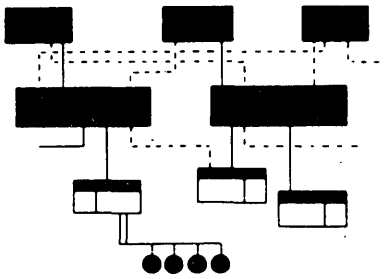
GENERAL ELECTRIC
COMPUTERS

GE-625/635 PERT/TIME



SYSTEM
SUPPORT
INFORMATION

GENERAL ELECTRIC

CPB-1192A

GE-625/635
PERT/TIME

Program Number
CD600K1.001

August 1966

Rev. June 1967

GENERAL ELECTRIC

INFORMATION SYSTEMS DIVISION

PREFACE

This manual is provided by General Electric Computer Equipment Department for programmers who use and modify the GE-625/635 PERT/TIME system. This is one of many similar publications available to programmers using GE systems.

After the general system description the manual describes the five sections of the GE-625/635 PERT/TIME system. Each section contains a description of that section, the links it contains, the subroutines it uses, and complete flowcharts at the end of each chapter. These flowcharts are drawn with the ASA standard symbols. The appendix contains a summary of these symbols.

Details of philosophy, purpose, operation, and interpretation of results are covered in the GE-625/635 PERT/TIME reference manual, CPB-1139.

GE-625/635 PERT/TIME is designed to operate under control of the General Comprehensive Operating Supervisor (GECOS). Information concerning GECOS may be found in the GE-625/635 Comprehensive Operating Supervisor, CPB-1195.

This revised manual includes information previously published in CPB-1192 and supplemented with information previously published in Technical Information Bulletins 600-138 and 600-149. In this revised edition, changes in technical content from the previous edition are identified with a bar in the margin opposite the change.

Suggestions and criticisms relative to form, content, purpose, or use of this manual are invited. Comments may be sent on the Document Review Sheet in the back of this manual or may be addressed directly to Documentation Standards and Publications, B-90, Computer Equipment Department, General Electric Company, 13430 North Black Canyon Highway, Phoenix, Arizona 85029.

(750 9-67)

CONTENTS

ILLUSTRATIONS

1. INTRODUCTION

GE-625/635 PERT/TIME is a computerized program which provides management with useful and timely control information. It was written for the GE-625/635 computer and utilizes network model techniques to define and schedule the execution of a complex work project.

COMPATIBILITY WITH USAF PERT

This program is compatible with USAF PERT specifications as outlined in the first two volumes of the U.S. Air Force PERT series:

Volume I, USAF PERT Time System Description Manual

Volume II, USAF PERT Time System Computer Handbook.

MINIMUM SYSTEM CONFIGURATION

The minimum system configuration for use of GE-625/635 PERT/TIME is:

- 32k core store (for PERT exclusive of software; the program can be recompiled for smaller core requirements)
- Card reader
- Printer
- 3 utility tapes and disc or 10 tapes with no disc (for PERT exclusive of system tapes)
- Card punch

2. SYSTEM DESCRIPTION

GE-625/635 PERT is composed of the following five sections:

FILMNT	-	File Maintenance
TOPOLO	-	Topology
SUMMRY	-	Summarization
REPGEN	-	Report Generator
SETOUT	-	Output Reports

FILMNT, TOPOLO, REPGEN, and SETOUT are separate sections which operate under control of a link control program, LCNT. Each of these sections prepares output that is used for input to succeeding sections. The SUMMRY section is under control of the TOPOLO section. This section, called by the user's option on Initial Card II performs network summarization. The relationships of these sections are shown in Figure 1.

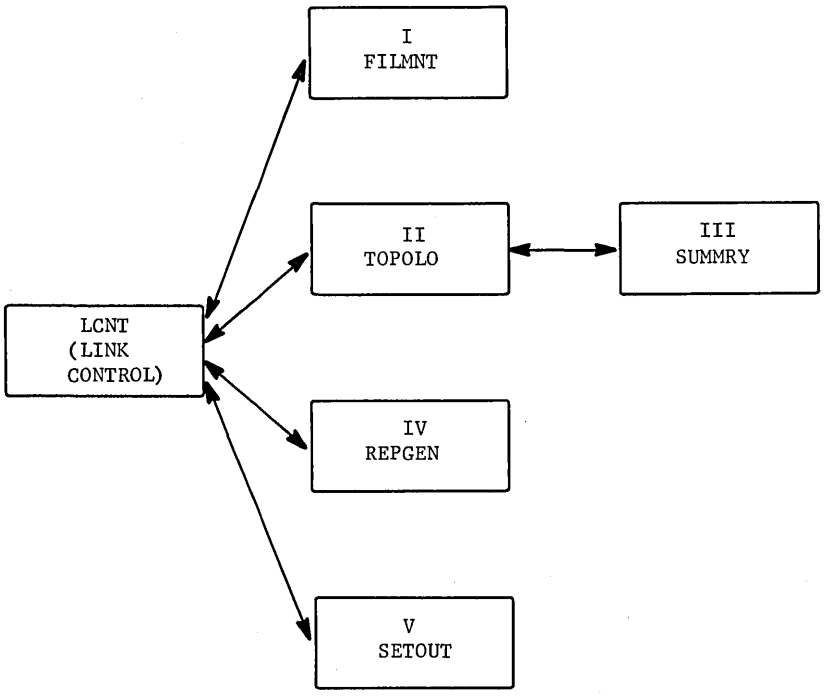


Figure 1. Program Structure

LINK CONTROL PROGRAM - LCNT

The link control program LCNT controls linkage of the File Maintenance, Topology, and output Sections.

LCNT does this by issuing calls to the system program LLINK followed by calls to the appropriate subroutines.

The following is a description of two COMMON areas used by LCNT.

COMMON Used:

Block Name	Data Names	Purpose
IOFB	LFILB (15)	Logical File Table
	IFCB (16)	Dummy Name
	LFA (22)	File Control Block 01
	LFB (22)	File Control Block 02
	LFC (22)	File Control Block 03
	LFD (22)	File Control Block 04
	LFE (22)	File Control Block I*
	LFF (22)	File Control Block P*
	LFG (22)	File Control Block 08
	LFH (6)	File Control Block 30
	BFE (321)	Buffer for I*
	BFF (321)	Buffer for P*
	BFH (321)	Buffer for 30
PRCH	LSVE	Last numeric data block saved
	NSMO	Summary option
	NCD	Network completion date
	NRDO	Report date option
	NRDT	Report date
	PEOF	End of file indicator (File 04)
	NSD	Network start date

Figure 2 is a flowchart for the LCNT program.

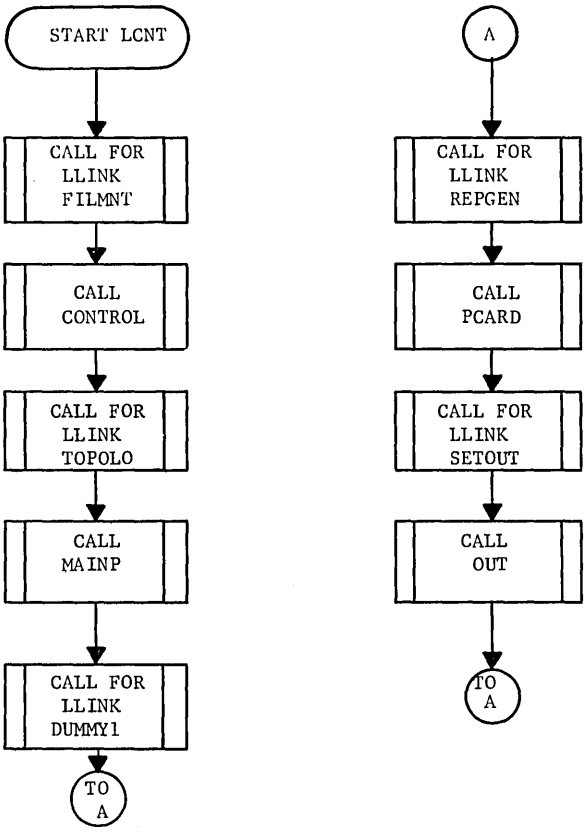


Figure 2. LCNT Flowchart

CORE ALLOCATION

Core allocation for GE-625/635 PERT/TIME is shown in Figure 3. Link DUMMY1 preserves labeled COMMON for use by links REPGEN, SETOUT, SEALNK, and REGLNK during multiple reports and thus provides for multiple execution of these links. Labeled COMMON KEEP carries data set by link REPGEN. Labeled COMMON ALLSRT, ISAVV, and UNSKED carry block data for the links involved.

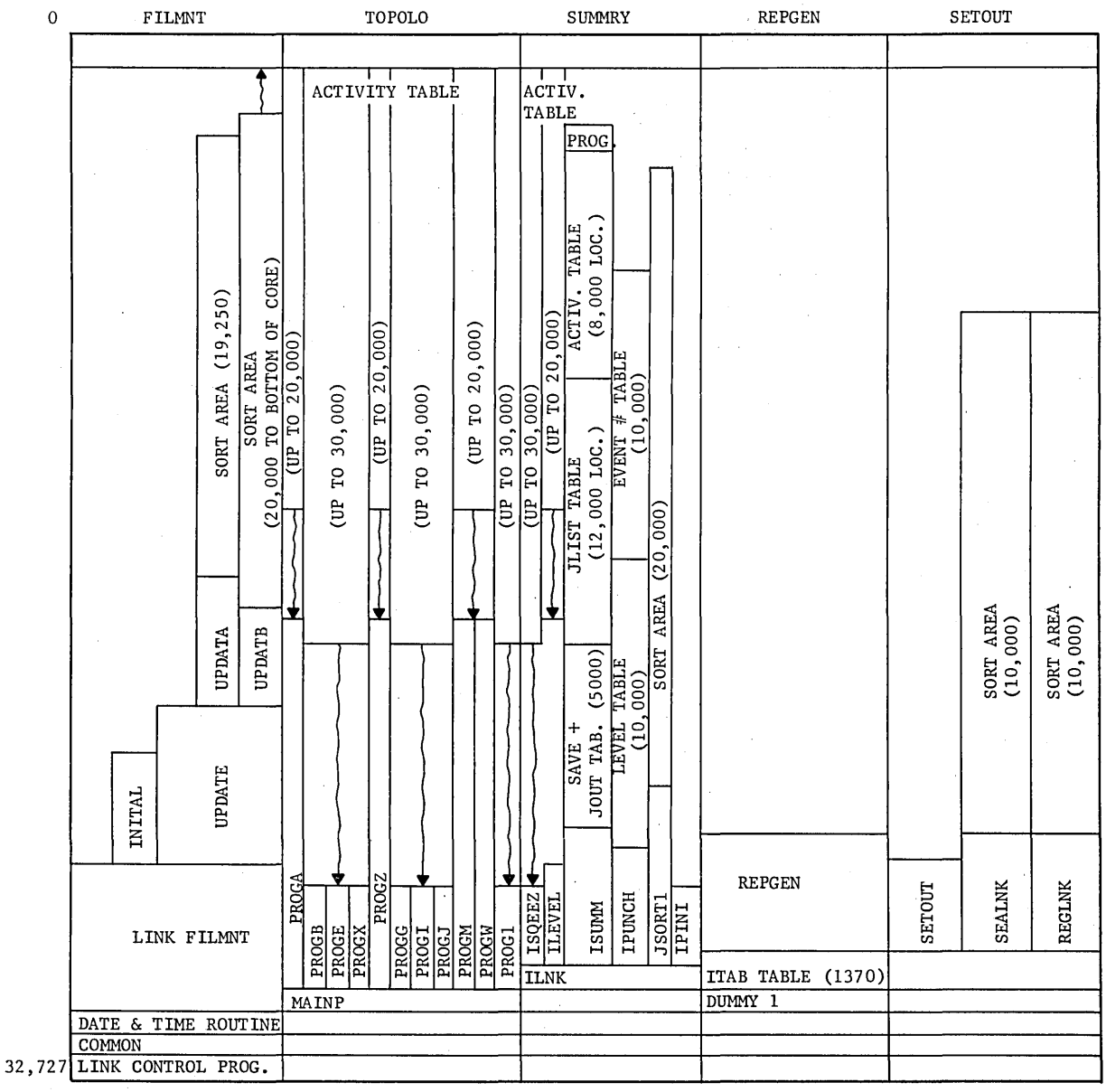


Figure 3. Core Allocation

3. FILE MAINTENANCE SECTION--FILMNT

The File Maintenance Section (FILMNT) consists of 16 subroutines contained in five links. These subroutines set the logical file tables and buffers for input/output operation, read the initial cards and holiday and data cards, update the master file, and generate master file reports and diagnostics.

FILE MAINTENANCE LINKAGE

The functions of the five links and the manner in which the subroutines perform these functions are described below.

Flowcharts for these links and subroutines are at the end of this chapter.

Control Link--FILMNT

Link FILMNT is the control link for the file maintenance section.

SUBROUTINE CONTRL

Subroutine CONTRL first uses FORTRAN I/O subroutines to define logical file units and buffers. (See FORTRAN IV Input/Output Library, CPB-1137.) It then calls each of the links for execution.

SUBROUTINES CALIN, MINT

The subroutine CALIN (the input calendar routine), the function MINT, and all the COMMON used in the section are also contained in the FILMNT link. CALIN is a calendar routine which converts a date to the number of work weeks from the network start date. It accepts a 5-, 6-, or 7-day work week with holidays defined by the Holiday and Vacation input cards. MINT is a BCD to binary conversion function which is used by most of the programs in the section.

Card Read Link--INITAL

Link INITAL reads Initial Cards I and II, and the Holiday and Vacation Cards, diagnoses all errors found on these cards, writes the Master File Summary Sheet, and writes the Holiday and Vacation Card Listing.

These functions are accomplished in three subroutines, INIT, HOLTAB, and ERCHK.

SUBROUTINE INIT

INIT reads the two initial cards and checks their identity against that of the old master file. (If the run number is a 1, a dummy old master file is generated before the check.) The number of days in the work week is checked for legality. If this number is not legal, a seven-day work week is assumed.

SUBROUTINE HOLTAB

Subroutine HOLTAB is then called. This routine reads the Holiday and Vacation cards and checks them for the following types of errors: nonexistent dates, dates which do not occur in ascending order, a blank begin or end date within a pair, and an overflow of the holiday table. The pairs of dates which pass the legality checks are converted to day values and stored in the holiday table. A listing of all the Holiday and Vacation cards is given with a listing of all the errors found on them. When an "E" is found in column 1, this subroutine is finished and subroutine ERCHK is called.

SUBROUTINE ERCHK

Subroutine ERCHK checks all of the fields on the initial cards for errors, and prints error diagnostics if any errors are found.

After all the fields are checked, control is returned to INIT where the Master File Summary Listing is written. "No Error" messages are written to indicate the absence of errors on the Initial cards or the Holiday and Vacation cards. Control is then returned to subroutine CONTRL.

Master File Update Links--UPDATE, UPDATA, and UPDATB

Links UPDATE, UPDATA, and UPDATB read, sort, and merge the transaction cards with the old master file creating an updated master file. Both the cards and the master file are checked for errors and a master file listing is generated if it has been requested on the initial cards.

Link UPDATE contains eight subroutines: UPDAT, ICDE, OCDE, LOCDE, INERR, HPSVE, DALG, and GLAD.

Links UPDATA and UPDATB are single subroutine links containing subroutines UPDAT1 and UPDAT2, respectively.

The external sort used in UPDAT2 is the 600/SM sort with user coding elements. (See GE-625/635 Sort/Merge, CPB-1005.) These user coding elements and the internal sort in UPDAT1, both call subroutines ICDE, OCDE, and LOCDE.

LINK UPDATE SUBROUTINES

Subroutine UPDAT

Subroutine UPDAT checks the sort option on the initial card and calls link UPDATA if an internal sort has been requested. If the external sort has been requested, link UPDATB is called.

Subroutine ICDE

Subroutine ICDE reads the transaction cards and checks the legality of the transaction codes and event numbers. If an error is found, an error message is written by using subroutine INERR. When a card is found which passes the error checks, control is returned to the calling sort routine. If a card with an "E" in column 1 is found, an "end of cards" indicator is set before the return.

Subroutine OCDE and INERR

Subroutine OCDE accepts a sorted card and merges it with the old master file. If an update to a nonexistent activity or a duplicate card is found, subroutine INERR writes an error message. A change code is attached to each updated record and control is sent to subroutine DALG.

Subroutine DALG

Subroutine DALG writes the master file record on the New Master File and writes the master file listing, if this listing has been requested. Control is then sent to subroutine GLAD.

Subroutine GLAD

Subroutine GLAD error checks all fields of the master file record, printing all errors found through the use of the subroutine INERR. As each error is found, a standard correction is made, to allow calculation to continue. When all fields have been checked, control is returned to DALG.

Subroutine HPSVE

DALG then saves the event numbers and all numeric data, such as time values, for the Topology section of the program. When the buffers are filled, these values are written on the Event Number File (NPAIR) and the Numeric Data File (NUMDAT), respectively. NUMDAT is written by the GECOS subroutine GESAVE which is called by HPSVE. (See GE-625/635 Comprehensive Operating Supervisor, CPB-1195.) Control is returned to OCDE.

When OCDE needs the next card, control is returned to the calling sort routine.

Subroutine LOCDE

When the sort being used no longer has any cards, subroutine LOCDE is called. LOCDE passes any records left on the old master file to subroutine DALG in the same manner as OCDE does. When all of the old master file has been passed to the new master file, LOCDE writes any data left in the buffers on NPAIR and NUMDAT and writes an "end-of-data" record on the new master file. Control is then returned to CONTRL.

CONTRL saves the holiday table for use by the calendar routine in the output section and rewinds all files. A return is then made to the Link Control Program.

THE HOLIDAY TABLE

The holiday table contains 501 locations; its values are integers representing the number of days from the base date of the calendar.

HIERARCHY OF CHANGE CODES

Change codes are attached to the master file for use by the output section. Only the most important change is used. The order of importance is as follows:

a) New Activity

b) Activity Code Change

c) Actual Date

d) Activity Time

e) Scheduled Date

f) Level Code

g) Activity Description

h) Event Description

i) Interface Flag

j) Short Path Flag.

FILE MAINTENANCE RECORD FORMATS

The File Maintenance Section produces the files, NPAIR and NUMDAT for use in the Topology Section, and a new master file.

NPAIR Record Format

The NPAIR record contains 318 words (159 pairs) in the following format:

First Word

Bits 0 - 8 Zero

9 - 35 Predecessor Event Number

Second Word

Bits 0 - 8 Zero

9 - 35 Successor Event Number

NUMDAT Record Format

The NUMDAT record contains 322 words in the following format:

Bits 0 - 1 Date Flag
2 - 3 Short Path Flag
4 - 17 Scheduled or Actual Date
18 - 21 Zeros
22 - 35 Activity Duration

Old and New Master File Format

The old and new master files have the same format. There are three types of records: the header record, the activity records, and the end of file record.

A) Header Record (1st Record) Format — File Maintenance Section

Word	Contents	Word Format*
1	Run Number	NNbbbbb
2	Report Date Day Value	NNbbbbb
3	Report Date Month Value	AAAbbbb
4	Report Date Year Value	NNbbbbb
5	Network Start Day Value	NNbbbbb
6	Network Start Month Value	AAAbbbb
7	Network Start Year Value	NNbbbbb
8	Network Completion Day Value	NNbbbbb
9	Network Completion Month Value	AAAbbbb
10	Network Completion Year Value	NNbbbbb
11	System ID of Last Run	CCCCCC
12	User's ID	CCCCCC
13-18	First 36 Characters of Network Title	CCCCCC....CCCCCC
19	Master File Report Option	Nbbbbbb
20	Summarization Option	Cbbbbbb
21	Report Date Option	Nbbbbbb
22	Number of Days in Work Week	Nbbbbbb
23-27	Last 30 Characters of Network Title	CCCCCC....CCCCCC
28	System ID for this Run	CCCCCC
29	Internal-External Sort Indicator	Nbbbbbb
30	Run Type Indicator	Abbbbbb

* b = Blank
N = BCD, Numeric
A = BCD, Alpha
C = BCD, Alpha or Numeric

B) Activity Records Format

Word	Contents	Word Format
1	Transaction Code	Nbbbbbb
2	Short Path Option	Nbbbbbb
3	Schedule Date Option	Nbbbbbb
4	Pred. Event Interface Flag	Abbbbbb
5-6	Pred. Event Number	NNNNNN - NNbbbbb
7	Pred. Event Summary Level	Abbbbbb
8	Succ. Event Interface Flag	Abbbbbb
9-10	Succ. Event Number	NNNNNN - NNbbbbb
11	Succ. Event Summary Level	Abbbbbb
12	Pessimistic Time Estimate	NNNNbb
13	Most Likely Time Estimate	NNNNbb
14	Optimistic Time Estimate	NNNNbb
15	Schedule Date	NNNNNN
16-21	Event Title	CCCCCC...CCCCCC
22	First 4 Characters of Activity Code	CCCCbb
23	Second 4 Characters of Activity Code	CCCCbb
24	Third 4 Characters of Activity Code	CCCCbb
25	Last 6 Characters of Activity Code	CCCCCC
26-31	Activity Title	CCCCCC...CCCCCC
32	Actual Date	NNNNNN
33	Change Code	Integer

C) End of File Record Format

Word	Contents	Word Format
1	End of File Indication**	E6666666
2-33	Anything	

** Not to be confused with an end of file mark.

FILE MAINTENANCE SECTION FLOWCHARTS

The following pages contain the flowcharts for the File Maintenance Section listed in the following order:

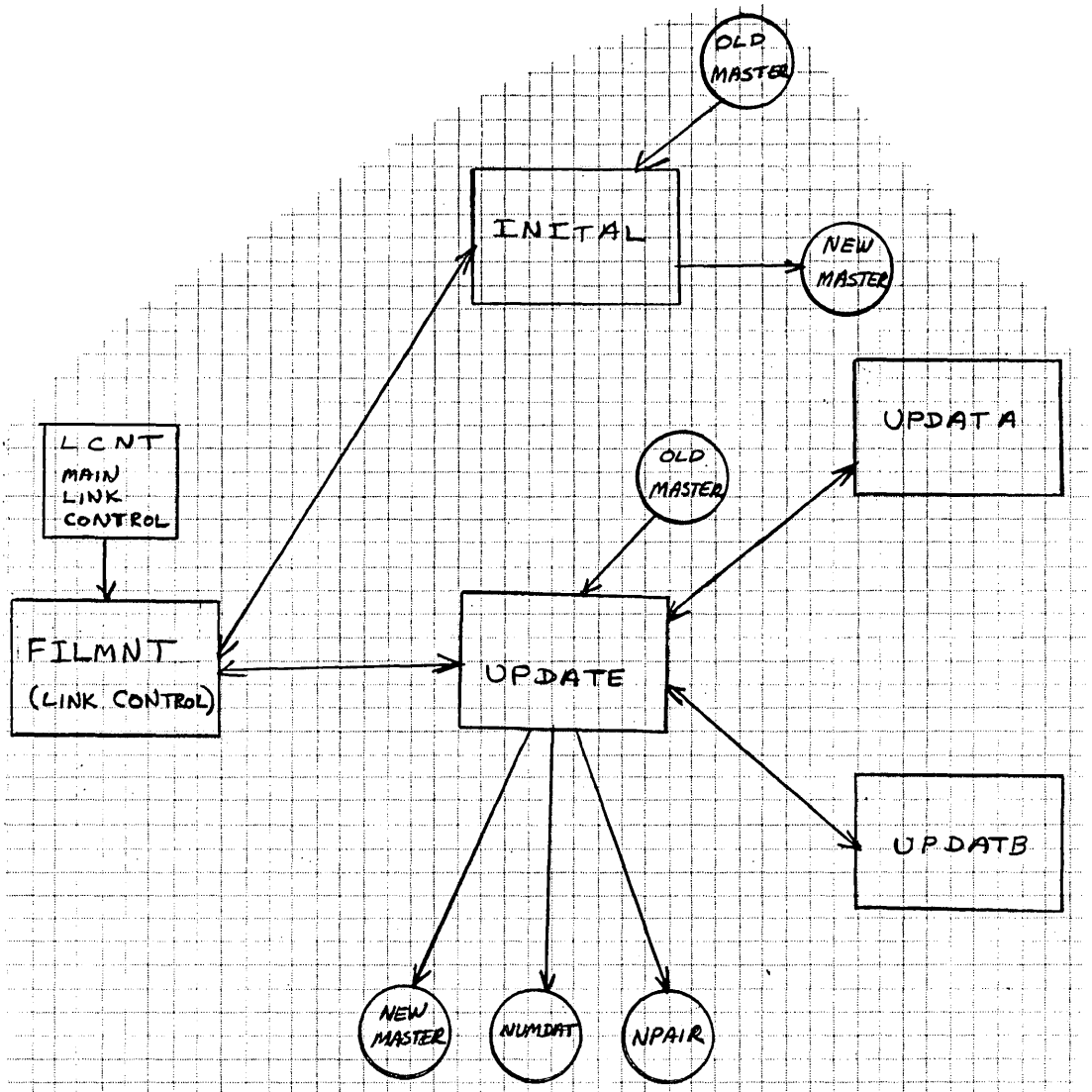
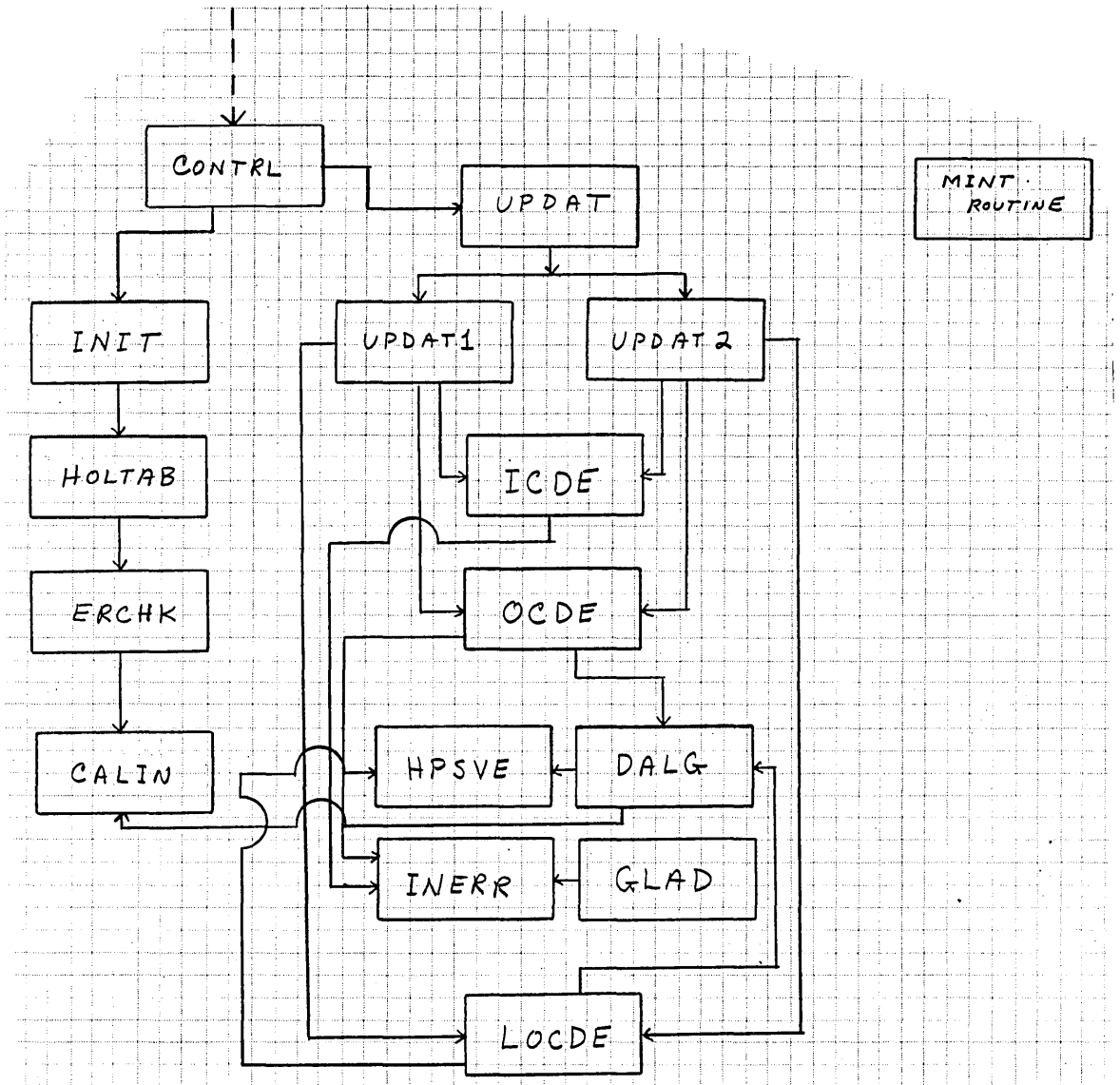


Figure 4. FILMNT Linkage and File Relationships



CONTR calls INIT, then UPDAT.
INIT calls HOLTAB.
HOLTAB calls ERCHK.
ERCHK calls CALIN.

