processing of the COMP-OP or Library Card, and if so, it is initialized and the program is routed back to G1. If no error occurred, the normal GEN1 processing continues.

G2-2

Checks for significant first words or source cards. LIB and LIBRARY test will only be true if a LIB card has been stored. SUBQUERY, QUERY, and REPORT cards are major break cards. MODIFY causes a change in the source input stream and SHL-SWT indicates the shorthand compiler will be involved.

G2-3

Error condition - LIBRARY card has been stored on the MIDMS Library.

G2-4

Causes SUMMARY card to be located at the front of a subquery.

G3

Checks for missing query card and causes error message to be written.

G3A

Writes source image on output work file.

G4

Initialize column counter.

G5

Look for first word on card.

G5A

Shift card image one position.

G6

Identify first word, DEFINE is sequenced by three.

G7

All source cards except SUMMARY and DEFINE are sequenced by twenty.

G10

Check for library name.

G10A

A LIB call must be preceded by a QUERY card - if not, process an error.

G11

Check for name of item being called from library.

G12

Look for word.

G12A

Shift source image and loop back.

G13

Invalid library name error processing.

G13A

Flag error and write out source image.

G13B

Process error message.

G14

The fifth character of the library name is checked to insure validity (Q, R, F or J) and if valid, processing continues; if not, a branch to G13 occurs and an error is processed. If the name ends in "J", J-SWT is set to indicate that object vectors are to be processed.

G14B

Library name must be five characters; if not, process an error.

G14A

Write out source image, initialize library segment number.

G15

An "R" is moved to operation to indicate a read (an "N" will be returned on a invalid condition) and JJ is incremented and moved to SUFFIX-NAME to indicate the name of the member on the library. If J-SWT is greater than zero it indicates object vectors are being processed.

G16

Operation is first checked to determine if a valid read has occurred (Operation = R) and if so, J-SWT is checked to see if an object vector is to be processed. If so, a branch to OB-CODE occurs. ITEM-SIZE is then divided by 80 and if equal to zero, an error is processed. If Q is greater than 125, it indicates the improper storage of data on the library member, and an error is processed. Q is then multiplied by 80, and if it is equal to ITEM-SIZE, it indicates no images were dropped due to the block not being a multiple of 80, and processing continues. If not, QUERY-SWT (QUERY) is set to one to indicate an error and an error message is processed.

G17

Q is the number of cards (source images) in the library member just read.

G18

Check for end of library cards.

G19

If object vectors are being processed (J-SWT = 1) and it is the first vector, COM-SAV is performed to insure the proper processing of this first vector and processing continues. If not, and if an object vector is being processed, the vector is written and control returns to normal processing.

G20

If first member is missing, name is erroneous.

G21

Process library error.

G22

Process Library error.

G30

Initialize NSAX switch.

G30-1

Has query card been read?

G30A

Process missing query and error.

G31

Determine NSAX source type.

X-DICK

Check for NSAX file name.

G32

Look for word.

G32-1

Process NSAX error message.

G32A

Shift left and loop back.

G33

Initialize for change in file name.

G34

Initialize new subquery headers.

G35

Check for optional word.

G36

Look for next word.

G36A

Shift and lookback.

G37

Save filename.

G40

CP is checked to see if this is a COMP-OP run, and if so, an error is processed. If not, the sort switch is set as required by the MODIFY statement.

G41

Look for MODIFY name.

G41A

Shift and loop back.

G42

Set error vector and process error message for bad MODIFY statement.

G43

Check next work on MODIFY card.

G43A

Reports have a high value "file name".

G44

Initialize for searching next word.

G45

Look for word.

G45A

Shift and loop back.

G46

Check for numeric subquery number.

G47

Check number.

G48

Check for numerics.

G50

Check for word.

G50A

Shift and loop back.

G51

Check for optional word.

G52

Look for file name.

G52A

Shift and loop back.

G53

Save filename.

G54

Process missing QUERY card error.

G54A

Sequence and write out card.

G60

Check for stored QUERY card.

G60A

Initialize a new query.

G70

Check for prior QUERY card. If none, process error.

G71

Initialize a shorthand language report.

G71A

Adjust card number to place at front and write out image.

G75

Is a sort required? If so, indicate with TYPE-ST (9).

G99

Go back.

G100

If no sort, set up to copy work file.

G101

Copy work file to align supervisor calls.

G102

Return.

COP.

Scan begins looking in C.C.8 for the COMP-OP name. If none is found, NO-NAME is performed and one is moved to ERR to indicate improper input and control is returned to G1. SCAN1 is then performed to load the COMP-OP name into the COMP-NAM Area in Working Storage. The name is then checked to insure it ends in 'R' and, if not, ERR is set and control is returned to G1. If all is O.K., COP-OUT is performed.

SCAN.

COMP2-1(I), a character in the COMP-OP card, is moved to a dummy field, CURR-NAM, until a non-space is found.

SCAN1

(Described in COP)

COP-OUT.

The sort-keys are set so the COMP-OP card is the first card in the output stream, the COMP-OP name moved into the Working Storage area and the entire source image is written out.

BAD-END.

An error message is written indicating that the COMP-OP name did not end in 'R'.

NO-NAME.

An error message is processed to indicate that no name was found on the COMP-OP card.

LIB-R.

Much the same theory of processing was used here that was used in COP: A COMP-OP run LIBRARY card is scanned to determine its "name" and store it. CURR-NAM, the dummy field, is initialized, and SCAN is performed until the first space is found after "LIB" or "LIBRARY" on the card. LIB-SCAN is then performed to load the library name into the working storage area LIB-NAME. Then, depending on the value of the fifth character in the library name ('R' or 'F') the appropriate element of CARD-ERR is loaded for later error checking. The fifth element of the library name is then checked to be certain it is an "R" or an "F"; if not, an error message is processed. If the fifth character is an "R" and CARD-ERR (3) = 1 (Report or SHL Card encountered) an error message is processed and control is returned to G1. LIB-CARD is then checked. If it is not equal to zero an 8 is stored in TYPE-ST(2) indicating both source and object are to be stored. LIB-NAME (5) is again checked, and if it is not equal to "R" at this point, it indicates a Fly-sheet (F) and a branch is effected to LIB-R5. The last processing consists of setting TYP-ST (2) to 7 to indicate that object only is to be stored.

LIB-R5.

One is added to LIB-CARD to indicate the successful processing of a Library Card, and to indicate to the LIB-R Routine that should another library card be encountered, it must end in an "F".

LIB-SCAN.

The name of the library member is taken character by character and stored in the area LIB-NAME.

LIB-ERR2.

The error vector is loaded and error 14 is written.

LIB-ERR1.

The error vector is loaded and error 16 is written.

LIB-ERR3.

The error vector is loaded and error 15 is written.

OB-CODE.

The processing theory here is much the same as in G16, except that object vectors are being processed rather than source images. Note that the branches on unsatisfactory conditions are to the same areas

OB-CODE (cont'd)

as occur in G16. Again, the ITEM-SIZE is divided by 108 (rather than 80) to enable the testing of Q for the proper blocking. If there are errors, branches to the appropriate error routines occur, and if correct, a branch to B1 occurs and normal processing continues; if not error 2 is written into the stream.

LIB-ERR.

Error message 13 is processed.

COM-SAV.

(See G19) This paragraph stores the compiler/logic processor communication data into the Linkage Section.

LIB-ERR4.

Error vector is loaded and error message 17 is processed.

LIB-ERR5.

Error vector is loaded and error message 12 is processed.

LIB-ERR6.

Error vector is loaded and the error message indicating the illegality of batching COMP-OP runs is processed.

LIB-ERR7.

SUBQUERY cards are not allowed in COMP-OP runs.

WRITE-CARD SECTION

Write work file record.

(3) GEN1A.

PROCEDURE DIVISION

Entry point for source statement sort Q-SWT (108) is reset.

G75

Initiate COBOL sort.

G99

Check for sort error; if one exists, indicate in Q-SWT (108).

SORTOK

Normal return.

INSRT SECTION

Open input file.

INSRT1

Provide records to the sort.

OUTSRT SECTION

Open output file.

OUTSRT1

Receive records from the sort, rearrange fields, and write out.

(4) GEN2.

PROCEDURE DIVISION

The first paragraph is the entry point of the load module, GEN2, which is called by the supervisor, GEMIDMS. DD-INPUT and DD-OUTPUT contain the names of the work files to be used as input and output for GEN2. These names are moved to the appropriate DCB's prior to opening the files.

HOUSE-KEEP

The files are opened and initialization is performed.

NEW-STATE

MISSING-SWT, if equal to one, indicates that a user did not submit a continuation card properly. Input records are sorted by file name. If FILE-INA is less than FILE-S the input file is out of sequence. TYPE-IN, if equal to one, indicates retrieval source statements. CARD (1), if equal to asterisk, indicates a note card.

NEW-STATE1

SUBQ-IN, if equal to zero, indicates a LIB or QUERY card.

SUBQ-S, if equal to zero, indicates object statement header not built.

SUBQ-S, if not equal to SUBQ-IN, indicates a change in subquery number.

QUERY-IN, if nct equal to QUERY-S, indicates a change in query number, causing also a change in subquery. FILE-INA indicates a new data file. If SKIP-SWT is equal to one, an error exists. SKIP-SWT remains one until the subquery changes. Its purpose is to indicate an error in processing of the current subquery.

INITIAL-REC

Initialize save areas. EXTRACT-WORD extracts a word from the source input card. If an error is detected, EXTRACT-WORD moves a one to BYPASS. The BYPASS switch is used throughout as an error-indicator switch. EXTRACT-WORD returns the length of the word, in characters, via IL. If IL is equal to zero, no word was found.

FIRST-WORD

The first word of a card determines the statement type. Most types result in a GO to C-name corresponding to the first word name. LIB and MODIFY cards are dropped, QUERY cards are trapped above by their header information. Notice that most, but not all, first words are followed by a blank. If the first word is not in the list, an error is processed and the next card image is read.

NEW-STATEQ

Reset error switch. This is done with each new query since errors are not carried across queries unless the ABORT is used. IC is set to one, to indicate card column 1 as the starting position for EXTRACT-WORD.

CHECK-SHL

END-STEP closes the previous subquery by writing object "Break" statements. TYPE-IN, if equal to four, indicates a SHL statement.

C-SHL1

TYPE 1 was trapped earlier, TYPE 4 now. If neither, no processing is necessary in GEN2 except that all other types of records must be written on the output work file for subsequent processing. If the query numbers are the same, the card is a continuation of the current SHL report being processed. If different, and each SHL report must have a unique query number, the number is saved and LIST-SWT is initialized.

C-SHL1A

Initializes switches, IC indicates starting card column. Extract the first word, check for error (BYPASS), look for LIST or FILE type card.

C-SHL2

Write a record.

C-SHL3

Loop back.

C-SHL4

To enter, the card must have been a SHL FILE card. Continuation card is not permitted. Initialize header part of record, save old filename. Extract filename and check for length between five and eight. If not OK, continue to next card; this error will be picked up later by the output compiler. If no change in filename, get next card. Value of one in LIST-SWT indicates a SHL file name has been encountered. The fifth character of the name must be an 'A'. Change the file name to an FFT name, set up LB calling sequence, call LR7. If LR7 has length of zero, it does not exist; issue warning.

C-SHL4A

Save the sizes of the logical records. Load LR-2, LR-8 and LR-9 into appropriate save areas. Check for a proper return from the call to the librarian.

C-SHL4B

If LR-9 exists, load it into the area LR99.

C-SHL4C

Change the FFT name back to the file name and save it.

C-SHL5

Process an error.

C-SHL6

Initialize switch for next statement and go back.

C-LOAD

Get the next word on the LOAD card. BYPASS, equal to one, is an error meaning no word was found. The word extracted should be the name of a convert routine or special operator surrounded by asterisks. CARDW8 is the name padded with trailing blanks to 8 characters. S806-TRAP determines whether the named load module exists. If not, BYPASS will equal one. ICNT contains the total number of characters in the constant pool, which can be user defines, constants, or literals. In this case, it indicates the starting position of the location at which the name of the subroutine will be stored. IL, equal to eight, is the length of the constant being stored by the perform LOAD-CONSTANT. The constant pool is built up as a series of records. CDYN is used for counting each character in the constant pool. When the counting reaches 90, the constant pool data is moved to the output area DATA-C and written out as a TYPE 3 record for further processing by the logic processor. The object statement is built up; A1 is the starting position of the name in the constant pool, A2 is the length of the name, A3 is the type (alpha or numeric), A4 is the set number which is always a one for the constant pool, and the op-code is twenty-four. The object statement is written out and the next statement is processed.

IF-TEST

IPAR is used as a pointer to the location of a parenthesis. IPAR is set to three to indicate the location of the beginning of the left parenthesis in the user source statement.

IF1

A null (no-op) statement is written for later filling, to resolve parenthetical logic. SWT4 will be equal to one in a conditional statement after a left parenthesis is encountered in the a-field, or in a simple conditional statement when there is one a-field before the logical operator. It is used to determine if another word in the a-field is to be decoded.

OR-TEST

XTRUE is a field in the object statement. SAVVE is a hold area. Check for parentheses.

FATAL-ERROR

Further processing cannot continue with reliability. Terminate execution of this overlay.

FRANK-D

Return to supervisor.

SUB Q-HK

WRITEV generates KEEP object statements if any are needed. Between subquery, housekeeping is accomplished. SKIP-SWT is reset for the next subquery. If LPAR is equal to zero, parentheses balance.

SH1

Create OP20 and OP21 statements; these are break, or separator statements. Check for summary file requirement (SUMMARY-SWT equals one). Do additional between-subquery initialization and housekeeping.

C-POOL

The size of the constant pool is limited to 9K. If exceeded, write an error message and set an error flag in the error vector for that query number.

CP1

Write the final segment of the constant pool. Do more reinitialization and return.

WRITE-4

Close work files.

QUERY-CHK

If there is no card sequence number, exit. Otherwise, set the error vector to indicate an error in that query, process an error message, and go do the next card.

C-SELECT

The SELECT statement is a secondary conditional statement. LM-SWT is set to one when sort flag or secondary logic is encountered. This switch is checked in WRITE-STATE. OP equals fourteen and LMODE equals eight are part of the select object statement. Get next word; it must be an IF. DECODE-A extracts the a-field and supplier description to object statement fields. SW2 is used to verify that the periodic data was used in a SELECT or FLAG statement for secondary logic processing. A4, greater than one, indicates a periodic field. Get next word; it must be a form of HIT.

CCS1

Get next word; it must be a constant. EXTRACT-NUM derives a constant from CARDW and indicates its value in N. BYPASS indicates an error. N in this case represents a number of subsets, which normally should not exceed 999 according to file structuring limitations. Place the value in the object statement and write it out.

CCS2

Process an error message without card column indicators. Go to next statement.

CCS3

Process an error message with card column indicators. Go to next statement.

C-KEEP

Identify statement type. Get next word, must have one.

CK1

The second word of the source statement must be delimited by a space. If not, write an error message.

CK2

SWT4 is turned on to indicate that the data name used in the KEEP statement must be a periodic or variable name. DECODE-A describes the a-field in the object statement. KEPS is used as an index when moving a KEEP field name to a hold area. One is added to KEPS for each field name encountered in a source KEEP statement. A maximum of fifty KEEP statements can be used in a subquery. Add the name to the list of KEEPs. Find out if there is another name. If so, go back; if not, go to next statement. Since they are not written out here, they must be the last statements in a subquery. Otherwise, the generated object statement will be lost.

CK3

If there are fifty-one KEEP statements, write an error. If more then fifty-one, the error has already been written, so first skip to next statement.

C-SUMMARY

CARD-CNT indicates the number of retrieval statements found in each subquery. It is checked to verify that a summary card, if used, is the card immediately following a subquery card.

C-SUMMARY-1

The constant pool record is written out to clear the area. The summary name must be in a separate record so it may be sorted with its proper key value. The 9's in FILE-S will cause it to sort after all other file names. The object statement is built and written, and the file name is restored.

C-SUMMARY-2

Get the next word. "FILE" is optional. The word should contain the name of the summary file to be created. The name must be between five and eight characters in length, and is treated as eight, padded with trailing blanks if necessary. The name is stored in the constant pool. SUMMARY-SWT indicates a summary file has been called for.

C-SUMMARY-3

Process an error message without card column indication. Go to next statement.

C-SUMMARY-4

Process an error message with card column indication. Go to next statement.

C-STOP

Get next word. Check for valid option. If OK, continue; otherwise, write an error.

C-STOP1

Simple stop statement has only one entry in object statement.

C-STOP2

Write out the object statement - process next card.

C-STOP3

This is a stop with an implied condition (limit).

C-LIMIT

Initialize, get next word. If none (IL equal to zero), process error. Word must be a constant; if not, get next word. The value 240 is the binary equivalent of a BCD 0 on the 360. Numerics have values above binary 239, alpha character equivalents are below 240. If there is a period, replace it with a space.

C-LIMIT2

Check each character in the word until a delimiter (space) is found. If any are not numeric, it is an error. Accummulate the value of the constant in the field A1.

C-LIMIT3

Write the object statement using OP and A1. Go to next card.

LIMIT-W

Write a warning message, do not set error vector. Go to next statement.

LIMIT-ERR

Write a error message that sets error vector.

C-SUBQUERY

This is a significant break point. Housekeeping is reinitializing values for a new file. Get the next word. There must be a file name specified and it must be between five and eight characters in length. If it is the same as the last one (FNAME equals CARDW5), skip remaining reinitialization. The fifth character of a MIDMS file name must be an "A". Change the A to a T and call the library to get LR-1 for this file. Obtain the lengths of LR-2 and LR-3. Load in LR-2 and modify the set numbers (FIX-F). Load LR-3, restore the file name and save it.

S1S

Initialize set-decode processing.

S2S

This is a loop executed for each set in the records format. Get a table entry, move it to a work area. Move the set-control position to A1 and the set length to A2. Compute a positive set number in A4 from the set-ID. The fixed set will be a one, and each subsequent periodic set will be incremented by one. Therefore, in later processing, SET equal to one will determine a fixed set, and SET greater than one determines periodic sets. A3 equal to one indicates a fixed set, for all others A3 equals two. When done, go to next statement.

S3S

Write the object statement with OP, A1, A2, A3, A4. Go back to next table entry.

S4S

Write an error message with column indication.

S5S

Write an error message without column indication.

S6S

Write an error message, set SKIP-SWT to indicate an error in processing of the current subquery.

C-MERGE

Sort and merge compilation is similar, except for the OP. Merge has OP equals seventeen.

C-SORT

Initialize object statement with OP equal to sixteen.

CS0

TYPE-ST is an indicator in the linkage section. It is used by GEN0 to determine the proper order of execution of load modules. The LM-SWT switch is checked in the WRITE-STATE section if periodic data is specified in source statements. It is used to indicate when the user has submitted sort flag parameters and/or secondary logic. Its purpose is to cause verification that primary logic is not mixed with periodic secondary logic. SORT-SWT indicates that a sort will be necessary. LMODE greater than eight indicates a sort.

CS1

Get the next word. LMODE equal to eight is ascending sort. LMODE equal to nine is descending sort. SWT4 equal to one indicates that DECODE-A is to extract a word before descending. If not set, the word is already available to DECODE-A. In this case, the switch is set and DECODE-A will build the A fields of the object statement from the data field specified in the next word.

CS2

Get the next word. The word "DESCENDING", if used, must precede the field name. If no more words, go to next card.

CS2A

Check first character for alpha or numeric. If alpha, skip ahead. If numeric, the word should be a partial notation. Extract the partial. K is used to verify that K is not greater than a user define name or FFT name. IW is the offset of the first half of the partial and IL is the length of the partial. The base A1 is incremented by the offset and the new length is placed in A2. Get the next word. If no word, process an error.

CS3

Check for the optional word "FLAGGED". If not there, skip ahead. The SFLAG-SWT indicates the periodic set involved in the SORT FLAGGED statement. It is initialized at zero and will contain the set referenced by a previous sort flag statement, if there was one. If it hasn't been used before, put the set number in SFLAG-SWT. If a different set is referenced, process an error.

CS3C

LMODE equal to eleven or twelve indicates the flagged option in the object statement. The set must be periodic, and therefore A4 will be greater than one.

GS3A

Get the next word. If none, go to next statement. NOTE*** The word "ALL" is not a valid part of a sort statement. It is meaningless and if used, the results are unpredictable. It should be removed. If the word is FLAGGED, go back and process. If the word is IN, continue.

CS3B

Process an error message.

CS4

Again, the word "ALL" should be removed, it is not a legal option. The word IN is mandatory.

CS5

Get next word. The word following IN should be a partial, representing positions in the sort key. IW is base zero; the positions must fall within the 85-character sort key. IN represents the second position of the partial. It must be to the right of the first value. The starting position IW is moved to B1, the length IL is moved to B2. The LMODE test should be changed - the legal values are nine, ten, eleven, and twelve. K, the ending position, should not be outside the sort key. If it is, issue a warning.

CS6

The lengths of the a-field and the position in the sort key must correspond, no truncation or padding is permitted. This has been done at user's request.

CS50

Process an error with card column indication. Go to next card.

CS51

Process an error without card column indication. Go to next card.

C-SEARCH

Get next word. If none, search mode is the default, SEARCH, and SMODE equal zero. Otherwise, move one, two, or three to SMODE as appropriate. If not a legal word, issue an error message. Go to next statement. No object statement is created from this statement. Before the processing of each user's source statement, SMODE is moved to LMODE.

C-FLAG

Initialize PSS-SWT. Decode the a-field. A4, equal to one, indicates fixed data, an error; FLAG may only be used with periodic fields. SWT2 is initialized to zero at the beginning of the DECODE-A section. A one is moved to SWT2 if the field is periodic. If a PSS control field is encountered, activate PSS-SWT. Get next word; there must be another one and it must be one of the four literals identified in this paragraph. Otherwise process an error.

F1C

LMODE equals one if a control field is used.

LMODE equals three if not. This is the flag LOW option.

F2C

Flag HIGH option. LMODE equals two if a control field is used. LMODE equal to four is not.

F3C

Get the next word. If none, process an error. It must be a constant and its value is returned in N. N must be a positive integer. If OK, move to B1 and write the object statement.

F5C

Process error with card-column indication. Go to next card.

F6C

Process error without card-column indication. Go to next card.

C-1F

Initialize SWT4 to indicate EXTRACT-WORD will not be performed in DECODE-A.

CF1

Get the next word. If no parenthesis, activate SWT4 and skip ahead. IPAR equal to one indicates a left parenthesis was found. Write an object NOP statement to indicate parenthesis.

CF3

Initialize OP to zero. Conditional OP's are derived from an accummulation of values, whereas most other OP's are finite and are moved. LPAR greater than nine is an overflow condition. PE-SWT, not equal to zero, indicates a parenthesis error has already been written. If this is the first parenthesis inbalance condition, write an error message and go to next statement.

CF3A

Beginning of intra-set logic statement processing.

CF4

Resolve the a-field. LMODE, equal to six or seven, indicates secondary logic.

CF5

A4, equal to one, indicates fixed set or define - these are not allowed.

CF6

Build the OP and b-fields portions of the object statement and write it out.

C--RECEIVE

Set up object statement OP code. Since statement is not a conditional, XTRUE equals one. Get next word, if none, process an error.

CR1

SWT4, equal to one, no-ops the extract word section in DECODE-A. If parentheses are found, process them. Generate a-field object vectors.

CR1A

Get the next word, check for and process parentheses. If no word, statement is complete. If anything is found, it should be a field partial. If so, adjust a-field addresses.

CR3

Get next statement.

C-ANY

LMODE, equal to four, indicates ANY type statement. Get next record; if none, process an error.

CAN1

Distinguish between primary and secondary logic statement.

CAN2

LMODE equals six for ANY HIT type statement.

LM-SWT equals one to indicate secondary statement has been encountered. Once LMODE is greater than five, all subsequent statements must have an LMODE greater than five or it is an error.

C-ALL

LMODE equals five for ALL type statement. Get the next word; if none, process an error.

CAL1

Distinguish between primary and secondary logic statements.

CAL2

LMODE, equal to seven, indicates ALL HIT type statement.

LM-SWT indicates initiation of secondary logic processing.

C-DEFINE

Initialize, assume alpha (DATA-TYPE equals four), unless overridden.

C1

Get next word, must have eight or less characters in the name.

C3

K, greater than DKD, indicates items not found in define name table. Loop through table to determine whether this name has previously been defined.

C4

Issue an advisory warning that this named field is being changed by a modify statement. If data-types do not match, issue an error message.

C4A

Set up error message.

C5

IL, equal to or less than five, could be FFT name.

C6

If the name exists, write an error message.

Loop through until done.

C7

If being modified, disregard further processing of this card, since the remainder is changed by the MODIFY card. Otherwise, build new entry for defined name table.

C8

Get next word. An asterisk indicates a convert routine so set switch and save the name of the convert routine.

C8B

LPARN indicates length parameter has been specified.

C9

Check matching parenthesis, adjust starting point of value.

C9A

Decode value and check for limits. If OK insert length and starting position into define table. Get next word; if none, done.

C10

Check length of literal. If convert routine is indicated, see if it exists. If it does, set up and execute it. EXIT-FLAG equal to one is OK, EXIT-FLAG equal to two is an error. Any other code is illegal.

C10A

Transfer results of conversion and check length.

C11

SWT3 means length was specified, compare with actual.

C13

Decode numeric constant.

C15

Check size of numeric field, insert leading zeros.

C16

Load defined value into constant pool, add spaces if necessary.

C17

Insert line into define table, check rest of card - should be nothing.

C25

Error message with column indicators.

C26

Error message without column indicators.

C-LIST through XDECODE-A

Generate output define statements to accommodate the LIST statement in the shorthand language. These pseudo source statements will be compiled by GEN3A.

WRITE-ERR1 SECTION

Bypass indicates an error was encountered. TYPE-S, equal to one, will print error message as a source statement.

XW1

Write out error message with record (card) sequence number one greater than previous record so message will print right after source statement. QUERY-SWT (100), equal to one, indicates ABORT. Change to two so ABORT error message will be printed in GEN6.

W-E1

Propagate 1's (error codes) through error-vector to cancel execution of all queries.

WRITE-ERR SECTION

BYPASS, equal to one, indicates error upon return from perform. Set up card column indication to identify word in source statement that caused error to be identified, move message, and write out error. Check for ABORT condition.

W-E1

Propagate error codes for ABORT option.

READ-OP SECTION

Read a continuation card.

R1

Read a record from the input work file (source statement card images).

R5

Copy to output work file.

R6

TYPE-IN, equal to one, indicates retrieval source statement. Continuation card header information must match previous source card. Check for possible error conditions.

R2

Process an error.

R3

Type 4 in shorthand. Trap errors.

R4

Continuation card must start in card column 4.

WRITE-STATE SECTION

Check for KEEP or SET DECODE OP's.

WS0

Check for numeric a-field.

WS00

Check for numeric b-field.

WS000

OP greater than twelve, indicates directive statements.

OP equal to six or twelve, indicates special operator.

A2 equal to B2 is a match of field lengths.

A2 must equal B2 for alpha compares.

WS1

Determine whether data-types match.

WS2

Check for primary conditionals following secondary conditionals.

WS2A

Determine whether periodic fields are referenced, which will cause GEN4 to be called rather than GEN4A.

WS2B

Indicate parentheses, check for previous errors, and check total number of generated object statements.

WS2C

Write the object record.

WS3

Provide space for the name and address of a special oeprator.

WS3A

Check whether sort statements follow conditional statements.

WS3C

Check whether sort flagged statements follow all other statements.

WS4

Write an error message.

WS5

Numeric a-field length cannot exceed fifteen digits.

WS6

Numeric b-field length cannot exceed fifteen digits.

WRITE-NOP SECTION

Write a no-op statement.

WN1

Determine level of parentheses.

WN2

Adjust image to clear parentheses.

FLD-PART SECTION

Check size of partial notation character string for excessive length.

F1F

Extract first number of partial set.

F2F

Check hyphen separator and set up for second number.

F3F

Extract second number.

F4F

Check for valid delimiter.

F4AF

Check length of field defined by partial limits for validity.

F5F

Process an error message.

F6F

Reset alpha/numeric check field.

DECODE-OP SECTION

D1D

Get next word, check for partial from the a-field. If one exists, decode and adjust a-field starting position and length.

D2D

Get next word. Check for noise word or missing operator.

D3D

An asterisk indicates a special operator. If none, skip ahead. Otherwise, check name and determine whether that special operator is on the library. If it is, save the name and increment the OP code.

D6

A negation of a compare operator increments the OP code by six.

D7

Check for legal operators and adjust the OP code accordingly.

T-SCAN

Set up for text-scan (CONTAINS) operator.

T-SCAN1

Get next word, must be one for valid statement.

D8

Negation of a SATISFY is illegal. A conditional OP may not exceed twelve.

D8A

LMODE less than four is a standard conditional, no ANYs or ALLs.

D9

Get next word; if a noise word, get another one.

DECODE-A SECTION

Initialize. Determine whether word is available (SWT4 not equal to zero).

A1A

Get a word.

A2A

Check validity, must be alpha. IL greater than five means word cannot be from FFT.

A3A

A defined name cannot exceed eight characters.

A4A

Determine whether a-name is in the FFT. The KEEP and variable sets are special cases.

A5A

If not in FFT, Check defined name table.

A6

Loop through define table. If not found here, it is an error.

A7

Move a-field, defined name descriptors to the object statement.

A14-1

For the variable set, describe the location of the variable set control field.

DECODE-B SECTION

Check for legitimate word.

B0

Check for OP's with mandatory multiple b-fields.

B1B

Check for and process parentheses, if found.

B2B

Check for textscan.

B3B

If a convert routine is called for, execute it here. Check for valid return code and process returned error message, if any.

B4B

Recover from successful convert routine call.

B4C

Type must be numeric.

B4D

Describe a defined field as the b-field.

B4A

Check length.

B4T

Extract a constant, set type as numeric.

B5

Length of b-field cannot exceed length of a-field.

B6

Describe defined b-field. Check for SATISFIES operator.

B7

Check type, 2 or 5 means numeric.

B8

If numeric, insert leading zeros.

B9

If alpha, fill with trailing spaces.

B10

Trap out field names as b-fields on BETWEEN and SATISFY operators.

B10A

Write an error.

B11

Check for legal conditionals.

B12

Initialize counter.

B13

Search FFT for name.

B14

Special oeprator may not have an FFT b-field.

B15

Check for data after a right parenthesis.

B16

Check for parentheses and partial notation.

B16A

Adjust parenthetical counters.

B16B

Get next word, check for required parentheses.

B18

Check defined name table for b-field name.

B19

When found, load address into object statement.

B20

Any more data after a parenthesis?

B20A

Process a parenthesis.

B20B

If not a special case, write out the object statement.

B21

Adjust vector pointer (XTRUE) to reflect parentheses.

B22

OP4-SWT is initialized at zero at the beginning of DECODE-B section. When decoding the BETWEEN statement, OP4-SWT is incremented by one for each range parameter and the corrector "AND" for a total value of three. The maximum value of OP4-SWT is, therefore, three and values other than zero and three are considered invalid.

B23

OP5-SWT is somewhat similar to the OP4-SWT, except that OP5 is concerned with extracting and accounting for the proper number of fields for the SATISFIES operator.

B24

Check number of b-fields.

B25

Value indicates state of progress through the required "satisfy" b-fields.

B26

The second b-field.

B27

The third b-field, also the number of remaining b-fields.

B28

The first of the last set of SATISFY b-fields; that is, the first search value.

B29

Successive search values.

B31

Starting point for SATISFY processing. Get the first b-field, the number of required bits.

B32

Check progression through the BETWEEN operator b-fields.

B33

The second b-field.

B34

The third b-field.

B35

Is parenthetical logic involved?

B36

Check type of a-field to determine type of b-field.

B50A-B51

Process error messages.

USER-ERROR SECTION

Extract and print a user-supplied error message from a convert routine.

EXTRACT-WORD SECTION

Initialize.

W1

IC is card column. Check for continuation card; if found, start scan in column 4.

W2

Check delimiters; if space, check next column.

W2A

Got something - find out what.

W3

Beginning of word.

W4

Save as W1.

W5

COM-SWT equal to A, indicates comma was found after a data name.

W6

Increment counters and loop back.

W6A

Check for required delimiter.

W6C

Write an error.

W7

Check delimiters.

W8

Fill with spaces.

W9

Initialize the query number checker switch. If RIA equals one, a new query number has been encountered.

HLD-REC SECTION

This is a temporary statement hold area used to process shorthand language statements requiring the generation of paragraph statements (HEADERS, etc.).

LEAD-ZERO SECTION

Inserts leading zeros into a numeric field to pad its length.

LOAD-CONSTANT SECTION

Moves a constant value to the constant pool record area and adjusts the pointers to the last used area in the constant pool.

TRAIL-SPACE SECTION

Adds training spaces to an alpha field to pad to the desired length. Sometimes executed on a numeric field after a LEAD-ZERO is performed, effectively NOPing the trail space execution.

WRITE-CDYN SECTION

Writes out a constant pool record of ninety useful characters. Also inserts appropriate key information into the record prior to writing.

EXTRACT-NUM SECTION

Extracts the value of a numeric field and returns that value in the field N.

RIGHT-PARN1 SECTION

Keeps track of how many parentheses have been encountered and the relative degree of imbalance between left and right parentheses.

FIX-F SECTION

Recompute values in the FFT Logical Record 2. Eliminate negatives and signed zeros.

TYPE-TEST SECTION

Check for a legal match of data types.

S806-TRAP SECTION

Execute a BLDL macro to determine whether a named load module exists on the system or not. Return code of zero is OK, others are errors.

KEEPP SECTION

Keep track of the number of KEEP statements associated with a SORT FLAGGED statement. Variable sets must be isolated.

WRITEV SECTION

Write an OP23 object statement for each valid KEEP encountered.

END-STEP SECTION

Wrap up an object statement set for a core-load (file).

DUP-CHK SECTION

Check for duplicate DEFINE'd names.

(5) GEN2X.

GEN2X is a subset of GEN2. GEN2X is GEN2 minus the capabilities that are not included in the current level of Non-Sequential Access (NSAX). The logical flow and narrative for GEN2 are the same for GEN2X except for some additional final processing.

GEN2X has no secondary processing since it does not flag subsets. The basic operators are LESS THAN, EQUAL TO, GREATER THAN, and their negations.

The additional final processing for GEN2X sorts the resultant output vecotrs to group all the vectors for a particular list. A check is also made on which sets have been used so that only the fixed set and/or one other set are used in the retrieval.

(6) GEN4X1.

PROCEDURE DIVISION

Initialize.

APAR

Read input file from GEN2X. If the record is a source record, type equal to one, go to source processing. Check to see if an error was encountered. If an error was encountered, go to APAR. If input record is a vector record, type equal to two, go to vector processing. If the record is a constant record, go to constant processing.

SOURCE-OUT

Write source record.

ASW

This is a switch paragraph. Its purpose is to initialize the reference list name. After the reference list name, RLSTNME, is initialized, the switch is altered to point to ACMP.

ACMP

Check for a list name change.

ACMPA

Check to see if there is room in the vector array.

BPAR

Move vector into vector array. Set selection element.

CNST-PROC

Check to see if there is room in the constant pool. Move constant string to constant pool.

CPAR

Constant pool overflow, initialize error message.

CCPAR

Vector array overflow, initialize error message.

CCCPAR

Close key file. Set switches to indicate error in NSAX processing. Check for ABORT processing.

LOP-3

Set Query Vector Error Elements to one.

LOP-4

Write error message.

PUT-OUT

Write source output file.

VECT-EOF

Set end-of-file flag, EOF-FLG, to indicate end of file. Close source input. If an error was encountered, go to final processing.

DPAR

Initialize list file DCB.
Update reference list name.
Open list file.

EPAR

Check to see if there are any active vectors.
Read list file.
Set addresses for any compare procedures.

GPAR

Get an active vector. Initialize sort key record.
Go to correct operator (op field of vector).

ERR

Invalid operation code.
Close list file.
Initialize error message.

HPAR

Check to determine if all active vectors have been applied to the list record.

FPAR

Close list file.
Check to see if there are nay more list files to process.
Close key file.

FINISH

Initialize selection vector.
Close output source file.

FINISHX

Return to GEN0.

LS thru LS-E

LESS Operator: Modified binary search until the number of search entries is less than sixteen. When there are less than sixteen list entries to be searched, a sequential search is used. Tests are made to de-activate the vector (i.e., when a vector is de-activated it is eliminated. List elements are no longer less than the test value). When a sequence of entries is known to satisfy the operator, the keys of those entries are written on the key file without comparing every entry's value.

EQ thru EQ-CC

EQUAL Operator: Modified binary search until the number of search entries is less than sixteen. When there are less than sixteen list entries to be searched, a sequential search is used. Tests are made to de-activate the vector (i.e., test value greater than list elements). When a vector is de-activated, it is eliminated.

GR thru GR-C

GREATER Operator: Modified binary search until the number of search entries is less than sixteen. When there are less than sixteen list entries to be searched, a sequential search is used. Tests are made to bypass this vector when the last list entry is not greater than the operator value. When a sequence of entries is known to satisfy the operator, the keys of those entries are written on the key file without comparing every entry's value.

NLS thru NLS-C

NOT LESS Operator: Modified binary search until the number of search entries is less than sixteen. When there are less than sixteen list entries to be searched, a sequential search is used. Tests are made to bypass this vector when the last list value is less than the search argument. When a sequence of entries is known to satisfy the operator, the keys of those entries are written on the key file without comparing every entry's value.

NEQ thru NEQ-A

NOT EQUAL Operator: Comparisons are made to determine not equal keys. When a sequence of entries is known to satisfy the operator, the keys of those entries are written on the key file without comparing every entry's value.

NGR thru NGR-D

NOT GREATER Operator: Modified binary search until the number of search entries is less than sixteen. When there are less than sixteen list entries to be searched, a sequential search is used. Tests are made to de-activate the vector (i.e., list elements are greater than search argument). When a vector is de-activated, it is eliminated. When a sequence of entries is known to satisfy the operator, the keys of these entries are written on the key file without comparing every entry's value.

COMPARE

Execute subroutine to compare list value and vector argument.

WRTC

Move key to output area. Write key on key file.

DE-ACT-VECT

Move 'X' to corresponding inactive element of the vector. Subtract one from the number of active vectors.

WRTNC

Move key to output area. Write key on key file.

FLSHD SECTION thru FLSHDX

This section extracts keys from a series of entries which are known to satisfy the current operator. FL is initialized by the calling routine. The process starts with the FL entry and terminates with the ST entry (usually the first list entry).

FLSHU SECTION thru FLSHUX

This section extracts keys from a series of entries which are known to satisfy the current operator. FL is initialized by the calling routine. The process starts with FL and terminates with the EN entry (usually the last list entry).

FLSHULS SECTION thru FLSHULSX

This section sequentially searches a series of list entries. Each list entry is tested to see if it satisfies its operator. As soon as the operator is not satisfied, the process ends and the procedure exits. The sequential search begins at the FL entry and continues upward until a bad comparison is made. FL is initialized by the calling routine.

FLSHDNLS SECTION thru FLSHDNLSX

This section sequentially searches a series of list entries. Each list entry is tested to see if it satisfies its operator. As soon as the operator is not satisfied, the process ends and the procedure exits. The sequential search begins at the FL entry and continues downward until a bad comparison is made. FL is initialized by the calling routine.

FLSAVEQ SECTION thru FLSHUEQX

This section sequentially searches a series of list entries. Each list entry is tested to see if it satisfies its operator. As soon as the operator is not satisfied, the process ends and the procedure exits. The sequential search begins at the FL entry and continues upward until a bad comparison is made. FL is initialized by the calling routine.

FLSHDEQ SECTION thru FLSHDEQX

This section sequentially searches a series of list entries. Each list entry is tested to see if it satisfies its operator. As soon as the operator is not satisfied, the process ends and the procedure exits. The sequential search begins at the FL entry and continues downward until a bad comparison is made. FL is initialized by the calling routine.

FLSHDER SECTION thru FLSHDERX

This section sequentially searches a series of list entries. Each list entry is tested to see if it satisfies its operator. As soon as the operator is not satisfied, the process ends and the procedure exits. The sequential search begins at the FL entry and continues upward until a bad comparison is made. FL is initialized by the calling routine.

FLSHUNGR SECTION thru FLSHUNERX

This section sequentially searches a series of list entries. Each list entry is tested to see if it satisfies its operator. As soon as the operator is not satisfied, the process ends and the procedure exits. The sequential search begins at the FL entry and continues upward until a bad comparison is made. FL is initialized by the calling routine.

FLSHUNEQ SECTION thru FLSHUNEQX

This section sequentially searches a series of list entries. Each list entry is tested to see if it satisfies its operator. As soon as the operator is not satisfied, the process ends and the procedure exits. The sequential search begins at the FL entry and continues upward until a bad comparison is made. FL is initialized by the calling routine.

FLSHDNEQ SECTION thru FLSHDNEQX

This section sequentially searches a series of list entries. Each list entry is tested to see if it satisfies its operator. As soon as the operator is not satisfied, the process ends and the procedure exits. The sequential search begins at the FL entry and continues downward until a bad comparison is made. FL is initialized by the calling routine.

PROCEDURE DIVISION

Initialize. Sort the Key File. Check to see if Sort was successful. Set switches to indicate an error in NSAX processing.

LOP

Initialize DCB's of source files. Open source files.

LOP1

Write all input source to output source.

LOP2

Write error message. Close source files. Check for ABORT processing.

LOP3

Set Query Vector Error Elements to one.

LOP4
FINALX

Return to GEN0.

INPAR SECTION

Open Key File for input.

INPAR-A

Read Key File. Move Key File record to sort record. Release sort record.

INPAR-B

Close Key File.

INPAR-E

Exit from section procedure.

OUTPAR SECTION

Open Key File for output.

OUTPAR-A

Return sort file record. Write key file from sort file record.

OUTPAR-B

Close key file.

OUTPAR-E

Exit from section procedure.

(8) GEN4X3.

PROCEDURE DIVISION

Initialization.

JPAR

Read key file. Eliminate duplicate records. Check for change in record number (record number is the relative record number of the standard sequential MIDMS record).

KPAR

Check for secondary key changes, go to processing required for that level of key change. Reinitialize reference keys (reference keys resemble previous key record). Initialize selection vector.

KKPAR

Initialize selection vectors. Change reference vector to trigger an automatic change.

KKKPAR

Set an element in both selection vectors. Check to see if a record is to be retrieved.

LPAR

Set an element in selection vector. Check to see if a record is to be retrieved.

MPAR

Move the number which represents the number of records which satisfied the user's request into the NSAX statistics field. Close any opened files.

MMPAR

Return to GENO.

NEPAR

Process error. Set switches to indicate fatal error in NSAX processing.

LOP

Initialize source files. Open source files.

LOP1

Write all input source to output source.

LOP2

Write error message. Close source files. Check for ABORT processing.

LOP-3

Set Query Vector Error Elements to one.

LOP-4

Go to return paragraph.

NPAR

Add one to the number which indicates the number of records which satisfy the user's request.

NSW

This paragraph is a switch which controls whether a summary file will be written and if it was opened. Initially, it is set so that a summary file will not be written. If a summary file is to be written, it is altered to point to the open paragraph. After the open paragraph is processed, the switch is altered to point to summary file processing.

NSWA

Initialize DCB's of the Pointer and Packed NSAX files. Open files.

IPAR

Read the Pointer file. Check to see if there are any valid entries.

NNPAR

Check to see if the record to be retrieved is on this particular pointer record. Compute subscript of entry.

OPAR

Check to see if the proper entry is available.

OOPAR

Initialize summary record length. Check to see if it is necessary to read a packed record. Initialize pack file record-ID.

RPAR

Read a packed file record (random file).

RRPAR

Check to see if the standard record is spanned (over several packed records). Move standard record element from the packed record to the summary record.

PPAR

Spanned record processing: move standard record element from the packed record to the summary record, initialize parameters to read and process another packed record.

SPAR

Write summary file record.

NRPAR

Read pointer file. Check for valid entries.

(9) GEN3A.

PROCEDURE DIVISION

This is the entry point for the module GEN3A, the shorthand statement compiler. Set up the appropriate DD names for the work files S1GEF1 and S1GEF2.

HOUSE-KEEP

Open the work files. Initialize the switches. Set the query number to one, the subquery number to one, and create the imbedded define names, sort key and subquery.

NEW-STATE

The missing switch is tested to determine if a new input record is to be read. If an invalid continuation statement or a required continuation source statement is not found in the READ OP Section, this switch is turned on.

NEW-STATE3

Check for generated source DEFINE statements.

NS4

Bypass all except TYPE 4 statements.

NEW-STATE1

Check for a change in a query number and bypass comment statements.

NS1B

Initialize skip switch. If logical record 1, 2, or 3 is not found while processing a file source statement, this switch is turned on. All subsequent statements which have the same query number as the file statement in which this error occurred are simply written to the output file, FILEOUT, with no processing performed.

NS1C

The item DKSC is used as a counter for the number of define names, internal and external, to be summed and/or counted.

END-ROUT

CLAS-SWT equal to zero indicates that a classification has not been indicated for this report. Classification is required entry. Switch 99 equal to one indicates that headers are present.

BEG-AGAIN

OP88 is a generated HEADER statement. OP99 is a generated TRAILER statement.

CREATE-LINE

Write out a statement.

SOP-6

SMODE of six indicates final output logic mode.

HDRS

Switch 99 equal to one indicates headers and OP11 is a MOVE statement.

WRT-REC

Switch A equal to a two is a period.

PRT-77

Write out a generated vector.

PAGE INTENTIONALLY LEFT BLANK PER CPT. RUSELLL OF DEFENSE INTELLIGENCE AGENCY AND MADELINE CRUMBACKER OF DDC.

PRT-7

Write the previous record. Check for a period and write the current record.

XXXXO

Write out a generated vector.

TRLS

Save the current object vector. Switch 99 with the value other than zero indicates that there are trailers to be generated.

TRL-1

Generate the OP34 and OP35 separator statements. An SMODE of two indicates a TRAILERS paragraph.

TRL-2

Save the generated object statement. Generate a MOVE statement.

SKIP-OUT

If no trailers are indicated generate a SKIP statement OP19. Write the record out and generate a PRINT statement.

HDR-SECT

Generate an OP34 and OP35 separator statement. An SMODE equal to one indicates headers.

MISSING-CLASS

Generate an error message.

XEND-ROUT

Determine whether the number of generated object statements for this report exceed the maximum allowable. If so, write an error message.

NS1CC

Check for overflow of the constant pool area.

NS1D

Write a constant pool record. Determine whether the default for PAGESIZE has been overridden.

NS1X

Generate an OP34 and OP35 object statement.

NS1Z

Write an error message.

NEW-REPORT

Initialize switches. Set up default values. Generate the imbedded names PAGENO, COUNT, and SUM.

INITIAL-REC

Save the header information from the record.

PPP

Initialize the statement area.

NS1

Get the next word. If none, go back for another.

FIRST-WORD

This is a "GO TO depending on" the value of the first word in a source statement. Notice that many of these statements are allowable in the standard longhand output language and are not available in the shorthand language. These include the MOVE statement, the variations of the MOVE statement, the arithmetic statements, and the LINE and BEGIN paragraph identifiers.

C-LABEL

Process a LABEL statement. A valid FFT name must be included as part of the statement. If not, process an error.

LAB-1

Mask into the FFT name the indicator LL to indicate that a label is being created for that FFT name. Continue processing as though this were a normal define name.

LO2

This is a return point from the routine GEN3A.

C-PSIZE

This paragraph is executed when the default value for PAGESIZE is overridden by a user's statement.

XPG1

Process an error message.

XP-WARN

Process a warning message.

C-FILE

This paragraph initiates the processing for a FILE Section. The OP34 and OP35 separator statements are generated and written out.

SOS

Perform syntax checking of the file name. It must not be greater than eight nor less than five characters. The fifth character must be an A.

SOSA

There may not be multiple FILE Sections with the same file name in the same report.

SOSB

Load in the FFT for this file name. Logical record 1 describes the lengths of logical records 2 and 3. Once this has been determined, load in logical record 2 and logical record 3 in their corresponding work areas.

S1S

Generate a SET-DECODE object statement.

S2S

Recompute the values in the logical record so that all fixed and periodic set numbers are positive integers.

S3S

Write an object statement.

S4S

Process an error.

S6S

The fatal error has been encountered. Further compilation is meaningless. Skip to the next report and put out a corresponding error message.

S8S

Recompute the TYPE 4 entries in the FFT logical record 2.

C-FINAL

Generate the OP34 and OP35 separator statements. Generate an EJECT statement.

C-LOAD

Generate an OP38 object statement which will cause the named subroutine to be loaded into core. This is intended to be used only with the MFT operating system and not with the MVT operating system.

C-FLY

Check for the end of the fly sheet and for proper format of the fly statements.

FLY-1

Write out the fly sheet statements as TYPE 8 records.

FLY-2

The line number specified on the fly sheet's source card must be numeric. If not, process an error message.

C-IF

Initialize processing of the conditional statement.

CF1X

Generate an object statement for a conditional source statement.

C-AND

The AND is synonymous with the IF.

C-OR

Other than the value of XTRUE, the rest of the object statement for an OR statement is identical with the AND statement.

C-OMIT

The OMIT statement may only appear in a FILE Section.

C-STOP

The STOP statement may only appear in a FILE Section.

C-RESET

Periodic set number must be specified and, of course, it may not exceed sixty.

C-SET

The second word in this statement is a number that denotes the periodic set to which this statement applies. The third word indicates which option of this statement is being chosen. An appropriate OP code is moved into the object statement.

CST1

Continue decoding the words on the SET statement. Check for set selection options with respect to flagged and unflagged statements.

CST2

The finally computed OP value must be between twenty-two and twenty-five.

CST2A

The 1000 indicates that all subsets are of interest.

C-SOURCE

A source direct generation is indicated by the value eight in the error vector for that particular report.

C-DEFINE

Processing for the DEFINE statements is identical with the processing for the DEFINE statements in the standard output compiler.

C-LIST

Initialize the values for list process generation. Check for a period and adjust values if one is encountered.

LST2

Decode the a-field and check for partial notation.

LST3

Check the limits of the partial notation.

LST4

Extract the value of the partial and determine whether the type is alpha or numeric.

LST5

Initialize the area to zero.

LST5A

Indicate that this LIST statement will be executed within a BEGINS paragraph.

LST6

If periodic fields are specified, perform the necessary generation.

WARN-MESSAGE

Process a warning message.

FATAL-ERROR

An error has been encountered from which there is no recovery. Close all files and stop.

C-LSIZE

Establish a default for line size based on the type device specified and if a specific size is mentioned, use that. Check to determine whether it is within the appropriate limits of twenty and one hundred thirty-two.

C-CLASS

Process a CLASS statement. Move the value into an appropriate area.

CENT-CLASS

Center the classification on the output page.

CL-1

The value 10032 is the first position in the output area. The center of the output line is computed from that starting address.

MOVE-PAGE

Generate a MOVE statement to handle the automatic generation of page numbers on the output page.

ADD-PAGENO

Generate a defined area called PAGENO for saving the current page number.

EMOV-PAGENO

Generate an edit MOVE statement so the page number will be printed with zero suppression.

CALL-SYSDATE

Get the current date from the operating system.

MOV-SDATE

Generate a MOVE statement to move this system date to the output area.

SPACE-26

Generate a SPACE statement to skip two lines.

PRT-TOP-LINE7

This will generate a PRINT statement.

HED-HN

Compute the size and location of a header line.

CNTER-HN

Center the value of the header information.

TRL-TN

Define a trailer line of which there may not be more than three.

C-DISPLAY

This statement may only appear in the FINAL OUTPUT Section.

C-DISP

The following series of DP paragraphs generate the appropriate statements to display specified literals or fields. User provided spacing is indicated by SP parameter.

C-COUNT

The COUNT and SUM operators are processed in the paragraphs starting with a CS. This series of paragraphs causes appropriate statements to be generated to cause either a COUNT or SUM of an appropriate field to be maintained while the file records are processed.

CLA-HED SECTION

Generate a MOVE statement for the classification header line.

MOV-CH SECTION

Load a value into the constant pool and generate the appropriate object statement indicators to identify where that value has been loaded.

WRITE-ERROR SECTION

Generate an error message with card column indication.

WRITE-ERR1 SECTION

Generate an error message without card column indication.

DECODE-OP SECTION

This section is virtually identical with the same named section in the GEN3 output compiler.

FLD-PART SECTION

This section is virtually identical with the same named section in the GEN3 output compiler.

DECODE-A SECTION

This section is virtually identical with the same named section in the GEN3 output compiler.

DECODE-B SECTION

This section is virtually identical with the same named section in the GEN3 output compiler.

READ-OP SECTION

This section is virtually identical with the same named section in the GEN3 output compiler.

EXTRACT-WORD SECTION

This section is virtually identical with the same named section in the GEN3 output compiler.

LEAD-ZERO SECTION

This section is virtually identical with the same named section in the GEN3 output compiler.

LOAD-CONSTANT SECTION

This section is virtually identical with the same named section in the GEN3 output compiler.

TRAIL-SPACE SECTION

This section is virtually identical with the same named section in the GEN3 output compiler.

WRITE-CDYN SECTION

This section is virtually identical with the same named section in the GEN3 output compiler.

EXTRACT-NUM SECTION

This section is virtually identical with the same named section in the GEN3 output compiler.

FIX-F SECTION

This section is virtually identical with the same named section in the GEN3 output compiler.

TYPE-TEST SECTION

This section is virtually identical with the same named section in the GEN3 output compiler.

USER-ERROR SECTION

This section is virtually identical with the same named section in the GEN3 output compiler.

S806-TRAP SECTION

This section is virtually identical with the same named section in the GEN3 output compiler.

DUP-CHK SECTION

This section is virtually identical with the same named section in the GEN3 output compiler.

(10) GEN3.

PROCEDURE DIVISION

This is the entry point for the program GEN3, the output compiler. The work file names are moved into the appropriate DCBs so the files can be opened.

HOUSE-KEEP

The work files are opened. Switches are initialized. Sort key is identified as a reserved word. It is defined with a length of eighty-five and a type of alphanumeric. Subquery is defined as a reserved word. It has a length of two and is numeric.