CALIN, HPSVE, INERR } do not make calls.

UPDAT calls UPDAT1 or UPDAT2.
UPDAT1 and UPDAT2 call ICDE, OCDE, and LOCDE.
ICDE calls INERR.
OCDE calls DALG and INERR.
LOCDE calls HPSVE and DALG.
DALG calls HPSVE, CALIN, and GLAD.
GLAD calls INERR.

MINT is a routine used by all the subroutines.

Figure 5. FILMNT Subroutine Relationships

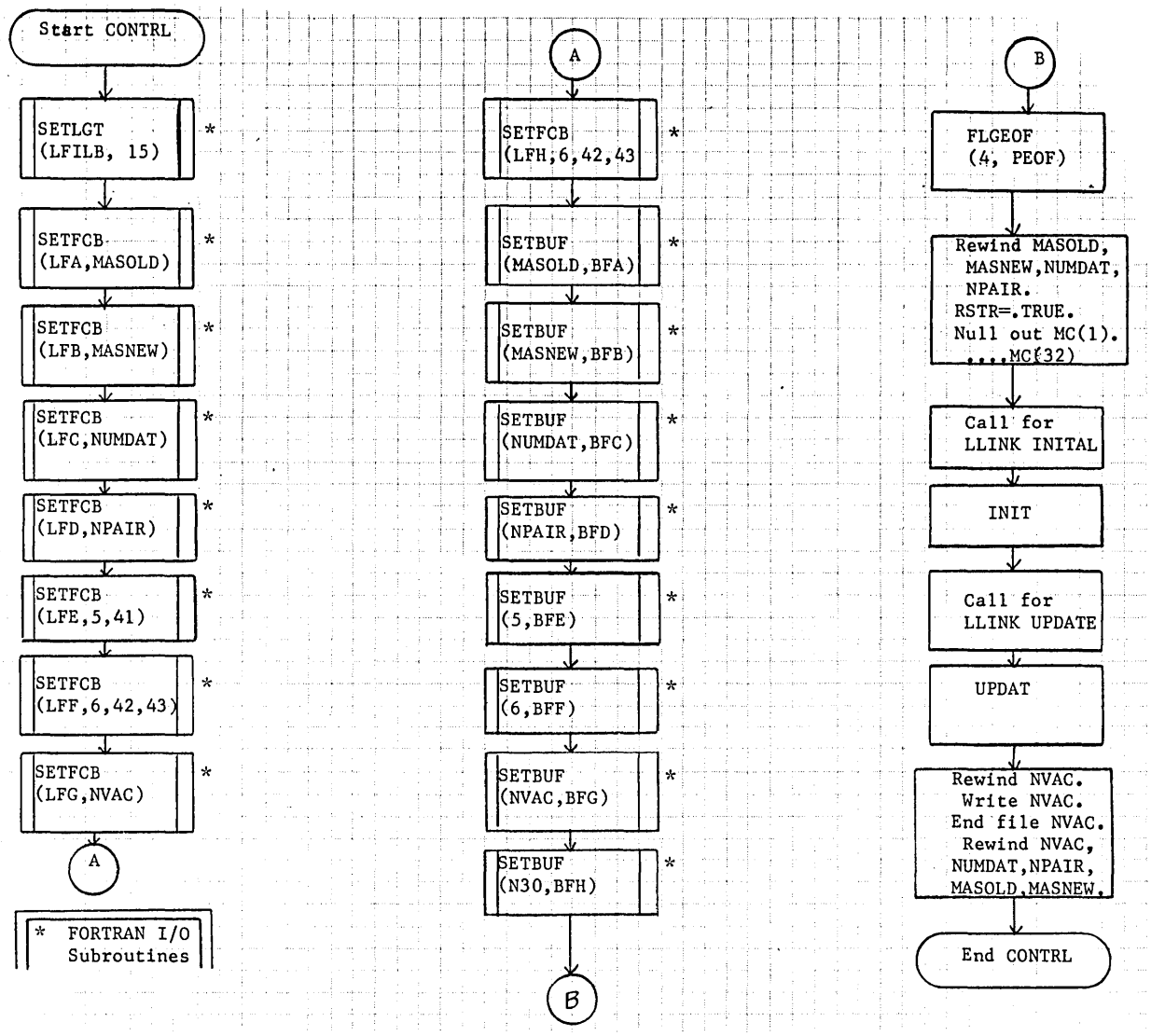


Figure 6. CONTRL Subroutine Flowchart

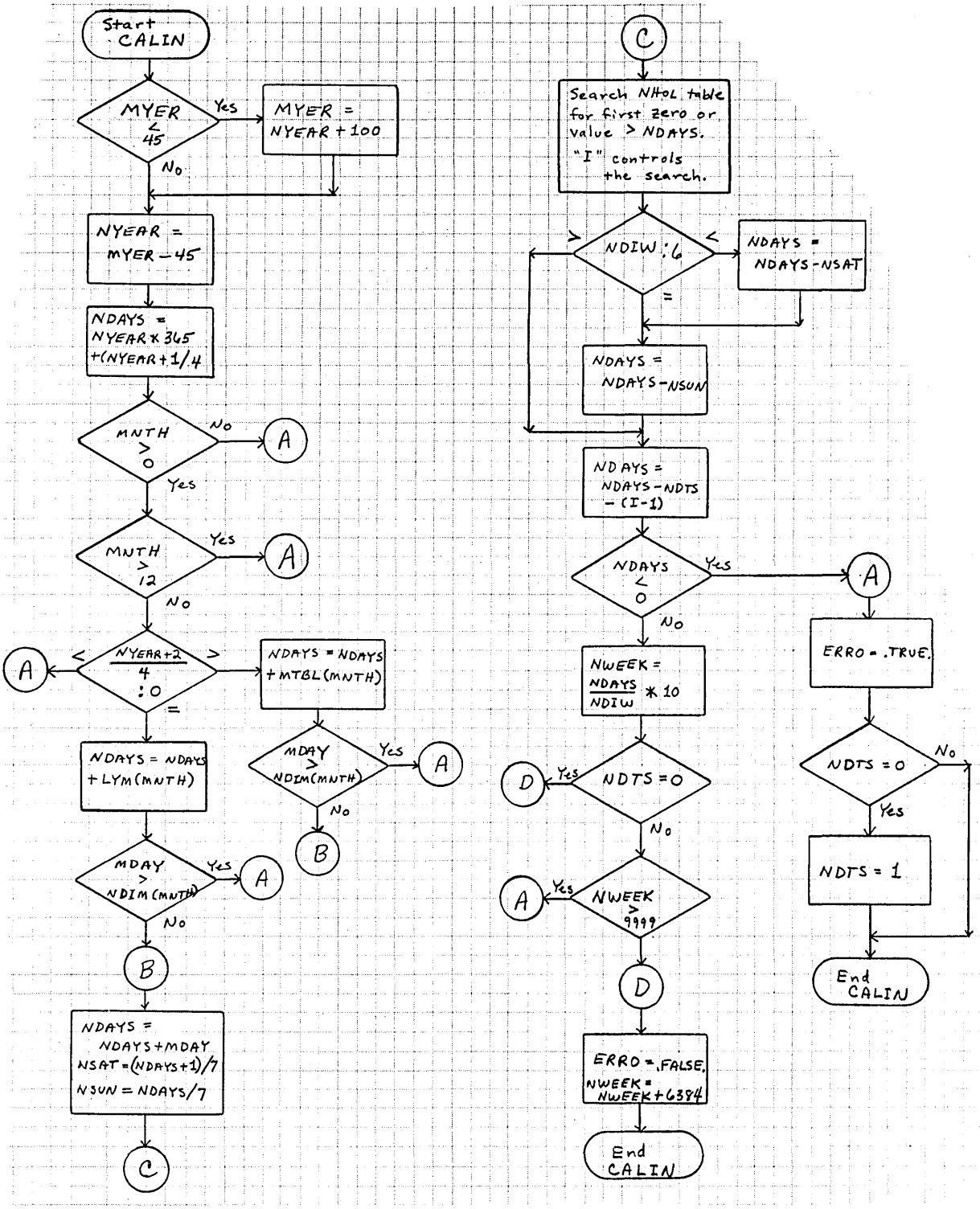


Figure 7. CALIN Subroutine Flowchart

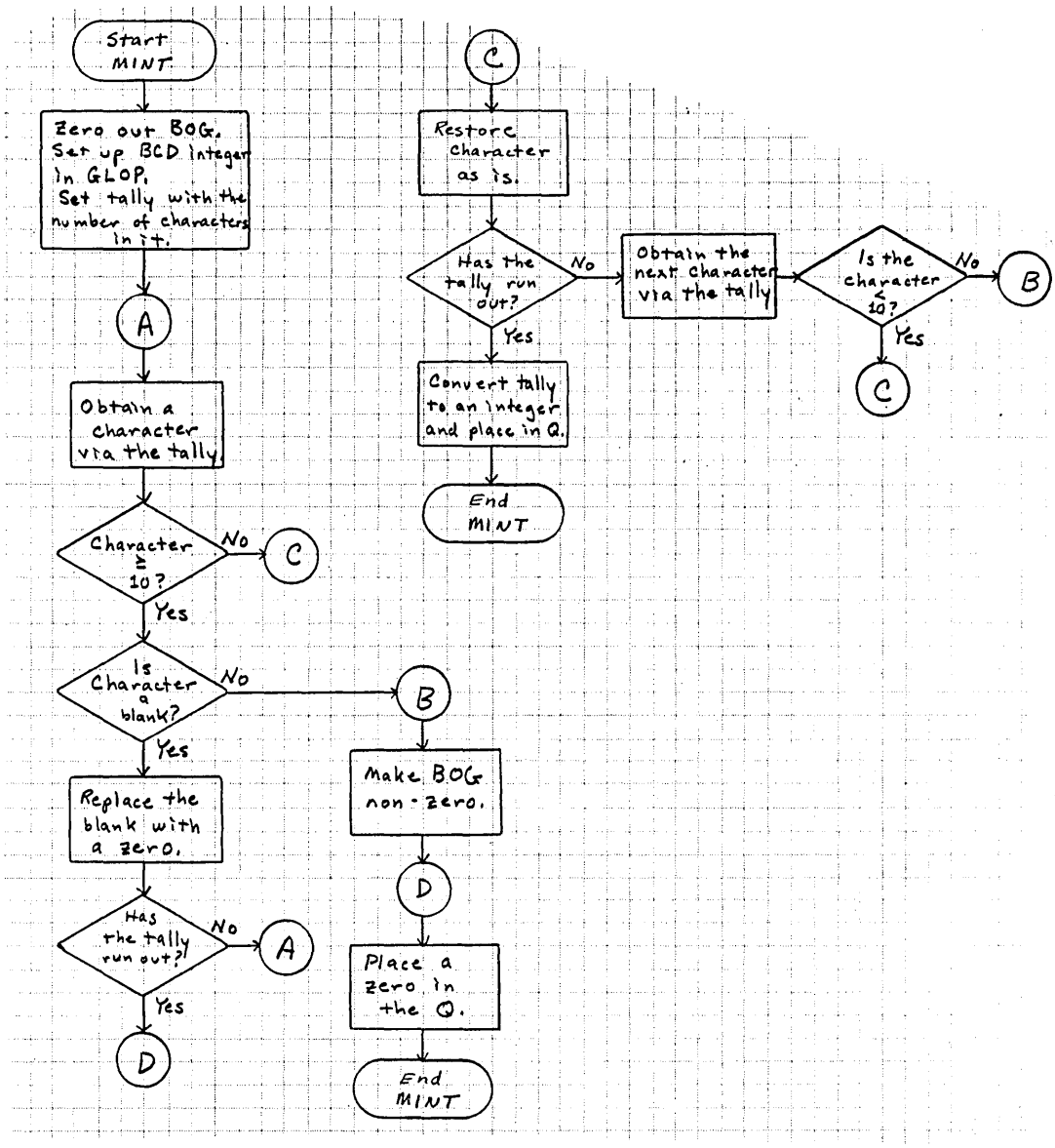


Figure 8. MINT Subroutine Flowchart

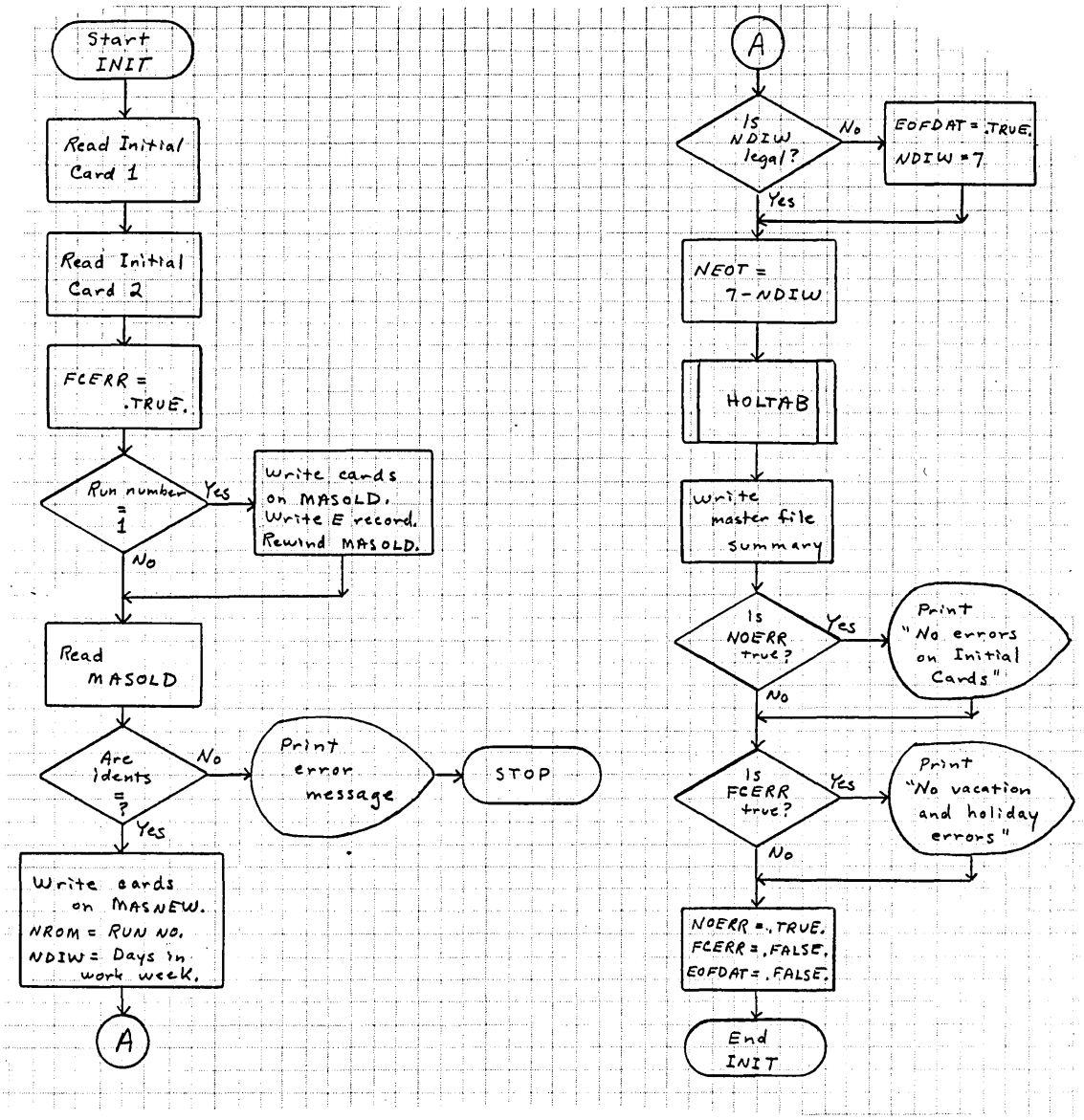


Figure 9. INIT Subroutine Flowchart

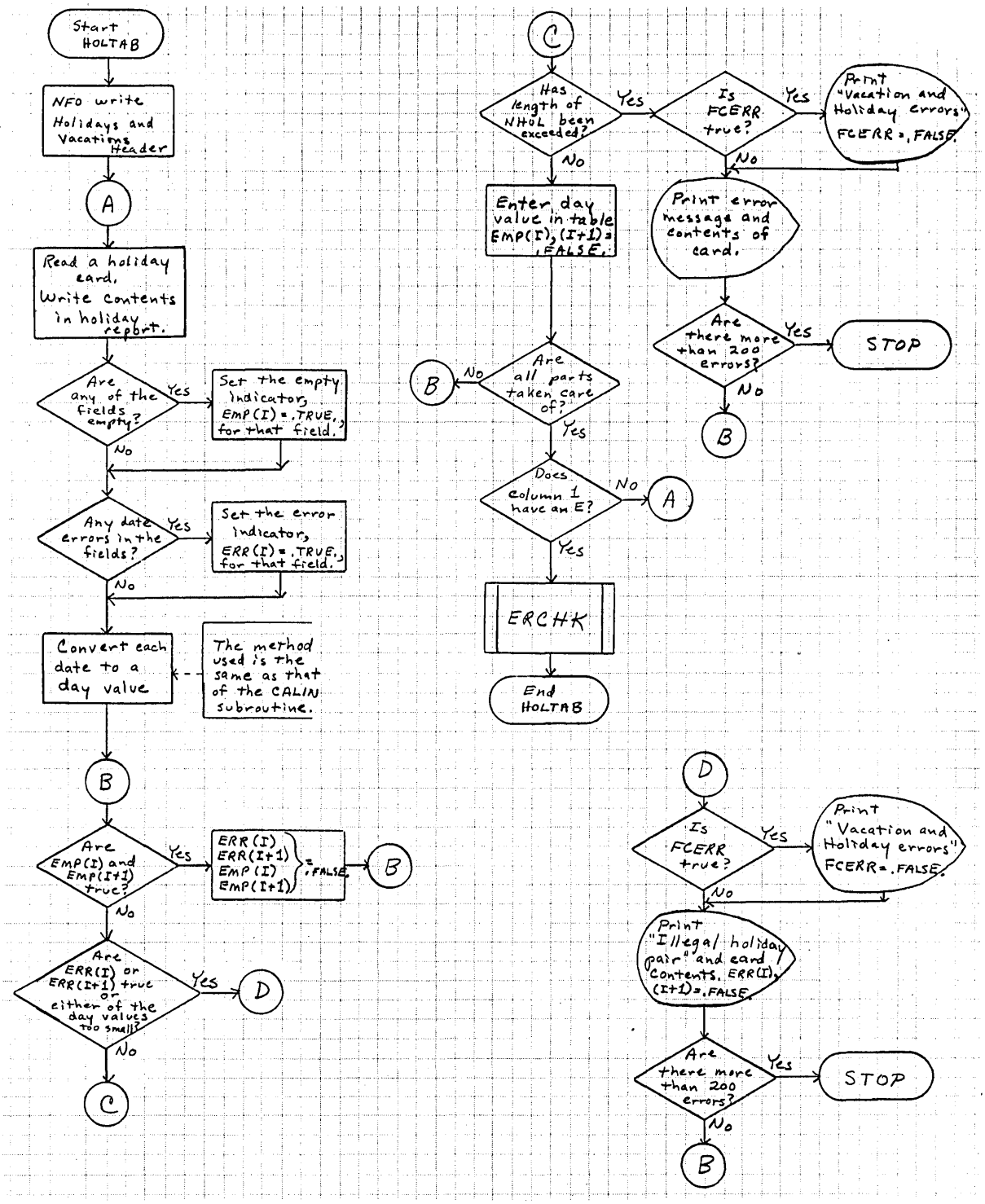


Figure 10. HOLTAB Subroutine Flowchart

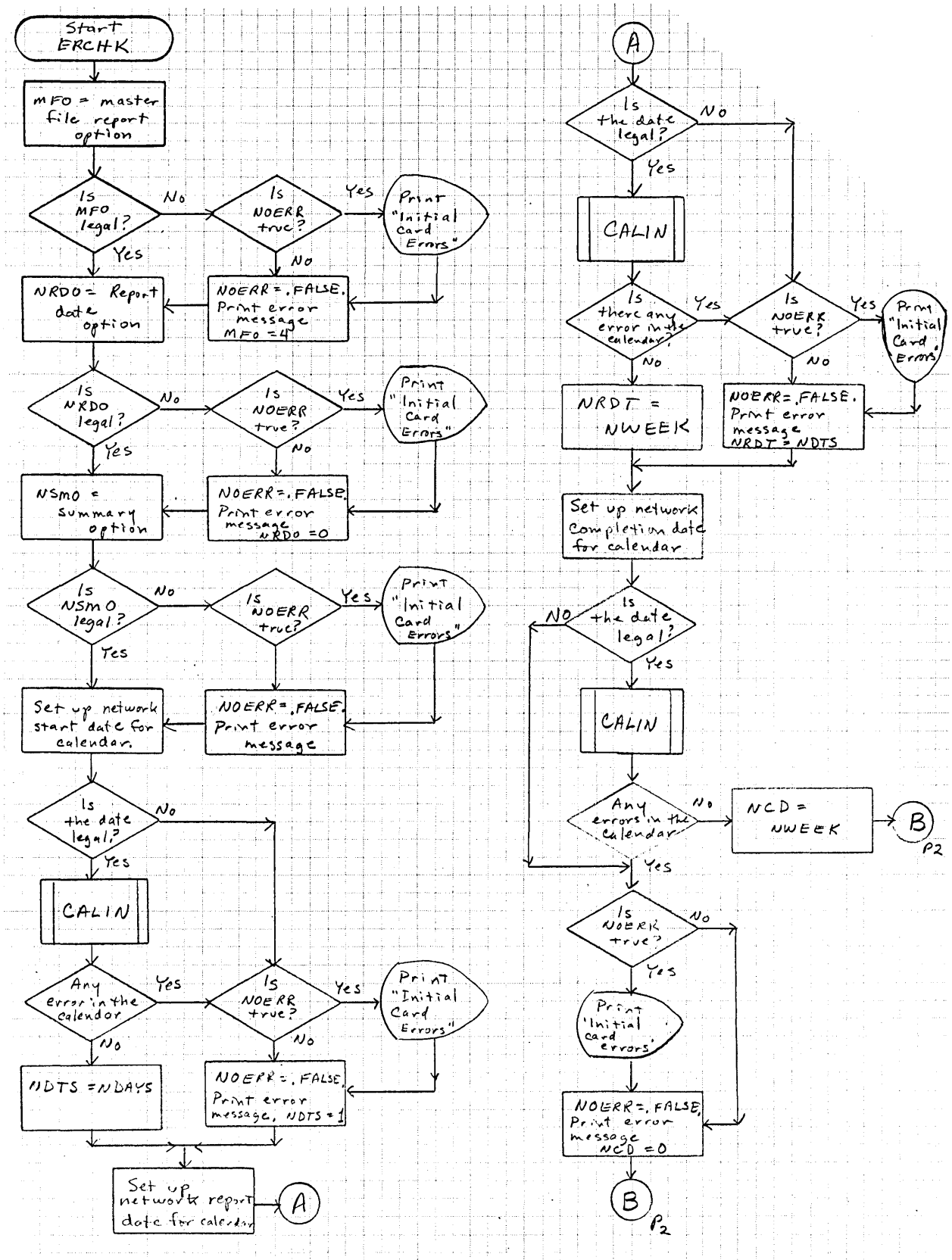


Figure 11. ERCHK Subroutine Flowchart P1

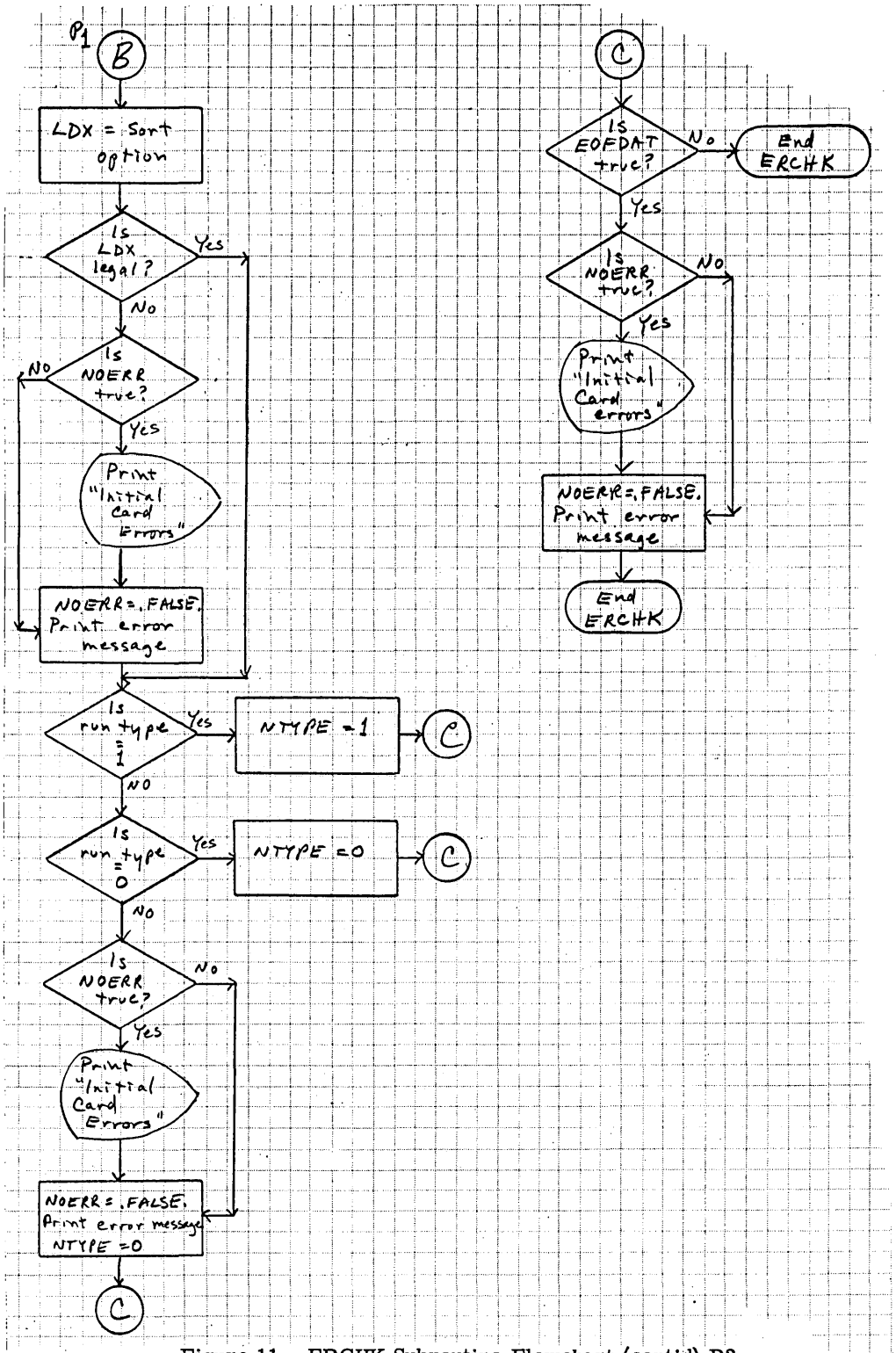


Figure 11. ERCHK Subroutine Flowchart (cont'd) P2

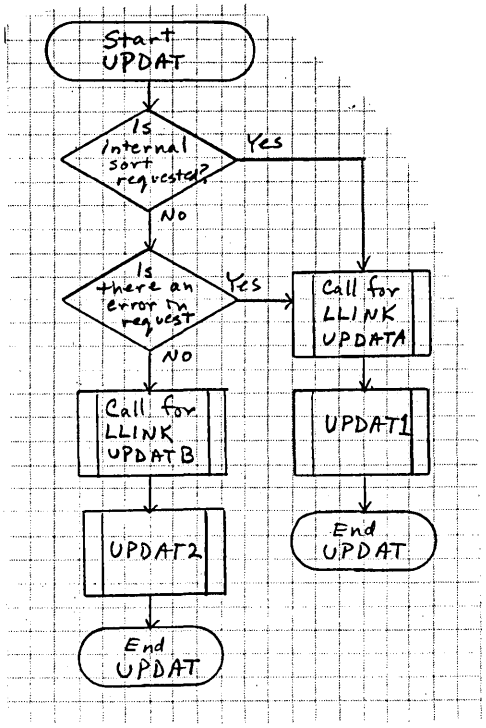


Figure 12. UPDAT Subroutine Flowchart

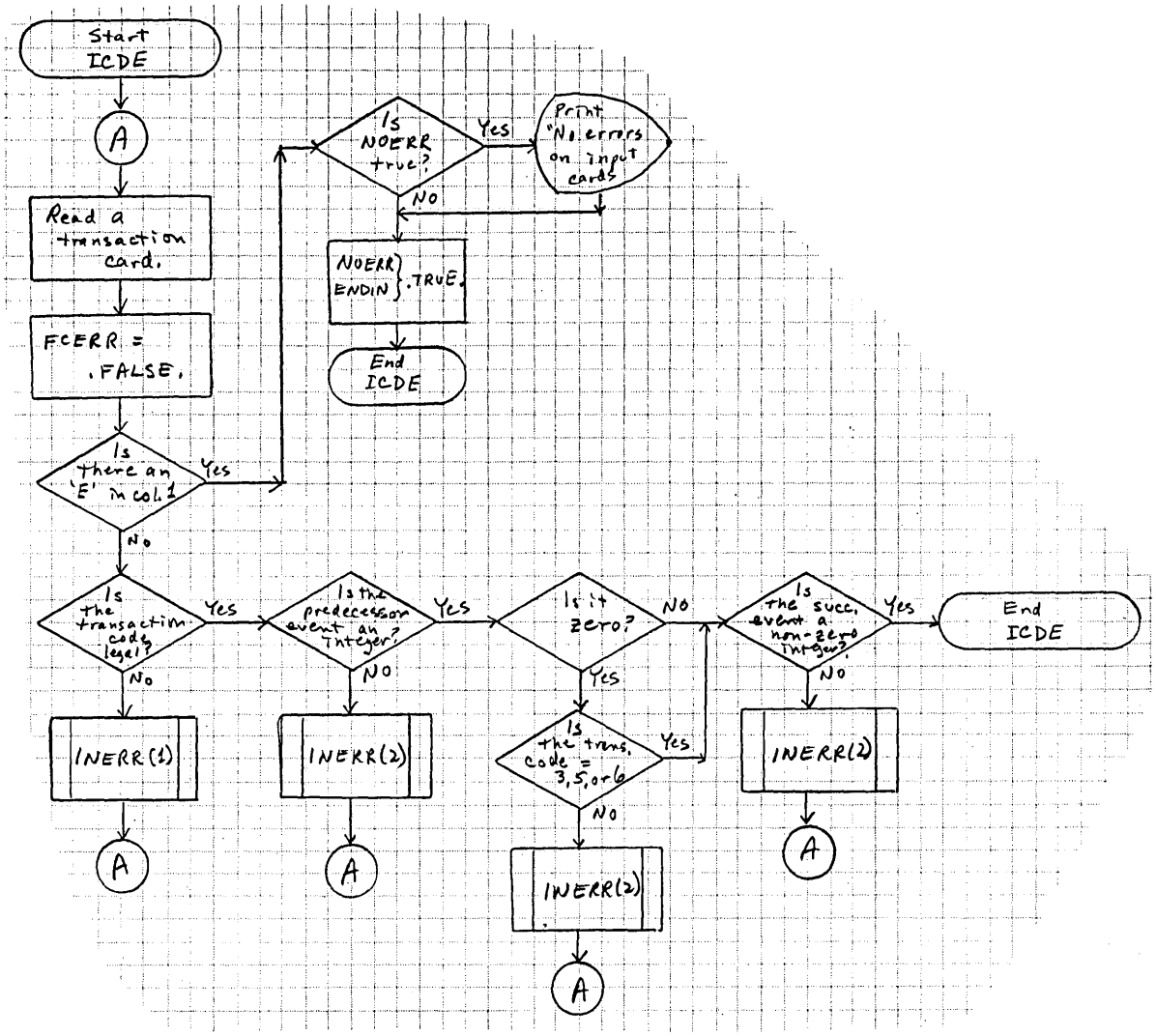


Figure 13. ICDE Subroutine Flowchart

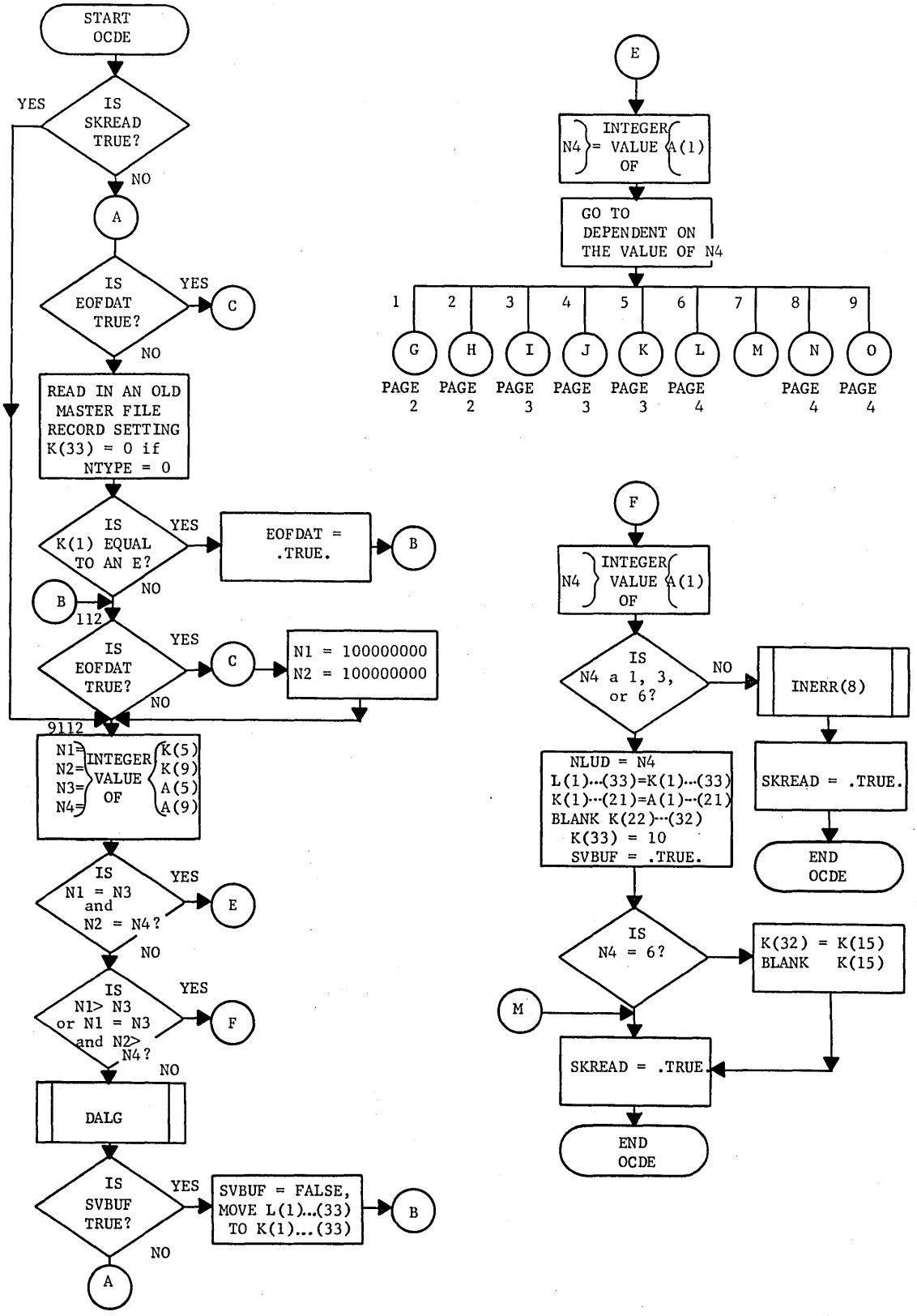


Figure 14. OCDE Subroutine Flowchart Page 1 of 4

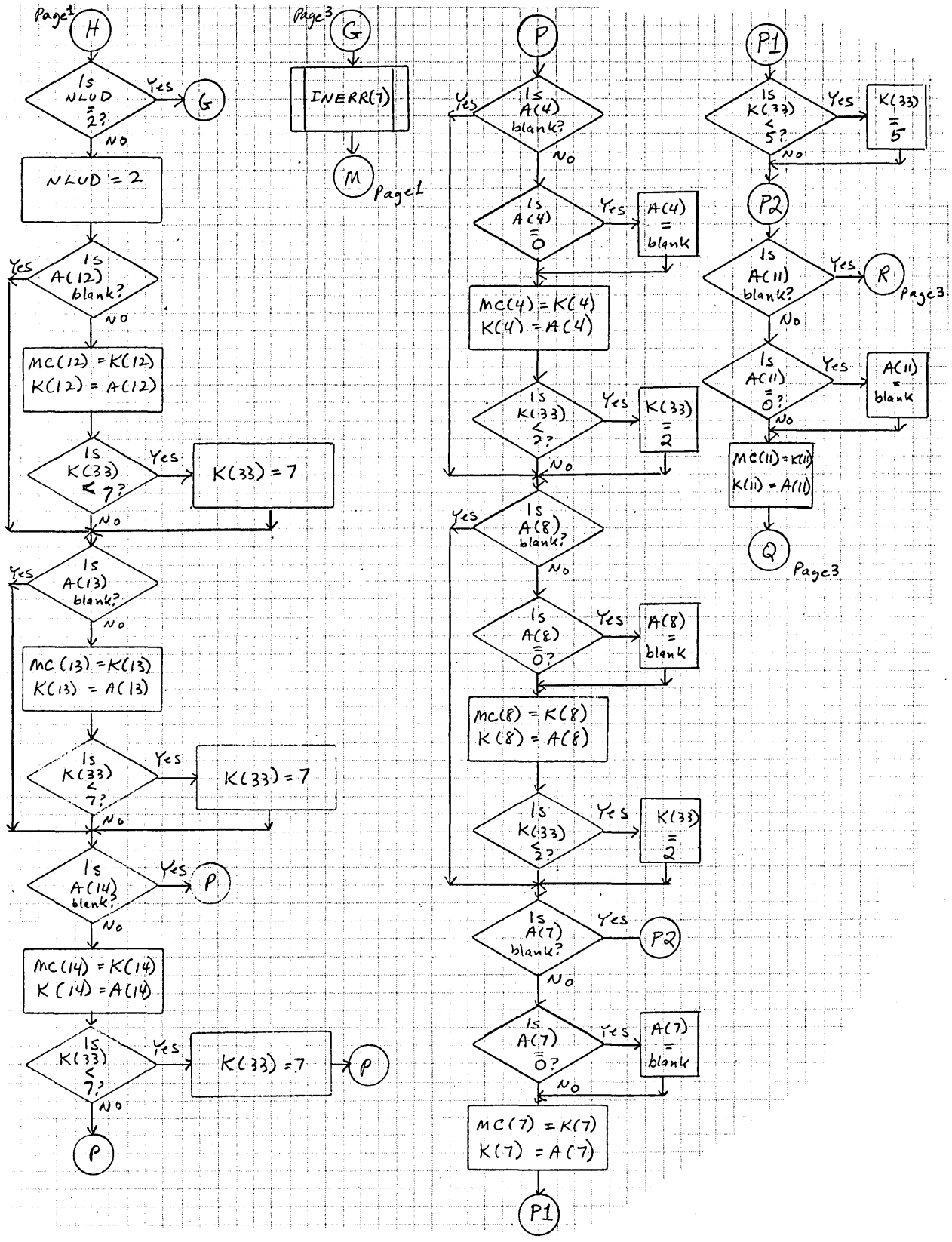


Figure 14. OCDE Subroutine Flowchart (cont'd) Page 2 of 4

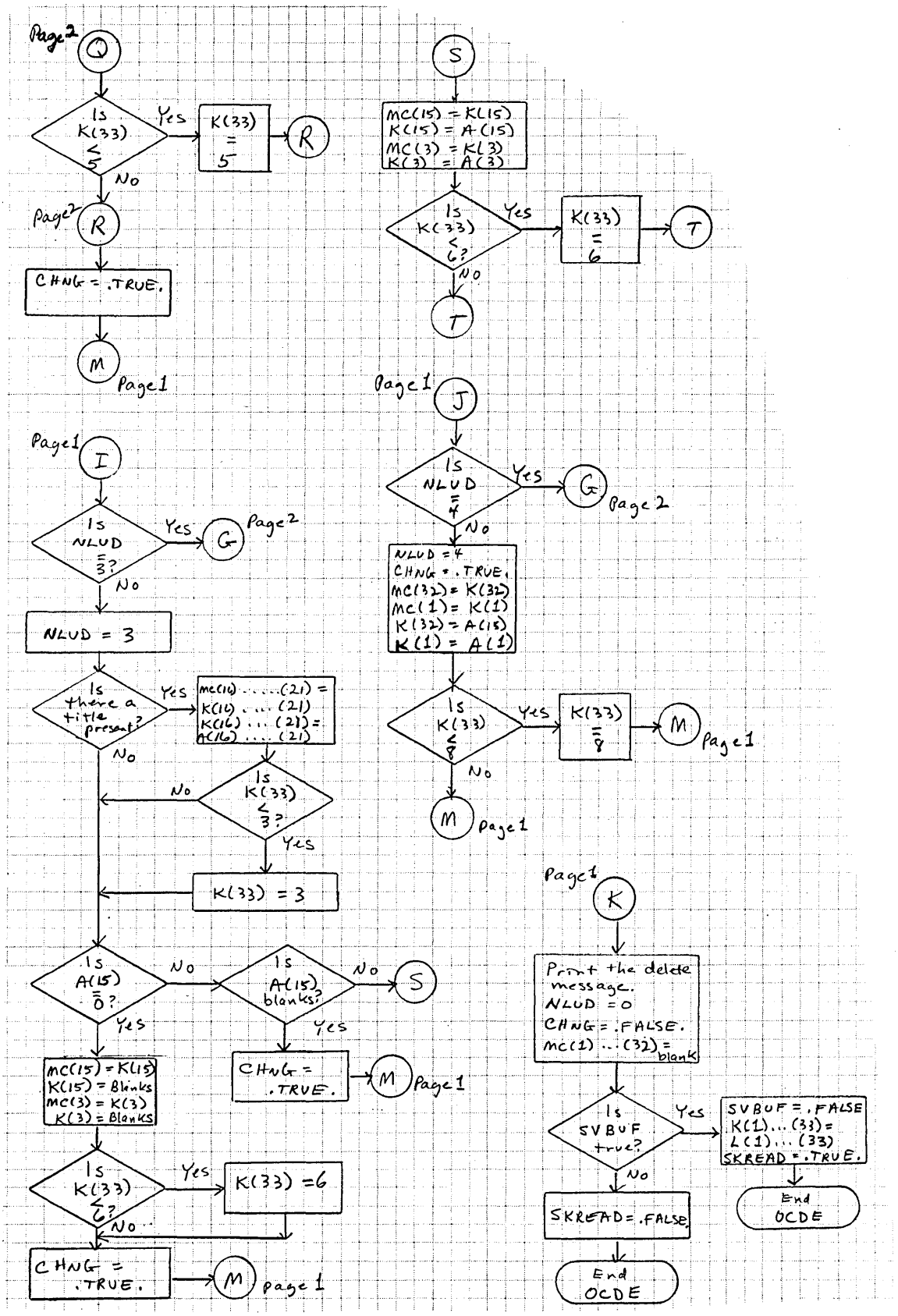


Figure 14. OCDE Subroutine Flowchart (cont'd) Page 3 of 4

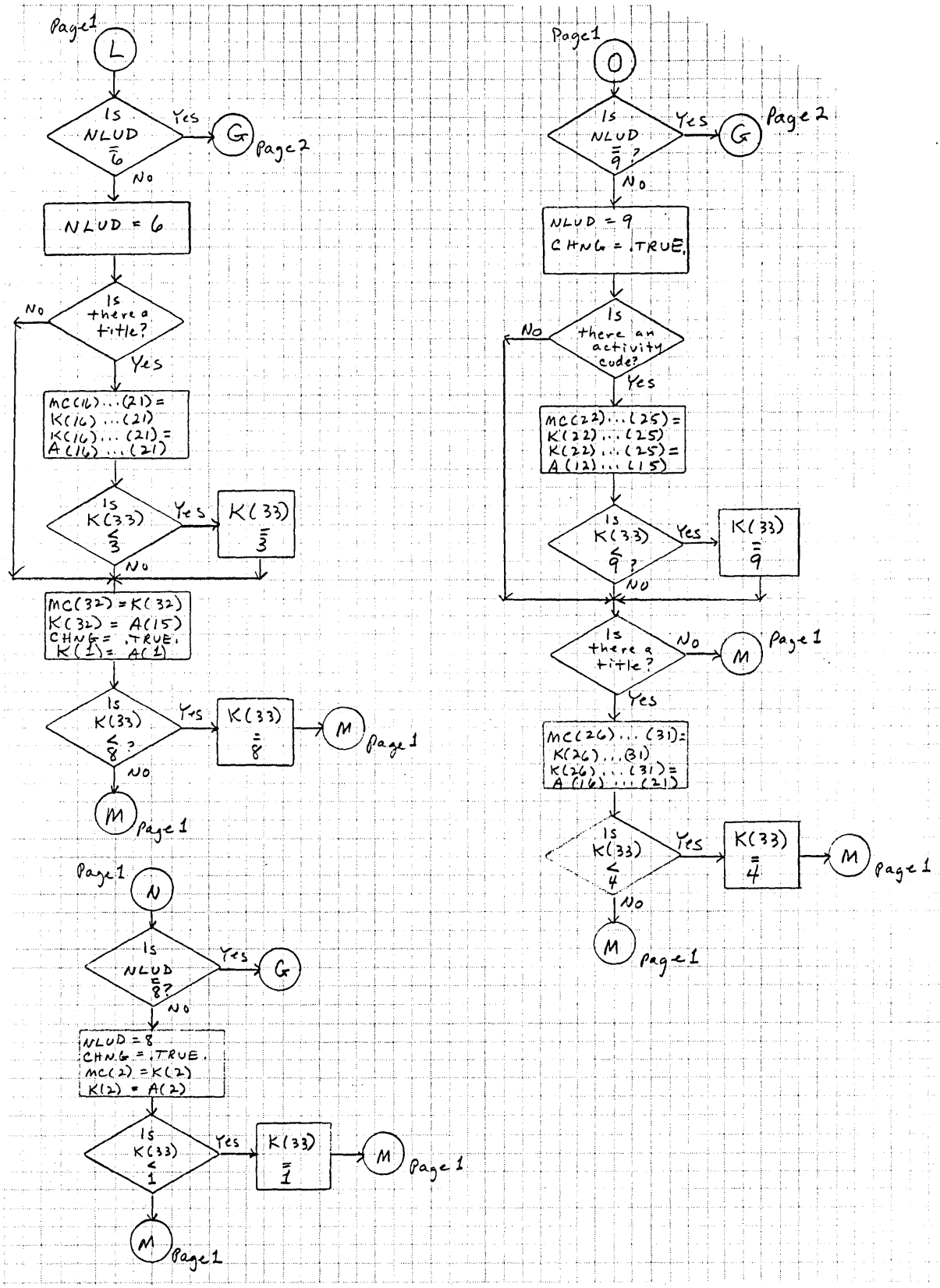


Figure 14. OCDE Subroutine Flowchart (cont'd) Page 4 of 4

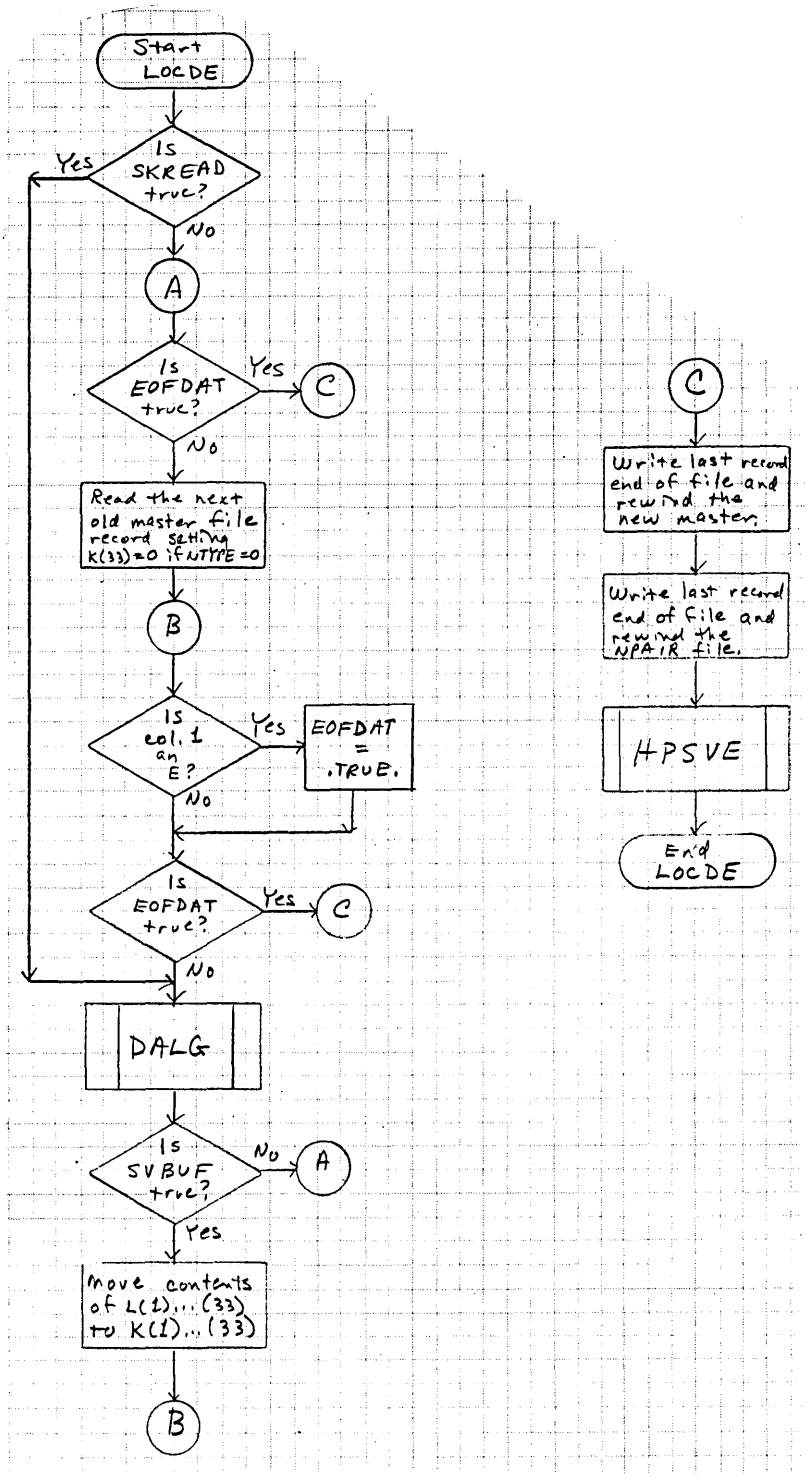


Figure 15. LOCDE Subroutine Flowchart

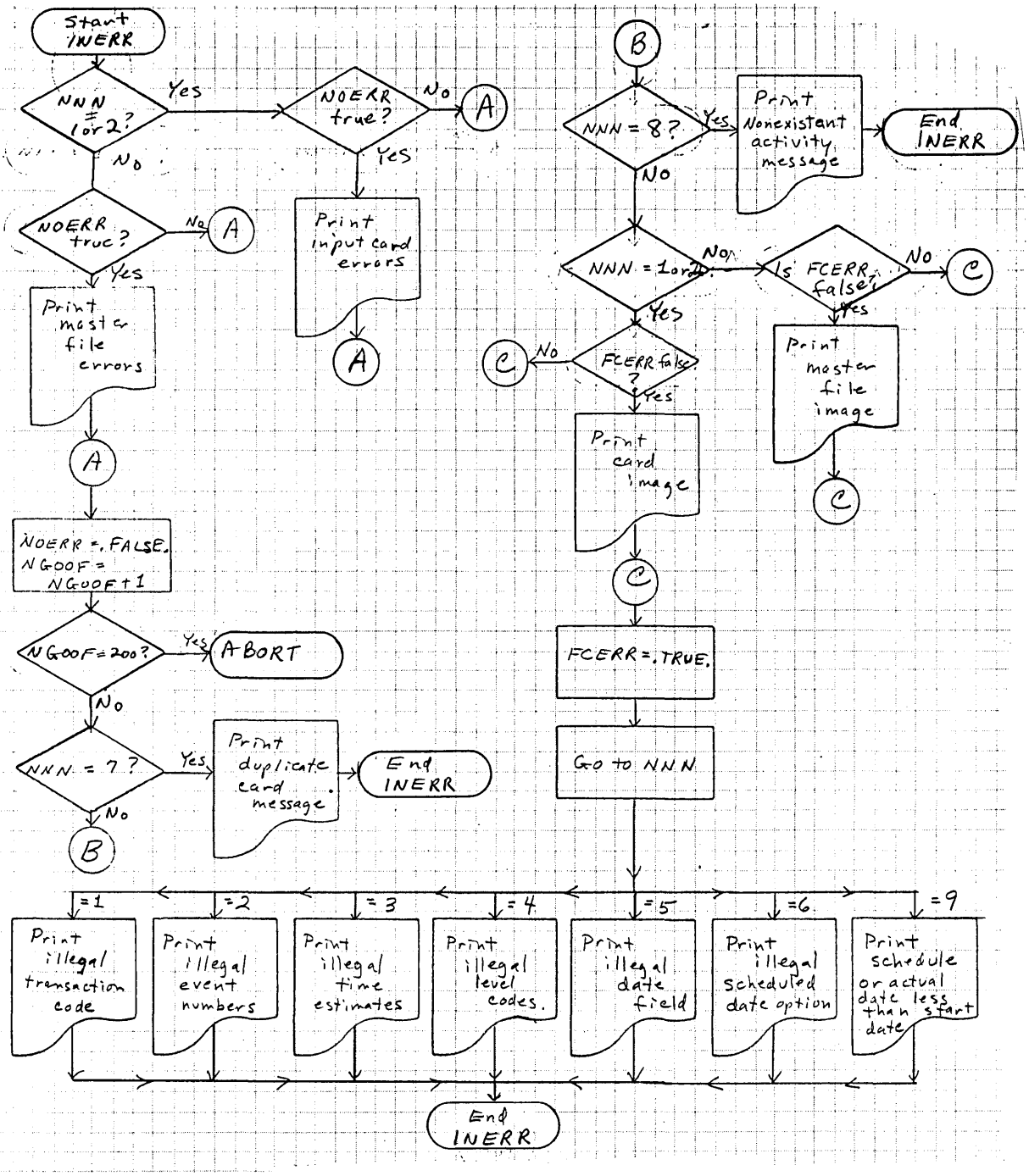


Figure 16. INERR Subroutine Flowchart

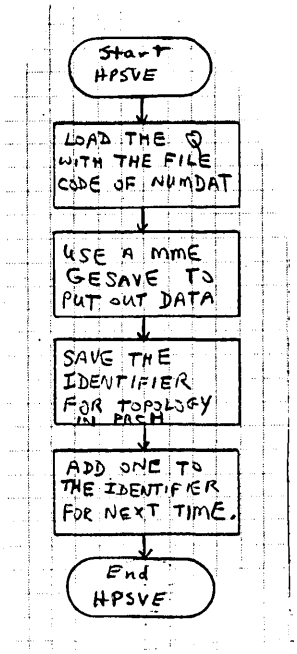


Figure 17. HPSVE Subroutine Flowchart

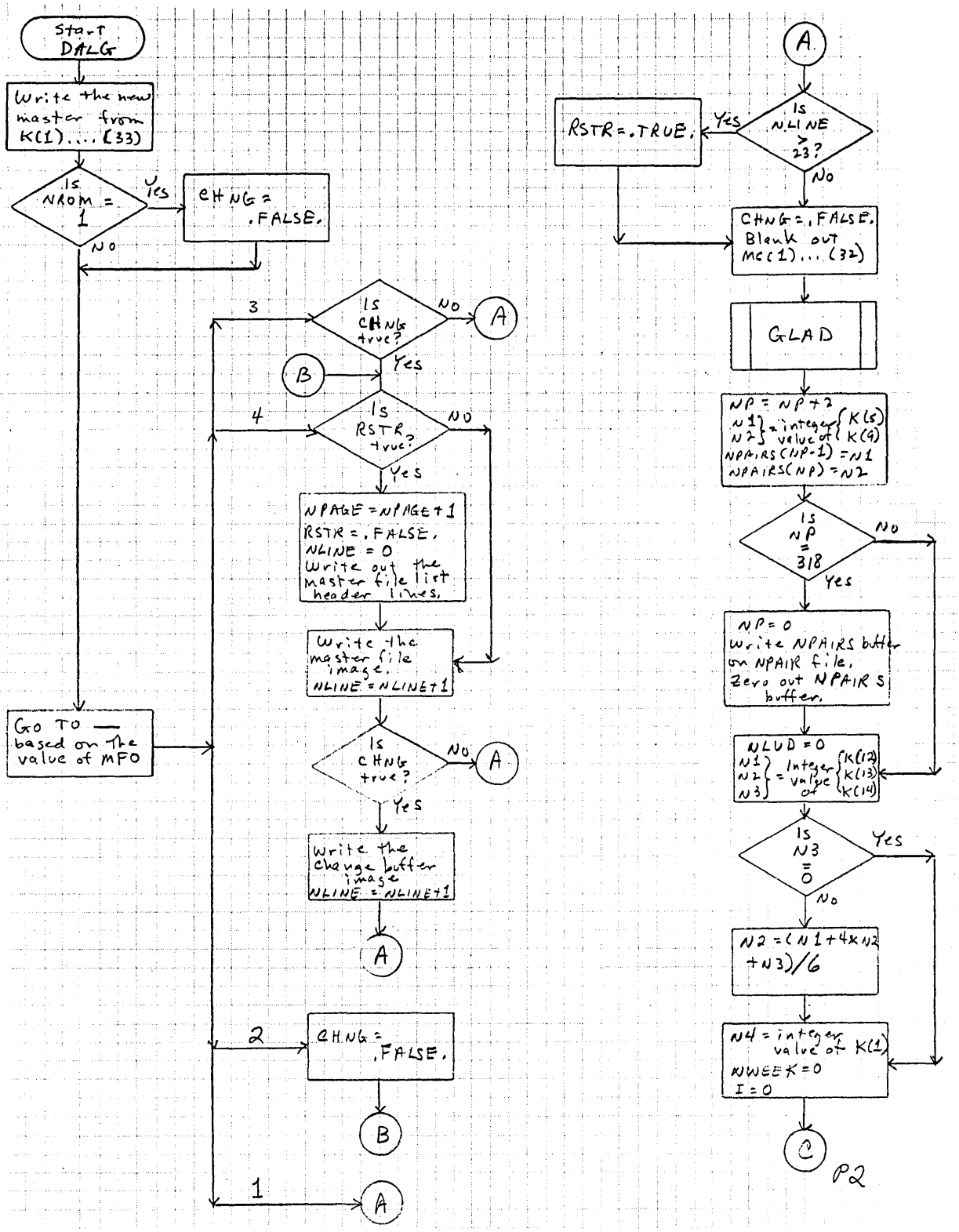


Figure 18. DALG Subroutine Flowchart P1

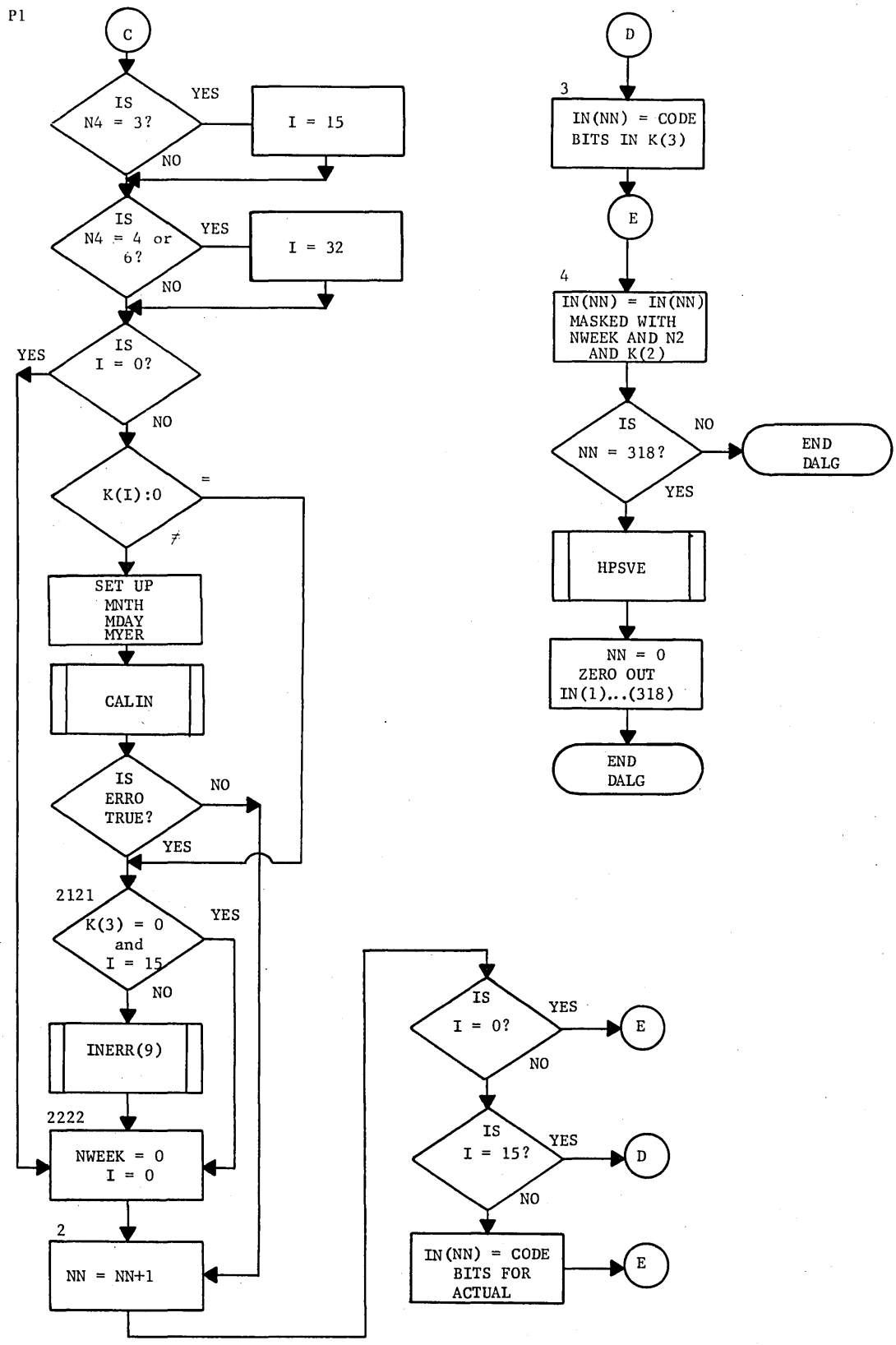


Figure 18. DALG Subroutine Flowchart (cont'd) P2

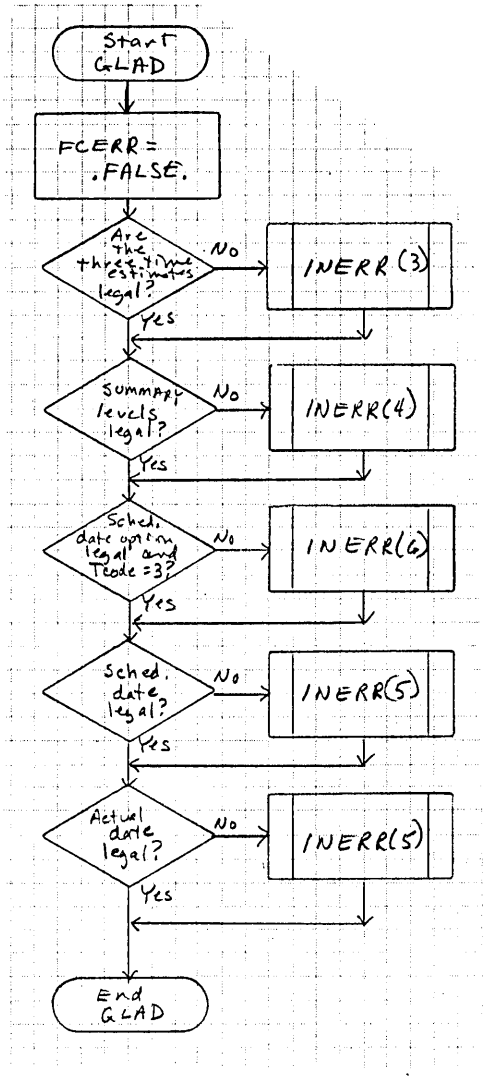


Figure 19. GLAD Subroutine Flowchart

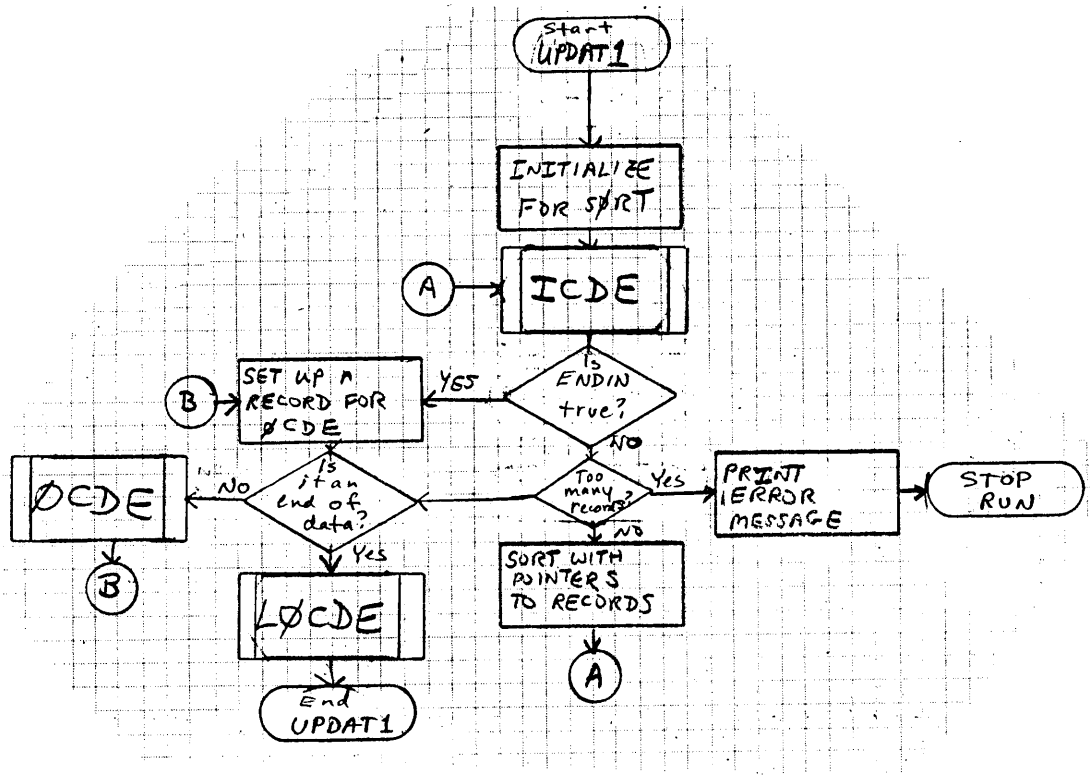


Figure 20. UPDAT1 Subroutine Flowchart

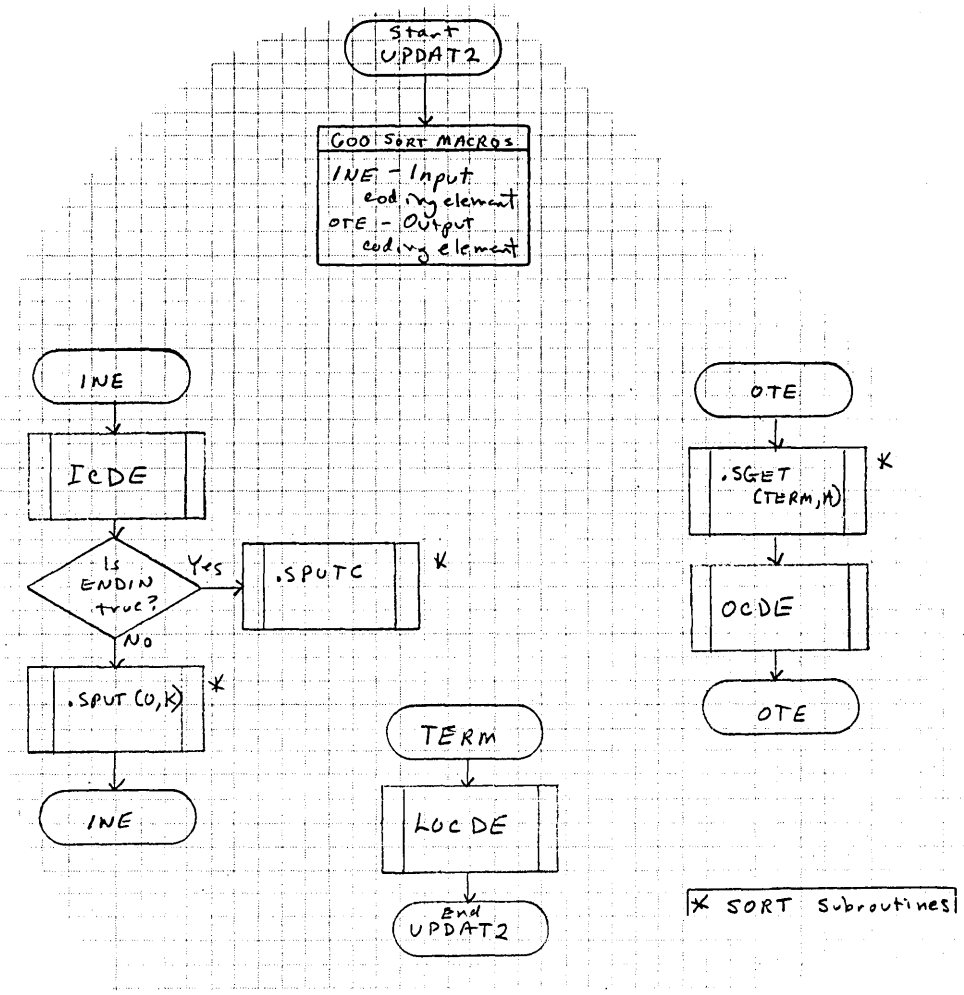


Figure 21. UPDAT2 Subroutine Flowchart

4. TOPOLOGY SECTION - TOPOLO

The Topology Section (TOPOLO) consists of 20 subroutines contained in 14 links. These subroutines perform the necessary operations on the data in the internal file NPAIR to calculate and produce the information used in performing Kahn's Method. (See "Topological Sorting of Large Networks", by Dr. A.B. Kahn, Communications of the ACM, November 1962.) This section also does the basic calculation of the expected completion date (TE) and the latest completion date (TL), adding them to the master file.

TOPOLO LINKAGE

The functions of the 14 links and the manner in which the subroutines perform these functions are described below. Flowcharts for these links and subroutines are at the end of this chapter.

Control Link--MAINP

MAINP is the link control program for both the TOPOLO and SUMMRY sections. If a circular network exists, it is detected after the subroutine STARTF; in this case, any sequenced activities are deleted, and the network is reversed. The process (through STARTF) is repeated, and the remaining activities constitute the circular subset.

If there is no circular part, the network is renumbered, sorted topologically, and the numeric data is added to the records. If summarization has been requested, it is done here. Then the TE and TL are calculated, and the results are added to the master file.

Read NPAIR Link--PROGA, Subroutine STARTA

Subroutine STARTA, contained in link PROGA, reads the activity pairs into COMMON storage and counts the number of pairs into NACT. The list may be terminated by an EOF or by a zero appearing as a successor. At the time the pairs are read, they are in order by predecessor, successor, and have been converted to binary (27 bits). If the number of activities exceeds program capacity, a comment is made.

The following is a description of the output record format for STARTA.

First Word

Bits
0 - 8 Zero
9 - 35 Predecessor event number

Second Word

Bits
0 - 8 Zero
9 - 35 Successor event number

Activity Scan Link--PROGB

Link PROGB contains the subroutines STARTB, STARTC, and STARTD.

SUBROUTINE STARTB

Subroutine STARTB, contained in link PROGB, performs the following three functions:

1. Adjusts NACT for indexing (multiplies by 3).
2. Expands the activities to occupy 3 words each instead of 2. As this expansion is made, STARTB rearranges and breaks up the contents into the forthcoming sort routine to obtain the greatest advantage.