NEW-STATE

The subscript switches are initialized. A record is read from the input work file.

NEW-STATE3

The input record is compied onto the output work file and written out. Only output source statements are used by the compiler. Any other type (i.e,, a type that will not equal five) is copied and bypassed.

NEW-STATE1

Determine if this statement is from the same report or another report. Bypass comment cards.

NS1B

If REPORT-SWT is greater than zero, at least one report has been encountered in a MIDMS batch job.

NS1C

CTR6 is the count of the number of generated object statements for report. It may not exceed 1333.

NS1CC

A constant pool cannot exceed 9000 characters. Since the initial point in the constant pool is 20030, 29030 indicates the 9000 increment.

NS1D

This paragraph checks to confirm that there is a program for each perform statement. If the WRITE-SWT is not equal to zero, it indicates that an output file is to be created. If this is so, then there must be a format statement describing the file name or record size.

NS1A

Check the legal values for the page size parameter. Default value is fifty. LINE-KT indicates the number of LINE paragraphs encountered. BEGIN-KT indicates the number of BEGINS paragraphs encountered. These numbers must be equal.

NS1X

This paragraph performs a wrap-up function for a report. It generates a OP34 and OP35 statement and provides page size, record size, and file name for the appropriate output files. ICNT contains the number of characters comprising the constant pool.

NS1Z

Write an error message.

NEW-REPORT

Initialize switches and counters for a new report. The two in DKD represents the two imbedded define names.

INITIAL-REC

If logical record 1, 2, or 3 is not found while processing a file source statement, SKIP-SWT is turned on. All subsequent statements which have the same query number as the file statement in which this error occurred are simply written to the output file with no processing performed.

PPP

Initialize the header portion of the output work file records.

NS1

Get a word from the last output source statement read in.

FIRST-WORD

Identify the first word on a source statement. All legal first words, not including continuation cards, are indicated by this list.

LO2

Close the work files and return to the supervisor program.

C-CAT

This paragraph generates an OP39 object statement.

P-SORT

This paragraph generates the object statement for the periodic sort.

C-FLYA

This processes a fly sheet statement. Get the next word. If none, the fly sheet is not numbered. If there is a word, check to determine whether it is numeric. If it is numeric, it must be in a range from one to ninety-nine. This number then constitutes the identification for this fly sheet.

C-FLYB

If that word was alpha, the only legal possibility is the word SHEET. If it was SHEET, go back and get the next word.

C-FLY

A string of fly statements is expected. The TYPE-IN should be five for all these statements. The query statement should not change until the end of the string is encountered. The statements are required to be in a fixed format where the fifth column is blank and the fourth column is either an R or an L.

FLY-1

The first three characters of a fly statement are the line number. They must be numeric. Set up the headers. If everything is OK, write out the record. Go back and get the next fly statement.

FLY-2

If an error is encountered, write out the record and an appropriate error message.

FLY-3

Indicate the end of a fly sheet.

C-FORMAT

Indicate that a format statement has been encountered.

CF0

Get the next word. There must be one.

CF1

That word must indicate either page size, output file, or record size or one of the abbreviations for those three. If not, process an error.

CF2

If page size is equal to zero, print the warning message.

CF3

Get the next word. If none, process an error. This word should be numeric and indicate the proper page size.

CF4

If record size is not equal to zero, it indicates that a DEFOME statement is being overriden by a MODIFY. If so, produce an appropriate warning.

CF5

Get the next word. There must be one and it must be numeric. This number indicates the record size specified. Record size may not be greater than 9999 characters.

CF6

If X file name is not equal to spaces, it means that a previously defined file name is being overridden by a MODIFY statement. If so, produce an appropriate warning message.

CF7

Get the next word. If none, produce an error. This word must not be greater than eight characters in length and must be alphabetic. This word indicates the file name being defined.

CF48

If a format statement was specified by format switch and no specifications were found, produce an error.

CF9

Get the next word.

CF49

Set up an error message.

CF50

Write an error message.

CF51

Write an error message.

C-LOAD

Get the next word. It should be delimited by an asterisk. If not, produce an error. Since this is a name, its length cannot be greater than eight characters. Perform the S806 trap to determine whether or not that routine exists. Load the name of that routine into the constant pool and check for constant pool records space overflow. Generate an OP38 and write it out. This statement should only be encountered with MFT operating system. It is not appropriate for MVT.

C-FILE

The presence of a file statement indicates the conclusion of the previous file section, therefore, generate an OP34 and an OP35.

SOS

Get the next word. There must be one and it must be a file name. Its length cannot be greater than eight characters nor less than five characters. To comply with standard MIDMS specifications, the fifth character must be equal to an A.

SOSA

DKX indicates the number of unique file sections found in a report. There may not be two file sections referencing the same file in the same report.

SOSB

Check the number of file sections referenced. It may not exceed sixty. Save the new file name in each file (DKX). Set up for a call to the librarian to get the FFT logical records for this file. LR1 contains the length for LR2 and LR3. Save these in DKF and DKS. Set up a call for logical record 2 to the library. Check the validity of that call. Generate the additional FFT names /RX and /RN. These each have a length of 9999. The first is alpha and the second is numeric. Next load in the logic record 3 of the FFT.

S1S

Generate an OP29 statement for each set in the FFT.

S2S

Fill out the a-fields and the object statement for the OP29 SET DECODE statement.

S3S

Write the object statement.

S4S

Write an error message.

S5S

Write an error message.

S6S

SKIP-SWT indicates that a fatal error has occurred and further compilation is meaningless. Therefore, generate an error message to indicate a skip to the next report.

C-HEADER

Generate an OP34 object statement. This identifies the beginning of a paragraph. Generate an OP35 object statement to indicate the end of the previous section.

C-FINAL

Generate OP34 and OP35 object statements to identify the beginning of the FINAL OUTPUT Section.

C-FIN1

Get the first word. Check for the noise word OUTPUT. If anything else, process a first word in a statement.

C-TRAILER

Generate an OP34 and OP35 object statement to indicate the beginning of the TRAILERS paragraph. The logic modes are as follows: HEADERS (LOGIC MODE 1), TRAILERS (LOGIC MODE 2), FILE (LOGIC MODE 3), LINE (LOGIC MODE 4), BEGINS (LOGIC MODE 5), FINAL OUTPUT (LOGIC MODE 6).

TRAIL-A

Get the next word. If none, OK. If there is one, determine whether it is the noise word CHANNEL. If it is, get the next word. If not, it must be numeric and must be between the values two and eleven.

TRAIL-B

Generate an error message.

TRAIL-C

N is the number of the carriage control channel.

TRAIL-D

OP75 indicates the carriage control TRAILERS option. If A1 is zero there is no change required.

TRAIL-E

Loop back.

C-FIN1

Get the first word. Check for the noise word OUTPUT. If anything else, process a first word in a statement.

C-TRAILER

Generate an OP34 and OP35 object statement to indicate the beginning of the TRAILERS paragraph. The logic modes are as follows: HEADERS (LOGIC MODE 1), TRAILERS (LOGIC MODE 2), FILE (LOGIC MODE 3), LINE (LOGIC MODE 4), BEGINS (LOGIC MODE 5), FINAL OUTPUT (LOGIC MODE 6).

TRAIL-A

Get the next word. If none, OK. If there is one, determine whether it is the noise word CHANNEL. If it is, get the next word. If not, it must be numeric and must be between the values two and eleven.

TRAIL-B

Generate an error message.

TRAIL-C

N is the number of the carriage control channel.

TRAIL-D

OP75 indicates the carriage control TRAILERS option. If A1 is zero there is no change required.

TRAIL-E

Loop back.

C-BEGIN

An LMODE of four indicates a line. This BEGINS paragraph must have been preceded by a LINE paragraph.

CB1

Update the LMODE to indicate that a BEGINS paragraph is being processed. LMODE equals five.

CB2

Process an error message.

C-LINE

A BEGINS paragraph may only follow a FILE section or a LINE paragraph. Otherwise, write an error.

CL1

Generate the OP34 separators and begin processing a LINE paragraph. LMODE equal to four.

C-IF

This is the entry point for compilation of conditional operators.

CF1X

Extract the a-field and set up the appropriate A1, A2, A3, A4 object fields. Decode the operator. If the OP is a 36 or a 37, there is no b-field. Otherwise, decode the b-field and write the object statement.

C-AND

AND is synonymous with IF.

C-OR

XTRUE equal to eleven signifies an OR operator. The IF and AND are denoted by a one in XTRUE.

C-WRITE

A WRITE statement cannot appear in a HEADERS or TRAILERS paragraph. Move a one to WRITE-SWT to indicate that an output file is to be created by this report. OP18 designates the WRITE statement.

C-PRINT

This is an OP16 object statement.

P1

Check for a period. If found, indicate in the XFALSE field. Get the next word. The only thing that can follow a PRINT statement is a PERIOD or an EJECT statement. Otherwise, generate an error message.

P2

Get the next word. If there is one, generate an error.

P3

Write an error message.

P4

Write an error message.

P5

Check for a numeric value. If it is not numeric, there must be a noise word. Go back and get another one. If it is a numeric value, it should be corresponding to the EJECT ON N statement. If the noise word is found, there must be a numeric value after it.

P5A

Extract a number. This is the EJECT ON number and it may not be greater than nine. The only legal expression that can follow this word is a period. If anything else, process an error message.

P5B

If a period is found, adjust the length of the word in CARDW. Check the length of N and loop back.

P6

Process an error message.

C--PUNCH

The PUNCH statement may not appear in HEADERS or TRAILERS paragraph. If it does, process an error message. Otherwise, indicate with an OP17 in the OBJECT statement.

C-SKIP

The SKIP statement may only be used in the TRAILERS paragraph. Logic mode equal to two. Otherwise, write an error message.

CS1X

This sets up the OP for the SKIP statement.

C-EJECT

An EJECT statement may not be used in a HEADERS or TRAILERS paragraph. Set up the OP21 for this OBJECT statement.

C-SPACE

SPACE statement has an OP20. Extract the next word. There must be a word and the length of that word should be either one or two depending on whether or not a period is found.

CS0

Check the word to determine whether or not it is numeric. If it is numeric, it is OK. Otherwise, write an error message.

CS1

Compute the actual value of that digit.

CS2

Process an error message.

C-OMIT

The OMIT statement may only be used in a file line or BEGINS paragraph. Set up the OP code of 27 for the object statement and write it out.

C-STOP

This statement may only appear in a file line or BEGINS paragraph. Set up an OP equal to 28 and write the object statement.

C-ADD

This is an OP7 statement. Set up the noise word COMPARE and skip ahead.

C-SUBTRACT

OP is equal to 8. Set up FROM as a noise word COMPARE. Skip ahead.

C-MULTIPLY

OP is equal to 9. Set up BY as a noise word COMPARE and skip ahead.

C-DIVIDE

OP is equal to 10. Set up the word BY as a noise word COMPARE.

CD1

Decode the a-field and determine whether or not the a-field is numeric. If it is numeric, field A3 will either be equal to a two or a five. If it is not, write an error message.

CD2

Get the next word. If there is none, process an error message.

CD3

The value A in AB-SWT indicates that the a-field is being checked for a subscript. Determine if the next word is numeric or alpha. If it is numeric, it is assumed to be a partial notation field. If so, decode the partial and readjust the object statement field that describes the a-field.

CD4

Check for a noise word that is required. If not present, write an error message. Get the next word. Decode the b-field and check the type. Again the type indicated by A3 must be either equal to a two or a five. Otherwise, write an error.

CD5

Write the object statement.

CD6

Write an error message.

C-DMOVE

Set up the OP to be equal to 40.

C-MOVE

This is the initial paragraph in processing a MOVE statement.

CM1

Get the next word. Check for possible use of a reserved word and the MOVE statement. SWT4 equal to one indicates that the EXTRACT word has already been performed and need not be done on entering DECODE-A.

CM2

Decode the a-field and insert the appropriate addresses in fields A1, A2, A3 and A4.

CM3

Get the next word. Check the delimiters. A delimiter must be either an asterisk or a space. Otherwise, process an error. Set up the index to search the a-field for a subscript. Check for partial notation. If found, adjust the a-field addresses appropriately. Check the limits on the partial notation. They cannot exceed the length of the original field.

CM3A

Get the next word.

CM4

Check for the noise word TO. It is not optional. The CMOVE and the TMOVE are the only legitimate operators that can have a field with asterisks as delimiters.

CM5

Decode the b-field. If OK, write out the object statement.

CM6

At this point only a CMOVE, TMOVE, or DMOVE are acceptable.

CM6A

Check the field to determine whether it is alpha or numeric. It must be alpha. It must not have a length greater than eight. This is a name of a called subroutine or a table. S806-TRAP determines whether or not this subroutine exists on a library and is available to the system. SOP NAME is the name of a save area where the special operator name will be held.

CM6D

Get the next word.

CM7

If this operator name is equal to spaces, produce an error message.

CM8

Get the next word. Decode the b-field. Set up the parameters for the object statement fields, B1, B2, B3 and B4. Save the name of the routine in a constant pool and set up the pointers to indicate its location in that constant pool. Check for overflow of a constant pool record.

CM9

Write an error message.

CM10

Write an error message.

CM11

This must be an OP11.(i.e., a standard MOVE statement). Save the OP code and get the next word.

CM6B

Initialize K.

CM6C

Search the FFT table to determine whether or not this field exists. If it does not, process an error message.

CM6F

Write an error message.

C-MOVE

CMOVE is an OP12.

C-TMOVE

TMOVE is an OP14.

C-EMOVE

EMOVE is an OP30.

C-VMOVE

VMOVE is an OP13. The VMOVE may only be used in a BEGINS paragraph. If found somewhere else, issue a warning message to indicate that there will be no repetition of the VMOVE. That is, it will be converted to a standard MOVE statement.

C-SET

Get the next word. It should be numeric. Add one to the set number. This is because the fixed set is considered to be SET1. The first periodic set is considered to be SET2 and so on. Therefore, SET1 is SET2 in the FFT. Get the next word and it must be one of those specified in the list, otherwise process an error message.

CST1

Get the next word. It should be numeric. Move its value to the object statement and extract the next word. If a word exists, compute the new OP corresponding to the value of the word found.

CST2

This recomputed OP for the set statement must be in a range between twenty-two and twenty-five, otherwise issue an error message.

CST2A

If no other value is found, a 1000 will cover all available subsets within a set since that number exceeds the maximum allowable 599.

CST3

Write an error message.

CST4

Write an error message.

C-REPORT

This will initialize a new report process that will be handled within a new STATE Section.

C-SOURCE

Get the next word. If there is none, it is an error. The eight in QUERY-SWT (Q-S) indicates that this is a source direct statement.

C-S1

Check for the valid words for the source initial statement word.

CS1A

Process an error message.

C-ANS1

Reset the source direct switch and continue to process a SOURCE QUERY type statement. Get the next word. The next word, if it exists, should be numeric. This number will indicate the number of the query in the answer file from which the answers will be provided for this report. Generate an OP77 statement and write it out.

C-ANS2

Process an error message for SOURCE QUERY.

C-S2

Get the next word.

C-TABLE

Add one to TABLE-SWT to indicate the number of tables included within this report. There must be another word. If not, process an error message. This word should be numeric. TABLE-SWT indicates which of the parameters are being processed that define the table.

C-TAB1

Load the table dimensions into the constant pool. Compute the size of the table. The combined length of the function and the argument may not exceed sixty-eight characters.

C-TAB2

Read in each of the table elements and check the total number at that point to determine it does not exceed the number identified within the size specification for that table. If there is a discrepancy, process an error message.

C-TAB2A

Write out the table entry record.

C-TAB3

Identify the table name in a list of defined names.

C-TAB4

Process an error message.

C-DEFINE

Initialize the DEFINE statement switches. Assume the type will be alpha unless otherwise specified.

C1

Get the next word. In any case the word must not exceed eight characters and might be one of the names included in a list of words checked for. If the word is a type indicator, go back and get the next word.

C3

Check to see whether the name exists in a define name table. If it does, set up an error.

C4

Process a warning message.

C5

Check the length. If the length is greater than five, it might be a defined name.

C6

Determine whether the field exists in the FFT.

C7

If the name is not a duplicate, set up an entry to create another line in the defined name table.

C8

Determine whether a convert routine will be required for this define.

C8B

Determine whether a length parameter has been specified for this define.

C9

If a left parenthesis has already been specified, there must be a corresponding right parenthesis. If not, process an error. Adjust IL to remove the parenthesis.

C9A

Check to determine whether this field is alpha or numeric. It must be numeric to be a length parameter. Compute the actual value of the numeric digits. This value will be found in N. A define field may have a length between one and three hundred sixty characters if it is alpha. Identify this field's location in a constant pool. Check to see whether matrix processing is invoked and whether this is an OCCURS type DEFINE statement.

C10

Check the length of the return value from a convert routine. Determine whether or not the convert routine exists. Call a convert routine and check the error code returned by that convert routine. If an error code was returned, an exit flag determines whether a returned error message is to be put out. If so, process.

C10A

Reposition a return value from the convert routine.

C11

SWT3 indicates whether a length parameter was specified in the DEFINE statement. The length returned by the convert routine must not exceed the length allowed for in the definition of that statement.

C13

If a numeric field is being defined and the length has not been specified, the next value found in that statement must be the numeric constant which represents the value of that field.

C15

If the field is numeric, its length cannot exceed fifteen characters. Insert leading zeros to fill the field to its proper length.

C16

Load the value in a constant pool. Check for overflow of the constant pool record. A card number less than one thousand indicates that this define is being redefined and already exists in the table.

C17

Move this new name entry into the define table.

C18

Get the next word. Save the value of N. Extract the number from that last word. Compute the size of the OCCURS field. Check for overflow in the constant pool. If the field is numeric, initialize at zero. Otherwise, initialize at spaces.

C20

The small card number indicates the MODIFY statement.

C-MATRIX

Determine whether a matrix has been defined previously. If so, process an error.

C-MAT1

Set up a define table entry for the matrix. Get the next word. The next string of characters must be a length parameter with parentheses as delimiters.

C-MAT2

Check the type of the matrix. If it is numeric, the length cannot exceed fifteen characters per element.

C-MAT2A

Get the next word. This next word must be numeric and will indicate the number of rows to be in a matrix defined. Get the next word and check for the noise word row or rows which is required.

C-MAT3

Get the next word. This must also be numeric and must indicate the number of columns to be present in a DEFINE matrix. Then check the next word for COL or COLS which are required words.

C-MAT4

Generate and write out the define matric object statement which is an OP76 and move the define matrix name to the table of define names.

CD-LIN

The SLANT LINE operator may be defined only once in a report.

C-LIN1

Set up the constant pool to accept the definition of the SLANT LINE operator. Fill an entry for a define name table and check the size of the occurrence on a slant line.

C-PROG

The program must appear after the last line BEGINS paragraphs before the FINAL OUTPUT Section. If not, write an error message.

CP1

Get the next word. This word should be numeric and indicate the number of the program that is to be performed. This number should be between one and ninety-nine, inclusive. If acceptable, generate an OP34 and OP35 OBJECT statement. Also check to see whether this program has been previously defined with the same number. If so, create an error message.

CP2

Flip a switch to indicate that this number has been used and may not be used again.

C-PERFORM

Check the LMODE. This statement may not be used in the HEADERS or TRAILERS paragraph. Check for the optional noise word PGM program. If found, skip back and get the next word. Set up an OBJECT statement for an OP33, the PERFORM statement. Extract the next word. This should be a program number that is to be performed. Try for another word. If it is found, this program is to be performed as a PERFORM statement. Requires execution of a program the number of times. This latest word that is extracted indicates how many times. Then check for the noise word TIMES or TIME.

CPP4

Get the next word and check for a period.

CPP2

Set up an error message.

CPP3

Write the OBJECT statement.

CPP5

Process an error message.

WARN-MESSAGE

Set up the error message fields to process a non-fatal warning message.

MOVE-WARNING

Move the warning message into the output area and set up card column indication for the word to cause the message to be generated.

WRITE-ERR SECTION

Set up card column indication for an error message. Increment the card sequence numbers so the message will come out directly after the source statement on the output listing. Write out the message. Set the error vector in QS. Check for an ABORT flag. If an ABORT flag exists in QUERY-SWT (100), propagate error vectors throughout the remaining queries in the query vector.

W-E1

This is the loop that propagates the error vector for the ABORT operator.

WRITE-ERR1 SECTION

Set up the headers for and write out an error message that does not have card column indication. Also check the ABORT switches the above paragraph does.

W-E1

Propagate the error vectors switch for the ABORT operator.

DECODE-OP SECTION

D1D

Get the next word. This should be an operator but first check to see whether there is a subscript involved in the a-field. After the subscript is checked for, get the next word and check for a numeric field which indicates partial notation on the a-field. If this occurs, process it at this point and readjust the object statement indicators for the a-field.

D2D

Check for the noise word IS. If it is found, bypass and get the next word.

D3D

Check for an asterisk delimiter. If found, indicate that special operators may not be used in the output module.

D6

Check for negation of the operator. If the word NOT is found, increment the OP code by three to indicate that negation will occur.

D7

Check for a legal operator. Increment the OP code correspondingly and continue. If not a legal operator, set up an error message.

T-SCAN

If the OP is equal to zero, this indicates that the TEXT SCAN is against the fixed or periodic field. If OP is equal to twenty-six, it indicates that the TEXT SCAN is against a variable field.

T-SCAN1

Get the next word. If none, process an error. Otherwise exit the DECODE-OP Section.

CHA-COM

Check the error vector for a compilation error preceding this statement. Establish a save area to be used by the CHANGE COMPLETE operators for comparing against the previous value.

D8

A valid COMPARE operator cannot have an OP with the value greater than six.

D9

Check for the noise word connector between the operator and the b-field. There must be one present.

D10

Write an error message.

D11

Write an error message.

FLD-PART SECTION

Initialize the subscripts for the field partial extraction section. Check the length of the combined partial. It may not exceed thirteen.

F1F

Check the first digit. It should be numeric. Compute the value of the numeric field. Loop back again for the next digit until a hyphen is encountered.

F2F

The only valid connector for partial notation fields is a dash. Set up for the decoding of the second half of the partial.

F3F

Flip through the digits in the last half of the partial and compute the actual value of the number found.

F4F

The only valid character at the end of this notation is either a space or a period.

F4AF

Check the value of the new length defined by the partial. It must be greater than zero.

F5F

Check to see the partial does not exceed the legal bounds.

F5FA

Process an error message.

F6F

Reset the alpha numeric checker.

DECODE-A SECTION

Initialize the DECODE-A switches.

A1A

Get the next word. If none, process an error.

A2A

Check the word for alpha or numeric. If the word is alpha and its length is greater than five, check its maximum length. If the word is five or less, it might be in the FFT.

A3A

Check for a maximum length of eight. If it exceeds that, process an error message.

A4A

If the field is in the FFT, set up the a-fields in the object statement to so indicate. Convert routine may only be used as the standard COMPARE operator with an operator code of six or less.

A5A

Loop through the FFT logical record 2 to determine whether the field exists or not.

A6

Loop through the table of define names to determine whether the field exists or not. If the end of the table is reached without making a hit, process an error message. This means the name has not been defined.

A7

Check for a MATRIX operator. If none, set up the define name entries in the object statement fields A1, A2, A3, A4. The A-MAT-SWT is used to decode the variable names used in the subscripts for matrix processing.

A-MAT1

Set up for a matrix operation in the object statement. Save this object statement as built so far. Another statement must be generated and written out before this one is completed.

A-MAT2

Get the next word. If there is none, process the error message.

A-MAT3

Write an error message.

A-MAT4

The subscripts for MATRIX operator must be bounded by parentheses. Shift the value left one position with the subroutine MARINM to eliminate the parentheses. Check to determine whether the subscript is a variable or constant. If it is a constant, extract the number to derive its actual value. Save this value in a constant pool and identify its location in the operators A1, A2, A3, A4.

A-MAT-ERR

Process an error message.

A-MAT10

Check the length to determine whether the field could be in the FFT.

A-MAT11

The set number in a MATRIX statement must be one. The variable name used as a subscript in a MATRIX statement must be numeric. If not, process an error.

A-MAT20

The RC-SWT indicates whether rows or columns are being processed. Check for a comma separating the two values within the double subscript. If there is not a second value or second subscript, process an error. Check for matching parentheses. Overlay the parentheses for this space. Check for a minimum length of the word. Determine whether that word is numeric or alpha. If numeric, save the value in a constant pool and indicate in the b-fields of the object statement where that constant pool location is.

A-MAT30

Save the a-field parameters to take advantage of previous coding and decoding and loading descriptions into the a-field area.

A-MAT31

Shift the values back so the a-field indicators point toward the first subscript and the b-field indicators point toward the second subscript.

A-MAT40

Set up the OP code of 47 and write out the statement. Now restore the object statement values to those built prior to processing the MATRIX statement.

A8

Write an error.

A9

Check for MOVE, ADD, or SUBTRACT operator.

A10

The search value cannot be greater than 360 characters in length. Save this value in a constant pool and indicate the pointers to it in the a-fields.

A11

Transfer a value from the input card area to the constant pool. Check the length of the constant pool record and write it out if necessary.

A12

The constant value can only be an a-field on a MOVE, ADD, SUBTRACT, or PERFORM statement. Otherwise, process an error.

A14

Check for a MATRIX, MOVE, or VMOVE statement. No other statement may access variable data.

A14-1

Generate the variable set control field information.

A15

The VMOVE may only reference variable data.

A16

The a-fields for the PERFORM and EDIT statements must be numeric.

A17

Check for a numeric field.

DECODE-B SECTION

Initialize switches. Determine that a word exists for the b-field. If not, produce an error. Check for a valid delimiter at the end of the word which must be either a space or a period.

B1B

Check for a period after the b-field name.

B2B

If the delimiter is not a space, then the b-field is a literal. All the OP codes mentioned may have literals as b-fields. Otherwise, an error is produced.

B3B

If SNAME does not equal spaces, then the value of SNAME indicates the name of the convert routine. Execute the convert routine. Check the return code indicated in EXIT-FLAG.

B4B

Recover and restore fields from the convert routine CALL.

B4D

To get to this paragraph, the OP code must have been a nine or a ten, which is a multiply or divide. In this case, the b-field must be numeric.

B4C

Save the b-field value in the constant pool area.

B4A

The length returned from a convert routine cannot exceed the length of the a-field.

B5

Determine whether the b-field constant length exceeds the ANAME length.

B6

Set up the b-field object vector to indicate that the value is in the constant pool.

B7

Determine whether the a-field is alpha or numeric.

B7A

If the a-field is numeric, the b-field must also be numeric.

B8

Insert leading zeros if necessary into a numeric b-field.

B9

Load the value into the constant pool and add trailing spaces if necessary.

B10

Determine whether the first character is an alpha character or a numeric digit. For the OP codes not mentioned within this paragraph, a constant is not legal as a b-field.

B11

If the length is five or less, it may be either an FFT item or a constant pool name. If the length is greater than five, yet less than nine, it may only be a constant pool name, as all FFT names must be five or less characters in length.

B12

Initialize the search for the name.

B13

Search the FFT table for the name.

B14

If the name has been found in the FFT table, move its description to the b-field vector.

B14A

Establish the location on the card source image.

B15

COM-SWT is used to determine if a comma was encountered when reading a word from the source statement in the EXTRACT WORD Section. If a comma was found prior to the word, a B is moved into the switch. If a comma was the last character of the word, then A is moved into the switch. When a range parameter is encountered, this switch is tested in DECODE-B to determine whether the range parameter is a partial or an output area.

B15A

Distinguish between a blank or a non-blank delimiter.

B16

Check for a subscripted b-field and/or partial notation.

B16D

Check for a period at the end of the statement.

B16A

If a period has not been found at the end of this statement, scan the rest of the card for a period.

B16B

Check for another word after a period has been found. This is an illegal condition.

B16C

Process an error message.

B17

Initialize a search for a name in the defined name table.

B18

If a name is not found, process an error.

B19

If a name has been found, set up the appropriate fields in the b-field of the object statement.

B20

Check for another word in that statement.

B20A

Check for a period in the statement.

B20B

Complete the rest of the fields in the object statement. Write it out. Go back and check for another b-field.

B21

Set the period indicator switch.

B22

Check for partial notation and if found, adjust the b-field object statement appropriately.

B-MAT1

Move the length and type of the matrix element to the b-field statement and save that statement. Additional object statements need to be generated for the subscripts on the matrix.

B-MAT2

Get the next word.

B-MAT3

Process an error.

B-MAT4

This paragraph decodes the double subscript. The first character must be a left parenthesis. If so, shift the field left to eliminate the left parenthesis. Determine whether the first matrix subscript is alpha or numeric. If numeric, it is a constant. If alpha, it is a field name. If it is numeric, identify in the object vector the location in the constant pool where the value will be saved and save that value.

B-MAT-ERR

Process an error message.

B-MAT11

The matrix subscript field must be numeric.

B-MAT20

The RC-SWT determines whether the row or column subscript is being processed on this pass through. The row subscript will be loaded into the a-field of the object statement. The column subscript descriptor will be loaded into the b-field of the object statement.

B-MAT21

Processing the second subscript of the matrix subscript fields, determine whether it is alpha or numeric and process similarily to the first one.

B-MAT30

The value 'C' in RC-SWT indicates that we are processing the column subscript.

B-MAT40

An OP48 indicates the b-field subscript. Write the matrix object statement out. Restore the statement previously generated.

WRITE-STATE SECTION

This paragraph is, in effect, a "GO TO depending on" the operation code of this statement.

A-SORT

The periodic sort must be in a BEGINS paragraph and must specify that a periodic field be sorted. Otherwise, process an error.

WSO

Determine that, where required, the ANAME is in fact numeric.

WSOA

Process an error message.

WSOO

Determine that, where necessary, the b-field is in fact numeric.

WSOOO

Separate the conditionals, directives, and move statements.

WS1

Determine that the a-field and b-field types match.

WS3A

A b-field TYPE 6 is equivalent to an alpha field.

WS3

The TYPE 6 statement is an output object statement.

WS33

Check for an error vector on this query will report. Then separate the generated object statements. Periodic set decode statements and break statements in A4 or D4, greater than one, indicate the periodic fields were referenced and that GEN6 will have to be called rather than GEN6A.

WS01A

Increment the object statement COUNT and write this statement out.

WS4

The lengths of the a- and b-fields must either have been equal or have been adjusted to equal before they get to this paragraph.

WS5

Validate the types of the a-fields and b-fields.

WS6

On a MOVE SPACES statement, the b-field must be alpha.

WS6A

Write an error message.

WS7

Trap out a variable MOVE and EDIT statement.

WS8

If the a-field is numeric, its length cannot exceed fifteen.

WS9

If the b-field is numeric, its length cannot exceed fifteen.

WS10

If the b-field is shorter than the a-field, truncation will occur. Issue a warning message.

WS11

Shorten the a-field length to the b-field length.

WS12

If the b-field is longer, trailing spaces will be added. Issue a warning message.

READ-OP SECTION

Read in an output source statement. Copy it onto the work file. Output source statements are TYPE 5. If the statement read is not a TYPE 5, go back and read another until one is encountered. An asterisk in column 1 indicates a comment card. Bypass and go back for the next card.

R2

Issue an error message.

R3

Issue an error message.

R4

Process an error.

EXTRACT-WORD SECTION

Initialize counters.

W1

Check for an end-of-card condition. If a continuation card was indicated on a previous statement, initialize the card columns search on column 4.

W2

Check for non-blank, non-delimiter type characters.

W2A

IC is the active column on the card.

W3

Save the delimiter.

W4

Check for an end-of-card condition.

W5

Check for the end of the word.

W6

Save the next character of the word.

W6A

Check for a delimiter.

W6B

Process an error.

W7

Check for a valid delimiter.

W7A

IW is the last character of the word.

W8

Add trailing spaces to the name to fill out to a fifteen-character name.

LEAD-ZERO SECTION

Insert leading zeros in a numeric field to pad the field to the specified length.

LOAD-CONSTANT SECTION

Perform a character move to load the value into the constant pool area. IL is the length of the value to be moved. CDYN is the subscript of the location of the constant pool to which that value is to be moved.

TRAIL-SPACE SECTION

Insert trailing spaces in the field whose value is less than its specified length.

WRITE-CDYN SECTION

Set up the headers for a constant pool type record. The type of this record is 7.

MOVE-CON

If an error has not been indicated in the query switch, write out the constant pool record. Initialize the subscript to the next available location in the constant pool record work area (the value of CDYN). Increment the sequence number in the header for the constant pool record.

EXTRACT-NUM SECTION

Initialize the counters.

MOV-CK

A numeric value is expected by the EXTRACT-NUM Section. If the binary value of the character is less than 240, that character is alpha and not numeric. Therefore, process an error message. Compute the actual value of the number digit-by-digit based on its binary value minus 240.

MOV-CHK1

Process an error message.

S806-TRAP SECTION

Execute a BUILD L macro to determine whether or not that named subroutine exists on the library.

S806-A

Process an error message.

S806-B

Process an error message.

S806-C

Process an error.

FIX-F SECTION

Recompute the values and logical record 2 of the FFT to give positive set numbers for all fields and eliminate plus and minus zeros. The fixed set will be set 1. Each periodic set will be incremented by one after that. Therefore, periodic set 1 will be considered to be set 2 and so on.

TYPE-TEST SECTION

If the types of the fields match, everything is OK. TYPE 1 or a TYPE 4 indicates an alpha field. A TYPE 2 or a TYPE 5 indicates a numeric field. A TYPE 6 is considered to be alpha.

USER-ERROR SECTION

Write out an error message returned by a user from a convert routine. Also set the error switch to indicate that a compilation error has occurred in processing this report.

PERIOD-SCAN SECTION

Search for and remove any periods found in the CARDW area.

CKCK SECTION

Determine whether a program exists for each PERFORM statement that requires a specific program number.

DUP-CHK SECTION

Initialize the counter. Search through the define name table looking for a duplicate name. If one is found, process an error message.

DCODE-PARITAL SECTION

Partial notation, if specified, must be within the boundaries of the field.

HYPHEN-SEARCH SECTION

If a hyphen character is found in the CARDW area, indicate with the value one in the hyphen switch.

BUILD-OCCURS SECTION

Create space in a constant pool for a DEFINE OCCURS type statement.

S-INDEX SECTION

A subscript must be preceded by a left parenthesis.

S-16

A subscript value must be followed by a right parenthesis and/or a period.

S-17

Process an error message.

S-10

A subscript may be used only on a conditional statement or a variation of a MOVE statement, except that it may not be used on a variable move.

S-100

Process an error message.

S-I1

Initialize the values for shifting the field left to eliminate the parenthesis.

S-I2

Determine whether the subscript is a constant or field name. If it is a constant, load the value of the constant into the constant pool and set up the a-field areas to correspond to the value of that constant and its location within the constant pool.

S-I3

The DECODE-A Section will process the name of the subscript field and insert its characteristics into the a-fields of the object statement. It must be numeric.

S-I4

The a-field must be saved since the subscript statement requires generating an additional statement just prior to the statement containing the subscript.

S-I8

Shift the field name left, character-by-character.

S-I9

Reset the AB-SWT.

INDEX-PROCESS SECTION

Set up an object statement for a subscript type operator.

INDEX-PROC2

B-SWT not equal to zero indicates that the b-field is being subscripted. This is an OP42.

INDEX-PROC3

Check the error vector for this report for a compilation error. If none, write out the subscript object statement.

INDEX-PROC4

Restore the saved object statement.

WRD-PAIR SECTION

Extract and edit partial notation fields. They must be separated by a hyphen delimiter.

WP-C

If followed by a period, indicate with a one value in XFALSE.

WP-A

If CARDW contains a period, indicate its presence by moving a one to XFALSE and replace the period with a space.

(10.1) GEN3B.

DATA DIVISION

| Name | Function |
|---|---|
| CONREC | Input record with its redefinitions to accommodate either input vectors or card images. |
| OUTOP | Output record. |
| BLANKLINE | Used to slew line. |
| COMP-OP-CARD-SAVE | Area used to save COMP-OP card. |
| QUERY-CARD-SAVE | Storage Area for QUERY card. |
| HOLD-DATE | Area used to store date |
| LAST-KEY | Area used to compare with CURRENT-KEY for valid sequence (Sort) check. |
| FILE-NAME-SAVE | Storage area for file name. |
| LINE-SEP | Print-Image for Separator pages. |
| LIB-CALL-SEQ | Parameters for library subroutine. |
| LIB-BUFFER | Buffer for source and vector images to be written to the library. |
| NAMER | Area for source report name. |
| NAMEJ | Area for object report name. |
| MSG1, MSG2 | Output messages to indicate addition or replacement of library member. |
| CONTROL-REC | Storage area used to build vector which holds communication data from the compilers for the logic processors (This is the data normally passed via the Linkage Section (i.e. QUERY-COMMUNICATIONS)). |

GEN3B SWITCH TABLE

| Switch Name | Variable/ Fixed | Values | Purpose |
|---|---|---|---|
| NAMER-SWT | V | "A" | Source member being added to library (initial value). |
| | | "R" | Source member replacing existing source member. |
| NAMEJ-SWT | V | "A" | Object member being added to library (initial value). |
| | | "R" | Object member replacing existing object member. |
| MCOUNT | V | Ø thru 99 | Contains the number of the last member written to library. |
| LENGTH | V | 80 | Multiplier to determine length of source library data. |
| | | 180 | Multiplier to determine length of object library data. |
| LINESUSED | V | Ø thru 56 | Counter of lines for page control |
| I | V | | Counter/index used for writing source and object vectors to the library. |

PROCEDURE DIVISION

The first paragraph is the entry point of the OP Library Maintenance program called by GENØ. After the input and output files are opened, the "MIDMS START PROCESSING (COMPILE OP)" banner page is written. The date subroutine is then called and the current date is stored in YYDDD format in the area HOLD-DATE.

G10-BEGIN-PROCESS

TYPE-S (2) of the Linkage Section is tested to determine the type of run to be processed. If TYPE-S (2) equals 9, the run is a normal RTOP execution and GEN3B should not have been called. The next test is for a COMP-OP card (Q-IN equal zero and TYPE-IN equal 1); if this is not found, a fatal error is processed and the run terminates. The name on the COMP-OP card is then saved in the areas NAMER (source member name) and NAMEJ (object member name) and a 'J' is placed in the fifth character position of the object member name. The next card is read and printed if it is an error message; otherwise QUERY card processing begins.

G15-PROCESS-QUERY-CARD

The input stream is then checked for a Query card and one is generated if not found. The "Query" banner page is printed with the query sequence number. DUPLICATE-CHECK is then performed to delete existing library members if a replacement operation is to occur.

G20-MAIN-SECTION

The main processing loop begins by checking for a possible sort sequence error by testing CURRENT-KEY against LAST-KEY. TYPE-IN is then tested to determine whether source or object images are to be processed.

G30-PROCESS-SOURCE

If TYPE-IN equal 4 or 5 source-image processing is indicated. When TYPE-S (2) equals 7, object vectors only are to be updated. If TYPE-S (2) = 8 testing is done to accommodate both source and object updating. When Q-SWT (1) equal one an error condition exists and the image is written out. Also, if the card image encountered is a library card this image is bypassed and a new image is read. The processing for source images then continues; I is incremented by one to index the source image and the image is then loaded into the library buffer. A check is then made to insure that the library buffer is not larger than 120 images and if true, the image is printed and a return to the main processing section is accomplished. Otherwise, the images are written to the library (PERFORM WRITE-LIBRARY).

G40-PRINT-CARD-IMAGE

The report source card images are printed.

G100-PROCESS-VECTORS

Begin the vector-buffer loading. If TYPE-IN indicates the image to be a compiled object vector (6), constant pool string (7) or a compiled fly-sheet object vector (8), any remaining source images will be flushed from the library buffer.

G110-FLUSH-SOURCE

This switch is reset after its initial execution to bypass flushing of source images from the library buffer.

G120-FSA

Reset the above switch and flush the source images from the library (PERFORM WRITE-LIBRARY). Compiler generated information is then saved in the first vector to be stored in the vector library buffer.

G140-SAVE-VECTORS

Error checking is first done and if passed, the necessary initialization takes place and a check is made to insure the integrity of the file name in each vector.

G150-SAVE-VECTORS

I is then incremented by one to index the vector being processed and the vector moved to the library buffer. If the number of vectors moved (I) is less than 90 a return to the main section is accomplished; if not, the vectors are written to the library and control is then given to the main processing section.

G200-END-OPSTATE

This paragraph is executed when an EOF is encountered on the input file. An immediate check is done to insure completion of the processing of vectors (i.e. flush buffer). The action message describing the disposition of the source report is then printed, if necessary.

G210-EO-MSG2

The action message describing the disposition of the object report is printed.

G220-EO-QUERY

The terminal QUERY page is printed.

G40-PRINT-CARD-IMAGE

The report source card images are printed.

G100-PROCESS-VECTORS

Begin the vector-buffer loading. If TYPE-IN indicates the image to be a compiled object vector (6), constant pool string (7) or a compiled fly-sheet object vector (8), any remaining source images will be flushed from the library buffer.

G110-FLUSH-SOURCE

This switch is reset after its initial execution to bypass flushing of source images from the library buffer.

G120-FSA

Reset the above switch and flush the source images from the library (PERFORM WRITE-LIBRARY). Compiler generated information is then saved in the first vector to be stored in the vector library buffer.

G140-SAVE-VECTORS

Error checking is first done and if passed, the necessary initialization takes place and a check is made to insure the integrity of the file name in each vector.

G150-SAVE-VECTORS

I is then incremented by one to index the vector being processed and the vector moved to the library buffer. If the number of vectors moved (I) is less than 90 a return to the main section is accomplished; if not, the vectors are written to the library and control is then given to the main processing section.

G200-END-OPSTATE

This paragraph is executed when an EOF is encountered on the input file. An immediate check is done to insure completion of the processing of vectors (i.e. flush buffer). The action message describing the disposition of the source report is then printed, if necessary.

G210-EO-MSG2

The action message describing the disposition of the object report is printed.

G220-EO-QUERY

The terminal QUERY page is printed.

G230-END-PROCESS

The MIDMS END OF PROCESSING page is printed and the files are closed.

G240-END

Return to GENØ.

WRITE-LIBRARY SECTION

This section writes individual members to the MIDMS Library. The member number and length are computed before the librarian subroutine (LB) is called. If the OPERATION code is set to N upon return, an error was encountered and a message is printed.

DUPLICATE-CHECK SECTION

This section deletes library members to prevent the occurrence of duplicately named members. If Q-SWT (1) equals one, an error condition exists and no addition/replacement will occur. If TYPE-S (2) is equal to 7 a replacement (or addition) of the object only will occur (DC-OBJECT) and if TYPE-S (2) equal 8 both source and object will be replaced (DC-SOURCE-LOOP). MEMBER-NAME is the area through which the member name is passed; MEMBER-SUFFIX indicates the number of successive member and is incremented at each library call.

WRITE-SEP SECTION

Separator pages are written.

EJECT-PAGE SECTION

Reset linecount and eject page.

PRINT-CARD SECTION

Print a source report card image.

ERROR SECTIONS

The remaining sections print error messages.

(11) GEN4 and GEN4A.

The program GEN4 is the logic processor of the Retrieval Subsystem of MIDMS. This is a brief description of the INPUT/OUTPUT Section.

FILE-CONTROL

There are five data files defined in GEN4. The first is CONFILE which is generated by the MIDMS compilers, GEN2, GEN3, GEN3A. This file contains all the constant pools generated by the compilers and vectors generated from statements the user has provided. Part of this data is used by GEN4 and part is placed in OPSTATE output file for further processing by the remaining modules of MIDMS Subsystem. The record formats of the CONFILE and OPSTATE file are identical. The logical records are of a fixed nature containing 108 characters and the blocking factor may be specified in the JCL cards. The third file is the INFILE which provides the input data files from which data will be retrieved. The logical records in this file area are of a variable nature and the first four characters in each logical record must be a binary control word. Maximum size of a logical record is 10,000 characters. The blocking factor could be determined by the size of the buffer and the block size that is specified in the DD cards. When records are retrieved for subquery, they may go as output in one of the two remaining files. First is the OUTFILE or answer file and contains fixed length records which are restructured by the output of the GEN4 program and contain sort information, rank, query number, and subquery number. The second possibility is that the records may go into the SUMMFILE which is identical to the input file. The answer records which are placed in a summary file are identical to the retrieval. However, the blocking factor may be changed at the user's discretion by the JCL cards.

DATA DIVISION

There are four groups of tables in the Data Division.
(1) Q-TABLE, (2) SET-TABLE, (3) STATEMENT TABLE, (4) MISCELLANEOUS.

## Q-TABLE

The Q-TABLES are a collection of nine vectors, each having 99 entries and are under the title of Q-TABS. The Q-TABLES contain information about each subquery that is to be processed per file. The following is the information in the Q-TABLE.

Q1 - is five characters long. First two characters are query numbers. Third character is a 2-type. The last two characters are subquery numbers.

Q2 - contains the periodic set number which means that in this subquery, a sort flag or a merge flag statement has appeared.

Q3 - is a subscript into the STATEMENT TABLE, which gives the location of the first statement of this subquery.

Q4 - is the location of the first sort/merge statement or if none appear, the end of the subquery.

Q5 - a one in Q5 will cause the subquery to stop processing when the limit of output has been reached. See STOP-ON-LIMIT statement.

Q6 - is not used at the present time.

Q7 - contains the number of input records that were processed for this subquery.

Q8 - is the number of answers that were created for this subquery.

Q9 - is the limit of answers that the user has specified for this subquery.

SET-TABLE

The SET-TABLE does the bookkeeping on the periodic sets. Each set is represented in this table by one row.

S1 - contains the starting position of the first subset, data set, or periodic set.

S2 - contains the length of a subset in that particular periodic set.

S3 - contains the number of subsets present in this periodic set.

S5 - is a flag switch which means that one of the subsets has been flagged.

S6 - indicates how many subsets have been processed at this time.

S7 - is a work switch which indicates what search mode was applied against this subset and if the logic was satisfactory or not.

STATE-FF

STATE-FF matrix is loaded with vectors that were created by the compilers for each user source statement. Each of these statements contain information about a-field and b-field operation and what search mode is being applied against the subsets to be processed. Information in a-field and b-field is starting position, length, type of the field, numeric or alphanumeric, set number to which it belongs, and operation that is to be performed (equal, greater than, not less than, etc.). See GEN2 compiler for detailed explanation of values of these fields.

PROCEDURE DIVISION

Entry point of GEN4 is EGEN4, using query communications which is an area defined in the supervisor GEN0. This area contains information about each subquery concerning presence of errors, summary file, if it is a source direct and quantity of statements in the output. GEN4 uses this area to obtain the data set names for both input and output and to print the processing of query errors.

HOUSEKEEPING - GEN4

A number of operations are performed on the DCB data control blocks of the input and output files. The logical record length on the DCB, the block size and the record format are all turned to zero. This facilitates the operating system to insert this parameter from the JCL control cards. The data control block of the INFILE is saved in DCB. The OPSTATE and OUTFILE are output and CONFILE is input.

STEP 1

Read a record from the CONFILE.

STEP 3-1

Housekeeping is done for a new data file for retrieval. Move zero to number of answers retrieved from this file, to number of sub periodic sets existing in this file, to the number of subqueries applied against this file and to the number of statements for all subqueries applied against this file. Also zero out the SIA, the starting position of a set, S2A, subsets processed and S7A, a work vector. Move spaces to FILE-NAME. Move spaces to subquery identification Q1(1). Move one to field N, to be used as a subscript for loading the vector statements to the statement table. Move one to I, to be used to load subquery information to the Q-TABLES. Move one to N1, to be used as a backup for N. Go to Step 4-1.

STEP 4

Read a record from the CONFILE.

STEP 4-1

The query number of the record just read is saved in field AL. Go to depends on TYPE-IN as there are nine types of statements coming from CONFILE. TYPE 1 is the source statements and the retrieval. TYPE 4 is source statements and shorthand language. TYPE 5 is source statements and output. TYPE 8 is fly sheets and TYPE 9 is not used. These statements are moved from the input buffer to the output buffer and written out on the OUTSTATE file in paragraph T1.

T6

TYPE 6 statement indicates that this vector is for the Output Subsystem. A new card number sequence is computed and inserted in the output record. If the logic mode is equal to zero, the previous logic mode is inserted into this statement.

T7

Q-SWT vector is an error vector. A one in position 1 means that query 1 is an error, a one in position 2 indicates that query 2 is an error. An error record is not written in the output file. If the query is not an error, go to T1 to place the record in the output files for the output module.

T2

This paragraph is executed when a TYPE 2 statement comes from the CONFILE. The query for this subquery is checked for an error. If the query is an error, the statement is not loaded. Go to STEP 4 and read another record.

STEP 5

Move spaces to EOJ switch to indicate no end of the CONFILE. Move the present name of the input file to the file name to save. Check if name of file for this statement just read compares with the file name loaded previously. If not equal, go to LINK-EDIT. If TYPE-IN is equal, go to STEP-6 if the type of record is a constant pool segment. Q1 (I) is the identification of the present subquery and Q1 contains query number and subquery number. If Q1 is equal to Q1 (I), go to STEP 5-1. Otherwise, a new I is computed. I is equal to QL + 1. QL represents the last number of subqueries that were loaded for this file. QL was set to zero in STEP 3-1. Set QL equal to I. Move the subquery identification to Q1 (I). Move 5555555 to Q9 (I). Q9 (I) now represents the system limit of answers to be retrieved for this subquery. It is a system default. Q7, Q8, Q4, Q2 and Q5 are initialized to zero. Place N in Q3 (I). N is the location where this vector statement will be loaded into the statement table STATE-FF.

STEP 5-1

Field DTRUE is filled in by the compilers with values for the logical connectors of the statement, IF, AND, OR, etc. An OR outside parenthesis is equal to eleven. If the statement was not an OR, go to STEP 5-L. If the statement was not of the primary logic class, go to STEP 5-L. If the previous statement was an IF or an AND outside parenthesis, go to STEP 5-LC. If the previous statement was an OR outside parenthesis, go to STEP 5-LC. Otherwise go to STEP 5L.

STEP 5-LC

If the previous statement was not an equal compare test, go to STEP 5-L. Add the previous b-field length to the starting position of the previous b-field. Test possibility of building a statement table. If ANAME equals one, five, seven, or nine, this can become a table and the statement will match that table. Check if the characteristics of the file such as position, length, type, set number for a-field and b-field, are equal. If any one of these characteristics do not match, go to STEP 5-L. If they all match, it is possible to do a table look-up here. Retrieve the previous statement from the statement table STATE (LL). LL is the last statement loaded. Increase the operation of that field by fifty. Reload the statement under STATE (LL) and go back to get another record.

STEP 5-L

The first half of the vector is converted from BCD numbers, which are generated from the compiler, to binary half-word numbers. If the operation (OP) is equal to twenty-two, the SUBQUERY has a user-provided limit of answers to be retrieved from the data file. The limit is moved to Q9 (I). It is not loaded into the STATE table. Go to to STEP 4 to read another record.

If the OP is not equal to eleven, go to STEP 5-LL. Eleven is the operation code for a STOP statement. There are two types of STOP statements. A conditional STOP and a STOP ON LIMIT. THE STOP-ON-LIMIT statement will contain a one in A1, otherwise A1 will be zero. The one is placed in Q5 (I) and the statement is no longer needed, therefore it is not loaded into the STATE table. Go to STEP 4 to read another record.

STEP 5-LL

The second half of the statement vector is converted from BCD to binary. If LMODE equals eleven, it is a flag statement and the set number is placed in Q2 (I). If LMODE equals twelve, it is a merged flag statement and the periodic set number is moved to Q2 (I). If the length and type of b-field (B2 and B3) are equal to zero, the length and type of a-field is moved to b-field. When OP equals nineteen, the statement contains information about coding the periodic set control words. If OP does not equal nineteen, go to STEP 5-2. Otherwise, if SL (the last subset from which information has been obtained) is greater than A4, go to STEP 4. The vector S4 contains the starting position of each periodic set control word in the file. This information is placed in A1 of the vector and moved to S4 (A4). A4 is the periodic set number. A2 contains the length of each of the subsets and is placed in S2A. SET-A is moved to SST-W (A4). It is placed into the SET-TABLE matrix. Also SL picks up the value of A4. Go to either of the statements in STEP 4.

STEP 5-2

Insert this vector statement into the STATE-FF matrix which is redefined as STATE (N). N being the number of statements at this particular time. Place N in the name field LL. LL indicates the last statement loaded. Increment N by one to be ready to load the next statement. If LMODE contains a value greater than eight, it means that the statement just loaded is a sort statement, a merge statement, or the last statement in the subquery. If that is the case, check if Q4 (I) contains any information. If Q4 (I) is equal to zero, it means that it is the first statement that has been encountered and the location is saved in Q4 (I). Go to STEP 4 to read another record.

STEP 6

The constant pool is loaded with the constants generated by the compiler. The CARD-IN field contains the subscript and the constant pool of this constant segment. The constant segment is ninety characters long and is moved to CONST64 (AL) which is a redefinition of the constant pool. Go to STEP 4 to obtain another record.

LL-TYPE8

This step will be executed when an END-OF-FILE is sensed in the input CONFILE. Set the E-O-J switch to X.

LINK-EDIT

At least two statements are required to exit a SUBQUERY. The user does not specify any condition logic or sort statement, but by heading the subquery card, the compiler will create two statements. They are OPCODE 20 and OPCODE 21. If two statements are loaded, go to LINK1. Otherwise, check if E-O-J switch is equal to X, which is the end of the input CONFILE and no more subqueries to be processed. If that is the case, go to GE-EOJ to terminate job. If E-O-J switch is not turned on, go to STEP 3-1 to read another file.

LINK1

A very important function of the retrieval is performed in steps LINK1 through LINK-END. After testing for a statement in the subquery, there is a condition of true or false. This condition determines the next statement to be executed. When the statements are embedded in parentheses and nested parentheses, then operation becomes more complex. This resolution is done at this time and at execution time later on. Load the number of statements which are contained in LL to N, SAVE-N, L7 (1), L7 (2), L7 (11), and L7 (12).