3. Adds one activity (high value) for the scan termination indicator.

The following is a description of the input and output record formats for STARTB.

Input

First Word

Bits
0 - 8 Zero
9 - 35 Predecessor

Second Word

0 - 8 Zero
9 - 35 Successor

Output

First Word

0 - 26 Successor
27 - 35 Predecessor (first part)

Second Word

0 - 17 Zero (used by sort)
18 - 35 Predecessor (second part)

Third Word

0 - 18 Index Value (used by sort)
19 - 29 Not Used
30 - 35 Index register 7 (used by sort)

SUBROUTINE STARTC

Subroutine STARTC sorts the activities by successor, predecessor. This is a logical sort, since no physical movement takes place; that is, physically, the activities are still in order by predecessor, successor. The logical sort is performed using a set of pointers in bits 0-17 in the third word of each record. The pointers are merged together using a buffer area in bits 0-17 in the second word of each record.

The following is a description of the record format for STARTC.

First Word

0 - 26 Successor event

27 - 35 Predecessor event (first part)

Second Word

0 - 17 Sort buffer area

18 - 35 Predecessor event (second part)

Third Word

0 - 17 Logical pointers

18 - 29 Not used

30 - 35 XR7, Sort parameter locator

SUBROUTINE STARTD

Input to the subroutine STARTD is the physical list of activities by predecessor, successor and the logical list by successor, predecessor. STARTD performs a "simultaneous step-by-step scan" to calculate the 4 columns of information necessary for the utilization of Kahn's method.

The following is a description of the record formats for STARTD.

Input

First Word

0 - 26 Successor event number

27 - 35 Predecessor event number (first part)

Second Word

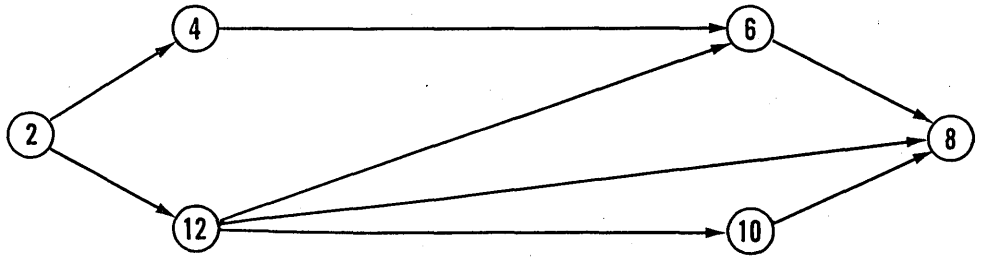
0 - 17 Logical pointers

18 - 35 Predecessor event number (second part)

Output is the third word in the following format:

0	Column A
1 - 15	Column B
16 - 20	Column C
21 - 35	Column D

Columns A and B are the activity list; columns C and D are the event list. Consider the network:



This network was used to compile the following list:

Pointer	Activities by Successor		Events	Activities by Predecessor		Location
0	2	4	2	2	4	0
6	4	6	4	2	12	3
15	12	6	6	4	6	6
9	6	8		6	8	9
12	10	8	8	10	8	12
18	12	8	10	12	6	15
21	12	10	12	12	8	18
3	2	12		12	10	21

The Activities by Successor List exists only through the logical pointers. The event numbers in the event list exist only at logic points during the "simultaneous step-by-step scan."

The following listings also refer to the network above. In these listings, Column A is a flag denoting the start of a group of activities having the same predecessor within the activity list. A "0" indicates the start of a new group. Column B locates each activity's successor event within the event list. Column C contains a count of the number of direct predecessor activities of each event. Column D locates within the activity list the first activity of a group having the associated event as a predecessor. If an event is terminal (has no successors), its column D entry contains a flag of 32767 (event #8 in the example).

Activity List					Event List			
Location	PRED No's.	Succ. No's.	A	B	Location	Event No's.	C	D
0	2	4	0	3	0	2	0	0
3	2	12	1	15	3	4	1	6
6	4	6	0	6	6	6	2	9
9	6	8	0	9	9	8	3	32767
12	10	8	0	9	12	10	1	12
15	12	6	0	6	15	12	1	15
18	12	8	1	9				
21	12	10	1	12				

Kahn's Method Link--PROGE

SUBROUTINE STARTE

Subroutine STARTE performs the basic manipulation as described in Kahn's Method.

Only the third word of the record is used and it is in the following format:

0	Column A
1 - 15	Column B
16 - 20	Column C
21 - 35	Column D

In the output record Column D is replaced by the event renumbering; Column C is replaced by a "sequenced flag" (31) for all events which are renumbered (hopefully, all); Column B is replaced by the renumbered value of each activity's predecessor event.

The following explanations refer to the example given in STARTD. The process is illustrated, step by step. At each stage, the interesting numbers are indicated by a square.

Frame 1

Location	A	B	C	D
0	0	3	0	0
3	1	15	1	6
6	0	6	2	9
9	0	9	3	32767
12	0	9	1	12
15	0	6	1	15
18	1	9		
21	1	12		

At the start, a search of column C in the event file reveals that location 0 is an event which can be sequenced (that is, an event whose column C entry is equal to zero).

Frame 2

Location	A	B	C	D
0	0	3	31	0
3	1	15	1	6
6	0	6	2	9
9	0	9	3	32767
12	0	9	1	12
15	0	6	1	15
18	1	9		
21	1	12		

The original column D entry "0" is stored, and the next available topological sequence number, 3, is inserted in column D. Column C is replaced by the "sequenced flag" (31).

Frame 3

Location	A	B	C	D
0	0	0	31	0
3	1	0	0	6
6	0	6	2	9
9	0	9	3	32767
12	0	9	1	12
15	0	6	0	15
18	1	9		
21	1	12		

Zero Table

3
15

The original column D entry located a group of activities in the activity list. The current topological sequence number replaces each column B entry within that group of activities. As each entry is replaced, its original contents locate in the event list a column C count whose contents may be reduced by 1. Two 0's were created; their locations, 3 and 15, are stored in the zero table in the order of their occurrence.

Frame 4

Location	A	B	C	D
0	0	0	31	0
3	1	0	0	6
6	0	6	2	9
9	0	9	3	32767
12	0	9	1	12
15	0	6	31	3
18	1	9		
21	1	12		

Zero Table

3
15

The zero table indicates that location 15 contains an event that can be sequenced. The original column D entry, 15, is stored, and the next available topological sequence number is inserted. Column C is replaced by the "sequenced flag" (31).

Frame 5

Location	A	B	C	D
0	0	0	31	0
3	1	0	0	6
6	0	6	1	9
9	0	9	2	32767
12	0	9	0	12
15	0	3	31	3
18	1	3		
21	1	3		

Zero Table
3
12

The original column D entry located a group of activities in the activity list. The current topological sequence number replaces each column B entry within that group of activities. As each entry is replaced, its original contents locate in the event list a column C count whose contents may be reduced by 1. A 0 is created; its location, 12, is stored in the zero table.

Frame 6

Location	A	B	C	D
0	0	0	31	0
3	1	0	0	6
6	0	6	1	9
9	0	9	2	32767
12	0	9	31	6
15	0	3	31	3
18	1	3		
21	1	3		

Zero Table
3
12

The zero table indicates that location 12 contains a sequencable event. The original column D entry, 12, is stored, and the next available topological sequence number is inserted. Column C is replaced by the "sequenced flag" (31).

Frame 7

Location	A	B	C	D
0	0	0	31	0
3	1	0	0	6
6	0	6	1	9
9	0	9	1	32767
12	0	6	31	6
15	0	3	31	3
18	1	3		
21	1	3		

Zero Table
3

The original column D entry located an activity in the activity list. The current topological sequence number replaces its column B entry. As this entry is replaced, its original contents locate in the event list a column C count whose contents may be reduced by 1.

Frame 8

Location	A	B	C	D
0	0	0	31	0
3	1	0	[31]	[9]
6	0	6	1	9
9	0	9	1	32767
12	0	6	31	6
15	0	3	31	3
18	1	3		
21	1	3		

Zero Table
[3]

The zero table indicates that location 3 contains an event which can be sequenced. The original column D entry, 6, is stored, and the next available topological sequence number is inserted. Column C is replaced by the "sequenced flag" (31).

Frame 9

Location	A	B	C	D
0	0	0	31	0
3	1	0	31	9
6	0	[9] →	[0]	9
9	0	9	1	32767
12	0	6	31	6
15	0	3	31	3
18	1	3		
21	1	3		

Zero Table
[6]

The original column D entry located an activity in the activity list. The current topological sequence number replaces the activity's column B entry. As this entry is replaced, its original contents locate in the event list a column C count whose contents may be reduced by 1. A 0 is created; its location, 6, is stored in the zero table.

Frame 10

Location	A	B	C	D
0	0	0	31	0
3	1	0	31	9
6	0	9	[31]	[12]
9	0	9	1	32767
12	0	6	31	6
15	0	3	31	3
18	1	3		
21	1	3		

Zero Table
[6]

The zero table indicates that location 6 contains an event which can be sequenced. The original column D entry, 9, is stored, and the next available topological sequence number is inserted. Column C is replaced by the "sequenced flag" (31).

Frame 11

Location	A	B	C	D
0	0	0	31	0
3	1	0	31	9
6	0	9	31	12
9	0	[12] →	[0]	32767
12	0	6	31	6
15	0	3	31	3
18	1	3		
21	1	3		

Zero Table
[9]

The original column D entry located an activity in the activity list. The current topological sequence number replaces the activity's column B entry. As the entry is replaced, its original contents locate in the event list a column C count whose contents may be reduced by 1. A 0 is created; its location, 9, is stored in the zero table.

Frame 12

Location	A	B	C	D
0	0	0	31	0
3	1	0	31	9
6	0	9	31	12
9	0	12	[31]	[15]
12	0	6	31	6
15	0	3	31	3
18	1	3		
21	1	3		

Zero Table
[9]

The zero table indicates that location 9 contains an event which can be sequenced. The original column D entry, 32767 is stored, and the next available topological sequence number is inserted. Column C is replaced by the "sequenced flag" (31).

Since the original column D entry was the terminal flag, no action is taken on the activity list.

Since the zero table is empty, a search of the entries in column C is continued. No more zero entries are found; thus, the process is complete.