LINK2

If N is less than one go to LINK6, as the first phase of linkage has been completed. The second phase of linkage will be performed in LINK6. Obtain vector from STATE (N) and move to STATE-FORM work area. If OP is equal to thirteen, it is a RECEIVE statement. Subtract one from N and go to LINK2 to obtain a new statement. No modification takes place in this case. ATRUE contains the logical connector of this statement, that is, a one for an IF, an eleven for an OR and if this statement is under parentheses, the level of parentheses is added to the values up to nine level of parentheses. If OP equals twenty-one, it is the end of subquery and this number is placed in SAVE-N. If the operation is a STOP statement, the SAVE-N is computed to its equal with the present subscript plus one, then change this code in ATRUE to a one. Move SAVE-N to ABM. If ATRUE is less than eleven, that is, if the statement has a logical connector of IF or AND, go to LINK4. However, if ATRUE is not less than eleven, subtract ten from I. Finally pick up pointer from L7 (I) and move it to AFALLS field.

LINK3

Move the location of the statement to vector L7 (I) (see step LINK2 for computation of I). Increment I by one. If I is less than twenty-one, go to LINK3 to repeat operation until value is propagated to vector L7 from I position to the 21st position. Reinsert the statement vector back to STATE (N) table. Decrement N by one and go to LINK2 to retrieve the next statement.

LINK4

The pointer for this connector is located in vector L7 (I) and is moved to AFALLS. Decrement I by one and note that ABM contains, at this point, the pointer where the statement is false. If I is greater than zero, the logic connector is under parenthesis and the TRUE NO-GO pointer is located in L7 (I). Move L7 (I) to ABM. Increment I by one.

LINK5

Propagate the NO-GO pointer which is located in the ABM from L7 (I) to this level of parenthesis. When propagation is finished, the vector is inserted back into STATE MATRIX (N). Subtract one from N and go to LINK2 to retrieve a new statement.

LINK6

Move the last subscript of the STATE table to N, SAVE-N, L7 (1), L7 (2), L7 (11), and L7 (12).

LINK7

N, with the value less than one, indicates that all of the statements have been processed. Obtain a vector from STATE (N) and move to the STATE-FORM work area. If OP is equal to eleven, it is a STOP statement. If I is less than eleven, then the statement is an IF or an AND type statement. However, if I is eleven or greater, it is an OR type statement.

LINK8A

Propagate the value of AL through L7.

LINK8

The number of the current statement goes to L7 (I). This number is then propagated through L7 (20).

LINK9

Check for a NO OP statement. If it is a NO OP statement, increment the TRUE and FALLS fields by one.

LINK10

Insure that there is a value in the AFALLS field and restore the statement to the table. If a LOAD statement is encountered, perform the load program section. Once this is done, the LOAD statement is logically deleted by decrementing the subscript.

LINK-END

The S fields, the set table and the set WTVL describe the characteristics of each of the sets within a record. Modify the data control block to identify the file name to be processed as the next data file. Display a start processing message on the console and open the input data file.

RECIN

Read a record. Call a COMPARE routine to initialize the values of the calling sequence areas. Do the same thing for the routine MARING. Shift the L-TAB7 left one character. Call a control field decode routine.

FILE-END

Close the data file and rewind.

NEXT-FILE

If a summary file is being created, close the summary file.

UNLOAD-PG

If any programs were loaded, now is the time to unload them.

UNLOAD

Increment through the object statement table looking for load statements. For each load statement a two in a B1 field indicates that the program is to be unloaded.

STEP100

Determine whether this is the end of job or end of file.

END-TOT

Write out a record containing the query number, subquery number, the number of input records processed and the number of answer records produced.

QSQ SECTION

Initialize the special operator switch.

TEST-FLAG

If flag switches were set, reinitialize them.

UN-FLAG

Restore the flagging indicators.

QSQ-1

Initialize switch ALL.

QSQ-2

Initialize for the next subquery. Move the SET-W-TBL to the SET TABLE area. Q3 (I) contains the subscript of the first statement of this subquery in the statement MATRIX.

CALARASI

When an entry in L-TAB7 is non-zero, the corresponding statement in the STATE-FF will not be executed. It will be replaced by a new statement which is located in STATE-FF at the subscript value of the L-TAB7 entry.

QSQ-3

Get the next statement. Extract from this statement the set numbers of the a-field and b-field. If a logic mode indicates a search all or search terminate type operation, initialize the number of subsets already processed in the S6A field to zero.

QSQ-5

"GO TO depending on" the operator for that particular statement. Check for invalid OP codes.

OP24

A load statement is not executed.

OP01

If number of subsets already processed equals or exceeds the number of subsets in a record for either the a-field or the b-field, the condition is not met. Otherwise, compute the address of the next subset for both the a-field and the b-field. Call a compare routine to determine whether the less than condition is true.

QSQ-2

Initialize for the next subquery. Move the SET-W-TBL to the SET TABLE area. Q3 (I) contains the subscript of the first statement of this subquery in the statement MATRIX.

CALARASI

When an entry in L-TAB7 is non-zero, the corresponding statement in the STATE-FF will not be executed. It will be replaced by a new statement which is located in STATE-FF at the subscript value of the L-TAB7 entry.

QSQ-3

Get the next statement. Extract from this statement the set numbers of the a-field and b-field. If a logic mode indicates a search all or search terminate type operation, initialize the number of subsets already processed in the S6A field to zero.

QSQ-5

"GO TO depending on" the operator for that particular statement. Check for invalid OP codes.

OP24

A load statement is not executed.

OP01

If number of subsets already processed equals or exceeds the number of subsets in a record for either the a-field or the b-field, the condition is not met. Otherwise, compute the address of the next subset for both the a-field and the b-field. Call a compare routine to determine whether the less than condition is true.

OP26

Set up the TEXT-SCAN operator to be treated as a standard EQUALS statement.

OP25

A TEXT-SCAN against a non-variable field is treated as a normal EQUAL operator except that a repetition occurs.

OP02

Check the number of subsets processed and compute the new starting addresses of the active subsets.

OP02R

Call a compare routine. The repeat switch indicates whether the address of the field is to be bumped and the next compare to be executed.

OP03

Check the number of subsets processed. Compute a new subset starting address. Call a compare routine and check for a greater than operator which has a return code of three.

OP04

Check the number of subsets processed. Compute the starting address of the next field. Call a compare routine. If the first half of between operators is OK, get the next statement. Compute the addresses of the next fields. Execute the compare again and check the second half of the between range.

OP05

SATISFIES operator must find an equal condition for the number of fields specified and of the values specified, each match must be a unique match.

OP06

Check the number of subsets processed. Compute the address of the next one. Set up the linkage for the call to the special operator. Provide the address of the record and of the constant pool. Link to the special operator and check the return codes. A return code of one is a hit. A return code of two is a miss. A return code of four is an error. If an error is encountered, adjust the pointer from the false part of the special operator statement to match that of the true. Move an eighteen to the OP code which will NO OP further execution of this statement.

RET-ERROR

The retrieval error paragraphs process an error message returned from the special operator. Provided along with the message are an indication of the record ID, the set number, the subset number, and the position within that subset of the data passed to the special operator which caused the error to occur.

LOAD-PG SECTION

Load the name of the program to be loaded into the calling sequence. Call MARIND which executes a load macro and restore the statement to the table.

COMPARE SECTION

Check the number of subsets processed. Compute the address of the next subset. Call a compare routine.

OP13 SECTION

OP13 is a received statement which accepts the return value from the special operator column. Check the subsets processed and check the length of the value to be returned. Move the data being returned to the record area as specified in the statement.

OP13-C

If the statement indicates the values to be returned to the constant pool, A3 will be greater than three. This move will be accomplished in the OP13-C paragraph.

OP07

Check the number of subsets executed. Compute the address of the next fields. Call a compare routine. If the NOT LESS condition is met, the statement is true.

OP08

Similarly check for a NOT EQUAL condition.

OP09

Check for a NOT GREATER THAN condition.

OP10

Check for a NOT BETWEEN condition.

OP11

If a summary file has been opened, close it. Produce the required counts of records read and accepted.

OP11-1

Once the STOP condition has been met the Q vectors are NO OP'ed.

OP11-2

If there are no more subqueries against this file, close the input data file. If this is the last subquery, go to end of job.

OP12

This is a negation of the special operator.

OP14

For the SELECT statement, determine whether the number of hit subsets matches the number required by this statement.

OP14-1

Loop through and count the number of hits in the subsets.

OP15

Initiate unconditional flagging depending on the logic mode specified in the object statement for either determination by location or value.

OP16

Directive statements are NO OP'ed at this time.

C-YES

Set the appropriate indicators for a true condition. Get the next statement.

C-NO

Set the appropriate indicators for a false statement. Get the next statement.

OP20

OP20 and OP21 are break points for a subquery. A limit check is made to determine the boundary of the statement table has not been exceeded and the flagging routine is executed.

LOGIC-ERROR

If there is an addressing problem on the statement table, an error message will be put out.

AG-1

The value of zero in S7A indicates that there is nothing special to consider on this subset. If S7A is equal to twenty-one, the subset was referenced but the logic was not satisfied.

BGO

Check the I/O switch to see whether there is a file to be written. Q2 is the set number that has a sort flag statement or a merge flag statement.

BG-1

Initialize the processing for a secondary logic.

SQSQ-5

This is a "GO TO depending on" for the secondary conditional statements. The execution of the secondary conditional statements, each paragraph of which is preceded by an SP with the OP code number is an SP24, SP01, SP02 and so on,, is virtually identical with the execution of the primary conditional logic paragraphs. the only real distinction is in the manner in which the S TABLES are set up to indicate the subsets available for processing. This similarity continues down to the WHAT-SOMETHING Section.

WHAT-SOMETHING SECTION

Logic mode of six indicates ANY HIT type logic. Under this condition, set up the address of the next statement for a true condition and loop back to execute it. For a logic mode of seven, the ALL HIT operator, additional passes will have to be made to validate the acceptance of all subsets of that record.

S-NO

The S-NO paragraph and the appendages to the S-NO paragraph process the different variations of a NO HIT on the execution of the statement depending on the logic mode, depending on the flagging mode and the position within that record.

SP17

The merge statement zeros out the subquery number so that when a sort is executed the sort will be performed by query.

SP16

The SP16 paragraphs set up the sort key. Determination is made as to whether the records should be written out. If it is to be written out, all processing has been completed against it prior to the sort statement. If all is satisfactory, then the field specified in the sort statement is moved into the sort key area.

SP14

This processes a select statement. Within paragraph 14-1 the B1 field contains a number of required hits against the subsets. If that number is greater than the number of available subsets, the condition is false.

SP23

This processes a keep statement. In creating a sort flagged answer file the fixed set in any periodic sets that are to be kept are moved to the high order position of the record area. Then each of the subsets within the sort flagged set are added on individually to the end of this fixed portion of the record. By processing in this manner, only the data unique to that particular copy of a record need be moved for each WRITE operation.

SP20

Q8 contains the number of output answer records. Q9 contains the limit of answer records if one was specified. If no limit was specified, the default value is 5,555,555.

SP20-1

Compute the number of segments required to write out this output record onto the answer file.

OUT-WRITE

In this paragraph the answer record is written out as a series of segments. The number of segments is a decrement from the value 99 so that when segment number 99 is reached, the last segment has been written. That is the end of that record.

OP-SUMMARY SECTION

This paragraph is executed if a summary file is being written. This paragraph will write a record onto the summary file. Data record is moved to the summary file buffer and written out.

SUM-OPEN

This paragraph sets up the DCB parameters for opening the output summary file.

SUM-CLOSE

This paragraph closes the summary file.

SHIFT-R SECTION

In this paragraph the data record is shifted so that the sort flagged record contains only the requested sets of data.

SHIFT-RR SECTION

This paragraph moves the segments of the record for the standard answer file to be written out.

SHIFT-R1

Compute the number of segments required for the output answer record.

MOUT-WRITE

Write out the answer record segments. When segment number is equal to 99, the last segment has been written out.

MQSQ

Determine whether another pass is required through the secondary logic.

GE-EOJ

Close the files and return to the supervisor program.

(12) GEN5.

PROCEDURE DIVISION

This is the entry point for the module GEN5. Q-SWT (108) with a value other than zero, indicates that the sort was no good. This is the sort of the answer records with the key as specified in the field FST-18. If an error has been encountered during the execution of this sort routine, the value five will be moved to Q-SWT (108). In addition to this, an error message will be displayed.

SORT OK

This is a normal return point from GEN5 routine.

INSRT SECTION

This opens the input file to the sort.

INSRT1

Read an input record and release that record to the sort routine.

INSRT2

At the end of the file, close the input file.

OUTSRT SECTION

The output file is opened.

OUTSRT1

The records received from the sort routine are written out onto the CARD OUT file.

OUTSRT2

This is a return point from the OUTSRT Section. Notice that the input file and output file are the same for this routine.

PROCEDURE DIVISION

This is the entry point for the GEN5A sort. This sort is a sort of the work file as opposed to the GEN5 sort of the answer record file. Q-SWT (108) is again an error indicator that shows whether or not an error has occurred in the execution of this sort. The call to MARINM sets up the DCB parameters for sorting the work file. If an error is encountered during the execution of the sort, move the value five to Q-SWT (108).

SORTOK

This is a normal return point from the execution of the sort.

INSRT SECTION

Open the input file.

INSRT1

Read an input record. Resequence TYP = 6 (OP Object Vectors) records. Test for FLY-SHEET line cards (TYP = 8). (Note: a FLY-SHEET line card is a data card of the form nnnLØ... or nnnRØ... which normally follows a FLY-SHEET statement.) For error-free (Q-SWT (1) equal Ø) COMP-OP runs (TYPE-S (2) not equal 9) release all line cards to the sort and cause them to be printed (at INSRT 11).

INSRT11

Convert FLY-SHEET line cards to standard report source statements (TYP equal 5) so that they will be printed by GEN6.

INSRT0

Sequence the TYPE 5 statements. Delete the TYPE 2 and TYPE 3 records. Release the remaining records to the sort program.

INSRT2

Close the input file. Check for a nonsequential type report. If the nonsequential switches are on, it indicates that the input X file will be the input work file, for the DCB parameters are modified to accept this file.

INSRT2X

Release nonsequential work file records to the sort.

OUTSRT SECTION

Open the output file from the sort which is the same file as was used for the input file.

OUTSRT1

Receive a record from the sort and write it on the output file.

OUTSRT2

This is a return point for the OUTSRT Section.

GEN5A is identical to the GEN5 sort routine except that the length of the answer records being sorted is different.

(14) GEN6 and GEN6A.

INPUT-OUTPUT SECTION

The MATRIS file is a dummy file used to acquire space for building a matrix. Nothing is written to or read from this file. The answer file contains the sorted answers produced by the retrieval module and as sorted by the GEN5 routine. The GEWORK3 file is used to acquire space to load in the object statements. Like the MATRIS file, there is no input or output on this GEWORK3 file. The S1GEF2 file is the input work file to the output logic processor. The REPORT OP print file is used as the printer output file. PUNCH OP is for a card punch output when required. FILEOUT is for a created output file in the output module with variable length for variable blocked records. FILEOUTF is a similar type file except that the DCB record form on this file is fixed or fixed blocked rather than variable or variable blocked. FILEDIR is used as an input file for source direct processing.

DATA DIVISION

The significant data division entries are as follows.

XMATRIX

This area is used as a save area for the object statement that generates a matrix.

C-X

This is a short table that contains the possible values for the carriage control channel options.

DCB-AREA

This space is used as a work area to modify DCB parameters.

PROGG

This table is used as a set of pointers for performed programs.

LAB-LINES

This area is used to save print lines contained within a LINE paragraph on a page eject so that the labels will be repeated at the top of the next page.

SAVE-LINE

This area is used to save a print line.

STATE-FORM

This area is used as a work area for the object statements. As they are extracted from the table, they are moved to the STATE-FORM area to be operated upon. The contents of the STATE-FORM area and output is the same as it was in retrieval. But the organization of the STATE-FORM object statement is somewhat modified. Notice that A4 and B4 fields are at the end of the statement rather than towards the center.

FILE-LIST

This table is used to save the file names to be processed by a report.

PROGRAM-DATA

This area contains the input data record, the output area, and the constant pool, in that order.

SET-TABLES

This table contains a description of each set within a record. It contains the starting position, the length of the subset, number of subsets and similar information concerning flagging. These values may change between each record and therefore are computed after each record is read.

R-H

This area is used to extract the binary value of a BCD character.

AREA LINKED

This area contains the calling sequence for user routines.

UUSS

This area is used to hold the name of the user-called routine.

PROCEDURE DIVISION

This is the entry point for the module GEN6. Upon entry to this routine the appropriate DCB modifications are made for the input file and the work files. TYPE S (1) equal to zero indicates that there are no answers. If there are answers, open the answer file. Get the first record.

BY-BY-ANSWER

Open the printer file. Eject to the top of a page and print the MIDMS start processing separator page.

PLACE-R

Read the next work file record.

PLACE-RR

If the query number has changed, write a separator page.

FIND1-R

Zero out the W area.

STEP-X

Initialize the S1000 and S10 fields for all possible periodic sets. Initialize the perform program address table.

NEW-RIT

If processing of a report has been completed, release the space occupied by the object statements. PHASE 1 equal to 8 indicates source direct.

START-RIT

At the beginning of each report a separator page is written which contains a source image of the query card for that query or report combination. Q-SWT (QN) equal to one indicates a compilation error and the report will not be executed. If the switch is not equal to one, it means that the execution will proceed. LQL (7) equal to a $ will suppress the printing of the source statements for the query. LQL (8) equal to a $ will suppress the printing of the source statements for the output report. Q-SWT (100) equal to two indicates that the ABORT option is to be exercised. In this case the ABORT message will be printed ten times.

NO-ERRORS

The default value for the carriage control channel is nine, as indicated by TCC-CTL (9). OPCOUNT (QN) contains the number of statements for this report.

PLACE8

Q-SWT equal to eight indicates a source direct process.

NEXT-RIT

Read all the object statements applicable to this report.

SAME-RIT

Within the SAME-RIT the input work file statements are processed according to their type. TYPE 1 statements, or retrieval source statements, are printed out if the printing has not been suppressed. The TYPE 4 and TYPE 5 statements are printed out if the printing of the output source statements has not been suppressed. Q-SWT equal to one indicates that an error has occurred in compilation and execution will not continue. If there has been no error, the TYPE 6 statements are loaded into the statement area. The TYPE 7 statements are loaded into the constant pool and the TYPE 8 statements are processed as fly sheet statements.

PRINT-CARD

This paragraph causes the source card images and error messages to be printed on the output listing. If a fly sheet is to be produced, the output lines are formatted and printed in these paragraphs.

LD-MATRIX

If a matrix is to be created, the appropriate computations are made to determine the size of the required matrix. These values are then loaded into the DCB parameters and the output file matrix is opened which acquires sufficient core in which to build this matrix.

OP48

S7 contains the starting position of the current subset for the periodic set indicated by the A4 or B4 fields in the object statement. The call to MARINZ executes the MATRIX statement.

LOAD-CONST

The constant pool record is loaded into the constant pool area.

LOAD-STATE

The appropriate DCB values are moved into the GEWORK3 DCB area so that space can be acquired for the object statements. SWT-W with a value of zero indicates that the GEWORK3 file is closed. A two is moved to SWT-W just prior to opening and immediately after the open, a one is moved to switch W. This is done to assist in debugging in cases where there is an abnormal termination caused by the open statement.

STEP99

This is a wrap-up paragraph for the printing of a report. Final output is performed, files are closed, and the final separator page is written.

UNLOAD

Upon completion of a report, any subroutines that were loaded by a load statement will now be unloaded. The same call to the subroutine will be used except that the value two is used rather than the value one. The value one causes a subroutines to be loaded; the value two causes a subroutine to be deleted.

STEP100

If a matrix has been created, delete the matrix.

GET-RIT SECTION

If the default value for the trailer channel of nine is being overridden, place the new value into the carriage control area of the output record. If a matrix has been defined, compute and acquire the space required for the matrix. An OP77 indicates a source query type report. If this is the case, the answer file is closed, rewound, and reopened.

FROM-ITS-Q

Determine whether there has been a change in the file name.

GET-RIT2

Convert the object statement fields to binary. Determine whether additional output files are required. For any additional class of output, indicate with an X in the appropriate switch. A logic mode of one indicates headers.

GET-RIT2A

If trailers are to be printed as indicated by a LMODE of two, move the address of that statement to the trailers save area.

GET-RIT2B

Final output indicated by an LMODE of six is indicated by moving the statement address to the final output save area.

GET-RIT2C

If the LMODE is greater than ten, it indicates that a program is being performed within a paragraph.

GET-RIT2D

Load the statement into the table and go back for another.

PASS2

SWTS indicates whether the work file is opened or closed. The value of zero means it is closed, the value of one means that the work file is open. SWTA performs the same function for the answer file. PHASE1 equal to eight indicates a source direct type process.

PASS21

The value of eight indicates source direct. Loop through the query vector looking for a source direct type flag.

PASS22

If the work file is closed, open it.

LINK EDIT

Q-SWT with a value of one indicates that an error has occurred and execution will not take place. However, if Q-SWT has a value of eight, it indicates that source direct processing will occur within this execution of the report. LL indicates the number of statements that exist for this report. For an executable report there must be more than two statements. The call to GEAZM initializes the address areas for the record, the matrix, and the object statement to be executed. The default for PAGESIZE if fifty lines. If it has been modified by user statements, it still may not be less than five. If PUNCHSWT contains an X, that indicates that a card deck is to be produced by this report. If the WRITE-SWT has a value other than spaces, an output file is to be created. (Note: The description for the remainder of this paragraph is IBM 360/ANS COBOL dependent.) Since the user may specify an output file having either a variable (RECFM = V or VB) or fixed (RECFM = F or FB) format, both file formats must be supported by GEN6. However, ANS COBOL (and its associated execute time subroutines) will, under certain conditions, override the user's specifications and, consequently, change the contents of the user's DD in the JFCB during execution of the OPEN. SETXLIST is an ALC subroutine which reads and saves the original contents of the JFCB and, during the OPEN, forces the user specifications into the DCB. By testing RECFM following the OPEN OUTPUT FILEOUT, the user specifications are determined. If the user specified a variable format, FILEOUT will be used; otherwise, FILEOUTF is used.

LOOP-A

This paragraph is executed if a source direct file is to be processed. The appropriate modifications are made to the data control block and the file is opened.

LOOP-AA

The space for saving the continuation label lines is initialized. The value of LL, the number of statements in this report which also is the subscript of the last statement, is moved to N so that the table may be stepped through from bottom to top.

LOOP-B

STATE (J) will be the last object statement in this stack.

LOOP-BB

The a-field and b-field data types are either one or four for alpha or two or five for numeric. Subtracting three from the four or five will convert it to the one or two which still maintains the integrity of the alphanumeric distinction. An LMODE of four indicates a LINE paragraph. The field AL now has a subscript of this last statement.

LOOP-C

Set up the subscripts in the true/false pointers. Check for a DMOVE operator.

LOOP-D

If a TMOVE is encountered, set up the address before this table can be loaded into the work area.

LOOP-L

Search through the list of the convert routine names to find a match for the operator in the TMOVE statement.

LOOP-LL

Move the address of the table to the instruction area. Change the OP code to a fifteen to indicate that this table is core resident.

LOOP-F

This paragraph is executed when a search of the convert routine name table does not result in a match. The failure to find a name in the table indicates the table has not yet been loaded. The paragraph includes a call to the MARINT routine which will issue a load macro and move that loaded table inside the GEN6 working storage area and then delete the loaded copy.

LOOP-DD

Once this table has been loaded, its name is added to the list of loaded tables. A maximum of sixty names may appear on this list.

LOOP-DDD

The object statement, as modified, is returned to the table of statements. An OP38 will cause a program to be loaded. An OP34 indicates the end of the paragraph. In this case the counters are reinitialized.

LOOP-E

If there are more statements to process, go back through the LINK EDIT cycle again.

END-STATE

The value zero in SWT S indicates that the S1GEF4 work file is closed. If there are more than two statements, perform a LINK EDIT process. Produce a separator page and continue.

GET-ANSWER SECTION

Read a segment of an answer record and add one to the count of the segments so that a check can be made to determine whether all sorted segments have been properly read.

ANSWER-END

The value of zero in SWT A indicates that the answer file has been closed.

EJECT-P SECTION

This will eject to the top of a page and print a blank line on the first line of that next page.

PRINT-IMAGE SECTION

This paragraph is used to print the source statements and the error messages, if any. The source listings will print fifty lines on a page.

STEP2 SECTION

Skip to the top of a page.

STEP4

If the input is a source direct file, read a direct file record. If the answer file is closed, go to the end type processing. If the query number of the last read answer is less than the current RIT number, get another answer. The source query statement will cause the answer file to be skipped through until a current query number is matched.

STEP5

If the file name is not changed, continue.

STEP6

If the file changes, find the appropriate file name and list. If K is greater than the FILEN (1), it indicates that the end has been reached.

STEP7

The OP35 is an end of paragraph indicator. Initialize for the processing of the next record and read it depending on whether it is a source direct record or an answer file record.

STEP44

If the answer file has been exhausted, go to the wrap-up section in STEP99.

SEQUENCE-ERROR

If a discrepancy has been encountered in reading the answer file, put out an error message.

STEP8

Initialize processing for the next record.

STEP8-RET

Read additional answer record segments until the entire data record has been built up. When record segment equals ninety-nine, this has occurred.

STEP9

Save the sort key and the subquery number. Identify the address of the first statement to be executed for that file's records.

READ-DIR

Read a record from the source direct file. Since this is a source direct file, there will be no segmentation or building of a record required. Save the file name, any entry point subscripts for the instructions.

STEP9-1

Compute the limits of the instruction addresses for this record.

STEP9-2

This is the number of subsets to be processed.

CALARASI

Get the next statement and check for an end of a paragraph. The end of paragraph is indicated by an OP34 except that if the OP is 34 and a logic mode is four, that indicates the beginning of a LINE paragraph. An OP of less than seven is a conditional operator. A non-conditional operator will cause the next statement to be executed.

MONTANA

This is a "GO TO depending on" paragraph. The OP code in the object statement determines which paragraph is to be executed. There is a distinction between the BPs and OPs. That is, the OPs cause direct execution of a statement based on the contents of the object statement pointers to the a-field and b-field. The BPs have a preprocessing involved to resolve subsets and flagging.

OP38

The OP38 is NO OP'ed at this point since during the LINK EDIT phase the required routine has already been loaded.

OLTENIA

Get the next statement and check for an end of paragraph.

BP44

This is a DSORT operator. Since it involves a sort of periodic subsets, if there is no more than one subset, there is no point in executing a sort and therefore the statement will be NO OP'ed. The field S3 contains the number of subsets within that periodic set.

BP43

Get the next statement. If there are multiple periodic sort statements, they are stacked up. If a different periodic set is involved, however, branch out and execute the previous one.

BP43-1

S1 contains the starting position of the periodic set to be sorted. The call to MARINPS will cause a periodic sort to be executed. R (A1) is the address of the periodic set. S3 (J) is the number of subsets in a record. STATE (I) is the address of the object statement. IX contains the number of sort statements against this periodic set and S2 (J) is the length of the subset.