Notice that the zero table is a "last-in, first-out" type. Its capacity is 100 entries; if it overflows, created zeros cannot be stored, and the normal scan must eventually be resumed.

To allow for more than 31 direct predecessors on an event, a column C entry of 30 has been chosen as a flag; this indicates that there are more than 29 direct predecessors. The actual count is found in column D of the next event entry. This involves no decrease in event capacity in the normal case, since a network generally has more activities than events. The provision for this must be made in STARTD.

SUBROUTINE STARTF

Subroutine STARTF checks each column C entry to determine if each event has been sequenced. (Column C = 31, if sequenced.) If an event has not been sequenced, a loop condition exists, and the program makes a note of this. Then all activities which were sequenced are physically deleted, and the predecessor and successor event numbers of the remaining activities are exchanged. The following is a description of the input and output record formats if a loop condition exists.

Input

First Word

0 - 26	Successor
27 - 35	Predecessor (first part)

Second Word

0 - 17	Not used
18 - 35	Predecessor (second part)

Third Word

0	Column A
1 - 15	Column B
16 - 20	Column C
21 - 35	Column D

Output (Only if loop condition exists)

First Word

0 - 26	Predecessor (exchanged)
27 - 35	Successor (first part) (exchanged)

Second Word

0 - 17	Successor (second part) (exchanged)
18 - 35	To be used by sort

Third Word

0 - 35	To be used by sort

Loop Condition Link--PROGX

The input data contains a loop. Prior to this program operation, sequenced activities have been deleted, and the event numbers have been reversed. The link PROGX contains subroutines STARTX and STARTY which sort the loop activities and put them into the form needed by STARTB.

SUBROUTINE STARTX

This program sorts the activities by the new predecessor, successor event numbers. Both input and output records have the same format.

Record Format

First Word

0 - 26	Predecessor
27 - 35	Successor, first part

Second Word

0 - 17	Successor, second part
18 - 35	Sort buffer area

Third Word

0 - 35	Sort buffer area

SUBROUTINE STARTY

Prior to the operation of STARTY, the activities containing a loop have been reversed and sorted. This program compresses them into two-word records to agree with the form needed by STARTB. After the second execution of this program, MAINP gives control to STARTZ to print the loop.

Circular Subset Print Link--PROGZ, Subroutine STARTZ

The subroutine STARTZ, contained in link PROGZ, prints the circular subset. The following actions must precede this program:

1. A loop is detected. Any activities which were sequenced are physically deleted and the network is reversed (the predecessor and successor event numbers are exchanged). (STARTF)

2. A physical sort arranges the activities in predecessor, successor sequence. (STARTX)

3. The activities are compressed into two words for normal STARTB (or STARTZ) input. (STARTY)

4. The sequencing process is repeated. (STARTB through STARTE)

5. The loop is detected a second time. Any activities which were sequenced are physically deleted. Step 1 "chops off" any activities which are predecessors to the circular subset. This step (5) "chops off" any activities which are successors to the circular subset. (STARTF performs this function through the reversal process.)

6. Another physical sort arranges the activities back into predecessor, successor sequence. Since they have been reversed twice, this is a subset of the activities as received from File Maintenance. (STARTX)

7. The activities are compressed into two words for normal STARTB (or STARTZ) input. (STARTY)

8. Since this was the second time STARTY was executed, MAINP releases control to STARTZ to print the circular subset.

Reassign Topological Numbers and Sort Link--PROGG

Link PROGG contains the subroutines STARTG and STARTH.

SUBROUTINE STARTG

The subroutine STARTG, contained in link PROGG, reassigns the topological numbering to the network. This is done in two steps. First, all successor event numbers are substituted for the user's numbers. During this process, a step-by-step scan is necessary, since the successor list is a logical list and the event list is a physical list. Second, column B is substituted as the new list of predecessor numbers.

The record formats for STARTG are:

Input

First Word

0 - 26	Successor event number
27 - 35	Predecessor (first part)

Second Word

0 - 17	Successor list pointers
18 - 35	Predecessor (second part)

Third Word

0	Column A
1 - 15	Column B (with predecessor substitution)
16 - 20	Column C
21 - 35	Column D (with event number substitution)

Output

First Word

0 - 14	Topological renumber (successor)
15 - 29	Topological renumber (predecessor)
30 - 35	Not used

Second Word

0 - 35	Not used (will be sorting buffer)

Third Word

0 - 17	Initialized for sorting pointers
18 - 29	Not used
30 - 35	Set in index register 7 for sorting

The renumbered activities are sorted (logically) by topological successor, topological predecessor, to facilitate later calculation of TE of successor event.

SUBROUTINE STARTH

The subroutine STARTH, contained in link PROGG, sorts the renumbered activities by successor, predecessor. This is a logical sort, since no physical movement takes place; that is, physically the order is the same as that on the master file. The logical sort is through a set of pointers in bits 0-17 in the third word of each record. The pointers are merged together using a buffer area in bits 0-17 in the second word of each record.

Record Description

First Word

0 - 14	Topological successor event
15 - 29	Topological predecessor event
30 - 35	Not used

Second Word

0 - 35	Sort buffer space

Third Word

0 - 17	Topological pointer
18 - 29	Not used
30 - 35	Index register (sorting parameter)

Read NUMDAT Link--PROGI, Subroutine STARTI

The File Maintenance Section has left the numeric data (that is, the date, date flag and duration for each activity) on a separate file (NUMDAT) consisting of one word per activity and arranged in the order in which the physical activities were received. The subroutine

STARTI, contained in link PROGI, takes this information off the file and distributes it in the activities file created in STARTG. The numeric data word contains all of the information necessary for the calculations for the expected completion date (TE) and the latest completion date (TL).

The formats for the NUMDAT record and the resultant activities file with this data incorporated into it are as follows:

Input

Numeric Data Word

0 - 1	Date Flag
2 - 3	Short path flag
4 - 17	Activity date
18 - 21	Not used
22 - 35	Duration

Output

First Word

0 - 13	Topological successor event
14 - 27	Topological predecessor event
28 - 29	Date flag
30 - 35	Activity date (first part)

Second Word

0 - 7	Activity date (second part)
8 - 21	Duration
22 - 35	Not used

Third Word

0	Short path flag
1 - 2	Not used
3 - 17	Topological pointer

TE and TL Calculation Link--PROGJ

Link PROGJ contains the subroutines STARTJ, STARTK, and STARTL which calculate the TE and TL and pack the records into lower core.

SUBROUTINE STARTJ

The subroutine STARTJ, contained in the link PROGJ, does the basic TE calculation. The programmer may use the topological predecessor and successor numbers as index values in entering an event file to determine previously calculated data about the event. The event file is initialized to show all events as complete. Any incomplete direct predecessor activity signifies that an event is incomplete. Thus, each complete activity's predecessor event must be indicated as complete, or a diagnostic flag will be set. Another diagnostic flag is set if the predecessor event of activity has had no associated calculated date. Also, at this time, the active activity flag is set. An incomplete activity is active if its predecessor event is flagged as complete.

The output record is in the following format:

First Word

0 - 13	Topological successor event
14 - 27	Topological predecessor event
28 - 29	Date flag
30 - 35	Activity date (first part)

Second Word

0 - 7	Activity date (second part)
8 - 21	Duration
22 - 35	Calculated activity TE

Third Word

0	Short path flag
1 - 17	Topological pointer
18	Flag, no predecessor diagnostic
19	Flag, active activity
20	Flag, constrained, complete diagnostic
21	Flag, event completion
22 - 35	Event date

SUBROUTINE STARTK

The subroutine STARTK, contained in the link PROGJ, does the basic TL calculation. The programmer may use the topological predecessor and successor numbers as index values in entering an event file to determine previously calculated data about the event. The event file is initialized to show all events occurring at a late time. Schedule finish dates are then entered. As each activity TL is handled, its late start is also calculated. If the late start is earlier than the occurrence of its predecessor event in the event file, the date of the predecessor event is replaced. Because of a lack of space, the calculated activity TL overlays the activity date field as it is calculated.

The output record is in the following format:

First Word

0 - 13	Topological successor event
14 - 27	Topological predecessor event
28 - 29	Date flag
30 - 35	TL (first part)

Second Word

0 - 7	TL (second part)
8 - 21	Duration
22 - 35	Calculation activity TE

Third Word

0	Short path flag
1 - 17	Topological pointer
18	Flag, no predecessor diagnostic
19	Flag, active activity
20	Flag, constrained, complete diagnostic
21	Flag, event completion
22 - 35	Event date

SUBROUTINE STARTL

The subroutine STARTL, contained in link PROGJ, is a FORTRAN program. To gain the added room necessary, the records are stripped of unnecessary information and packed into lower core. The input and output record formats are as follows:

Input

First Word

0 - 13	Topological successor event
14 - 27	Topological predecessor event
28 - 29	Not used
30 - 35	TL (first part)

Second Word

0 - 7	TL (second part)
8 - 21	Not used
22 - 35	TE

Third Word

0 - 17	Not used
18	Flag, diagnostic, incomplete activity with no predecessors
19	Flag, active activity
20	Flag, diagnostic, constrained but complete activity
21 - 35	Not used

Output

Output records are 60-bits and are packed starting at location BLOCK.

0 - 13	Topological successor event
14 - 27	Topological predecessor event
28 - 41	TE
42 - 55	TL
56	Flag, new successor group
57	Flag, diagnostic, incomplete activity with no predecessors
58	Flag, active activity
59	Flag, diagnostic, constrained but complete activity

Modified Master and Diagnostics Link-PROGM

Link PROGM contains the subroutines STARTM and UNP74.

STARTM reads the master file, attaches the topology information to this file, and writes the modified master. UNP74 unpacks the records and distributes the information in the records properly into each activity record. Since TOPOLO has never changed the physical order of its records, they are still in the order in which they are read from the master file. All diagnostic messages as found by the Topology Section are also printed.

SUBROUTINE STARTM

Input to the subroutine STARTM is the master file and the internal records from STARTL. Output is a modified master and diagnostics written on an output file.

SUBROUTINE UNP74

UNP74 unpacks the next internal record and distributes the necessary information into the master file record currently held by the FORTRAN program STARTM. There is a one-to-one correspondence between the master file records and the internal records, and both files are in the same order. The input record format is the same as the output record format of STARTL.

Output Start and Terminal Events Link--PROGW, Subroutines STARTW and GETSE

Prior to operation of subroutine STARTW, STARTD has written all the start and terminal events on a file. STARTW reads that file back in and writes the events on the output tape. They could not have been written initially, because of the stringent core requirements of STARTD. An auxiliary subroutine, GETSE, performs the actual record retrieval.

GETSE supplies STARTW with a pair of numbers representing a start or terminal event. Terminal events are in the form 0,X and start events are in the form X,0. The records are blocked 318 words per physical record. The formats for the modified master file are as follows:

INITIAL RECORD OF MODIFIED MASTER FILE

Word	Contents	Format
1	Run Number	BCD chars 0,1
	Number of Activities	Binary Bits 18-35
2	Report Date - Day	BCD chars 0,1
	Number of Events	Binary Bits 18-35
3	Report Date - Month	BCD chars 0-2
4	Report Date - Year	BCD chars 0,1
5	Start Date - Day	BCD chars 0,1
6	Start Date - Month	BCD chars 0-2
7	Start Date - Year	BCD chars 0,1
8	Network Completion Date - Day	BCD chars 0,1
9	Network Completion Date - Month	BCD chars 0-2
10	Network Completion Date - Year	BCD chars 0,1
11	Network I.D.	BCD 6 chars
12	Users I.D.	BCD 6 chars
13-18	Network Title (First 36 chars)	BCD 36 chars
19	Master File Report Option	BCD char 0
20	Summarization Option	BCD char 0
21	Run Date Option	BCD char 0
22	Number of Days in Work Week	BCD char 0
23-27	Network Title (Final 30 chars)	BCD 30 chars
28	New Network I.D.	BCD 6 chars
29	Sort Option	BCD char 0
30	Run Type Option	BCD char 0
31	Number Days in Work Week	BCD char 0

MODIFIED MASTER FILE RECORD

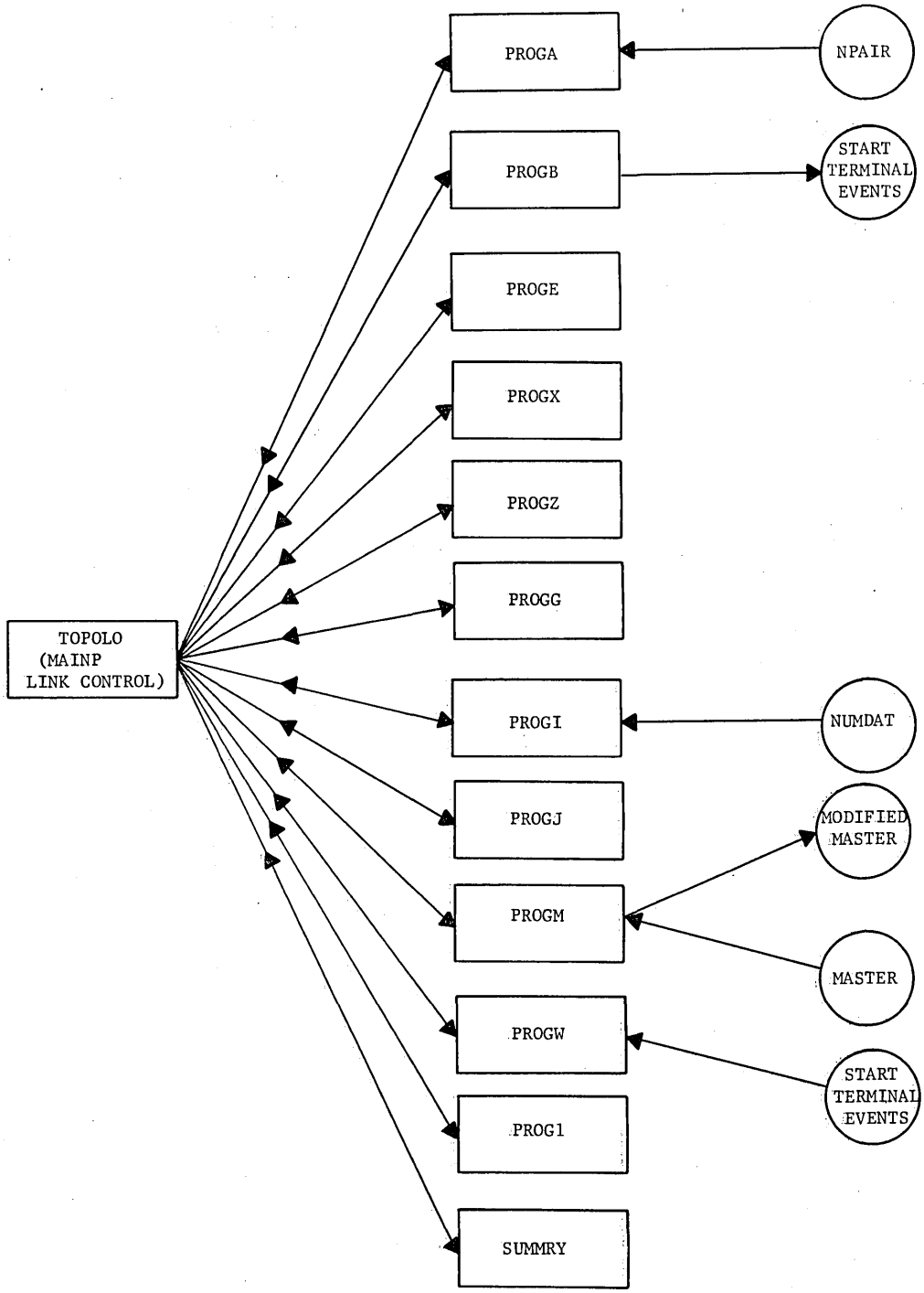
Word	Contents	Format
1	Transaction Code	BCD char 0
2	Short Path Flag	BCD char 0
3	Scheduled Date Option	BCD char 0
4	Beginning Event Interface	BCD char 0
5	Beginning Event #, First 6 chars	BCD, 6 chars
6	Beginning Event #, Final 2 chars	BCD, chars 0,1
7	Beg. Evt. Level Code	BCD, char 0
8	Succ. Evt. Interface	BCD, char 0
9	Succ. Evt. #, First 6 chars	BCD, 6 chars
10	Succ. Evt. #, Final 2 chars	BCD, chars 0,1
11	Succ. Evt. Level Code	BCD, char 0
	Succ. Evt. TE	Binary Bits 6-35
12	Optimistic Time	BCD chars 0-3
13	Most Likely Time	BCD chars 0-3
14	Pessimistic Time	BCD chars 0-3
15	Scheduled Date	BCD, 6 chars
16-21	Event Title	BCD 36 chars
22	Activity Info. First 4 chars	BCD chars 0-3
	Current Activ. Flag	Bit 35
23	Activity Info. Second 4 chars	BCD chars 0-3
	Actual Indicator	Bit 35
24	Activity Info. Third 4 chars	BCD chars 0-3
	Run Date Option Char	BCD char 5
25	Activity Info. Final 6 chars	
26-31	Activity Title	BCD 36 chars
32	Actual Date where MM=month, DD=day, YY=year	BCD MMDDYY
33	Change Code	Binary
34	Topological Sequence	Bits 0-13 Pred # Bits 18-31 Succ #
35	TE	Bits 0-13
	TL	Bits 18-31
	Beg. Evt. Flag	Bit 35

Sort for Summarization Link--PROG1, Subroutine SPEC

Prior to the operation of this program, summarization has been requested. The activities are in core by successor, predecessor (topologically). The subroutine SPEC, contained in the link PROG1, sorts the activities in predecessor, successor order for input to the Summarization Section. The data area is saved and is restored by MAINP at the conclusion of summarization.

TOPOLOGICAL SECTION FLOWCHARTS

The following pages contain the flowcharts for the Topological Section listed in the following order:


TOPOLO
(MAINP
LINK CONTROL)
PROGA
PROGB
PROGE
PROGX
PROGZ
PROGG
PROGI
PROGJ
PROGM
PROGW
PROG1
SUMMRY
NPAIR
START
TERMINAL
EVENTS
NUMDAT
MODIFIED
MASTER
MASTER
START
TERMINAL
EVENTS