BP01

If the a-field is not periodic, bypass this processing. The BP01 paragraphs resolve the flagging on the subsets and indicate which subsets are qualified and non-qualified for processing of the statements based on either flagging or on a set selection statement.

BP01-2

Once the subset qualification processing has been completed, the "GO TO depending on" for the execution of the OP code can be accomplished.

BP02

Subscripting at this point will cause a statement to be NO OP'ed. A conditional need not be executed since there is no possible way for the condition to be met. This includes a condition based on the CHANGE/COMPLETE operators.

OP41

This paragraph, with its included call to MARINZ, will resolve the subscript address on the data field.

OP01

For those OP codes executed by the MARINZ routine, the address of the b-field is incremented to indicate the address of the next available periodic subset.

OP14

Since the OP14 and OP15 statements may not have a periodic b-field, they are injected between the increment for the b-field periodic subset updated address in the a-field address updating. The call to MARINZ will execute this statement and return for another statement.

OP12

The OP12 is a CMOVE. Adjust the address to the BSTART field to a base of one. Move the name of the routine into the save area. Compute the starting address of a-field. Move the contents of the a-field to the calling sequence area. Set up the remaining calling sequence parameters. Execute a call to the MARINL routine which will cause the name convert routine to be called and executed. Check for a valid return code from the convert routine and check the length of the value return from that convert routine. Then move that value to the proper b-field location.

OP16

The OP16 is a print statement. If it is found in either the HEADERS or TRAILERS Section, skip ahead. If within the body of the page, check the number of lines printed and if there is space remaining, skip ahead. Otherwise, save the label lines and execute trailers and headers. If the page is exhausted and if it is in the middle of a periodic set processing, eject to the page and put out the continued message.

OP16-A

If there are save label lines, move these out and print them.

OP16-C

Save the print line.

OP16-2

If within a LINE paragraph, add a blank line to the label lines saved if there are less than six.

OP16-16

Print the data line and blank out the print line image. Go get the next statement.

OP17

The PUNCH statement will print a card using the stacker select indicator.

OP18

The OP18 will write a record to the output file. If the file has a record form of variable or variable blocked, the record will be written in this paragraph.

OP18-F

This paragraph is used to write out an output file record that has either a fixed or fixed block format.

OP26

This is a move of variable set data. The field IX contains the subscript of the variable set control field. If the control field is spaces, it means that there is no variable set for this record. If there is a variable set for this record, move the variable set control field to a work area. Replace the value of the set control field in a record with spaces. The control field contains the starting position and the length of the variable field within this record.

OP13-1

Backspace across the area to be printed and check for truncation of the word. Wherever possible only full words will be printed toward the end of the variable print line.

OP13-2

Restore the updated control field to the record area.

OP26-1

Move a segment of the variable field to the b-field area.

OP19

The OP19 is a SKIP statement. It will skip to the bottom of a page assuming that carriage control channel 12 of the value C indicates the bottom of a page on the printer.

OP20

This is a space operator. The ASTART field contains a number of lines to be spaced. If at the bottom of a page, spacing is not necessary, continue.

OP20-1

Loop through writing blank lines with either one, two, or three lines spacing depending on the value of ASTART. Continue looping until the appropriate number of spaces have been generated.

OP21

This is the EJECT operator. The value in B1 indicates that the EJECT is only to be executed if there are B1 lines or less remaining at the bottom of the page.

OP21-1

Execute the EJECT by performing trailers and headers.

OP22

The OP22 is a set selection statement. S6 indicates the first subset of interest, and S5 indicates how many subsets are to be made available.

OP23

This is similar to the OP22 except that the indication in S6 means that only flagged subsets are to be considered.

OP24

The OP24 will select the last N subsets whether flagged or not. If there are fewer actual subsets than are called for in the statement, this statement is synonymous with the OP22; that is, they will be made available. If there are more subsets than are requested, then computation is performed to determine the first desired subset.

OP25

This is similar to the OP24 except that only flagged subsets are to be considered.

OP27

The OMIT statement causes further processing of that record to terminate.

OP28

The STOP statement causes further processing of this file for the report to be terminated. The answer records from this same query number are read and looped through until the end of the answers for this query are reached. At this point the branches are made to STEP99 to wrap up the report.

OP29

An OP29 is the set decode type object statement. One of the set decode statements is generated for each set that is identified in the FFT. This includes the fixed sets and the periodic sets. Begin by initializing the set table work area.

OP29-A

If the set is not periodic, skip ahead. Each entry in a set table is based on the set number as a subscript. S1 contains the high order position of the set. This can vary from record to record. The value of this high order position must be obtained from the set control field which is located within the fixed portion of each record. S2 contains the length of the subset and S3 contains the number of subsets in a record which also must be derived from the periodic set control field information.

OP29-1

Loop through the OP29 statements and update the addressing for each set within this record.

OP33

An OP33 is a PERFORM statement. Decode the number of the program to be performed and the number of times the program is to be performed. Check for a valid program number. Identify the address of the first statement in the program to be performed. Extract that statement. The last statement in a program being performed must also be extracted and the GO TO fields must be updated to indicate the return point from the performed program. If an error condition is encountered, issue an error message and execute a STOP statement so that further processing of the report will not continue.

OP34 SECTION

This is an end of the paragraph. The next paragraph to be executed depends on the logic mode of the current statement.

LINE-START

If the logic mode is four, that indicates that this is the beginning of a LINE paragraph. The periodic set tables are reinitialized so that additional processing can continue without being affected by any previous LINE paragraphs, if any.

LINE-END

Each time a LINE paragraph is executed, the DUPPY-SWT is incremented. If the user invokes an internal loop which could be caused by modification of control fields, he is not permitted to execute a paragraph more than 11,000 times.

OP30-2

At the end of a LINE paragraph, if periodic processing is involved, a determination must be made as to whether that line is to be executed again for the next qualified subset. This determination is made in the OP30- paragraphs.

DUPPY-LOOPY

Issue an error message. If the user is in a loop, stop the execution of this report.

OP36

This is a CHANGE operator. The change condition is true if either this is the first value in a string of values or if this current value is different from the previous value.

OP36-1

Initialize R (BSTART) for a base one type subscript.

OP36-37

If the COMPLETE/CHANGE statement has been found to be true, the current value must now be saved for further comparisons and later executions.

OP37

The COMPLETE operator will not cause a condition to be met on the first occurrence of a data element, otherwise it is identical to the OP36 statement.

OP37-1

Set up for a call to MARINZ with a NOT EQUAL operator to determine whether a change has in fact occurred between this element and a previous element.

OP39

This is a /TOPCAT operator. /TOPCAT will cause the input record to be moved to the output record area and written out as a variable length record.

OP46

This is a TEXT SCAN operator. The pointers in this object statement are initially to the variable set control field rather than to the actual field itself since this is variable from record to record. Once the variable set control field has been moved to a work area the value of set high order position will indicate the number of characters in this variable set. If the value being searched for has a greater length than the value of the variable field, there cannot be a match. A call to MARINZ will execute the TEXT SCAN.

LOAD-PG SECTION

The object statement contains the address of the name of the program to be loaded. This name is moved to the calling sequence area. From there a call is made to the MARIND routine which executes a load macro.

HEADERS SECTION

Initialize the body line count. Check the header switch. If HEAD-S is equal to zero, it means that no headers are included in this report; therefore, bypass. Eject the page and set up the pointers so that the next statement to be executed will be the statements that are logic mode 1 or headers type statements. Upon return from executing the headers statements the saved pointers to the body lines paragraph statements are restored and the section is exited.

WRIT-SEP SECTION

This will write a separator line.

TRAILERS SECTION

The SWT TR equal to spaces indicates the trailers have not yet been executed. During the execution of a report, a normal sequence is to execute trailers and then headers, however, this should not be done on the first page. If the SWT TRAIL-S is equal to zero, it indicates that there is no TRAILERS paragraph for this report and, therefore, the section is bypassed. If there are trailers, set up the address of the TRAILERS paragraph statements, that is logic mode 2 statements. Execute the statements.

TRAIL-RETURN

Upon return from executing the TRAILERS statements, restore the pointers to the statements as they were originally for the printing of the body of the text.

FINAL-OUTPUT SECTION

The FINAL-S switch indicates whether there is a FINAL-OUTPUT Section or not. If there is one, save the pointers to the statements, adjust them to execute the logic mode 6 statements and perform the necessary executions.

FINAL-RETURN

Restore the pointer and exit the processing for this report.

NO-DATA

If there are no answer records and if there are no source direct statements, create MIDMS end-of-processing separator page.

ANSWER-ERR

Once all reports have been published or if there is an error encountered in the answer file, determine that all files are closed and returned to the supervisor.

(15) GEAA.

```
DATETIME START  0
         SAVE   (14,12)
         BALR   10,0            GET SYSTEM DATE
         USING  *,10
         ST     13,SAV+4        SAVE R13
         LA     13,SAV          PUT SA ADD IN R13
         L      2,0(1)          GET ADD OF PARAM LIST
         TIME   DEC
         ST     1,0(2)          STORE PACKED DATE BACK INTO CALLING PROGRAM
         ST     0,4(2)          STORE PACKED TIME BACK
         XC     HOLD1,HOLD1     CLEAR OUT HOLD1
         SR     4,4             CLEAR R4
         LR     5,1             MOVE PACKED DATE INTO R5
         SRA    5,12            SHIFT RIGHT 12 BITS TO DROP DAYS
         O      5,=F'15'        SET PACKED SIGN TO PLUS
         ST     5,HOLD1+4       PUT IT INTO HOLD AREA
         CVB    5,HOLD1         SO THAT YOU CAN CONVERT IT
         UNPK   HOLD1,HOLD1     UNPACK IT FOR OUTPUT
         MVZ    HOLD1+7(1),HOLD1+6  MOVE ZONE OF F
         MVC    13(2,2),HOLD1+6  MOVE IT INTO CALLING PROGRAM
         D      4,=F'4'         DIVIDE BY 4
         CL     4,=F'0'         IS THERE A REMAINDER
         BE     SETLEAP NO REMAINDER- IT IS A LEAP YEAR-
         LA     6,NONLEAP       SET UP ADD OF NONLEAP TABLE
         B      CONT
SETLEAP  LA     6,LEAP          SET UP ADD OF LEAP YEAR TABLE
CONT     N      1,MASK          CLEAR OUT YEAR
         XC     HOLD1,HOLD1
         ST     1,HOLD1+4       MOVE DAY TO HOLD
         CVB    3,HOLD1         INTO R3 AS BINARY
COMPARE  L      4,0(0,6)        LOAD MONTH VALUE FROM TABLE
         CR     3,4             R3 HAS DAYS R4 HAS TABLE VALUE
         BH     NEXT ACTUAL DAYS STILL HIGHER
         S      6,=F'8'         DEC TO GET LAST MONTH
         L      5,0(0,6)        GET LAST MONTH
         SR     3,5             SUB TO GET DAYS OF MONTH
         CVD    3,HOLD1         CVD IT
         UNPK   HOLD1,HOLD1     UNPACK IT
         MVZ    HOLD1+7(1),HOLD1+6
         MVC    8(2,2),HOLD1+6  BACK INTO CALLING PROGRAM
         MVC    10(3,2),12(6)   MOVE FULL ABBR OF MONTH BACK
```

```
         L       13,SAV+4        LOAD R13 BACK
         RETURN  (14,12)
NEXT     A       6,=F'8'         BUMP TO NEXT TABLE VALUE
         B       COMPARE
         CNOP    0,4
SAV      DS      18F
HOLD1    DS      D
MASK     DC      X'0000FFFF'
NONLEAP  DC      F'000'
         DC      CL4'XXX'
         DC      F'031'
         DC      CL4'JAN'
         DC      F'059'
         DC      CL4'FEB'
         DC      F'090'
         DC      CL4'MAR'
         DC      F'120'
         DC      CL4'APR'
         DC      F'151'
         DC      CL4'MAY'
         DC      F'181'
         DC      CL4'JUN'
         DC      F'212'
         DC      CL4'JUL'
         DC      F'243'
         DC      CL4'AUG'
         DC      F'273'
         DC      CL4'SEP'
         DC      F'304'
         DC      CL4'OCT'
         DC      F'334'
         DC      CL4'NOV'
         DC      F'365'
         DC      CL4'DEC'
         DC      F'000'
         DC      CL4'XXX'
         DC      F'031'
         DC      CL4'JAN'
         DC      F'060'
         DC      CL4'FEB'
         DC      F'091'
         DC      CL4'MAR'
         DC      F'121'
         DC      CL4'APR'
         DC      F'152'
         DC      CL4'MAY'
         DC      F'182'
         DC      CL4'JUN'
```

```
DC      F'213'
DC      CL4'JUL'
DC      F'244'
DC      CL4'AUG'
DC      F'274'
DC      CL4'SEP'
DC      F'305'
DC      CL4'OCT'
DC      F'335'
DC      CL4'NOV'
DC      F'366'
DC      CL4'DEC'
END     DATETIME
```

(16) GEAB.

```
MARINB    START  0
          SAVE   (14,12)
          BALR   10,0           DETERMINE EXISTENCE OF LOAD MODULE
          USING  *,10
          LR     3,1            ADDRESS OF ADDRESS OF NAME
          L      2,0(3)         LOAD 2 WITH ADDRESS OF NAME
          MVC    NAMEADDR(8),2(2)  MOVE NAME TO SAVE AREA
          ST     13,REG13       SAVE BACK ADDRESS
          LA     13,SREG        LOAD ADDRESS OF CURRENT SAVE AREA
          BLDL   0,LISTADDR     EXECUTE BLDL MACRO
          L      13,REG13       RESTORE BACK ADDRESS
          L      2,4(3)         LOAD ADDRESS OF RETURN CODE FIELD
          STH    15,0(2)        STORE BLDL RETURN CODE IN CALLING PROGRAM
          RETURN (14,12),T      GO BACK
          DS     0F
REG13     DS     1F
SREG      DS     18F
LISTADDR  DC     X'0001'
          DC     X'0040'
NAMEADDR  DS     CL8
          DS     28H
          END    MARINB
```

(17) GEAC.

```
MARINCTL START  0
         SAVE   (2,5)
         BALR   15,0          DECODE PERIODIC SET CONTROL WORDS,
         USING  *,15          CONVERT TO BINARY AND LOAD IN SET-W-TBL.
         LM     2,4,0(1)      PICK UP ADDRESS OF RX,SET-W-TBL,SL.
         LH     4,0(4)        LOAD VALUE OF SL IN REG. 4.
         AH     2,2(3)        COMPUTE ADDRESS OF FIRST PSC.
         BCT    4,LOOP        CHECK TO SEE IF FILE HAS ANY PERIODICS.
         B      RETX          IF NO RETURN TO THE CALLING PROGRAM.
LOOP     LA     3,12(3)       LOAD ADDRESS OF NEXT ROW IN SET MATRIX.
         PACK   FLDA,0(4,2)   PACK FIRST HALF OF PSC, NUMBER OF SUBSETS
         CLI    FLDA+7,X'04'  CHECK FOR BLANK
         BNE    AOK           IF NOT, SKIP AHEAD
         MVI    FLDA+7,X'0F'  REPLACE BLANK WITH ZERO
AOK      CVB    5,FLDA        CONVERT TO BINARY, NUMBER OS SUBSETS
         STH    5,4(0,3)      STORE RESULT IN THIRD COLUMN OF MATRIX
         PACK   FLDA,4,(4,2)  PACK SECOND HALF OF PSC, STARTING POSITION
         CLI    FLDA+7,X'04'  CHECK FOR BLANK
         BNE    BOK           IF NOT, SKIP AHEAD
         MVI    FLDA+7,X'0F'  REPLACE BLANK WITH ZERO
BOK      CVB    5,FLDA        CONVERT TO BINARY
         STH    5,0(0,3)      STORE RESULT IN FIRST COLUMN OF MATRIX
         LA     2,8(0,2)      COMPUTE ADDRESS OF THE NEXT PSC.
         BCT    4,LOOP        IF THERE IS ONE BRANCH TO LOOP
RETX     RETURN (2,5),T       GO BACK
         DS     0D
FLDA     DS     CL8
         END    MARINCTL
```

(18) GEAD.

```
MARIND   START  0
         SAVE   (14,12)
         BALR   10,0          LOAD OR DELETE LOAD MODULE
         USING  *,10
         L      12,4(1)       LOAD ADDRESS OF REGISTER SAVE AREA
         ST     13,4(12)      SAVE RETURN POINT IN SAVE AREA
         ST     12,8(13)      SAVE ADDRESS OF SAVE AREA IN CALLING PROGRAM
         L      3,8(1)        LOAD ADDRESS OF LOAD/DELETE SWITCH
         L      3,0(3)        LOAD VALUE OF LOAD/DELETE SWITCH
         L      2,0(1)        LOAD NAME OF SUBROUTINE
         BCT    3,DELX        IF LOAD/DELETE SWT = 1, DO A LOAD,
                                OTHERWISE DELETE
LOADX    LOAD   EPLOC=(2)     EXECUTE LOAD MACRO
         B      RETX          DONE
DELX     DELETE EPLOC=(2)     EXECUTE DELETE MACRO
RETX     L      13,4(12)      RESTORE BACK CHAIN
         RETURN (14,12),T     GO BACK
         END    MARIND
```

(18.1) GEAE.

```
SETXLIST  OSHK                                SET UP SAVE/RESTORE
          USING   IHADCB,R2
          L       R2,0(R1)                    GET ADDRESS OF DCB (FIXED)
          MVC     FXLSAVE,DCBEXLST+1          SAVE COBOL EXIT LIST ADDRESS
          MVC     DCBEXLST+1(3),LISTADDR      SET EXLIST TO DDOVRIDE
          L       R2,4(R1)                    GET ADDRESS OF DCB (VARIABLE)
          LA      R2,0(R2)                    CLEAR HIGH ORDER BYTE
          ST      R2,DCBADDR                  SAVE ADDRESS OF VARIABLE DCB
          MVC     RJFDCBAD(3),DCBADDR+1       MOVE DCB ADDR TO RDJFCB PLIST
          MVC     VXLSAVE,DCBEXLST+1          SAVE COBOL EXIT LIST ADDR
          MVC     DCBEXLST+1(3),JFCBLADR      STORE RDJFCB EXLIST ADDR
          STM     14,12,12(13)                STORE REGISTERS
          CNOP    0,4
          BAL     1,*+8                       BRANCH AROUND PARAM
          DC      AL1(128)                    TO SET HIGH ORDER BIT ON
RJFDCBAD  DS      AL3                         AREA TO HOLD DCB ADDRESS
          SVC     64                          RDJFCB SVC
          LM      14,12,12(13)                RESTORE REGISTERS
          MVC     DCBEXLST+1(3),LISTADDR      SET EXIT LIST ADDR TO DDOVRIDE
          B       OKRET                       RETURN
          DROP    R2,R13                      THE EXLST SET ABOVE WILL CAUSE
          USING   *,R15                       /FOLLOWING CODE TO BE EXECUTED
          USING   IEFJFCBN,R4                 /DURING THE OPEN
          USING   IHADCB,R1                   R1 HAS DCB BASE ADDRESS
DDOVRIDE  LA      R4,JFCBAREA                 LOAD R4 WITH JFCB ADDRESS
          MVC     DCBBUFL(2),JFCBUFL          SET BUFL
          MVC     DCBRECFM(1),JFCRECFM        SET RECFM
          MVC     DCBBLKSI(2),JFCBLKSI        SET BLKSIZE
          MVC     DCBLRECL(2),JFCLRECL        SET LRECL
          LA      R1,0(R1)                    CLEAR HIGH ORDER BYTE
          C       R1,DCBADDR                  IS VARIABLE FILE BEING OPENED
          BE      VARDCB                      BRANCH IF YES
          MVC     DCBEXLST+1(3),FXLSAVE       RESTORE EXLIST ADDR (FIXED)
          BR      R14                         RETURN
VARDCB    MVC     DCBEXLST+1(3),VXLSAVE       RESTORE EXLIST ADDR (VARIABLE)
          BR      R14                         RETURN
          LTORG
          DS      0D
DCBADDR   DS      F                           DCB ADDR (FOR COMPARE)
OVEXLIST  DC      X'85'                       DCB EXLIST OPTION FLAG
          DC      AL3(DDOVRIDE)               ADDRESS OF DCB EXIT ROUTINE
JFCBLIST  DC      X'87'                       RDJFCB EXLIST OPTION FLAG
          DC      AL3(JFCBAREA)               ADDR OF JFCB AREA
JFCBAREA  DS      44F                         44 WORD JFCB HOLD AREA
LISTADDR  DC      AL3(OVEXLIST)               DCB EXLIST ADDRESS
JFCBLADR  DC      AL3(JFCBLIST)               JFCB READ EXLIST ADDRESS
```

(18.1) GEAE. (CONTINUED)

```
VXLSAVE     DS      AL3                     SAVE COBOL EXLIST (FIXED)
FXLSAVE     DS      AL3                     SAVE COBOL EXLIST (VARIABLE)
IEFJFCBN    DSECT
            IEFJFCBN                        DSECT FOR JFCB
            DCBD    DSORG=PS,DEVD=TA        DSECT FOR DCB
            END
```

(19) GEAG.

```
GEAGM    CSECT                 ENTRY POINT FOR INITIALIZATION
         BALR   15,0
         USING  *,15           SET ANALYSIS ROUTINE
         MVC    SAVEADDR(32),0(1)  INITIALIZE BY SAVING ADDRESSES
                                      OF PARAMETERS
         BR     14             RX,SET-TABLE,SL,ABM,FLAG-AREA,OP,SWTGO
MARINGO  SAVE   (14,12)        ENTRY POINT FOR EXECUTION
         ENTRY  MARINGO
         BALR   15,0
         USING  *,15
         LM     2,8,SAVEADDR   PICK UP ADDRESSES
         LH     1,0(4)         PICK UP NUMBER OF DATA SETS IN FILE
         CLI    1(7),X'15'     IF SUBQUERY LOGIC WAS FALSE
         BE     OP21           GO TO OP21
         AH     6,0(5)         POSITION FLAG-AREA ADDRESS WITH
         AH     6,0(5)         THE VALUE OF ABM
LOOP1    MVI    11(3),X'00'    MOVE ZERO TO S7 - SET STATUS
         BCT    1,LOOP2        BRANCH IF A PSET REMAINS TO BE PROCESSED
         B      RETX           OTHERWISE RETURN
LOOP2    LA     3,12(3)        COMPUTE ADDRESS OF PSET ROW IN MATRIX
         CLI    11(3),X'00'    IF THERE WAS NO SEARCH ON SET THAN
         BE     LOOP1          GO TO GET NEXT DATA SET.
         CLI    11(3),X'01'    IF TYPE OF SEARCH IS SEARCH
         BE     G01            BRANCH TO G01
         CLI    11(3),X'02'    IF TYPE OF SEARCH IS SEARCH-ALL
         BE     G02            BRANCH TO G02
         CLI    11(3),X'03'    IF TYPE OF SEARCH IS SEARCH-TERMINATE
         BE     G03            BRANCH TO G03
         CLI    11(3),X'15'    IF SET WAS ACTIVE BUT LOGIC WAS FALSE
         BE     G021           GO TO GET NEW SUBSET.
         B      LOOP1          ANY OTHER STATUS IS NOT SIGNIFICANT HERE
G02      BAL    14,FLAG        LOAD RETURN ADDRESS AND GO TO FLAG
G021     LH     7,8(3)         PICK UP ACTIVE SUBSET NUMBER
         LA     7,1(7)         ADD 1 TO IT
         STH    7,8(3)         STORE IT IN S6
         CH     7,4(3)         IF THERE IS ONE MORE SUBSET
         BL     SWTGO          BRANCH TO SWTGO
         B      LOOP1
SWTGO    MVI    1(8),X'01'     INDICATE PROCESSING NOT COMPLETE
         B      LOOP1
G03      CLI    7(3),X'00'     IF PSET NOT PREVIOUSLY FLAGGED
         BE     G02            BRANCH TO DO IT NOW
         CLI    9(3),X'00'     IF THIS IS THE FIRST SUBSET
         BE     G02            BRANCH TO FLAG IT, OTHERWISE
         LH     7,8(3)         PICK UP SUBSET NUMBER
         BCTR   7,0            COMPUTE ADDRESS OF PREVIOUS SUBSET
```

```
         MH      7,2(3)          FOR SEARCH-TERMINATE LOGIC
         AH      7,0(3)
         AR      7,2
         CLI     0(7),X'F0'      IF THE PREVIOUS SUBSET IS FLAGGED
         BL      GO2             BRANCH TO THIS ONE ALSO
         B       LOOP1
GO1      CLI     7(3),X'00'      IF THE PSET IS ALREADY FLAGGED ONCE
         BNE     LOOP1           DO NOT FLAG IT AGAIN, OTHERWISE
         LA      14,LOOP1        LOAD RETURN ADDRESS TO LOOP1 AND PROCEED.
FLAG     LH      7,2(3)          PICK UP LENGTH OF SUBSET IN THIS PSET, AND
         MH      7,8(3)          MULTIPLY IT BY NO. OF PRECEDING SUBSETS, AND
         AH      7,0(3)          ADD TO IT THE STARTING POSITION OF PSET
         LA      7,1(0,7)
         STH     7,0(6)          SAVE RELATIVE ADDRESS OF FLAG IN BUFFER
         BCTR    7,0
         LA      6,2(6)          POSITION SAVE AREA FOR NEXT FLAG.
         AR      7,2             AND FINALLY UPSET IT BY ADDRESS OF IN-BUFFER
         OI      0(7),X'C0'      MASK OUT STRAY ZONE BITS
         NI      0(7),X'CF'      INSERT FLAG IN FIRST POSITION OF SUBSET
         MVI     7(3),X'01'      INSERT FLAG IN S5 - PSET FLAG STATUS
         LH      7,0(5)          PICK UP VALUE OF ABM
         LA      7,1(7)          ADD 1 TO IT AND
         STH     7,0(5)          STORE IT IN ABM - NUMBER OF FLAGS IN BUFFER
         BR      14              RETURN VIA 14
OP21     MVI     11(3),X'00'     MOVE ZERO TO SET STATUS
         BCT     1,LOOP3         BRANCH IF A PSET REMAINS TO BE CHECKED
         B       RETX            OTHERWISE RETURN
LOOP3    LA      3,12(3)         COMPUTE ADDRESS OF PSET ROW IN SET MATRIX
         CLI     11(3),X'15'     IF SUBSET DID NOT NEGATE THE LOGIC
         BNE     OP21            BRANCH TO CHECK NEXT PSET, OTHERWISE
         LH      7,8(3)          ADD 1 TO ACTIVE SUBSET NUMBER
         LA      7,1(7)
         STH     7,8(3)
         CH      7,4(3)          IF SUCH A SUBSET IS PRESENT THEN
         BL      LOOP4           BRANCH TO LOOP4, OTHERWISE
         B       OP21            FORGET IT
LOOP4    MVI     1(8),X'01'      INDICATE THAT FALSE LOGIC IS NOT EXCLUSIVE
         B       OP21
RETX     RETURN  (14,12),T
SAVEADDR DS      16F
         END     MARINGO
```