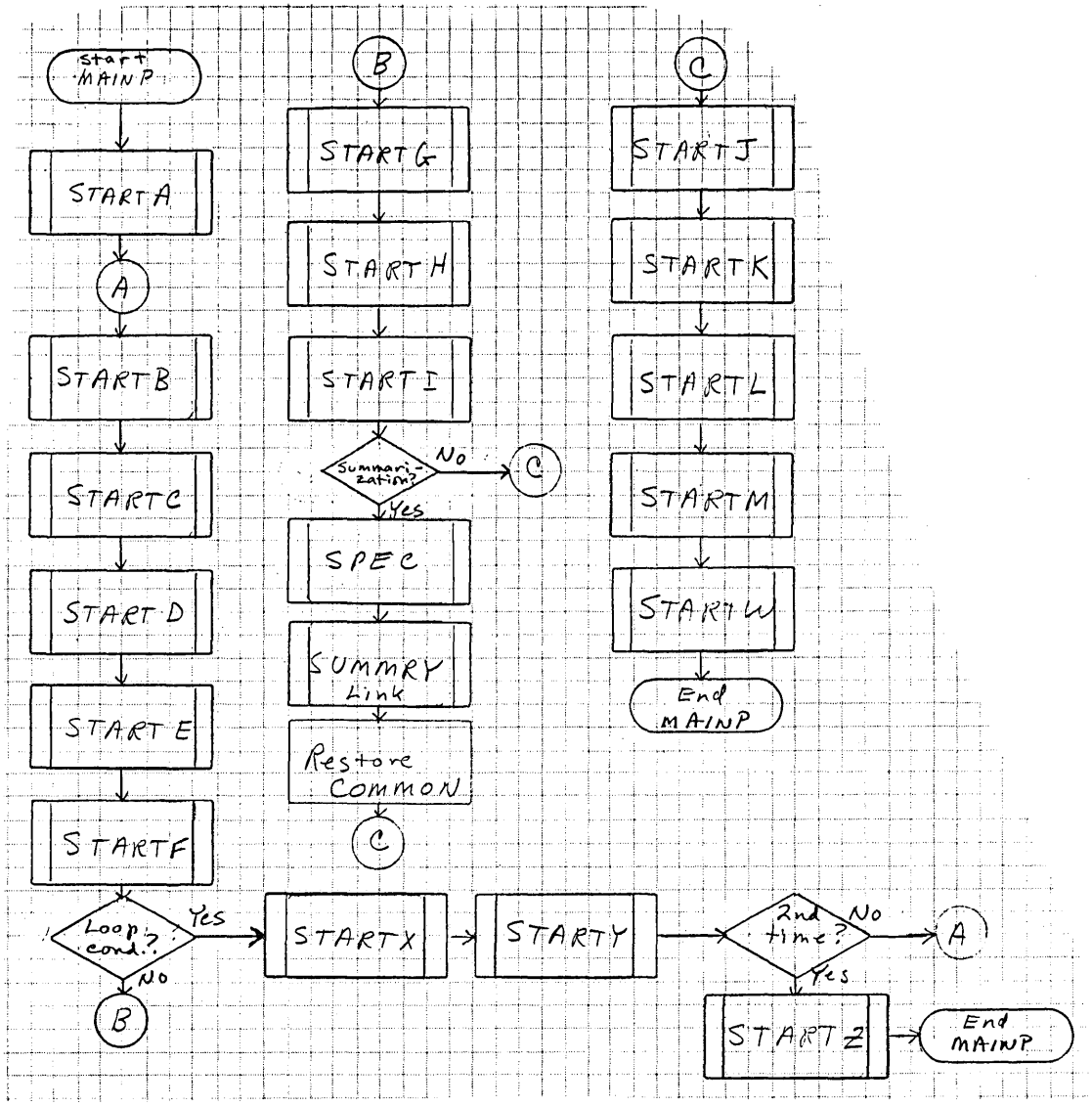


Figure 23. Link MAINP Control Flowchart

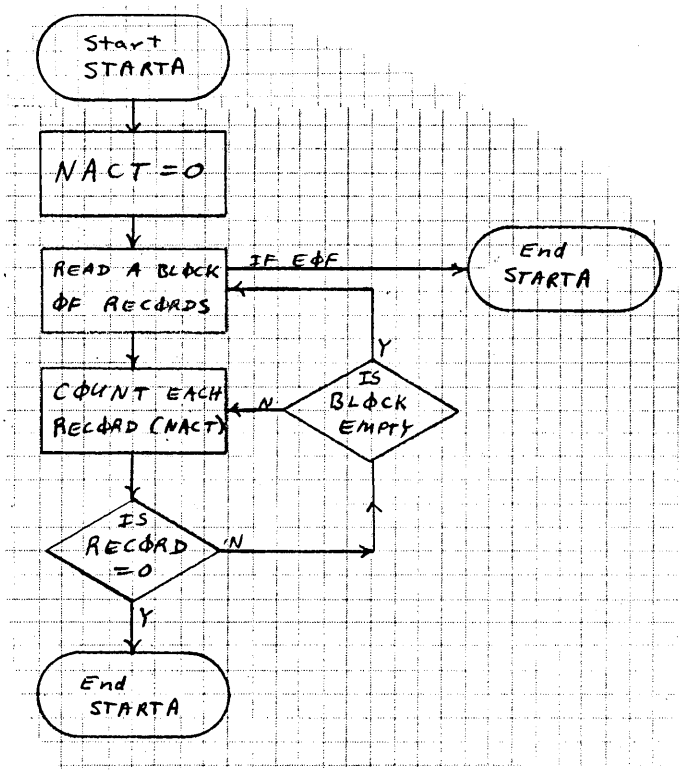


Figure 24. STARTA Subroutine Flowchart

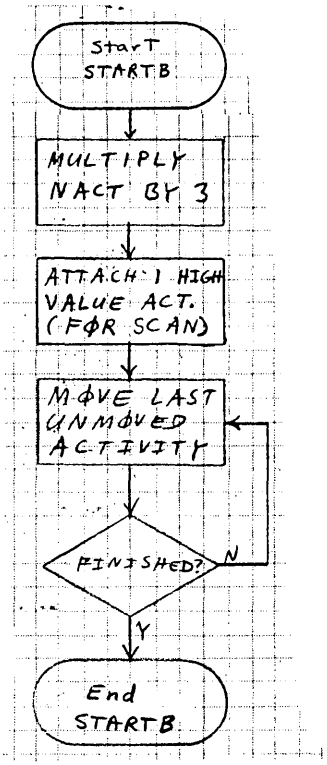


Figure 25. STARTB Subroutine Flowchart

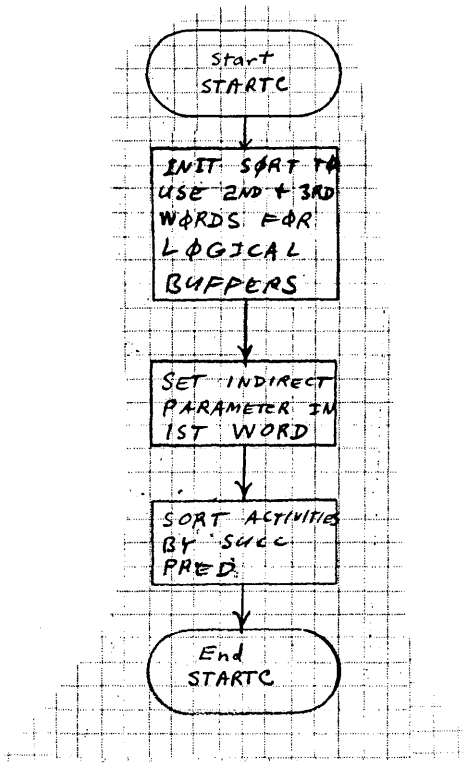


Figure 26. STARTC Subroutine Flowchart

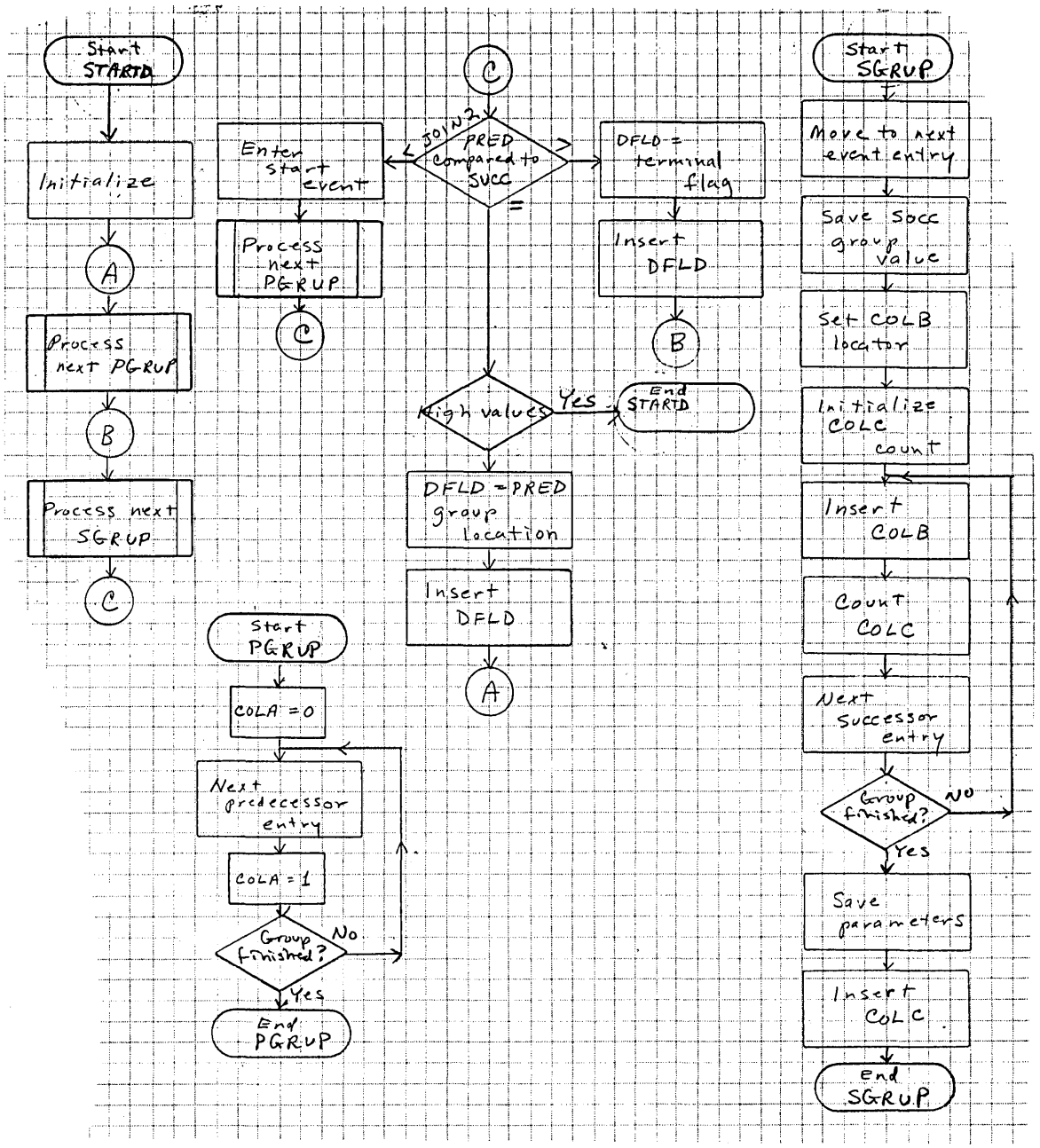


Figure 27. STARTD Subroutine Flowchart

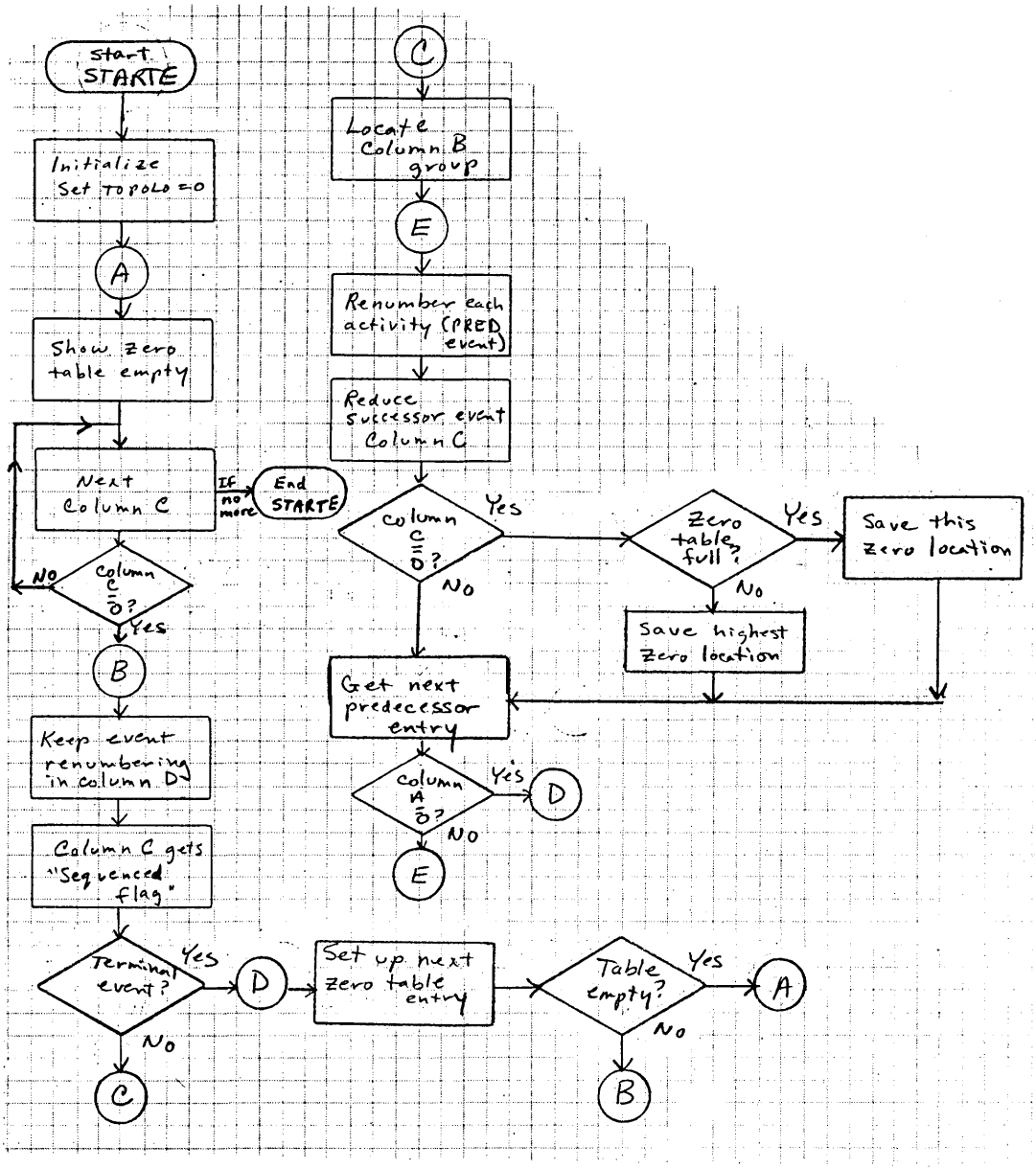


Figure 28. STARTE Subroutine Flowchart

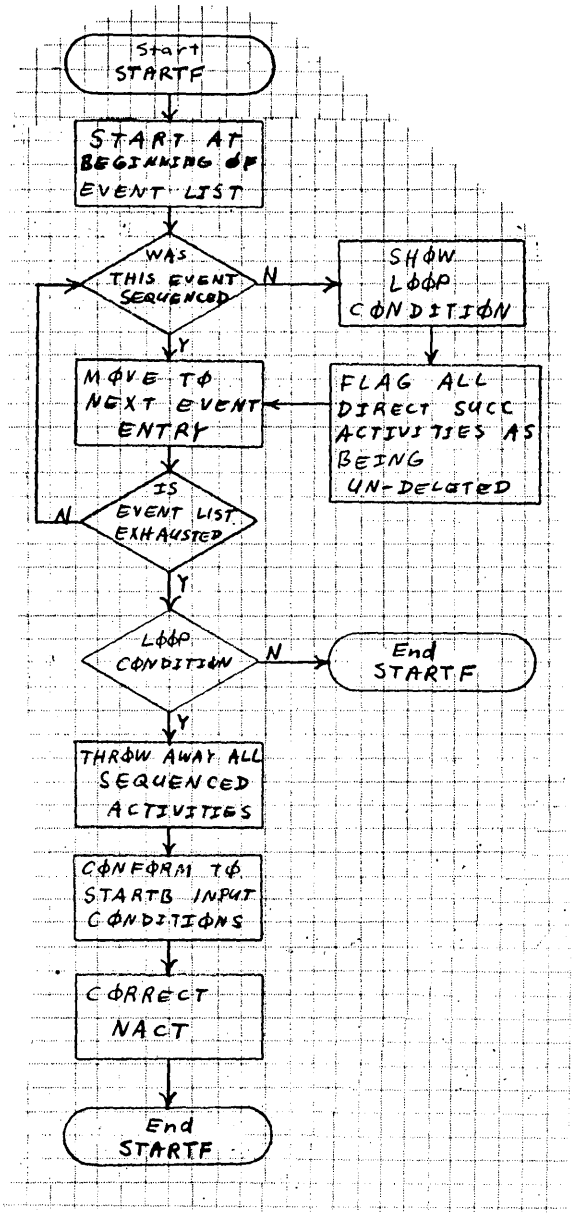


Figure 29. STARTF Subroutine Flowchart

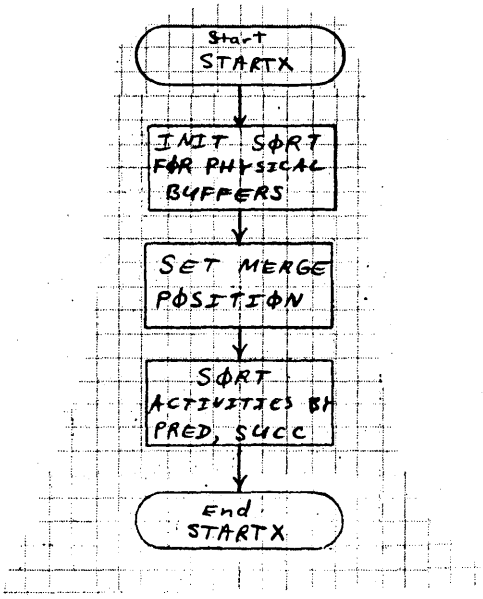


Figure 30. STARTX Subroutine Flowchart

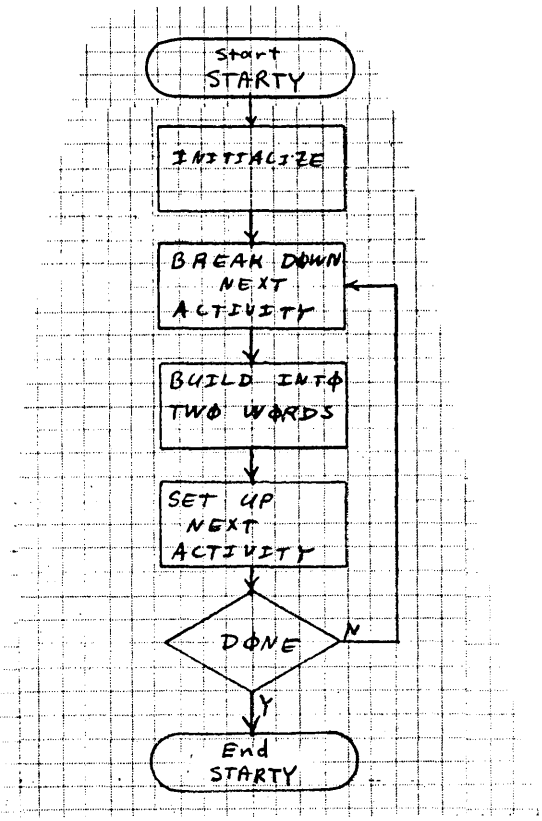


Figure 31. STARTY Subroutine Flowchart

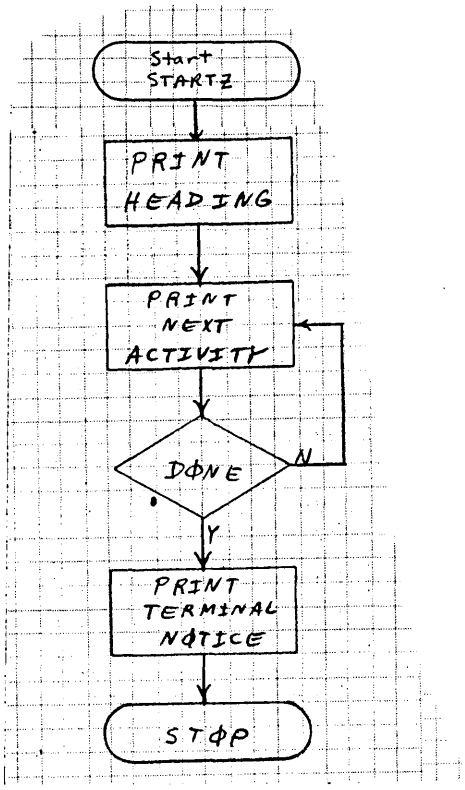


Figure 32. STARTZ Subroutine Flowchart

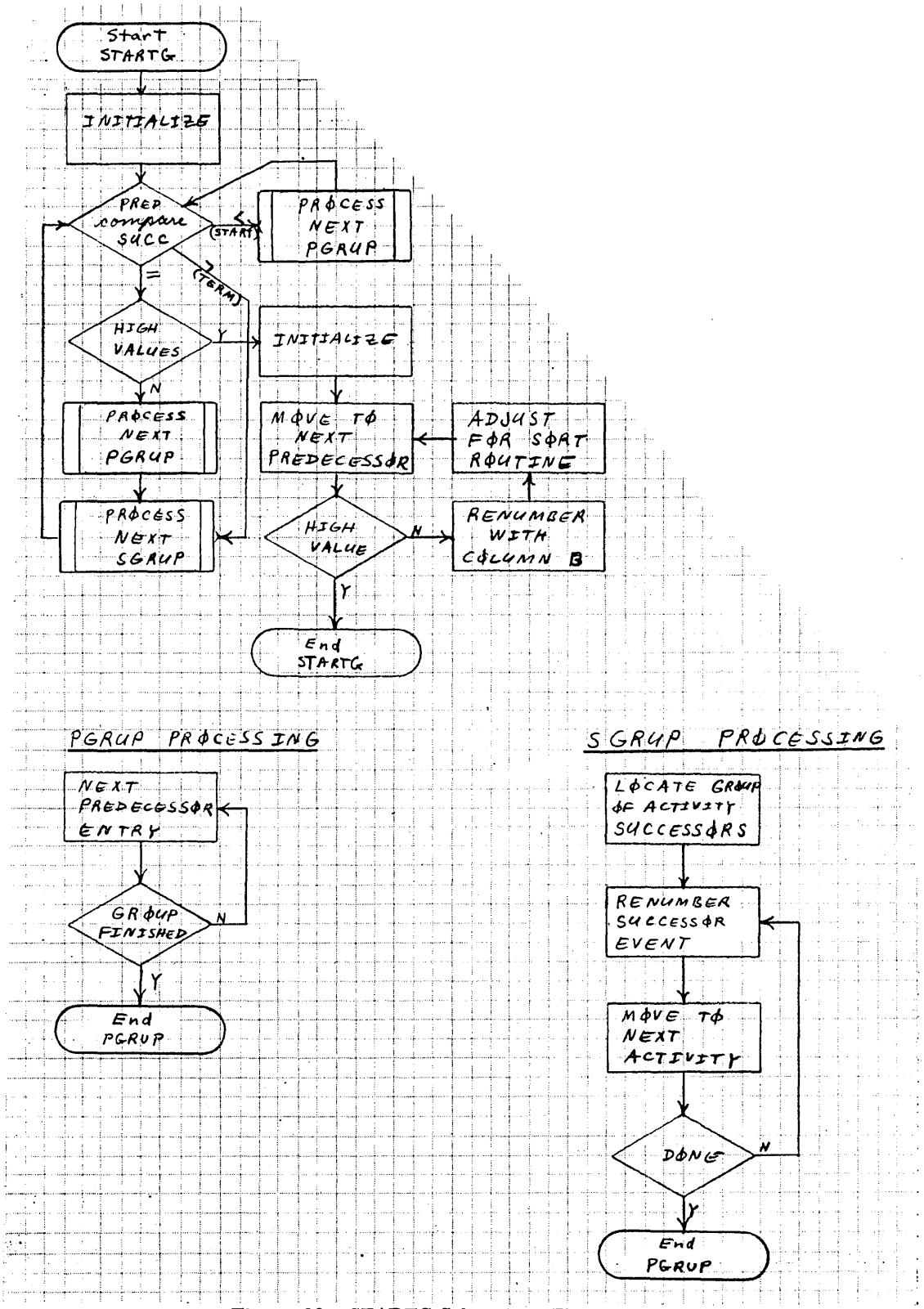


Figure 33. STARTG Subroutine Flowchart

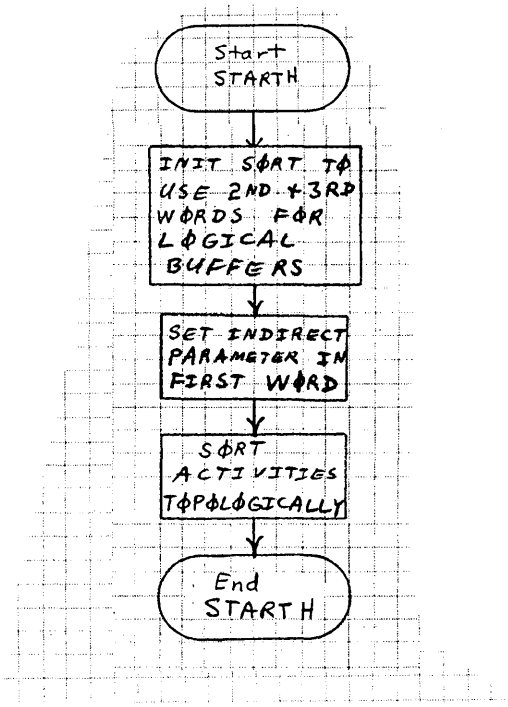


Figure 34. STARTH Subroutine Flowchart

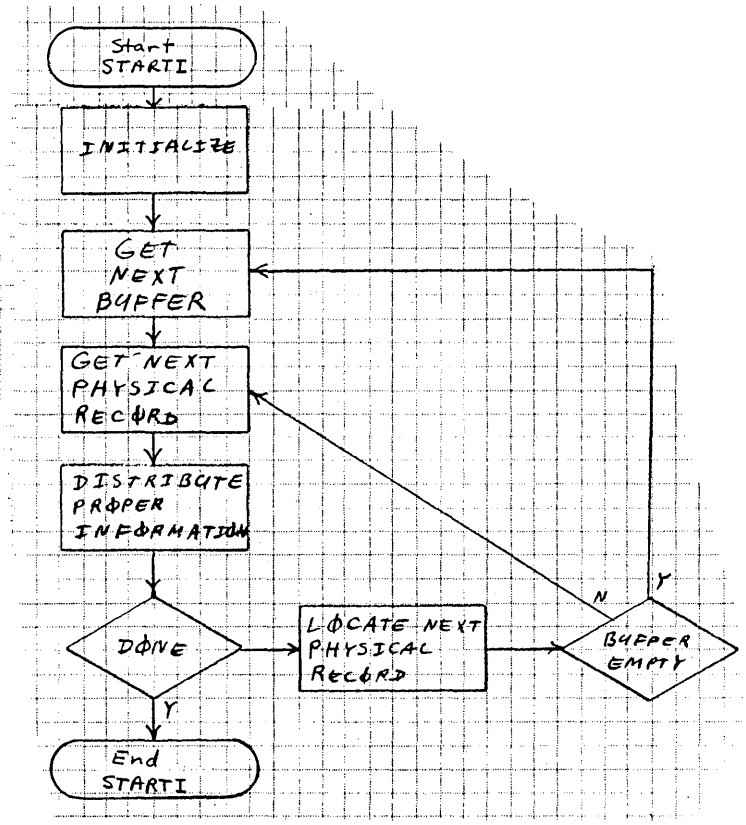


Figure 35. STARTI Subroutine Flowchart

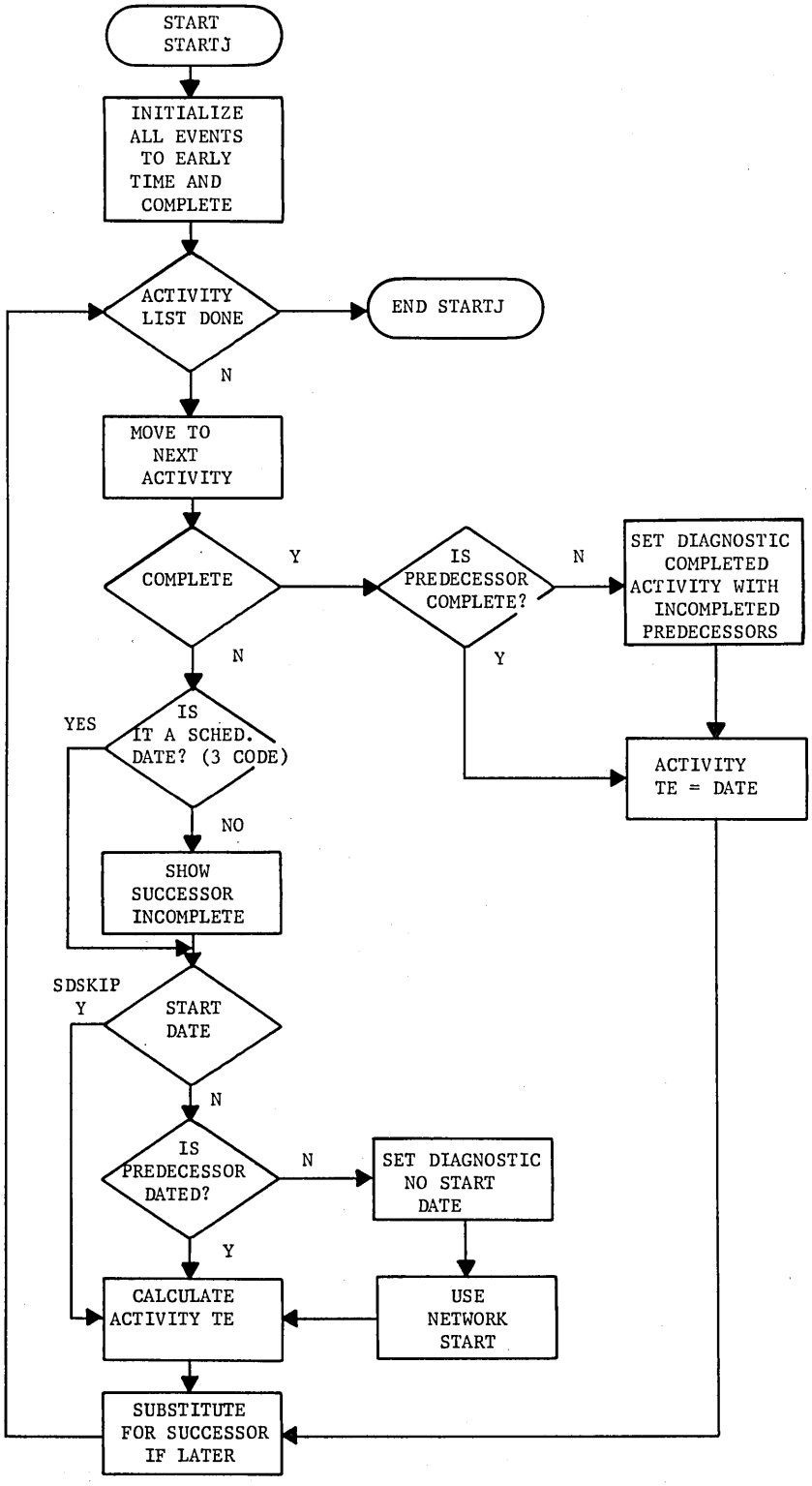


Figure 36. STARTJ Subroutine Flowchart

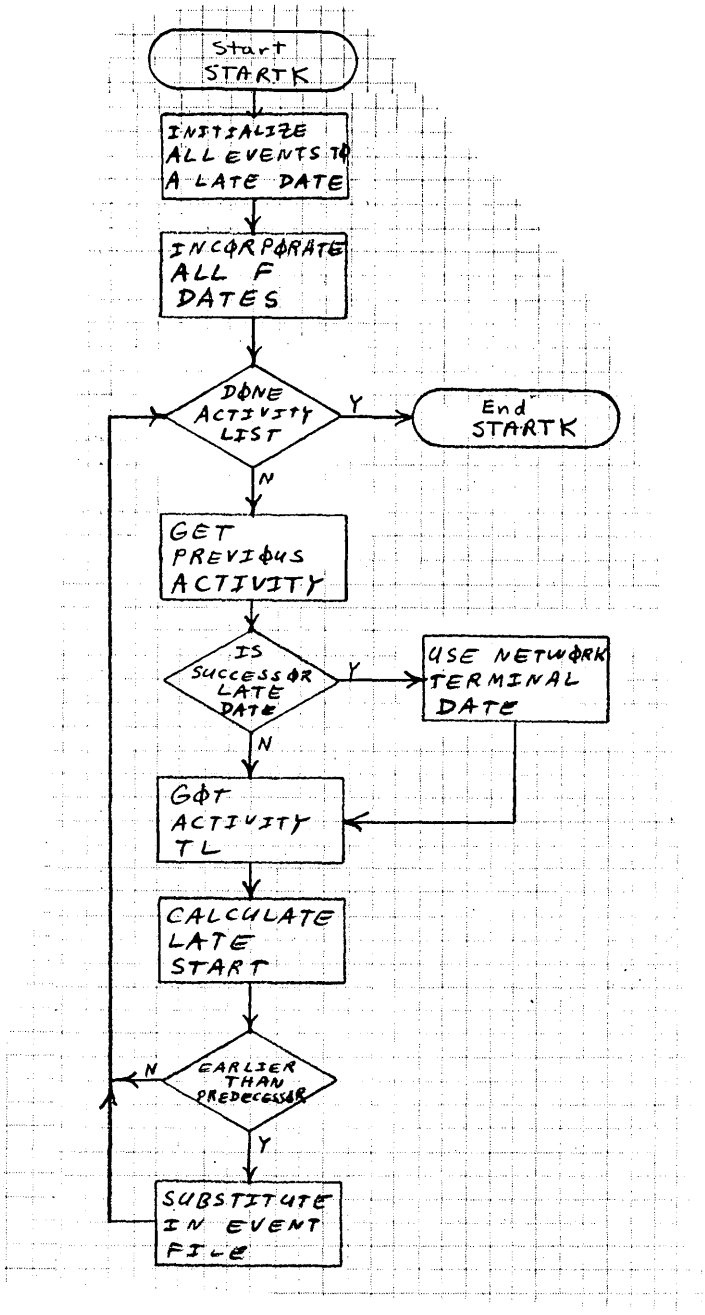


Figure 37. STARTK Subroutine Flowchart

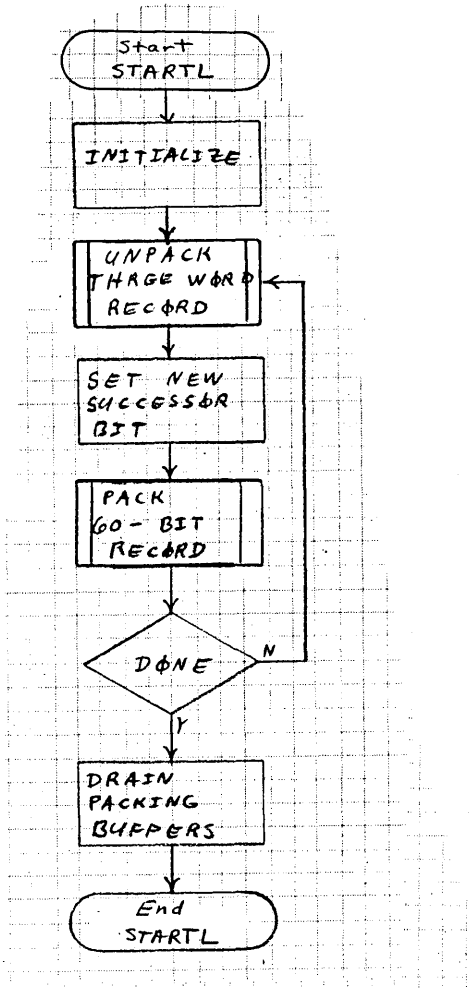


Figure 38. STARTL Subroutine Flowchart

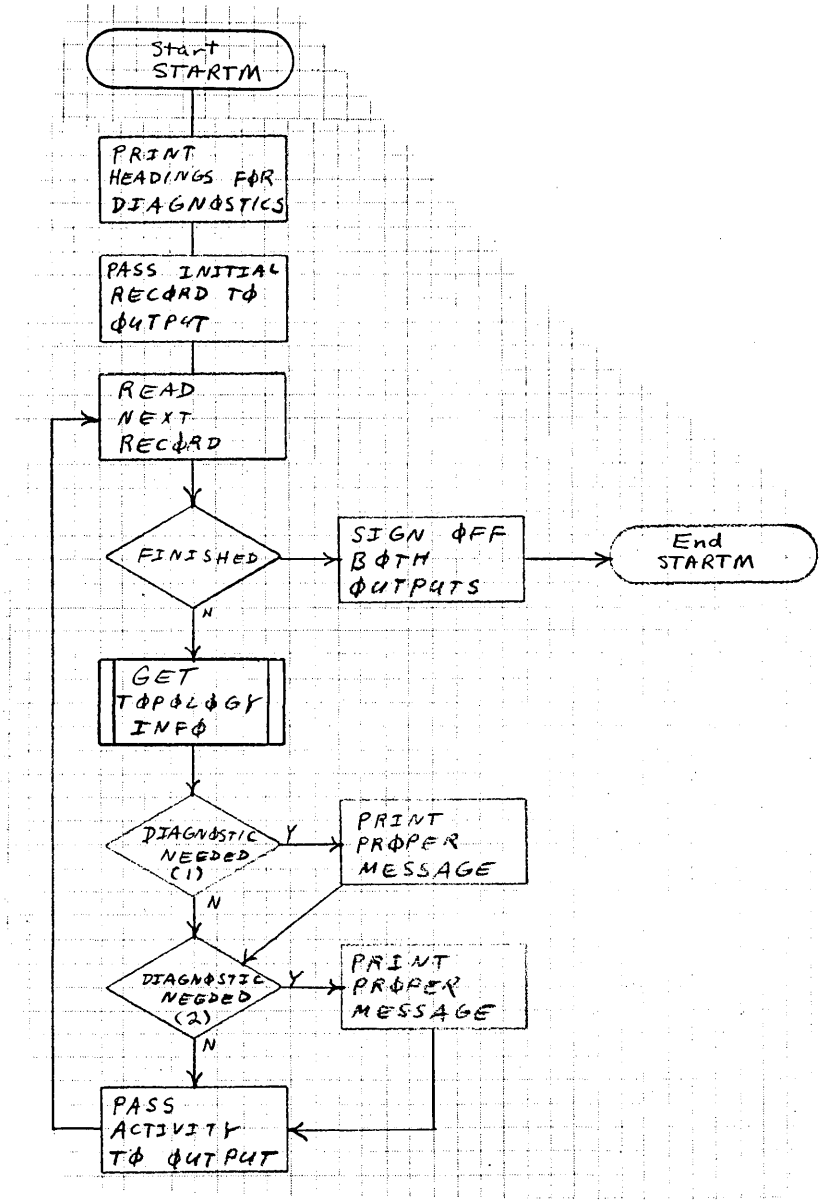


Figure 39. STARTM Subroutine Flowchart

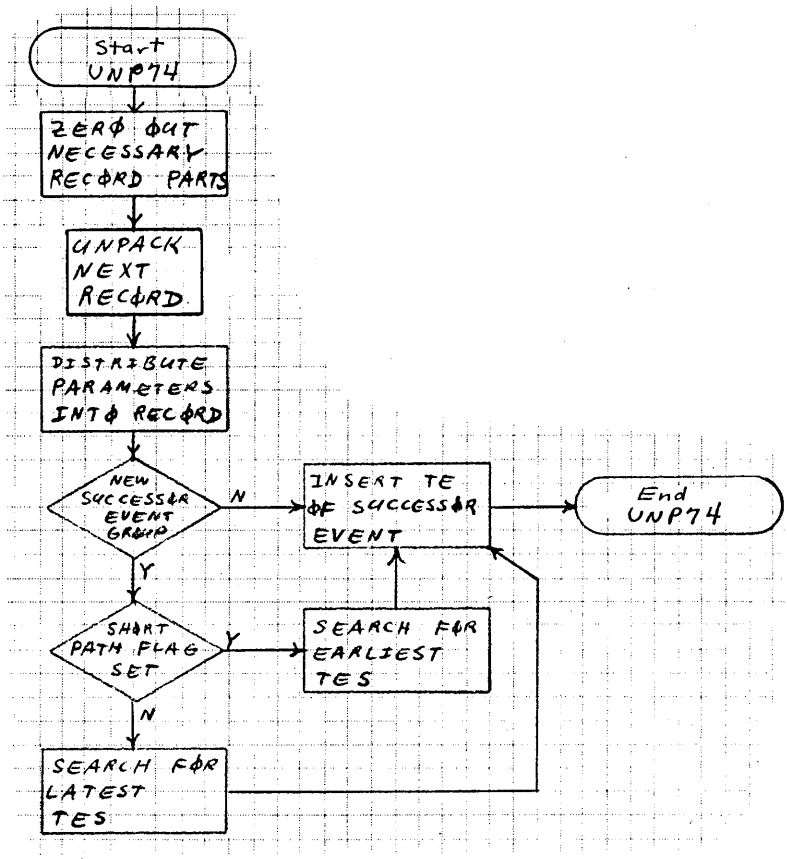


Figure 40. UNP74 Subroutine Flowchart

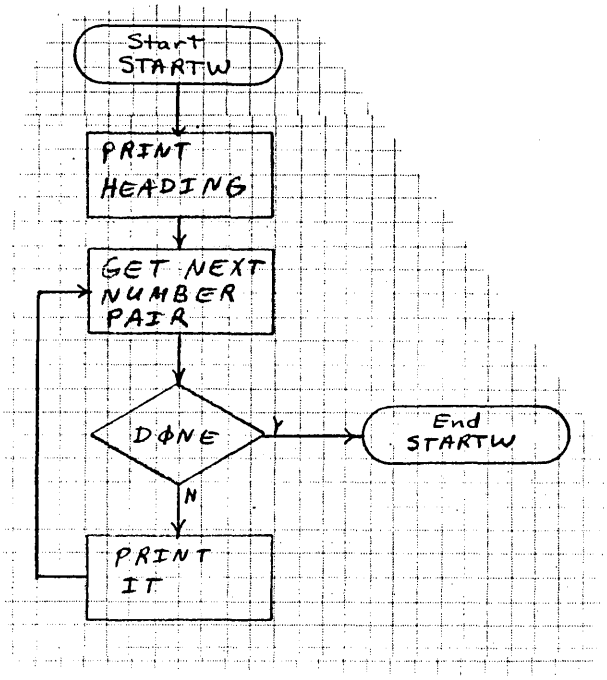


Figure 41. STARTW Subroutine Flowchart

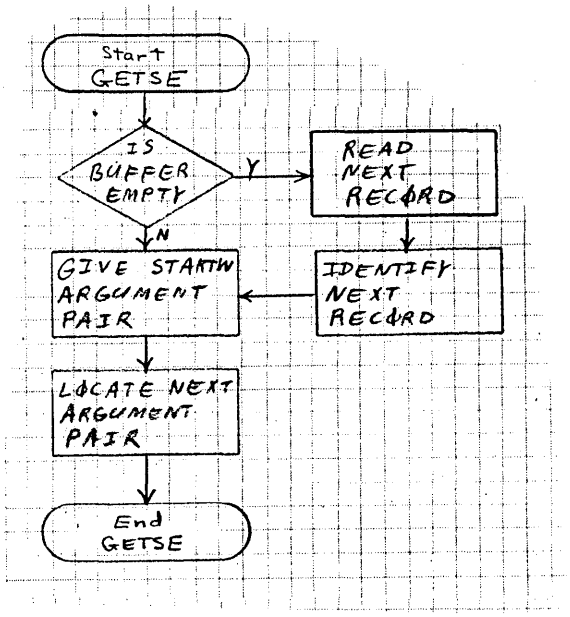


Figure 42. GETSE Subroutine Flowchart

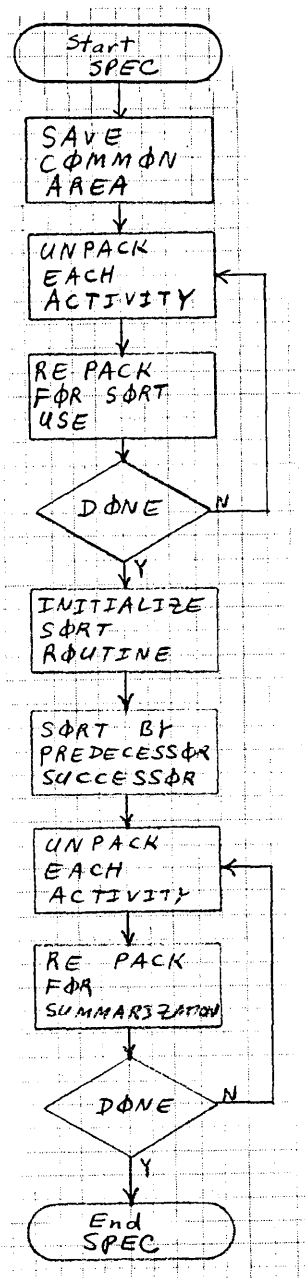


Figure 43. SPEC Subroutine Flowchart

5. SUMMARIZATION SECTION - SUMMRY

The Summarization Section takes a detailed network and constructs a summarized version based on user selected events and events which clarify the work in progress. The program is compatible with USAF PERT specifications,[1] and the following four cases are stipulated for program selected events:

Case 1: If an event which is a start event, a preselected event, or a completed event has both complete and incomplete successor activities, then each incomplete successor becomes a program-selected event under Case 1.

Case 2: All completed events with at least one incomplete successor activity are program-selected under Case 2.

Case 3: Case 3 applies when a completed activity and an incomplete activity merge. If the predecessor event of the incomplete activity lies on a path which constrains the predecessor event of the completed activity and there are no intervening selected events on this path, then the completed activity's predecessor event becomes a program-selected event under Case 3.

Case 4: All events which represent the merging of complete and incomplete activities are program-selected under Case 4.

SUMMARIZATION SECTION LINKAGE

The basis for the SUMMRY Link is a list of network activities in topological sequence. In this listing activities are renumbered by pseudo-event numbers as follows: (1) for each activity the pseudo-predecessor event number is less than the pseudo-successor event number and (2) activities are sequenced in ascending pseudo-predecessor/pseudo-successor order.

The program consists of six links under control of the ILNK program:

ISQEEZ, ILEVEL	Preparation for summarization
ISUMM	The actual summary
IPUNCH, JSORT1, IFINI	Conversion to card format and assimilation of master file information.

[1] USAF PERT, Volume II, September 1963, page II-18.

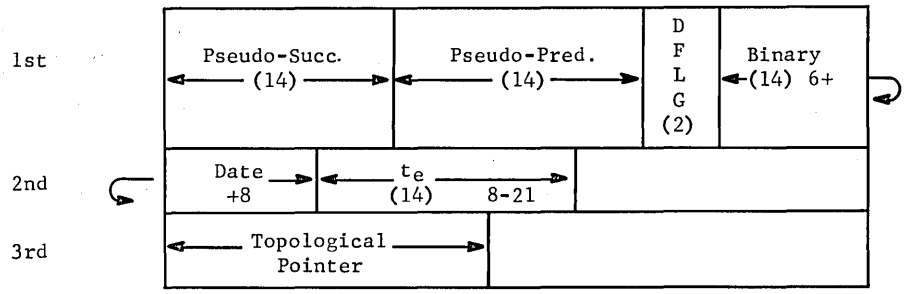
Output consists of a deck of cards which are in proper format to be run as a separate network or combined with other activities to form an enlarged network covering several projects.

The functions of the six links and the manner in which the subroutines perform these functions are described below. Flowcharts for these links and subroutines are at the end of this chapter.

Compress Table Link--ISQEEZ, Subroutine SQUEEZ

The first link of the SUMMRY Section, the ISQEEZ link, is initiated with an activity list left in core by the Topology Section. The list contains three computer words for each activity; therefore, for maximum size networks of 10,000 activities, computer core space is severely limited. The ISQEEZ link compresses the table to two words per activity and retains only the information required for summarization.

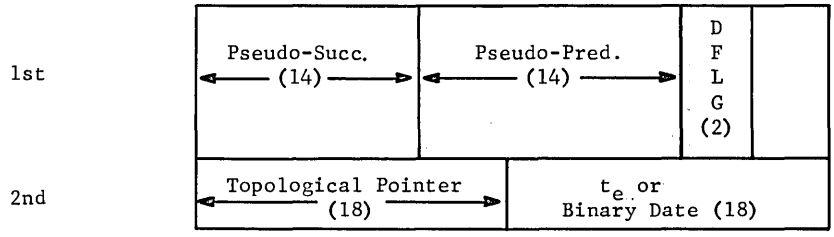
Prior to compression, interesting fields for each entry of the activity list appear, as shown below:



This table, in Block Common YYYYY, location DUMMY, is in master file order. This preserves the correspondence between the pseudo-event numbers and the original user numbers. The topological pointer of the first activity entry points to the location in core (DUMMY + pointer) of the first activity of the topological sequence. Thus, the pointer for the nth activity of the list points to the nth activity of the topological sequence.

The DFLAG field indicates whether the activity is complete or incomplete. For summarization, the expected duration time (t_e) is of no interest after the activity is completed. Thus, if testing shows that DFLAG contains a 3 (signifying completion), the t_e field can be omitted and the binary date field retained for completed activities.

The compressed table appears in core as follows:



Observe that the ISQEEZ Section must adjust the topological pointers to fit the compressed table:

2/3 x original topological pointer = new topological pointer

The compressed table in Block Common YYYYY, location DUMMY, is left in core memory for further processing by the next section.

Input to SQUEEZ is the activity table (3 words per activity) in core in location DUMMY. The table may be as large as 30,000 locations.

Output is a compressed activity table (2 words per activity) in core at location DUMMY. The table may be as large as 20,000 locations.

Summarization Preparation Link--ILEVEL

The ILEVEL link performs the following functions:

- Determines the level of summarization in accordance with user requirements.
- Transfers the event level codes for each activity from the master file to the activity table in core.
- Writes a file (EVE) of event information giving user event numbers, pseudo-event numbers, and event level codes for each activity.
- Writes an activity file (ATAB) with eight words per activity in preparation for the actual summarization to follow.

Input to the link ILEVEL is the holiday file (NVAC) and the new master file. Output is the event file (EVE) and the activity file (ATAB).

SUBROUTINE LEVOUT

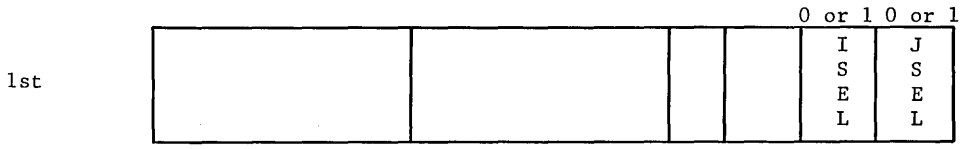
The link begins in subroutine LEVOUT by reading into core memory the calendar information from file NVAC. This releases the file for use as the event (EVE) file and retains the calendar information in core for use in the final section.

SUBROUTINES SETFLG, GETIJ, DIB, TSTWR and SETACT

The following procedure is accomplished through the use of subroutines: SETFLG, GETIJ, DIB, TSTWR and SETACT.

If the user indicates a minimum level summary, the activity table in core memory does not depend upon event level codes in deciding which events are a part of the summary. However, when the user indicates an alphabetic summary level (A to Z), each record of the master file is examined and event level codes which are alphabetically less than the summary level are noted on the activity list in core.

Two selection flags are set by SETFLG in the first word of each activity as follows:



The ISEL flag refers to the user selection of the predecessor event; the JSEL flag refers to the user selection of the successor event.

The event file (EVE) consists of 6-word records organized as follows:

Word	
1	(IPRIME - BINARY) Pseudo-Predecessor #
2	(I1 - BINARY) Predecessor #
3	(JPRIME - BINARY) Pseudo-Successor #
4	(J1 - BINARY) Successor #
5	(ILEV - 1BCD, LEFT-JUSTIFIED) Predecessor Level
6	(JLEV - 1BCD, LEFT-JUSTIFIED) Successor Level

This file preserves the correlation between the pseudo-event numbers and the user's event numbers for use in converting data to its original form for punched output.

After the entire master file has been examined, information from the activity list in core is transposed to eight-word records and written on the activity file (ATAB) in the following format:

Word	
1	Last Activity for This Pred. Flag — 0 or 1
2	Select Pred. Flag (0 or 1)
3	Complete/Incomplete (2 / 1)
4	Select Succ. Flag (0 or 10)
5	Pseudo-Predecessor
6	Pseudo-Successor
7	t_e or Binary Date
8	(Empty)

The writing of the activity tape completes preparation of data for the actual summarization performed in the next link (ISUMM).

Actual Summary Link--ISUMM

The ISUMM link is initiated by reading up to 1000 records from the activity file (ATAB) into the ACTIV array. These activities (eight words each) are in topological sequence. The topological sequencing assures that when an activity is considered, all predecessor activities have already been processed and all successor activities will follow this one. The program performs summarization by stepping from one activity to the next, examining each for the conditions of selection, and passing pertinent information for successor event selection to successor activities further down the list. This information is stored in the JLIST array and a pointer (or subscript) indicating its position is passed to successor activities.

The formats for the ACTIV entries and the JLIST entries are as follows:

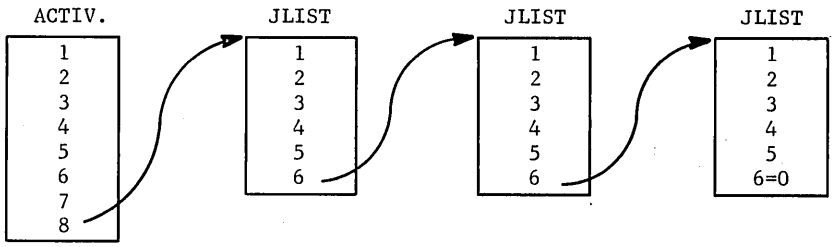
ACTIV

1	0 or 1	{ 1 Indicated last activity for this Pseudo-Predecessor
2	Select Flag (Predecessor Ev.)	
3	Complete/Incomplete	{ 2 = Complete 1 = Incomplete
4	Succ. Select Flag	{ 10 indicates user level selection of Suc. Event
5	Pseudo-Predecessor	
6	Pseudo-Successor	
7	t_e of Binary Date	
8	0 or → (Pointer)	