(20) GEAL.

```
MARINL   START    0
         SAVE     (14,12)        USER MAY ENTER THIS ROUTINE AT WILL.
         BALR     10,0
         USING    *,10
         L        12,4(1)        18 WORD AREA PROVIDED BY USER AS
                                 SECOND PARAM.
         ST       13,4(12)       SAVE BACK CHAIN ADDRESS
         ST       12,8(13)       SAVE ADDRESS OF USER SAVE AREA
         L        2,0(1)         ADDRESS OF CALLED PROGRAM NAME,
                                 FIRST PARAM.
         LR       13,12          ADDRESS OF SAVE AREA TO BE PASSED ON.
         LA       1,8(1)         THIRD PARAM TO BE PASSED ON AS
                                 ADDRESS OF LIST
         LINK     EPLOC=(2)      EXECUTE SUBROUTINE
         L        13,4(12)       LOGIC RETURN CHAIN
         L        14,12(3)       RESTORE R14
         RETURN   (0,12),T       GO BACK AND PRESERVE RETURN CODE
                                 IN R15
         END      MARINL
```

(21) GEAM.

```
MARINM   START  0
         SAVE   (2,7)
         BALR   15,0          MOVE LOGICAL STRING
         USING  *,15
         LM     2,6,0(1)      ADDRESSES OF SENDING FLD, SUBSCRIPT, REC. FLD.,
         LH     6,0(6)        SUBSCRIPT, LENGTH - LOAD LENGTH
         BCTR   6,0           SUBTRACT 1 FROM LENGTH
         AH     2,0(3)        BASE + DISPLACEMENT OF SENDING FIELD
         BCTR   2,0           SET TO BASE 0
         AH     4,0(5)        BASE + DISPLACEMENT OF RECEIVING FIELD
         BCTR   4,0           SET TO BASE 0
MOVEEX   EX     6,MOVE        MAX MOVE IS 256 ON ONE EXECUTE MVC
         SRDL   6,8           SHIFT TO SUBTRACT NUMBER MOVED IN REG6
         SLA    6,8           AND SAVE THAT NUMBER IN REG7
         BZ     RETM          DONE
         SRL    7,24          SLIDE NUMBER TO LOW END OF REGISTER
         LA     2,1(7,2)      ADJUST NEW SENDING ADDRESS
         LA     4,1(7,4)      ADJUST NEW RECEIVING ADDRESS
         BCT    6,MOVEEX      IF ADJUSTED  LENGTH   1 LOOP BACK
MOVE     MVC    0(0,4),0(2)   MVC WITH 0 LENGTH WILL MOVE 1
RETM     RETURN (2,7),T       GO BACK
         END    MARINM
```

(22) GEAN.

```
MARINN   START 0              TRANSFERS 8 CHARS TO DCB OF A DIRECT FILE
         SAVE  (2,4)          CALL SEQUENCE CHARS, FILENAME
         DS    0H
         STM   2,4,28(13)     SAVE REGISTERS
         BALR  2,0            SET UP BASE REGISTER
         USING *,2
         L     3,0(1)         LOAD REGISTER WITH ADDRESS OF DDNAME
         L     4,4(1)         LOAD REGISTER WITH ADDRESS OF DEC
         L     4,8(4)         RELOAD REGISTER WITH DCB ADDRESS
         MVC   40(8,4),0(3)   MOVE DDNAME TO DCB DDNAME
         RETURN(2,4),T        RETURN
         LM    2,4,28(13)     RESTORE THE REGISTERS
         MVI   12(13),X'FF'   SET RETURN INDICATION
         BR    14             RETURN
         END   MARINN
```

(23) GEAP.

```
MARINPS  START  0
         SAVE   (14,12)        CALLING SEQUENCE IS AS FOLLOWS
                               R(S1(A4)),S3(A4),STATE(N),NFIELDS,S2(A4)
                               THIS SORT PROGRAM IS A REENTRANT SUBROUTINE
                               USE OF GENERAL REGISTERS IS AS FOLLOWS
                                0  NUMBER OF SORT FIELDS
                                1  LENGTH OF SUBSET
                                2  ADDRESS OF THE PERIODIC SET
                                3  NUMBER OF SUBSETS IN THIS SET
                                4  ADDRESS OF THE FIRST SORT STATEMENT
                                5  NUMBER OF SORT FIELDS
                                6  LENGTH OF SUBSET
                                7  NUMBER OF SUBSETS
                                8  ADDRESS OF A-SUBSET
                                9  ADDRESS OF B-SUBSET
                               10  ADDRESS OF SORT STATEMENT
                               11  ADDRESS OF A-SORT-FIELD
                               12  ADDRESS OF B-SORT-FIELD
                               13  ADDRESS OF SAVE AREA FROM CALLING PGM
                               14  LENGTH OF SORT FIELD
                               15  BASE REGISTER, BAD NEWS
                                   CALLING SEQUENCE PARAMETERS ARE
                                    1  ADDRESS OF PERIODIC SET-1
                                    2  ADDRESS OF NUMBER OF SUBSETS
                                    3  ADDRESS OF FIRST SORT FIELD PARMS
                                    4  ADDRESS OF NUMBER OF FIELDS TO SORT
                                    5  ADDRESS OF SUBSET LENGTH
         BALR   15,0
         USING  *,,15
         LM     2,6,0(1)       LOAD ADDRESSES OF PARAMETERS
         LH     3,0(0,3)       NUMBER OF SUBSETS
         LA     2,1(0,2)       ADDRESS OF SET BUMPED TO BASE 1 FROM BASE 0
         LH     5,0(0,5)       NUMBER OF FIELDS TO SORT
         LH     6,0(0,6)       LENGTH OF SUBSET
         LR     0,5            NUMBER OF FIELDS TO SORT
         LR     1,6            LENGTH OF SUBSET
PS040    LR     7,3            NUMBER OF SUBSETS
         BCTR   7,0            SUBTRACT 1
         LR     8,2            ADDRESS OF A-SUBSET
         LR     9,2            ADDRESS OF B-SUBSET IS ADDRESS OF A-SUBSET
         AR     9,6            PLUS SUBSET LENGTH, OR NEXT SUBSET
PS050    BCT    3,PS100        IF THERE ARE SUBSETS, CONTINUE, OTHERWISE
         B      PSEXIT         GO TO EXIT
```

```
PS100    LR      10,4           ADDRESS OF SORT STATEMENT
PS110    LH      11,0(0,10)     FIELD A1 OF STATE (START POS OF REC)
         LR      12,11          IS BASE ADDRESS FOR SORT FIELD
         AR      11,8           INCREMENT BASE BY SUBSET POSITION TO
         AR      12,9           GET ACTUAL SUBSET ADDRESSES
         LH      14,2(0,10)     FIELD A2 OF STATE (LENGTH OF SORT KEY FIELD)
         EX      14,COMPX       COMPARE 1ST SORT FIELD WITH 2ND SORT FIELD (A-B)
         BE      PS400          CHECK FOR ANOTHER SORT FIELD
         BL      PS150          IF LOW, SUBSET FIELDS ARE IN ASCENDING SEQUENCE
         CLI     7(10),X'2C'    IS DESCENDING SORT CALLED FOR (OP44)?
         BE      PS200          IF SO, SUBSETS ARE OK
         B       PS160          CONTINUE
PS150    CLI     7(10),X'2B'    IS ASCENDING SORT CALLED FOR (OP43)?
         BE      PS200          SUBSETS ARE OK
PS160    LR      8,9            SHIFT B-SUBSET TO A-SUBSET
PS200    AR      9,6            COMPUTE NEXT B-SUBSET
         BCT     7,PS100        IF MORE SUBSETS, LOOP BACK
         CR      2,8            HAS A-SUBSET BEEN INCREMENTED
         BE      PS300          IF NOT, NO FLIP-FLOP NEEDED
         LR      9,2            SET UP FOR FLIP-FLOP
PS222    BCTR    6,0            SET LENGTH TO BASE 0
         EX      6,MVC1         EXCLUSIVE OR'S ACCOMPLISH
         EX      6,MVC2         THE FLIP-FLOP OF THE
         EX      6,MVC3         SUBSETS IN PLACE
         SRDL    6,8            SUBTRACT NUMBER OF BYTES OR'D AND SAVE IN REG7
         SLA     6,8            LO ORDER 8 BITS ARE WIPED OUT
         BZ      PS300          IF NOT 0 THE SUBSET IS LONGER THAN 256
         SRL     7,24           SET UP FOR NEXT OR PASS
         LA      8,1(7,8)       COMPUTE NEW A-SUBSET STARTING POSITION
         LA      9,1(7,9)       COMPUTE NEW B-SUBSET STARTING POSITION
         B       PS222          DO IT AGAIN
MCV1     XC      0(0,8),0(9)    THESE THREE EXCLUSIVE OR'S WILL
MVC2     XC      0(0,9),0(8)    SWAP UP TO 256 BYTES
MVC3     XC      0(0,8),0(9)    OF DATA WITHOUT A BUFFER
COMPX    CLC     0(0,11),0(12)  SORT KEY COMPARE
PS300    LR      6,1            RESTORE LENGTH OF SUBSET
         AR      2,1            INCREMENT TO NEXT SUBSET
         B       PS040
PS400    LA      10,24(0,10)    LOCATION OF NEXT SORT STATEMENT
         BCT     5,PS110        IF ANOTHER SORT FIELD EXISTS, PROCESS IT
PS410    LR      5,0            RESTORE SORT STATEMENT COUNT
         B       PS200          PROCESS NEXT SUBSET
PSEXIT   RETURN  (14,12),T      RETURN
         END     MARINPS
```

(24) GEAS.

```
MARINS   START  0
         SAVE   (1,5)          BUILD SORT KEY AND INVERT IF CALLED FOR
         BALR   15,0
         USING  *,15
         LM     2,5,0(1)       RX,CONX,STATE-FORM,KSRT
         LH     1,0(4)         BASE ADDRESS OF SORT FIELD (A1)
         CLI    5(4),X'04'     A3 FIELD OF STATEMENT (FIELD TYPE)
         BL     AREC           IF LESS THAN 4, FIELD IS FROM RECORD
         AR     1,3            SORT FIELD IS FROM CONSTANT POOL.
         B      ADDR
AREC     AR     1,2            SORT FIELD IS FROM RECORD BUFFER POOL
ADDR     LH     2,10(4)        SORT KEY BASE ADDRESS (B1)
         LA     5,0(2,5)       ACTUAL SORT KEY ADDRESS
         LH     2,2(0,4)       PICK UP LENGTH OF FIELD
         BCTR   2,0            SUBTRACT 1 TO ADJUST TO BASE 0
         EX     2,MOVEX        MOVE FIELD TO SORT KEY
         CLI    19(4),X'0A'    SORT/MERGE DESCENDING
         BE     INVERT         GO TO INVERT DATA
         CLI    19(4),X'0C'    IF SORT/MERGE FLAGGED DESCENDING
         BE     INVERT         GO TO INVERT DATA
         B      RETX
MOVEX    MVC    0(0,5),0(1)    ACTUAL MOVE OF FIELD IN SORT KEY
TRANS    TR     0(0,5),TBL     ACTUAL INVERSION BY BINARY TRANSLATION
INVERT   EX     2,TRANS        INVERT DATA
RETX     RETURN (1,5),T
TBLE     DC     X'FFFEFDFCFBFAF9F8F7F6F5F4F3F2F1F0'
         DC     X'EFEEEDECEBEAE9E8E7E6E5E4E3E2E1E0'
         DC     X'DFDEDDDCDBDAD9D8D7D6D5D4D3D2D1D0'
         DC     X'CFCECDCCCBCAC9C8C7C6C5C4C3C2C1C0'
         DC     X'BFBEBDBCBBBAB9B8B7B6B5B4B3B2B1B0'
         DC     X'AFAEADACABAAA9A8A7A6A5A4A3A2A1A0'
         DC     X'9F9E9D9C9B9A99989796959493929190'
         DC     X'8F8E8D8C8B8A898887868584838281 80'
         DC     X'7F7E7D7C7B7A79787776757473727170'
         DC     X'6F6E6D6C6B6A69686766656463626160'
         DC     X'5F5E5D5C5B5A59585756555453525150'
         DC     X'4F4E4D4C4B4A49484746454443424140'
         DC     X'3F3E3D3C3B3A39383736353433323130'
         DC     X'2F2E2D2C2B2A29282726252423222120'
         DC     X'1F1E1D1C1B1A19181716151413121110'
         DC     X'0F0E0D0C0B0A09080706050403020100'
         END    MARINS
```

(25) GEAT.

```
MARINT   START  0              DYNAMICALLY LOAD TABLE INTO
         SAVE   (14,12)        WORKING STORAGE
         LM     2,6,0(1)       BASE ADDR OF REC AREA, DISP OF REC AREA, SIZE
                                 OF REC AREA, ADDRESS
         BALR   12,0           OF STATEMENT, ADDRESS OF RECORD
         USING  *,12
         LH     8,8(0,5)       RELATIVE ADDR OF BFLD (B1)
         AR     8,6            ACTUAL ADDR OF BFLD
         MVC    TADDR(12),0(8) GET NAME TABLE AND NEW ADDRS
         L      11,TADDR       ADDRESS OF NAME
         ST     13,REG13       SAVE BACK - CHAIN
         LA     13,MASK        ADDRESS OF SAVE AREA FOR REGISTERS
         LA     0,TNAME        NAME OF TABLE TO BE CALLED
         LOAD   EPLOC=(0)      GET TABLE
         LR     13,0           ADDRS OF TABLE
         LH     7,0(13)        LENGTH OF FIRST PARAM (LENGTH OF ARGUMENT)
         AH     7,2(13)        LENGTH OF SECOND PARAM (LENGTH OF FUNCTION)
         MH     7,4(13)        COMPUTE LENGTH OF TABLE (NO OF OCCURRENCES)
         LA     7,6(7)         ADD LENGTH OF PARAMS TO GET TOTAL TABLE LENGTH
         C      7,0(4)         IS THERE ROOM FOR TABLE IN WORKING STORAGE
         BH     RETX           VERY SORRY....
         MVC    16(2,5),10(5)  SAVE BLENGTH IN NOGO
         STH    11,14(0,5)     SAVE ADDRESS OF NAME
         A      2,0(3)         COMPUTE ACTUAL STORAGE ADDRESS
         ST     2,8(5)         LOAD IT IN BSTART OF STATE-FORM
         L      8,0(4)         LENGTH OF STORAGE PASSED AS NOT USED
         SR     8,7            NEW AMOUNT OF STORAGE FREE
         ST     8,0(4)         RECORD IT IN LENGTH PARAM
         LR     8,7            LENGTH OF TABLE
RETM     BCTR   8,0            SUBTRACT 1 TO ADJUST TO BASE 0
         EX     8,MOVEX        MOVE TABLE IN WORKING STORAGE
         SRDL   8,8            SHIFT RIGHT THE LENGTH MOVED
         SLA    8,8            REPOSITION THE LENGTH LEFT
         BZ     DONE           BRANCH IF TABLE MOVED
         SRL    9,24           COMPUTE LENGTH MOVED
         LA     2,1(9,2)       RECOMPUTE RECEIVING ADDRESS
         LA     13,1(9,13)     RECOMPUTE SENDING ADDRESS
         BCT    8,RETM         BRANCH IF MORE THAN ONE BYTE
MOVEX    MVC    0(0,2),0(13)   ACTUAL MOVE INSTRUCTION
DONE     MVI    7(5),X'0F'     SET NEW OP CODE FOR MACRO **15**
RETX     LA     13,MASK        PROVIDE SAVE AREA FOR MACRO CALL
         LA     0,TNAME        LOAD ADDRESS OF NAME TABLE
         DELETE EPLOC=(0)      WIPE OUT LOADED COPY OF TABLE
         L      13,REG13       RESET REGISTER 13
         RETURN (14,12),T      GO BACK
         DS     0D
         DS     CL4
TADDR    DS     CL4
TNAME    DS     CL8
MASK     DS     20F
REG13    DS     2F
         END    MARINT
```

(26) GEAX.

```
GEAXM    CSECT                    COMPARE ROUTINE
         SAVE    (1,7)
         BALR    15,0             INITIALIZATION SECTION
         USING   *,15
         LM      2,5,0(1)         RX, CONX, STATE-FORM, N
         STM     2,5,SAVEADDR     SAVE PARAMETER ADDRESS
         RETURN  (1,7),T          GO BACK
MARINX   SAVE    (1,7)            EXECUTION SECTION
         ENTRY   MARINX
         BALR    15,0
         USING   *,15
         LM      2,4,SAVEADDR     ADDRESSES OF PARAMETERS
         LH      5,4(4)           LOAD TYPE OF A-FIELD (FIELD A3)
         LH      6,14(4)          LOAD TYPE OF B-FIELD (FIELD B3)
         LH      1,0(4)           STARTING POSITION OF A-FIELD (A1)
         LH      7,10(4)          STARTING POSITION OF B-FIELD (B1)
         CLI     5(4),X'04'       IF FIELD A3 (TYPE) IS LESS THAN 4,
         BL      AREC             THE A-FIELD IS FROM THE RECORD.
         AR      1,3              A-FIELD IS IN CONSTANT POOL
         BCTR    5,0              SUBTRACT 3 FROM THE A-FIELD TYPE
         BCTR    5,0              TO MAKE TYPE EQUAL TO 1 (ALPHA)
         BCTR    5,0              OR 2 (NUMERIC)
         B       ADDRS
AREC     AR      1,2              A-FIELD IS IN RECORD BUFFER POOL
ADDRS    CLI     15(4),X'04'      IF FIELD B3 (TYPE) IS LESS THAN 4,
         BL      BREC             B-FIELD IS IN RECORD BUFFER POOL
         LR      2,3              B-FIELD IS IN CONSTANT POOL
         BCTR    6,0              SUBTRACT 3 FROM TYPE TO MAKE
         BCTR    6,0              TYPE EQUAL TO 1 (ALPHA) OR
         BCTR    6,0              2 (NUMERIC)
BREC     AR      2,7              COMPUTE ABSOLUTE ADDRESS OF B-FIELD
         CLI     9(4),X'19'       OP 24 IS A CONTAINS OPERATOR
         BE      TEXTSC           GO SET UP FOR TEXT SCAN
         CLI     9(4),X'1A'       OP 25 IS A CONTAINS OPERATOR
         BE      TEXTSC           GO SET UP FOR TEXT SCAN
         BCT     5,COMPN          BRANCH TO COMPARE NUMERIC (TYPE = 2)
         LH      6,2(4)           COMPARE ALPHA-NUMERIC HERE - ADD DISPLACEMENT
LOOP     BCTR    6,0              SET A-FIELD ADDRESS TO BASE 0
         EX      6,COMPX          ALPHA COMPARE
         BL      LONE             BRANCH IF A-FIELD IS LOW
         BH      LTHREE           BRANCH IF A-FIELD IS HIGH
         SRDL    6,8              SAVE NUMBER OF CHARS. COMPARED IN REG 7
         SLA     6,8              SUBTRACT NO. OF CHARS. COMPARED FROM REG 6
         BZ      LTWO             IF DONE GET OUT OF LOOP
         SRL     7,24             MOVE SAVE NO. TO LO ORDER END OF REGISTER
         LA      1,1(1,7)         COMPUTE NEW A-FIELD STARTING POSITION
         LA      2,1(2,7)         COMPUTE NEW B-FIELD STARTING POSITION
         B       LOOP             DO SOME MORE
```

```
LTWO      MVI     9(4),X'02'      EQUAL COMPARE, RETURN CODE OF 2
          B       RETX            ALL DONE, FIELDS ARE EQUAL
LTHREE    MVI     9(4),X'03'      A-FIELD IS GREATER, RETURN CODE OF 3
          B       RETX            ALL DONE A   B
TEXTSC    LH      7,2(4)          LENGTH OF A-FIELD
          LH      5,12(4)         LENGTH OF B-FIELD
          SR      7,5             IF THE A-FIELD IS SHORTER THAN THE B-FIELD
          BL      LONE            THERE CAN BE NO MATCH, SO GET OUT.
          BCTR    5,0             ADJUST B-FIELD LENGTH TO BASE 0
          STC     5,LOOPT+1       INSERT THE B-FIELD LENGTH INTO THE LOOPT
                                    CLC INSTRUCTION
          LA      7,0(1,7)        THE LAST HIGH ORDER A-FIELD CHAR. THAT CAN
                                    MATCH
          LA      6,1(0,0)        VALUE 1, THE BXLE INCREMENT
LOOPT     CLC     0(0,1),0(2)     COMPARE A-FIELD AND B-FIELD ACCROSS THE
                                    B-FIELD LENGTH
          BE      LTWO            ITS A HIT
          BXLE    1,6,LOOPT       BUMP THE A-FIELD CHARACTER, AND IF NOT DONE
                                    LOOP BACK
LONE      MVI     9(4),X'01'      A-FIELD IS LESS.  RETURN CODE OF 1
          B       RETX            ALL DONE  A  B, OR NO HIT IF TEXTSCAN
COMPX     CLC     0(0,1),0(2)     COMPARE LOGICAL A-FIELD WITH B-FIELD
PACKA     PACK    AFLD,0(0,1)     PACK A-FIELD
PACKB     PACK    BFLD,0(0,2)     PACK B-FIELD
COMPN     LH      6,2(4)          PICK UP A-FIELD LENGTH *A2*
          BCTR    6,0             ADJUST TO BASE 0
          N       6,FLDOF         ACCEPT 15 DIGITS OR LESS
          EX      6,PACKA         PACK THE A-FIELD
          CLI     AFLD+7,X'04'    BLANKS?
          BNE     LOADB           IF NOT, CONTINUE
          MVI     AFLD+7,X'0F'    REPLACE BLANK WITH 0
LOADB     LH      6,12(4)         LENGTH OF B-FIELD (B2)
          BCTR    6,0             ADJUST TO BASE 0
          N       6,FLDOF         ACCEPT 15 DIGITS OR LESS
          EX      6,PACKB         PACK THE B-FIELD
          CLI     BFLD+7,X'04'    BLANKS?
          BNE     COMPP           IF NOT, CONTINUE
          MVI     BFLD+7,X'0F'    ACCEPT BLANKS AS ZEROS
COMPP     CP      AFLD,BFLD       COMPARE PACKED DECIMAL
          BL      LONE            BRANCH IF A-FIELD IS LESS
          BH      LTHREE          BRANCH IF A-FIELD IS GREATER
          B       LTWO            BRANCH IF EQUAL COMPARE
RETX      RETURN  (1,7),T         GO BACK
          DS      0D
SAVEADDR  DS      16F
AFLD      DS      CL8
BFLD      DS      CL8
FLDOF     DC      X'0000000F'
          END     MARINX
```

(27) GEAZ.

```
GEAZM    CSECT                    EXECUTES MOST OUTPUT, CONDITIONAL, AND
                                    DIRECTIVE STATEMENTS
         SAVE    (14,12)
         BALR    15,0             INITIALIZE BY SAVING PARAMETERS
         USING   *,15             N,RX, MATRIX, STATE-FORM
         LM      2,5,0(1)         ADDRESSES OF PARAMETERS
         ST      4,MATRIX         SAVE MATRIX ADDRESS
         LR      4,3              RECORD ADDRESS
         STM     2,5,SAVEADDR     N, RX, RX, STATE-FORM
         BR      14               EXIT
MARINZ   SAVE    (14,12)          SECOND ENTRY POINT WITHOUT PARAMS
         ENTRY   MARINZ           NORMAL EXECUTION ENTRY POINT
         BALR    15,0
         USING   *,15
         LM      1,4,SAVEADDR     N, RX, RX, STATE-FORM
         LH      5,6(0,4)         GET OPERATION CODE
         SLA     5,2              MULTIPLY IT BY 4, EACH TBLOP ENTRY IS 4 BYTES
         LA      6,TBLOP-4        GET LIST-ADDRESS OF OP-SECTIONS (TBLOP BASE 0)
         L       14,0(5,6)        GET ADDRESS OF OP-SECTION (OP NO TO BE EXEC.)
         BR      14               EXECUTE THE OP-SECTION
ADDRS    LH      5,0(0,4)         A-FIELD, RELATIVE ADDRESS (A1 OF STATE-FORM)
         LH      6,2(0,4)         A-FIELD, LENGTH (A2 OF STATE-FORM)
         LH      7,4(0,4)         A-FIELD, TYPE OF DATA (A3 OF STATE-FORM)
         LH      8,8(0,4)         B-FIELD, RELATIVE ADDRESS (B1 OF STATE-FORM)
         LH      9,10(0,4)        B-FIELD, LENGTH (B2 OF STATE-FORM)
         LH      10,12(0,4)       B-FIELD, TYPE OF DATA (B3 OF STATE-FORM)
         CLI     21(4),X'00'      CHECK FOR SET NO. ZERO IN A-FIELD (A4)
         BE      AMAT             MEANS ADDRS IS IN MATRIX
         AR      5,2              ABSOLUTE ADDRESS OF A-FIELD
RECB     CLI     23(4),X'00'      B-FIELD SET NO. (B4) EQUAL ZERO MEANS
         BE      BMAT             B-FIELD IS A MATRIX
         AR      8,2              ABSOLUTE ADDRESS OF B-FIELD
BDONE    BCTR    6,0              ADJUST A-FIELD LENGTH TO BASE 0
         BCTR    9,0              ADJUST B-FIELD LENGTH TO BASE 0
         BR      14               RETURN
AMAT     A       5,MATRIX         COMPUTE A-FIELD MATRIX ADDRESS
         B       RECB             CHECK B-FIELD
BMAT     A       8,MATRIX         COMPUTE B-FIELD MATRIX ADDRESS
         B       BDONE
APK      N       6,FLDOF          ACCEPT 15 DIGITS OR LESS FOR A NUMBER
         EX      6,PACKA          PACK THE A-FIELD
         CLI     AFLD+7,X'04'     CHECK FOR A BLANK
         BNE     0(0,14)          IF NOT, RETURN
         MVI     AFLD+7,X'0F'     OTHERWISE CHANGE IT TO ZERO
         BR      14               AND RETURN
BPK      N       9,FLDOF          ACCEPT 15 DIGITS OR LESS FOR A NUMBER
         EX      9,PACKB          PACK THE B-FIELD
         CLI     BFLD+7,X'04'     CHECK FOR A BLANK
```

```
         BNE    0(0,14)          IF NOT, RETURN
         MVI    BFLD+7,X'0F'     OTHERWISE, REPLACE WITH A ZERO
         BR     14               AND RETURN
PACKA    PACK   AFLD,0(0,5)      ACTUAL PACK INSTRUCTIONS
PACKB    PACK   BFLD,0(0,8)
CONV     BCT    7,AOK            HAS A-FIELD BEEN CHECKED FOR BLANKS & PACKED?
         BAL    14,APK           IF NOT, CHECK AND PACK
         B      CONVB            NOW DO THE B-FIELD
AOK      BAL    14,AZP           GO TO THE LEADING ZERO FILL SECTION
CONVB    BCT    10,CONVG         ARE WE READY TO EXIT YET?
         B      ATRUE            IF SO, SET UP RETURN
CONVG    BCT    10,BOK           HAS B-FIELD BEEN CHECKED AND PACKED?
         BAL    14,BPK           IF NOT, CHECK AND PACK
         BR     12               RETURN TO 12
BOK      BAL    14,BZP           GO TO LEADING ZERO FILL SECTION
         BR     12               RETURN VIA 12
AZP      N      6,FLDOF          ACCEPT 15 OR LESS DIGITS AS A-FIELD LENGTH
         EX     6,ZAPA           ZERO FILL A-FIELD
         BR     14               RETURN VIA 14
BZP      N      9,FLDOF          ACCEPT 15 OR LESS DIGITS AS B-FIELD LENGTH
         EX     9,ZAPB           ZERO FILL B-FIELD
         BR     14               RETURN VIA 14
ZAPA     ZAP    AFLD,0(0,5)      INSERT LEADING ZEROS IN AFLD
ZAPB     ZAP    BFLD,0(0,8)      INSERT LEADING ZEROS IN BFLD
OP01     BAL    14,ADDRS         DECODE INSTRUCTION (LESS THAN)
         BCT    7,OP01N          BRANCH IF A-FIELD NOT ALPHA-NUMERIC
         EX     6,COMPX          COMPARE ALPHA-NUMERIC, LOGICAL
         BL     ATRUE            A-FIELD IS LESS THAN B-FIELD
         B      AFALLS           IT IS NOT LESS
COMPX    CLC    0(0,5),0(8)      LOGICAL COMPARE OF UP TO 256 CHARACTERS
OP02     BAL    14,ADDRS         DECODE INSTRUCTION (EQUAL)
         BCT    7,OP02N          BRANCH IF A-FIELD NOT ALPHA
         EX     6,COMPX          COMPARE A-FIELD WITH B-FIELD
         BE     ATRUE            A-FIELD IS EQUAL TO B-FIELD
         B      AFALLS           IT IS NOT EQUAL
OP03     BAL    14,ADDRS         DECODE INSTRUCTION (GREATER THAN)
         BCT    7,OP03N          BRANCH IF A-FIELD NOT ALPHA
         EX     6,COMPX          COMPARE A-FIELD WITH B-FIELD
         BH     ATRUE            A-FIELD IS GREATER THAN B-FIELD
         B      AFALLS           IT IS NOT GREATER
OP04     BAL    14,ADDRS         DECODE INSTRUCTION (NOT LESS THAN)
         BCT    7,OP04N          BRANCH IF A-FIELD NOT ALPHA
         EX     6,COMPX          COMPARE A-FIELD WITH B-FIELD
         BL     AFALLS           A-FIELD LESS THAN B-FIELD, CONDITION FALSE
         B      ATRUE            CONDITION TRUE
OP05     BAL    14,ADDRS         DECODE INSTRUCTION (NOT EQUAL)
         BCT    7,OP05N          BRANCH IF A-FIELD NOT ALPHA
         EX     6,COMPX          COMPARE A-FIELD WITH B-FIELD
         BE     AFALLS           A-FIELD IS EQUAL TO B-FIELD, CONDITION FALSE
         B      ATRUE            CONDITION TRUE
```

```
OP06     BAL    14,ADDRS     DECODE INSTRUCTION (NOT GREATER THAN)
         BCT    7,OP06N      BRANCH IF A-FIELD NOT ALPHA
         EX     6,COMPX      COMPARE A-FIELD WITH B-FIELD
         BH     AFALLS       A-FIELD GREATER THAN B-FIELD (CONDITION FALSE)
         B      ATRUE        CONDITION TRUE
OP07     BAL    14,ADDRS     DECODE INSTRUCTION (ADD)
         LR     11,10        B-FIELD DATA TYPE
         BCT    7,OP07N      BRANCH IF A-FIELD NOT ALPHA
         B      ATRUE        NOP THE INSTRUCTION
OP07N    BAL    12,CONV      PACK FIELDS IF NECESSARY
         AP     BFLD,AFLD    ADD A-FIELD TO B-FIELD IN WORKING STORAGE
STRB     LH     9,10(0,4)    B-FIELD LENGTH
         BCTR   9,0          ADJUST TO BASE 0
         LR     10,11        B-FIELD DATA TYPE
         BCT    10,STRBN     BRANCH IF B-FIELD NOT ALPHANUMERIC
         B      STRBNX       STORE AN ALPHA B-FIELD
STRBN    BCT    10,STRBZ     IF GOING TO A WORK AREA, BRANCH
STRBNX   SLA    9,4          ADJUST LENGTH OF REC. AREA FOR UNPK INSTR.
         EX     9,UNPKB      UNPACK THE B-FIELD FOR DISPLAYING
         B      OPXX         ON THE WAY BACK
UNPKB    UNPK   0(0,8),BFLD  CONVERT B-FIELD TO BCD IN BFLD
STRBZ    SLA    9,4          ADJUST LENGTH PARAMETER FOR ZAP INSTRUCTION
         EX     9,ZAPBB      ZERO AND PACK B-FIELD
         B      OPXX         ON THE WAY OUT
ZAPBB    ZAP    0(0,8),BFLD  INSERT LEADING ZEROS AND PACK INTO SAVE AREA
OP08     BAL    14,ADDRS     DECODE INSTRUCTION (SUBTRACT)
         LR     11,10        B-FIELD DATA TYPE
         BCT    7,OP08N      BRANCH IF A-FIELD NUMERIC
         B      ATRUE        NOP THE INSTRUCTION
OP08N    BAL    12,CONV      PACK THE FIELDS IF NECESSARY
         SP     BFLD,AFLD    SUBTRACT AFLD FROM BFLD
         B      STRB         RESTORE THE B-FIELD TO PROPER FORM
OP11     BAL    14,ADDRS     DECODE INSTRUCTION (MOVE)
         LR     11,10        B-FIELD DATA TYPE
         BCT    7,OP11N      BRANCH IF A-FIELD NUMERIC
         CR     6,9          COMPARE LENGTHS
         BE     OP11X        NO PADDING NECESSARY
         MVI    0(8),X'40'   INSERT BLANK IN RECEIVING AREA
         ST     6,SAVE6      SAVE ADDRESS OF LENGTH OF A-FIELD
         ST     8,SAVE8      SAVE ADDRESS OF B-FIELD
         LR     6,9          B-FIELD LENGTH
OP11F    EX     6,MOVEF      PROPAGATE BLANKS IN RECEIVING FIELD
         SRDL   6,8          SAVE NUMBER MOVED
         SLA    6,8          SUBTRACT NUMBER MOVED
         BZ     OP11XC       IF DONE, SKIP AHEAD
         SRL    7,24         RIGHT JUSTIFY NUMBER MOVED
         LA     8,1(7,8)     COMPUTE NEW STARTING ADDRESS
         BCT    6,OP11F      IF GREATER THAN ONE, BO BACK
```

```
MOVEF    MVC    1(0,8),0(8)    PROPOGATE CHARACTER
OP11XC   L      6,SAV6         RESTORE A-FIELD LENGTH
         L      8,SAV8         RESTORE B-FIELD ADDRESS
OP11X    EX     6,MOVEX        ALPHA MOVE
         SRDL   6,8            SAVE NUMBER MOVED
         SLA    6,8            SUBTRACT NUMBER MOVED
         BX     OPXX           IF DONE, THIS IS THE WAY OUT
         SRL    7,24           RIGHT-JUSTIFY NUMBER MOVED
         LA     5,1(7,5)       COMPUTE NEW SENDING ADDRESS
         LA     8,1(7,8)       COMPUTE NEW RECEIVING ADDRESS
         BCT    6,OP11X        IF MORE THAN ONE LEFT, GO BACK
MOVEX    MVC    0(0,8),0(5)    MAX STRING LENGTH IS 256, MIN IS 1
         B      OPXX           WAY OUT
OP11N    BCT    7,OP11Z        IF DEFINED AREA, BRANCH OUT
         BAL    14,APK         RECORD FIELD MIGHT HAVE BLANKS
         B      MOVEA          MOVE
OP11Z    BAL    14,AZP         LEAD ZEROS AND PACK, NO BLANKS
MOVEA    MVC    BFLD, AFLD     MOVE AFLD TO BFLD
         B      STRB           MOVE BFLD TO THE B-FIELD
OP01N    BAL    12,CONV        PACK
         CP     AFLD,BFLD      COMPARE
         BL     ATRUE          A-FIELD LESS THAN B-FIELD
         B      AFALLS         A-FIELD NOT LESS
OP02N    BAL    12,CONV        PACK
         CP     AFLD,BFLD      COMPARE
         BL     ATRUE          A-FIELD LESS THAN B-FIELD
         B      AFALLS         A-FIELD NOT LESS
OP03N    BAL    12,CONV        PACK
         CP     AFLD,BFLD      COMPARE
         BH     ATRUE          A-FIELD GREATER THAN B-FIELD
         B      AFALLS         A-FIELD NOT GREATER
OP04N    BAL    12,CONV        PACK
         CP     AFLD,BFLD      COMPARE
         BL     AFALLS         A-FIELD LESS
         B      ATRUE          A-FIELD NOT LESS THAN B-FIELD
OP05N    BAL    12,CONV        PACK
         CP     AFLD,BFLD      COMPARE
         BE     AFALLS         A-FIELD EQUAL
         B      ATRUE          A-FIELD NOT EQUAL
OP06N    BAL    12,CONV        PACK
         CP     AFLD,BFLD      COMPARE
         BH     AFALLS         A-FIELD GREATER
         B      ATRUE          A-FIELD NOT GREATER THAN B-FIELD
OP31     BAL    14,ADDRS       MOVE SPACES
         MVI    0(8),X'40'     MOVE BLANK TO B-FIELD
```

```
OP323    LR     5,8              B-FIELD ADDRESS
         LA     8,1(0,8)         ADD 1 TO B-FIELD ADDRESS
         LR     6,9              B-FIELD LENGTH
         SLA    9,8              WILL WE NEED TO PROPAGATE?
         BZ     ATRUE            NO, WE'RE DONE
         BCTR   6,0              ADJUST LENGTH TO BASE 0
         B      OP11X            GO DO A MOVE
OP32     BAL    14,ADDRS         MOVE ZEROS.
         MVI    0(8),X'FO'       MOVE ZERO TO B-FIELD
         B      OP323            PROPAGATE ZEROS
ATRUE    MVC    0(2,1),14(4)     LOAD ADDRESS OF NEXT STATEMENT TO BE
                                   EXECUTED IN N
         B      OP99             GET OUT
AFALLS   MVC    0(2,1),16(4)     LOAD ADDRESS OF NEXT STATEMENT TO BE
                                   EXECUTED IN N
         B      OP99             GET OUT
OP30     BAL    14,ADDRS         EDIT SECTION (EMOVE)
         LR     11,8             B-FIELD ADDRESS
         BCT    7,OP36N          IF A-FIELD IS NUMERIC, BRANCH
         B      OP11             IF ALPHA, DO A REGULAR MOVE
OP36N    BCT    7,OP36P          IF DEFINED A-FIELD, BRANCH
         BAL    14,APK           PACK THE RECORD FIELD
         LH     6,2(0,4)         LENGTH OF A-FIELD
EDIT     LA     5,MASK           EMOVE - THE MIDMS MASK IS
         MVC    0(0,5),0(11)     CONVERTED TO THE HEX CONFIGURATIONS
         MVC    1(31,4),0(5)     ACCEPTED AS CONTROL SYMBOLS
         LA     7,16(0,5)        BY THE 360 ED (EDIT) INSTRUCTION.
         SR     7,6
         LR     10,6             THE MASK IS ASSUMED TO
         EX     9,MOVEM          ALREADY EXIST IN THE B-FIELD
         LA     8,0(0,0)         STARTING WITH POSITION B1.
         IC     8,9(0,5)
         LA     6,TBLE           THIS MIDMS EMOVE CLOSELY
         AR     6,8              RESEMBLES THE 1410 EDIT
         CLI    0(5),X'FO'       FEATURE.
         BNE    EDITP
         MVI    1(5),X'21'
EDITP    MVC    TBLES,0(6)
         MVI    0(6),X'20'
PRCED    LA     8,16(0,0)
         SR     8,10
         AR     8,9
         EX     8,TRAN
         EX     8,EDITX
         EX     9,MOVEE
         MVC    0(0,6),TBLES
         B      ATRUE
```

```
MOVEM   MVC     0(0,7),0(11)
TRAN    TR      1(0,5),TBLE
EDITX   ED      0(0,5),AFLD
MOVEE   MVC     0(0,11),0(7)
OP36P   BAL     14,AZP
        LH      6,2(0,4)
        SLA     6,1
        BCTR    6,0
        B       EDIT
OP09    BAL     14,ADDRS        DECODE INSTRUCTION (MULTIPLY)
        LR      11,7            A-FIELD TYPE
        BCT     7,OP09N         BRANCH IF NUMERIC
        B       ATRUE           NOP IF ALPHA
OP09N   BAL     12,CONV         PACK
        MVC     QFLD,ZFLD       MOVE ZEROES TO QFLD (HIGH HALF OF QA PAIR)
        MP      QFLD(16),BFLD   MULTIPLY QFLD BY BFLD (AFLD BY BFLD)
STRA    LH      6,2(0,4)        LENGTH OF A-FIELD
        BCTR    6,0             ADJUST TO BASE 0
        LR      7,11            A-FIELD TYPE
        BCT     7,STRAN         IF NUMERIC, BRANCH
        B       ATRUE           NOP THE INSTRUCTION
STRAN   BCT     7,STRAZ         IF A DEFINED AREA, BRANCH
        SLA     6,4             ADJUST POSITION OF LENGTH FOR EX UNPK
        EX      6,UNPKA         CONVERT TO BCD
        B       ATRUE           THE WAY BACK
UNPKA   UNPK    0(0,5),AFLD     ID TO EBCDIC
STRAZ   SLA     6,4             ADJUST LENGTH POSITION IN REGISTER
        EX      6,ZAPAA         ZERO FILL AND PACK THE A-FIELD
        B       ATRUE           TIME TO EXIT
ZAPAA   ZAP     0(0,5),AFLD     INSERT  LEADING ZEROES
OP10    BAL     14,ADDRS        DECODE INSTRUCTION (DIVIDE)
        LR      11,7            TYPE OF B-FIELD
        BCT     7,OP10N         BRANCH IF NUMERIC
        B       ATRUE           NOP THE INSTRUCTION IF ALPHA
OP10N   BAL     12,CONV         PACK
        CP      SFLD(8),BFLD    DIVISION BY ZERO?
        BNE     CONT            IF NOT, CONTINUE
        MVC     AFLD,SFLD       DIVISION BY ZERO RESULTS IN ZERO
        B       STRA            MOVE AFLD TO A-FIELD
CONT    MVC     QFLD,ZFLD       MOVE ZEROS TO QFLD (HIGH HALF OF QA PAIR)
        DP      QFLD(16),BFLD   DIVIDE AFLD BY BFLD, RESULT IN Q
        MVC     AFLD,QFLD       MOVE RESULT TO AFLD
        B       STRA            MOVE AFLD TO A-FIELD
OP15    L       8,8(4)          B1 FIELD OF STATEFORM
        LH      5,0(4)          A1 FIELD OF STATEFORM
        AR      5,2             ABSOLUTE ADDRESS OF A-FIELD
        MVC     BFLD(6),0(8)    GET TABLE PARAMS
        LH      6,BFLD          LENGTH OF ARGUMENT
        LH      9,BFLD+2        LENGTH OF FUNCTION
```

```
        LH      10,BFLD+4       NUMBER OF OCCURRENCES
        LA      8,6(8)          STARTING POINT IN TABLE
        BCTR    6,0             ADJUST LENGTH TO BASE 0
OP15C   EX      6,COMPX         COMPARE ARGUMENT WITH SEARCH VALUE
        BE      OP15B           BRANCH ON MATCH
        LA      8,1(6,8)        ADD LENGTH OF ARGUMENT (BASE 1) TO STARTING
                                  POINT
        LA      8,0(9,8)        ADD FUNCTION LENGTH TO GET ADDRESS OF
        BCT     10,OP15C        NEXT ARGUMENT FOR NEXT PASS
        MVC     8(4,4),14(4)    NOT IN TABLE-SET UP FOR A
        B       OP11            REGULAR MOVE
OP15B   BCTR    9,0             ADJUST FUNCTION LENGTH TO BASE 0
        LA      5,1(8,6)        FUNCTION SUBSCRIPT
        LH      8,14(4)         SET UP FOR FUNCTION MOVE
        AR      8,3             ACTUAL B-FIELD ADDRESS
        EX      9,MOVEX         MOVE THE FUNCTION TO THE B-FIELD
OPXX    LH      2,0(1)          N, SUBSCRIPT OF CURRENT INSTRUCTION
        LA      2,1(2)          ADD 1
        STH     2,0(1)          N = N + 1
        B       OP99            EXIT
OP14    LH      8,8(0,4)        TMOVE
        LR      7,15            SAVE REGISTERS
        ST      1,REG1
        AR      8,2             COMPUTE ACTUAL B-FIELD ADDRESS
        MVC     TADDR(12),0(8)ADDRESS OF NAMED TABLE
        ST      13,REG13
        LA      13,MASK
        LA      2,TNAME
        LOAD    EPLOC-(2)       LOAD THE TABLE
        LR      15,7            RESTORE REGISTER
        L       1,REG1
        LR      13,0            LOCATION OF TABLE
        LH      5,0(0,4)        RELATIVE AFLD ADDRESS
        AR      5,3             COMPUTE ACTUAL AFLD ADDRESS
        MVC     8(2,4),TADDR+2          SAME ADDRESS OF TABLE
        LH      6,0(0,13)       LENGTH OF ARGUMENT
        LH      9,2(0,13)       LENGTH OF FUNCTION
        LH      10,4(0,13)      NUMBER OF ENTRIES IN TABLE
        LA      8,6(0,13)       FIRST ARGUMENT IN TABLE
        BCTR    6,0             ADJUST TO BASE 0
LOOK1   EX      6,COMPX         COMPARE VALUE WITH ARGUMENT
        BE      FOUND           IT'S A HIT
        LA      8,1(6,8)        COMPUTE NEXT ARGUMENT
        LA      8,0(9,8)        ADDRESS
        BCT     10,LOOK1        IF MORE ENTRIES, GO BACK
        LA      13,MASK         NO HIT
        DELETE  EPLOC=(2)       WIPE OUT TABLE
        LR      15,7            RESTORE REGISTERS
```

```
         L       1,REG1
         L       13,REG13
         LR      2,3             SET UP FOR REGULAR MOVE
         B       OP11            AND EXECUTE IT
FOUND    CH      9,10(0,4)       DO LENGTH MATCH?
         BE      LOOK2           SAME, OK
         BL      LOOK2           FUNCTION SHORTER, OK
         LH      9,10(0,4)       TRUNCATE FUNCTION
LOOK2    LA      5,1(8,6)        SET UP FOR MOVE
         BCTR    9,0             OF FUNCTION TO
         L       8,TADDR         THE B-FIELD AREA
         AR      8,3
         EX      9,MOVEX         MOVE THE FUNCTION
         LA      13,MASK         SAVE REGISTER
         DELETE  EPLOC=(2)       WIPE OUT TABLE
         LR      15,7            RESTORE REGISTER
         L       1,REG1
         L       13,REG13
         B       OPXX            SET UP STATEMENT SUBSCRIPT FOR NEXT
                                   ONE AND RETURN
OP40     LH      8,14(4)         GET RELATIVE ADDRESS OF TABLE
         MVC     14(4,4),8(4)    SHIFT BFLD PARAMS
         AR      8,3             COMPUTE ABSOLUTE ADDRS OF TABLE
         ST      8,8(4)          SET ADDRESS AS FOR TABLE LOOKUP, OP15
         B       OP15            DO A REGULAR TABLE LOOKUP
OP42     LH      5,0(4)          SUBSCRIPT ADDRESS - SUBSCRIPT STATEMENT
         AR      5,2             ABSOLUTE ADDRESS OF SUBSCRIPT FIELD
         LH      6,2(4)          LENGTH OF SUBSCRIPT FIELD
         BCTR    6,0             ADJUST TO BASE 0
         BAL     14,APK          PACK THE SUBSCRIPT VALUE
         CVB     14,AFLD         CONVERT THE OCCURRENCE TO BINARY
         LA      6,2(0,0)        VALUE 2
         CR      14,6            CHECK FOR NEGATIVE SUBSCRIPT
         BL      OP421           BAD SUBSCRIPT
         BCTR    14,0            SUBTRACT ONE FROM OCCURRENCE
         MH      14,10(0,4)      NO. OF OCCURRENCES TIMES LENGTH OF OCCURRENCE
OP420    STH     14,0(4)         NEW STARTING ADDRESS
         B       ATRUE           RETURN
OP421    SR      14,14           SET TO 0, INDICATING NO SUBSCRIPT OFFSET
         B       OP420           SUBSCRIPT HAS BEEN NOPED
OP46     LR      3,1             ADDRS OF N
         LH      1,0(0,4)        RELATIVE TEXT ADDRESS
         AR      1,2             ABSOLUTE TEXT ARDR
         BCTR    1,0             TEXT ADDR MINUS 1
         LH      8,8(0,4)        RELATIVE ADDRS OF VALUE
         AR      8,2             ABSOLUTE ADDRS OF VALUE
         LH      6,2(0,4)        LENGTH OF TEXT
         LH      9,10(0,4)       LENGTH OF VALUE
         BCTR    9,0             SUBTRACT 1
```

```
         BCTR   6,0            SUBTRACT 1
         SR     6,9            SUBTRACT FROM TEXT LENGTH VALUE
         BL     OP46N          BRANCH IF TEXT SHORTER
         LA     10,0(1,6)      ADDRS OF LAST TRANSLATE
         SR     5,5            ZERO OUT RX5
         IC     5,0(0,8)       PICK UP FIRST CHAR OF VALUE
         LA     2,T635         GET ADDRS OF TABLE
         AR     5,2            COMPUTE ADDRS IN TABLE
         MVI    0(5),X'FF'     SET BITS ON
OP46F    EX     6,OP46T        EXECUTE TRANSLATE AND TEST
         BZ     OP46A          BRANCH IF NOT FOUND
         BH     OP46H          BRANCH IF LAST CHAR IN LENGTH
         EX     9,OP46C        EXECUTE COMPARE
         BE     OP46Y          BRANCH IF VALUE FOUND
         LR     6,10           LOAD LAST ADDRS TEXT
         SR     6,1            SUBTRACT PRESENT ADDRS TEXT
         B      OP46F          GO TO LOOP
OP46A    SRDL   6,8            SHIFT RIGHT DOUBLE LOGICAL
         SLA    6,8            SHIFT LEFT
         BZ     OP46N          BRANCH IF END OF SEARCH
         SRL    7,24           SHIFT RIGHT LOGICAL
         LA     1,1(1,7)       RECOMPUTE TEXT ADDRS
         BCT    6,OP46F        BRANCH TO LOOP SUBTRACTING 1
OP46H    EX     9,OP46C        EXECUTE COMPARE
         BE     OP46Y          BRANCH IF VALUE FOUND
OP46N    LA     4,2(0,4)       END OF TEXT NO MATCH
OP46Y    MVC    0(2,3),14(4)   VALUE MATCHED
         MVI    0(5),X'00'     RESET TABLE TO 0
         B      OP99           GO TO EXIT
OP46T    TRT    1(0,1),T635    TRANSLATE AND TEST
OP46C    CLC    0(0,1).0(8)    COMPARE TEXT TO VALUE
OP48     BAL    14,ADDRS       DECODE MATRIX INSTRUCTION
         BAL    14,APK         PACK A-FIELD AND COMPUTE SUBSCRIPT
         BAL    14,BPK         PACK B-FIELD AND COMPUTE SUBSCRIPT
         CVB    14,AFLD        CONVERT A-FIELD ADDRESS TO BINARY
         BCTR   14,0           ADJUST TO BASE 0
         STH    14,0(4)        PLACE ADDRESS IN FIELD A1 OF STATEFORM
         CVB    14,BFLD        CONVERT B-FIELD ADDRESS TO BINARY
         BCTR   14,0           ADJUST TO BASE 0
         STH    14,8(4)        PLACE B-FIELD ADDRESS IN B1 OF STATEFORM
         B      ATRUE          EXIT
T635     DC     256X'00'
         DS     0D
SAVEADDR DS     16F
         DS     CL4
TADDR    DS     CL4
```

```
TNAME    DS      CL8
MASK     DS      CL160
REG1     DS      2F
FLDOF    DC      X'0000000F'
FLD70    DC      X'00000070'
ZFLD     DC      X'0000000000000000'
SFLD     DC      X'000000000000000F'
MATRIX   DS      1F
         DS      0D
QFLD     DS      CL8
AFLD     DS      CL8
BFLD     DS      CL8
REG13    DS      4F
OP99     RETURN  (14,12),T       GO BACK
TBLE     DC      X'000102030405060708090A0B0C0D0E0F'
         DC      X'101112131415161718191A1B1C1D1E1F'
         DC      X'202122232425262728292A2B2C2D2E2F'
         DC      X'303132333435363738393A3B3C3D3E3F'
         DC      X'404142434445464748494A4B4C4D4E4F'
         DC      X'505152535455565758595A5B5C5D5E5F'
         DC      X'606162636465666768696A6B6C6D6E6F'
         DC      X'707172737475767778797A7B7C7D7E7F'
         DC      X'808182838485868788898A8B8C8D8E8F'
         DC      X'909192939495969798999A9B9C9D9E9F'
         DC      X'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'
         DC      X'B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF'
         DC      X'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'
         DC      X'D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF'
         DC      X'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF'
         DC      X'21F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'
TBLES    DC      X'20'
SAVE6    DS      1F
SAVE8    DS      1F
TBLOP    DC      A(OP01)         01
         DC      A(OP02)         02
         DC      A(OP03)         03
         DC      A(OP04)         04
         DC      A(OP05)         05
         DC      A(OP06)         06
         DC      A(OP07)         07
         DC      A(OP08)         08
         DC      A(OP09)         09
         DC      A(OP10)         10
         DC      A(OP11)         11
         DC      A(OP12)         12
         DC      A(OP13)         13
         DC      A(OP14)         14
```

```
DC      A(OP15)         15
DC      A(OP99)         16
DC      A(OP99)         17
DC      A(OP99)         18
DC      A(OP99)         19
DC      A(OP99)         20
DC      A(OP99)         21
DC      A(OP99)         22
DC      A(OP99)         23
DC      A(OP99)         24
DC      A(OP99)         25
DC      A(OP99)         26
DC      A(OP99)         27
DC      A(OP99)         28
DC      A(OP99)         29
DC      A(OP30)         30
DC      A(OP31)         31
DC      A(OP32)         32
DC      A(OP99)         33
DC      A(OP99)         34
DC      A(OP99)         35
DC      A(OP99)         36
DC      A(OP99)         37
DC      A(OP99)         38
DC      A(OP99)         39
DC      A(OP40)         40
DC      A(OP42)         41
DC      A(OP42)         42
DC      A(OP99)         43
DC      A(OP99)         44
DC      A(OP46)         45
DC      A(OP46)         46
DC      A(OP48)         47
DC      A(OP48)         48
DC      A(OP99)         49
DC      A(OP99)         50
DC      A(OP99)         51
DC      A(OP99)         52
DC      A(OP99)         53
DC      A(OP99)         54
DC      A(OP99)         55
DC      A(OP99)         56
DC      A(OP99)         57
DC      A(OP99)         58
DC      A(OP99)         59
END     MARINZ
```