JLIST

1	Source (Pred.)
2	Σt_e or Largest Binary Date
3	Select/Complete/ Incomplete
4	Last Complete Event (or Empty)
5	Binary Date for Last Complete Event
6	0 or →(Pointer)

Observe that the last word for each of these is reserved for a pointer. A zero in the last word of the ACTIV entry represents the absence of predecessor linkage; that is, a start event for the network. A pointer in this position locates predecessor linkage. A pointer in the last word in the JLIST entry represents additional information for the activity; a zero represents the end of the linkage. Graphically, the linkage appears as follows:



Input to the ISUMM link is the activity file (ATAB) built by the previous link.

Output is the summary activity (SOUT) file.

SUBROUTINE SUMM

The entire ISUMM Link is controlled by the subroutine SUMM. This subroutine examines each activity, determines what action is needed, and calls other subroutines to provide any missing information.

For a start event of the network, SUMM initiates the JLIST entry, passes the JLIST pointer to successor activities, and initiates the step to the next activity.

For activities other than starts, subroutine SUMM examines all the conditions for user selection or program selection and takes the appropriate action--merging parallel paths when possible, linking additional JLIST entries to those already established, separating summary activities, and initiating new summary activities as a result of event selection.

SUBROUTINE SELSAV

This subroutine is called whenever a start event, a completed event, or a user-selected event is encountered. It determines whether any incomplete paths proceed from the event and sets a switch to be interpreted by subroutine SUMM. This provides for program selection of events under Case 1 (page 83). It also prepares for selection under Case 3 by signaling the SUMM subroutine to save, for completed activities, the predecessor event and its actual date, utilizing words four and five of the JLIST format.

SUBROUTINE BUMP

This subroutine steps to the next activity and determines when the activity list in core must be replenished and/or when all activities have been processed.

SUBROUTINE MAKMTY

When JLIST entries are merged as being parallel paths which can be combined, or when a summary activity is defined and written on tape for further processing, portions of the JLIST array can be released for use by the program. The MAKMTY subroutine releases the JLIST locations by delinking them from their previous chain and linking them to the location called EMPTY. Thus, the JLIST array contains only active entries; those no longer in use become immediately available to the program.

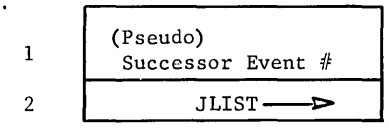
SUBROUTINE SETLOC

This subroutine works with subroutine MAKMTY, to assure the fullest use of the JLIST array. Subroutine SETLOC accesses location EMPTY for the next available JLIST location, delinks the location from the EMPTY chain, and returns a JLIST pointer (1) to the main program. There is a limit of 2,000 JLIST locations available for linkage. Since the program constantly reuses the JLIST locations, exceeding this capacity seems to be virtually impossible. However, if this ever occurs, a message reading "JLIST SPARE

LOCATION EXCEEDS LIMIT OF 2,000" will appear on the printer, and the program will cease execution. This limitation can be improved by increasing the size of the JLIST array, and decreasing the size of the ACTIV array; however, this will cause a sacrifice in timing for extremely large networks.

SUBROUTINE IPASS

The IPASS subroutine scans the activity list and its linkage in an effort to link a JLIST entry to its successor event. When the successor event is not found in the activity list, its linkage is preserved for later consideration by storing the successor event number and the JLIST pointer in the SAVE array as follows:



When the successor event is found within the activity list, the IPASS subroutine compares each item of the JLIST linkage being passed with each item of the JLIST linkage already in existence for the successor event. Merging and linking of the JLIST entries continues until one chain is established for the successor activity.

SUBROUTINE LOOP

This subroutine handles the functions associated with the divergence of the paths of a network from an event. The linkage passed from predecessor activities must be duplicated for each diverging path. Subroutine LOOP builds a new JLIST entry corresponding to the old entry and passes the new chain to its successor through the IPASS subroutine.

In completing the new linkage, the subroutine reads in more activities if the core list is exhausted. There are three returns from this subroutine in addition to the normal one:

RETURN	-	Normal return to next instruction after call
RETURN 1	-	Last activity encountered
RETURN 2	-	Further JLIST linkage to be duplicated
RETURN 3	-	More diverging activities necessitate further processing.

SUBROUTINE IOUT

This subroutine is called when a new list of activities is read into the ACTIV array. It completes the tasks left after the processing of the previous list. In particular, it must examine the SAVE array to determine whether any successor events not found in the last activity list can be found in the new list. Any successor events found are passed to the JLIST linkage. Those successor events not found are returned to the SAVE array unless they can be identified as end events; in this case, they are moved to the JOUT array, ready to be processed as summary activities.

SUBROUTINE MRJ

Subroutine MRJ examines the JOUT array of end events in an attempt to merge parallel paths using the IPASS subroutine. When the JOUT linkage to the JLIST entries is reduced

as much as possible, the subroutine PUNCH is called to write the summary activity file. Upon completion of the JOUT list, control is returned to the main program.

SUBROUTINE PUNCH

This subroutine writes a file of summary activities to be punched on cards by a later program section. The record format for the summary file SOUT is:

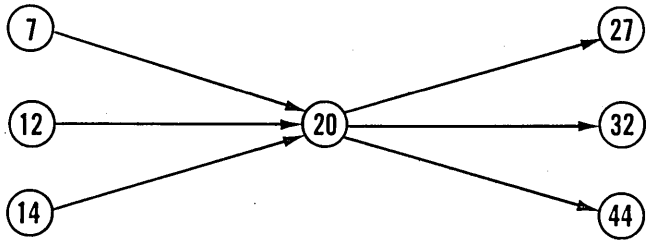
1	Pseudo- Pred-Event # (Source)
2	Pseudo- Succ-Event #
3	Σt_e or Largest Bin. Date
4	Complete (2) or Incomplete (1)

SUBROUTINE NCOUNT

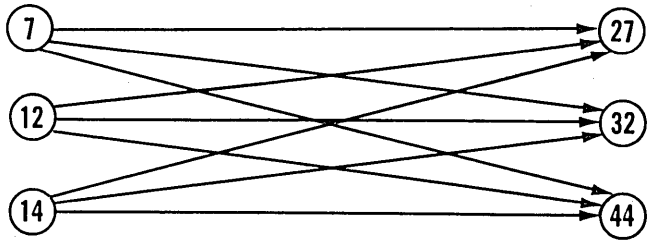
Subroutine NCOUNT is optional to the Network Summarization Program. The programmer may remove this entire routine by deleting the call to subroutine NCOUNT from the SUMM subroutine (the instruction immediately preceding statement 2004 SUMM).

Subroutine NCOUNT determines whether the exclusion of an event from the network summary will simplify the network or greatly complicate it.

Consider, for example, the network section shown below:



If events 7, 12, 14, 27, 32, and 44 are all selected events (either user or program selected), the summary network would yield:



Thus, seven events are reduced to six; however, six activities are replaced by nine. Obviously, this is not an effective summary, and the logic of the original relationship between the selected events is essentially lost. The number of summary activities within

a network section can be held to a minimum, and the logical relationship maintained by including within the summary the following types of events:

1. Events in which more than two summary activities merge and more than one activity diverges.

2. Events in which two summary activities merge and more than two activities diverge.

The NCOUNT subroutine counts the merging and diverging summary activities and sets the IFIX flag when selection of the event under examination will clarify the summary network.

Generate Card Format Link--IPUNCH, Subroutines WRITEP and CALOUT

The IPUNCH link of the Summarization Section reads the EVE file and builds an event table and a level table in core memory. This facilitates the conversion of pseudo-event numbers to the user's event numbers. It also assures that an event level code indicated on any activity record will be present on the summary activity. The preliminary file (SOUT) of summary events written by the ISUMM section is, thus, converted to the user's original format; binary calendar days are converted to actual calendar days through the CALOUT subroutine, and the summary activities are written in revised form for file POUT:

1	Succ. Event # (J1)
2	Pred. Event # (I1)
3	Transaction Code (TC)
4	Pred. Level Code (ILEV)
5	Succ. Level Code (JLEV)
6	Σt_e (M)
7	Actual Date (DATE)

Input to the IPUNCH link is the summary activity (SOUT) file built by last link and the event (EVE) file built by ILEVEL link.

Output is the punch summary (POUT) file containing level indicators and user event numbers.

Sort Link--JSORT1, Subroutines JSORT, SRTMAS, and IFIXJ

The JSORT1 link uses the GE-625/635 Sort/Merge Generator to sort first the POUT file of summary activities by employing JSORT and then the master file by using SRTMAS--both according to the successor event numbers of the activity. (The input coding element for SRTMAS is IFIXJ.) The sorts permit descriptions and scheduled dates from the master file to be punched on the output punched cards for the summary activities.

Input to the JSORT1 link is the punch summary (POUT) file built by IPUNCH link and the master (OMAS) file.

Output is the new punch summary (IOUTS) file in successor event sort order and the master file in successor event sort order (NMAS) cards for Summarized Network Link-IFINI, Subroutine GETD.

The IFINI link combines information from the master file with that of the summary activity file to punch cards in PERT format for the summarized network. This concludes the Summarization Section.

Input to the IFINI link is the new punch summary (IOUTS) file built by JSORT1 link and the master file (NMAS) built by JSORT1 file in successor event order.

Output is the holiday (NVAC) file restored and the summary cards punched in PERT format.

SUMMARIZATION SECTION FLOWCHARTS

The following pages contain the flowcharts for the Summarization Section listed in the following order:

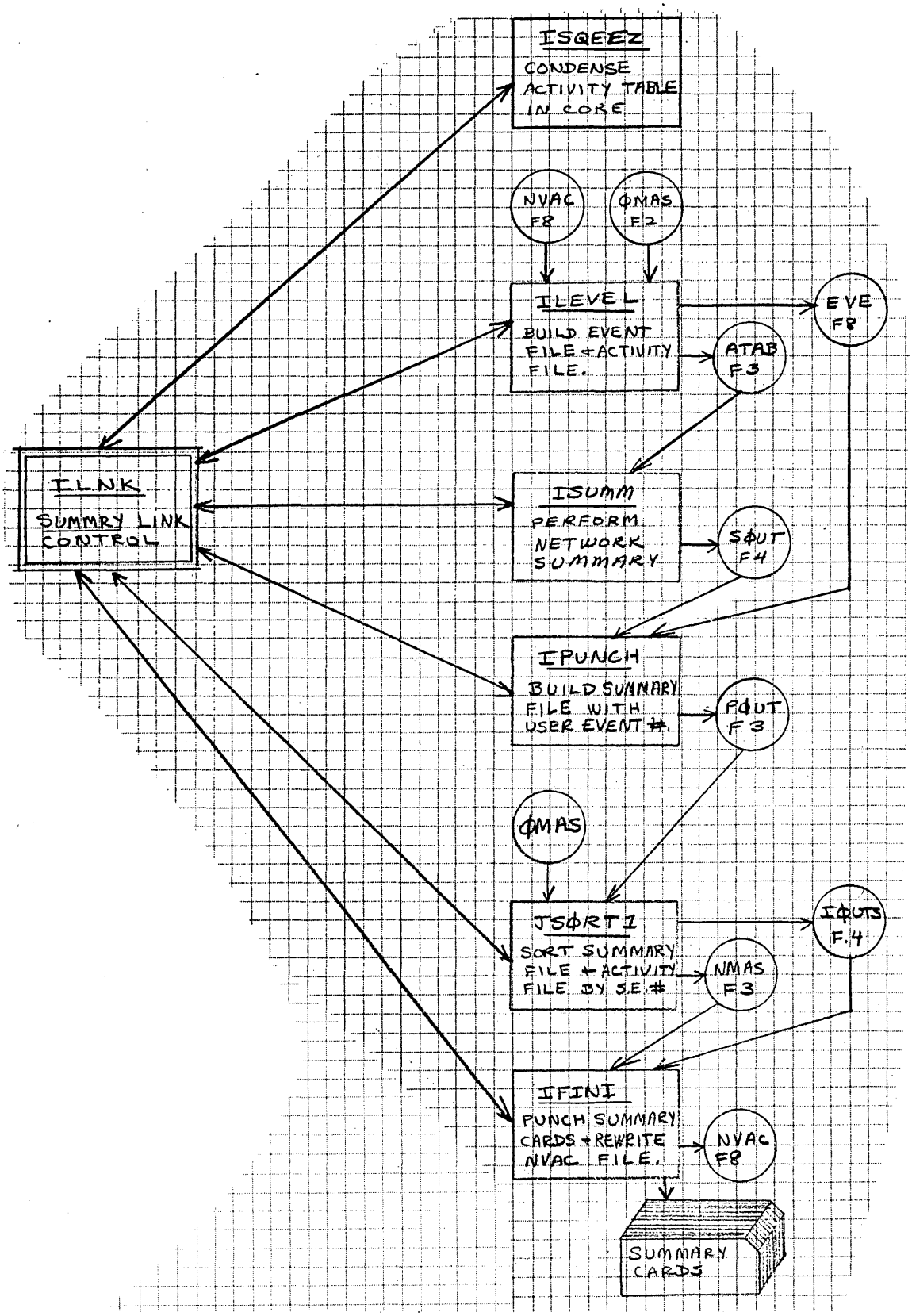


Figure 44. SUMMRY Link Control and Intermediate Files

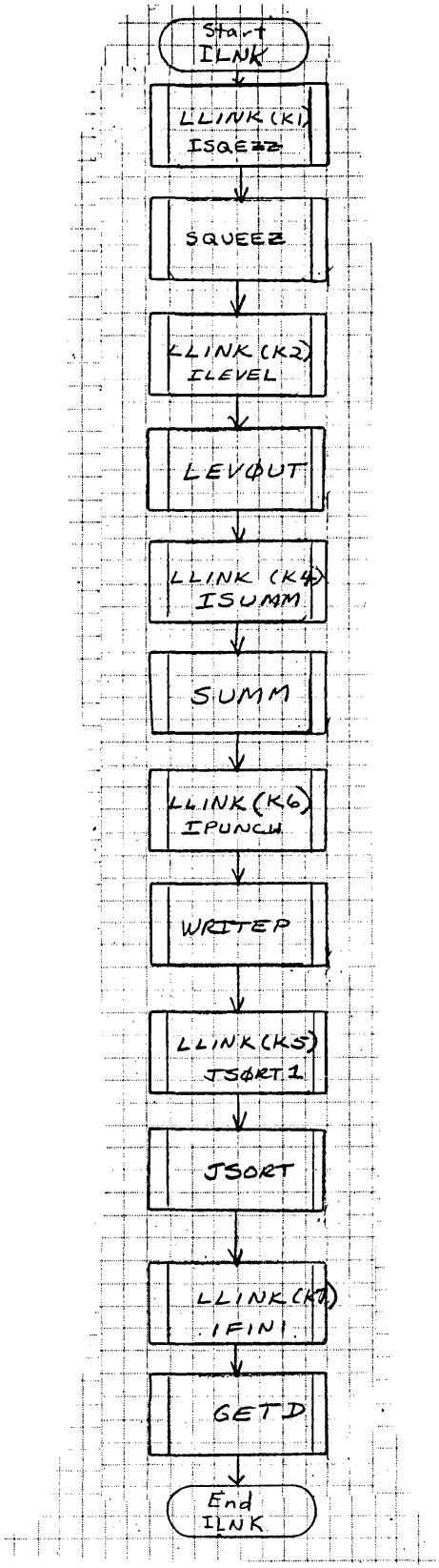


Figure 45. Link ILNK Flowchart

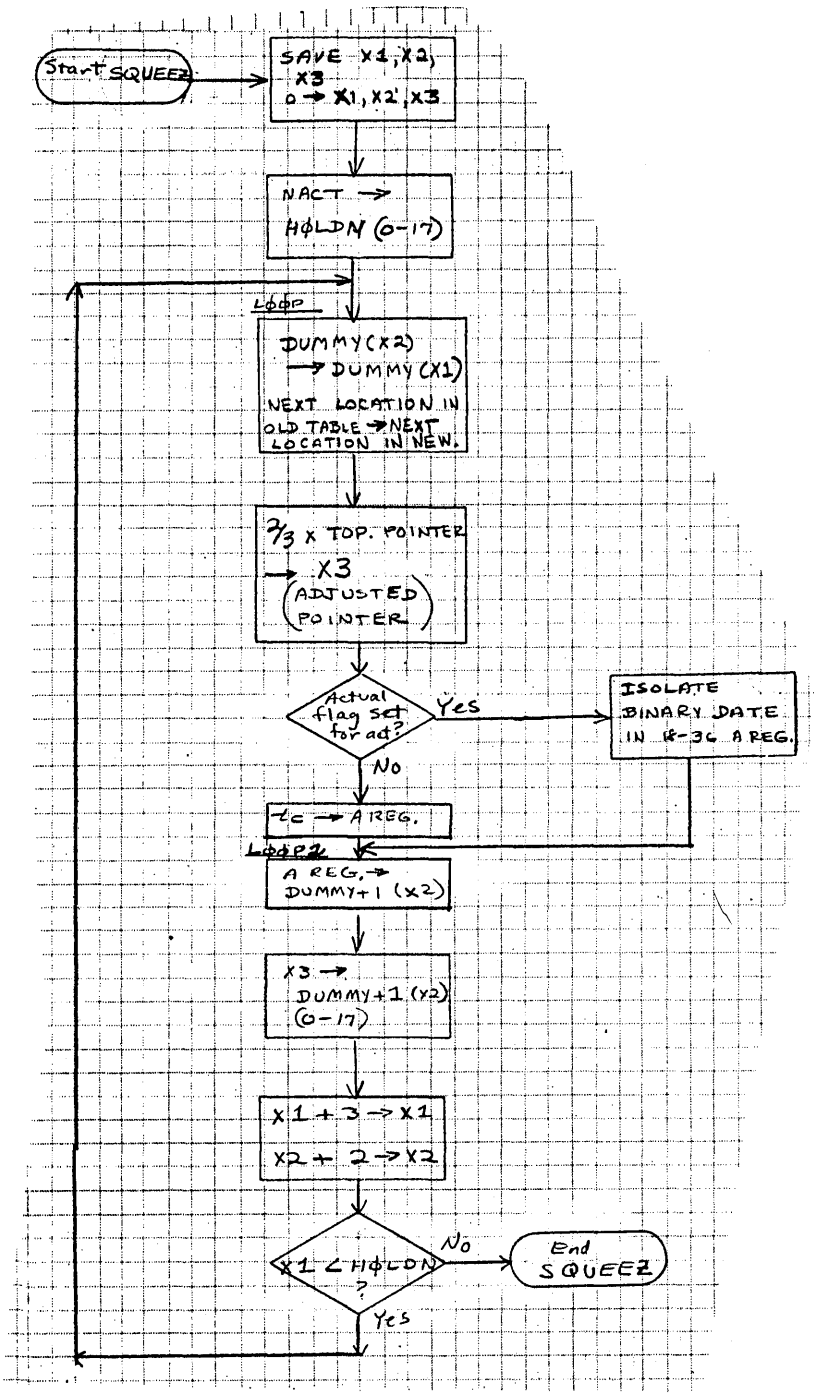


Figure 46. SQUEEZ Subroutine Flowchart

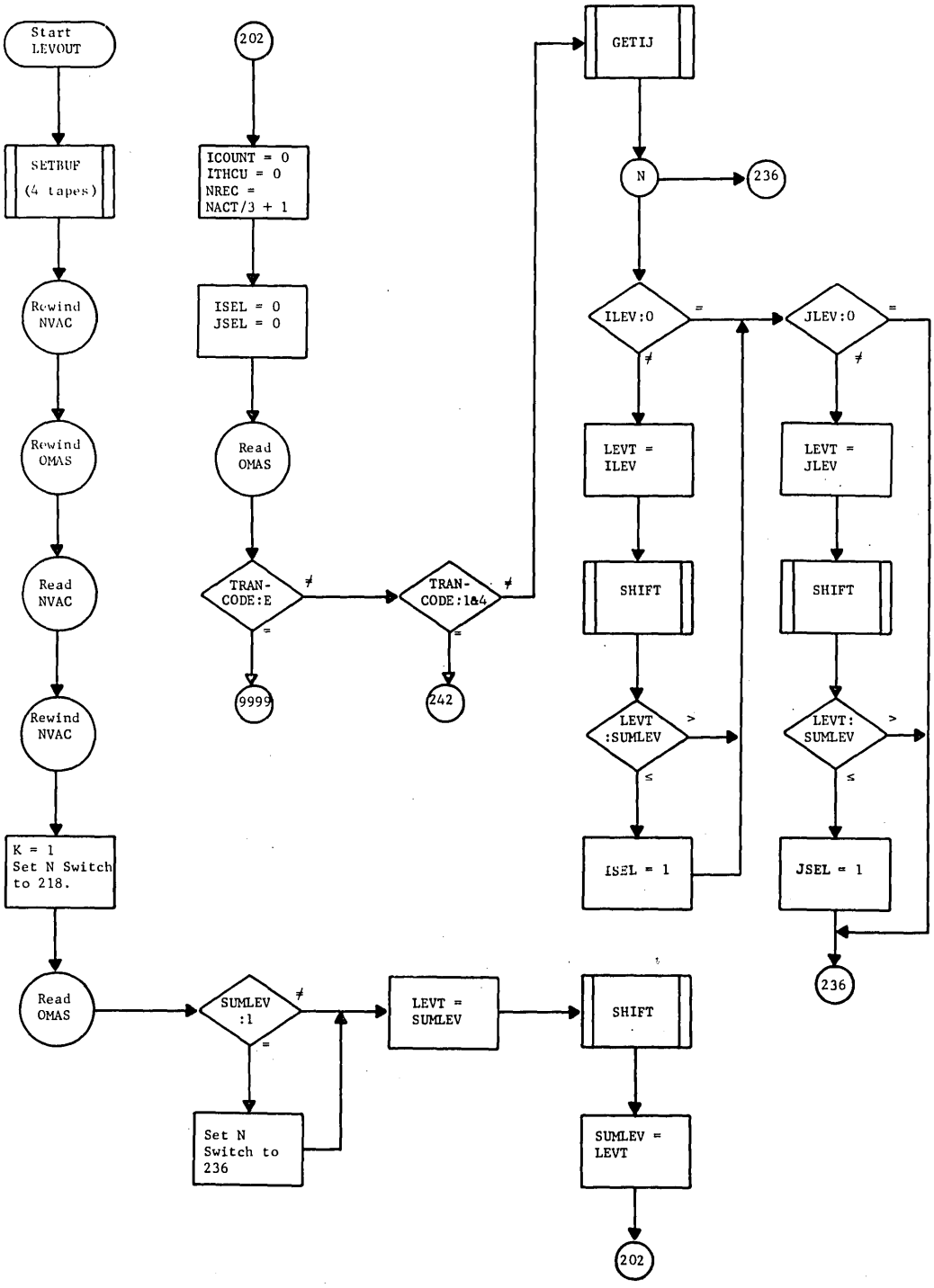


Figure 47. LEVOUT Subroutine Flowchart P1

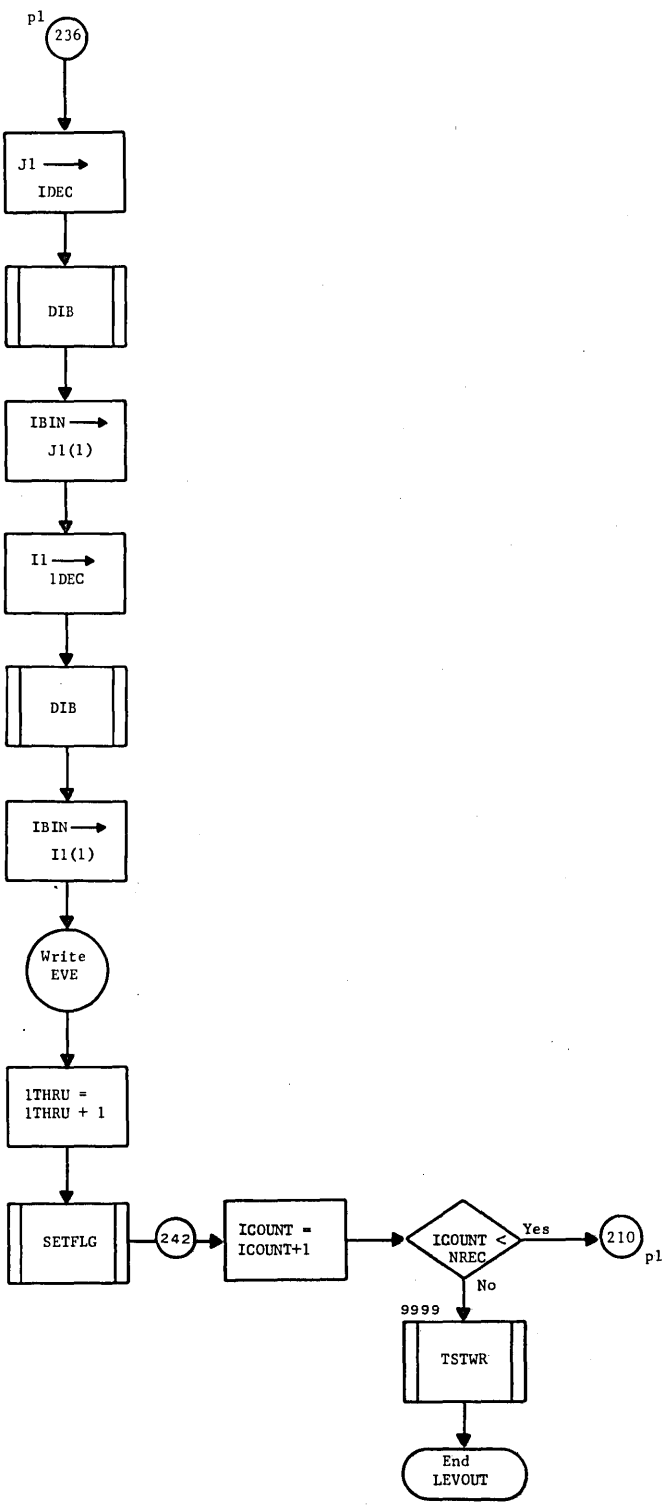


Figure 47. LEVOUT Subroutine Flowchart (cont'd) P2

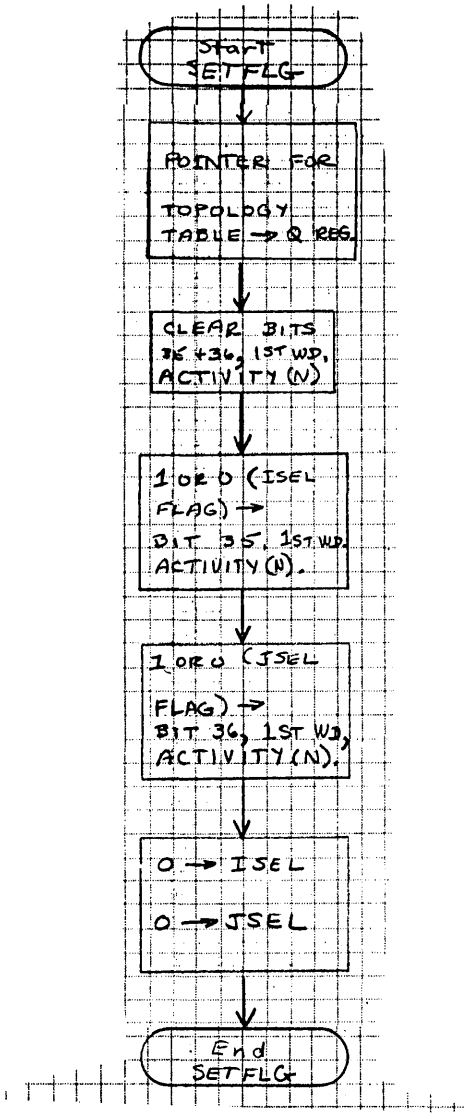


Figure 48. SETFLG Subroutine Flowchart

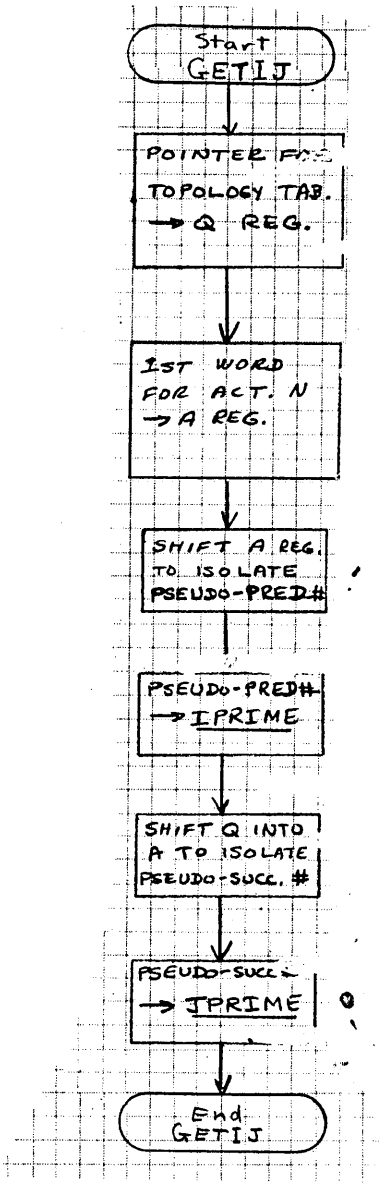


Figure 49. GETIJ Subroutine Flowchart

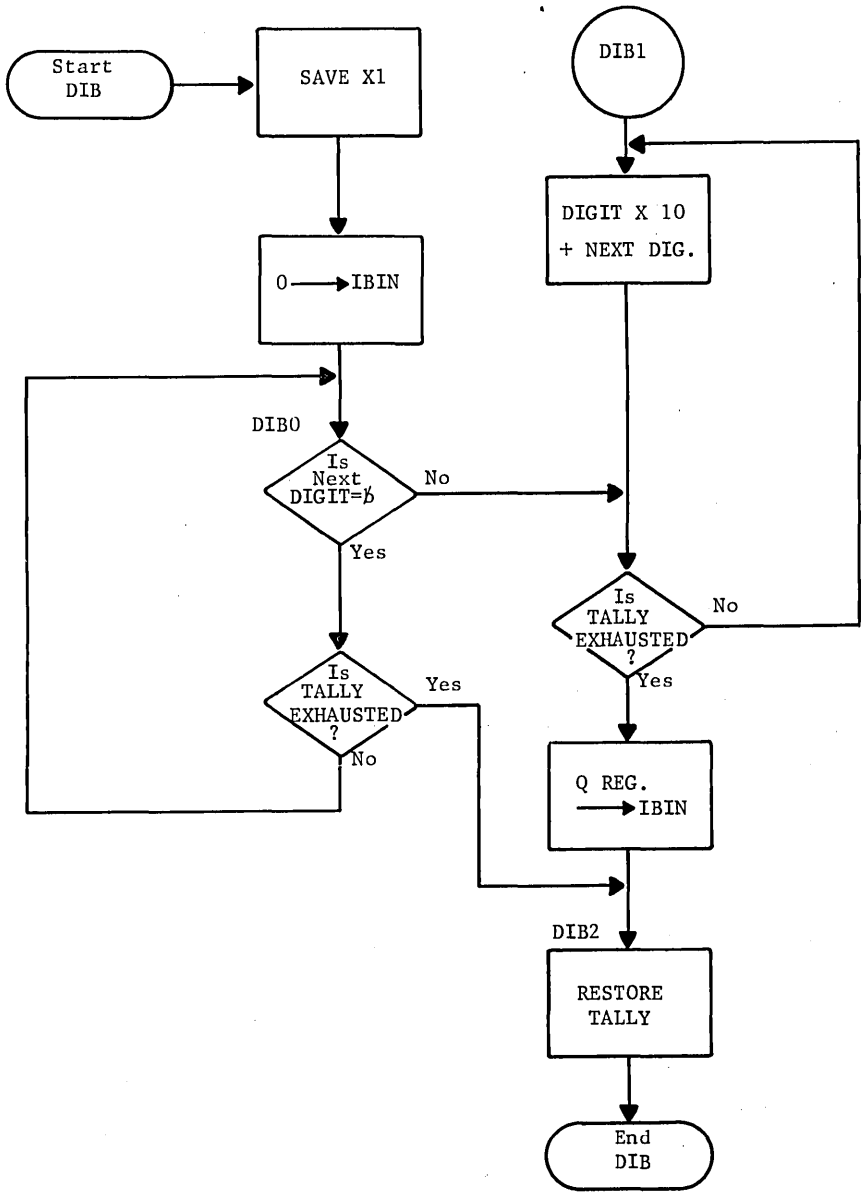


Figure 50. DIB Subroutine Flowchart

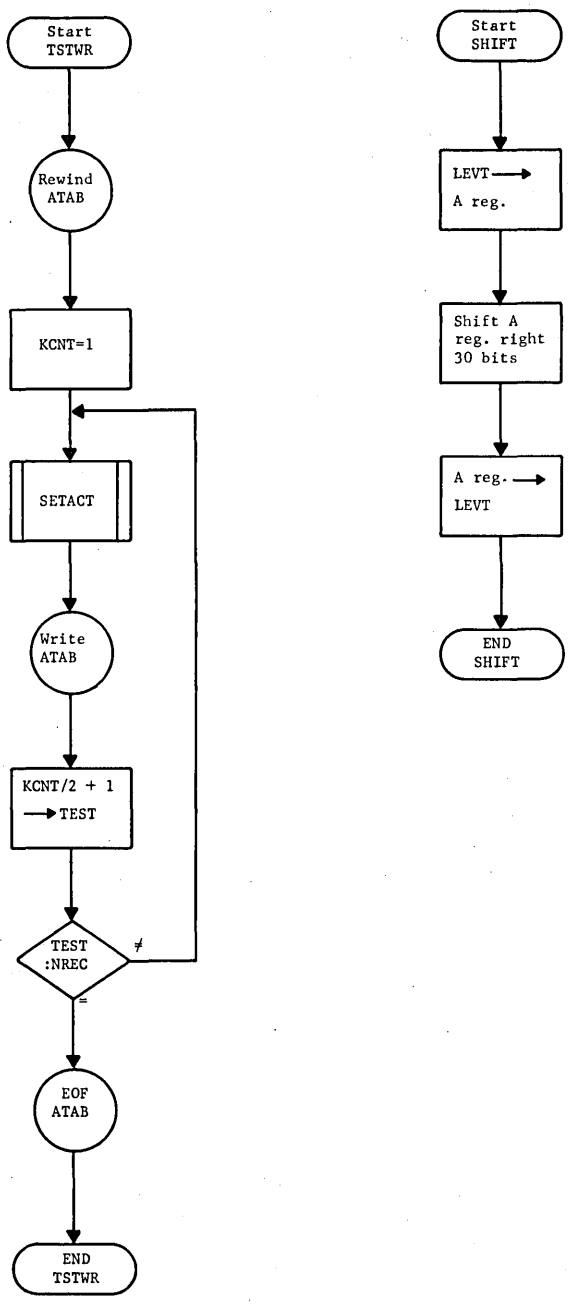


Figure 51. TSTWR Subroutine Flowchart

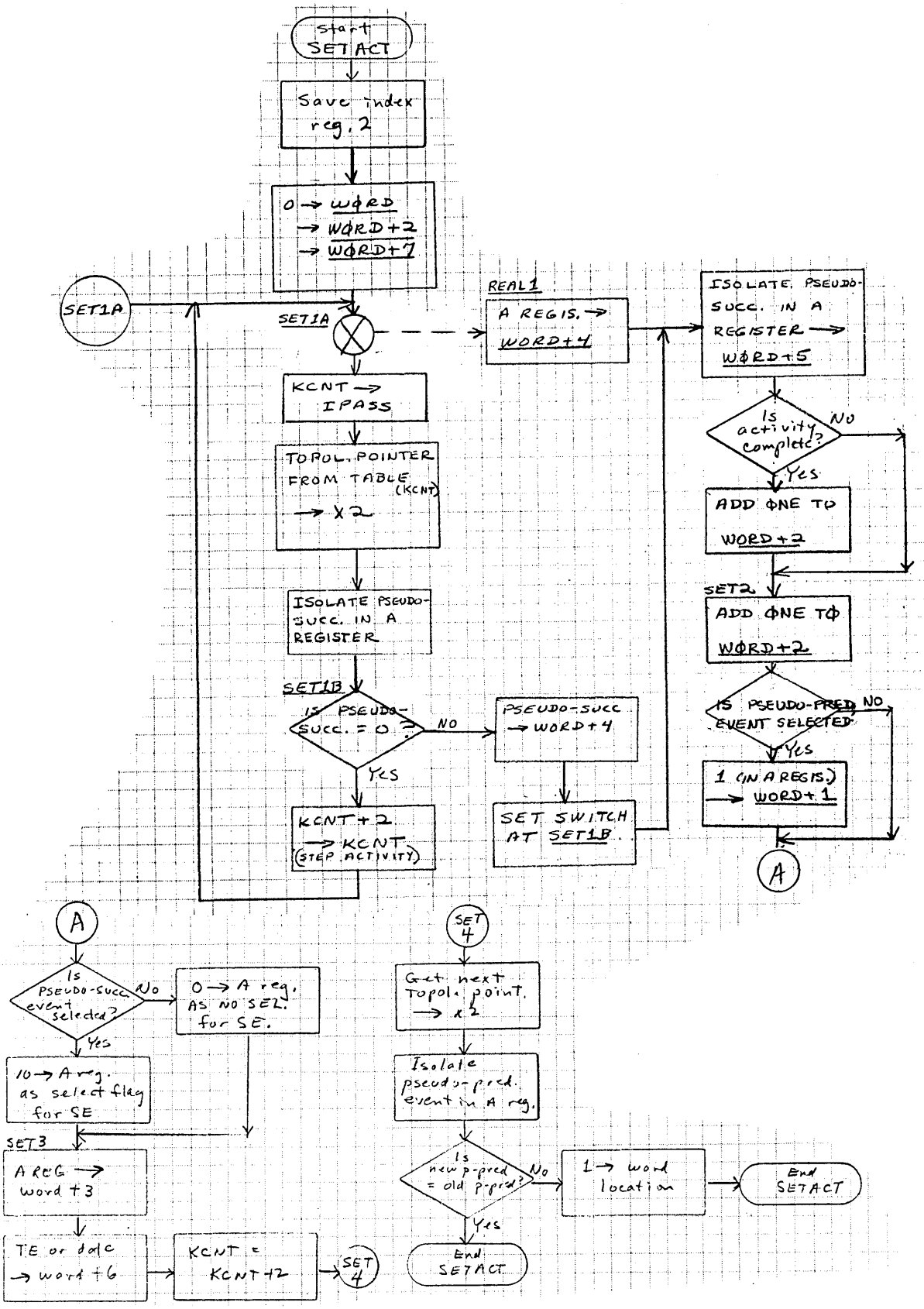


Figure 52. SETACT Subroutine Flowchart

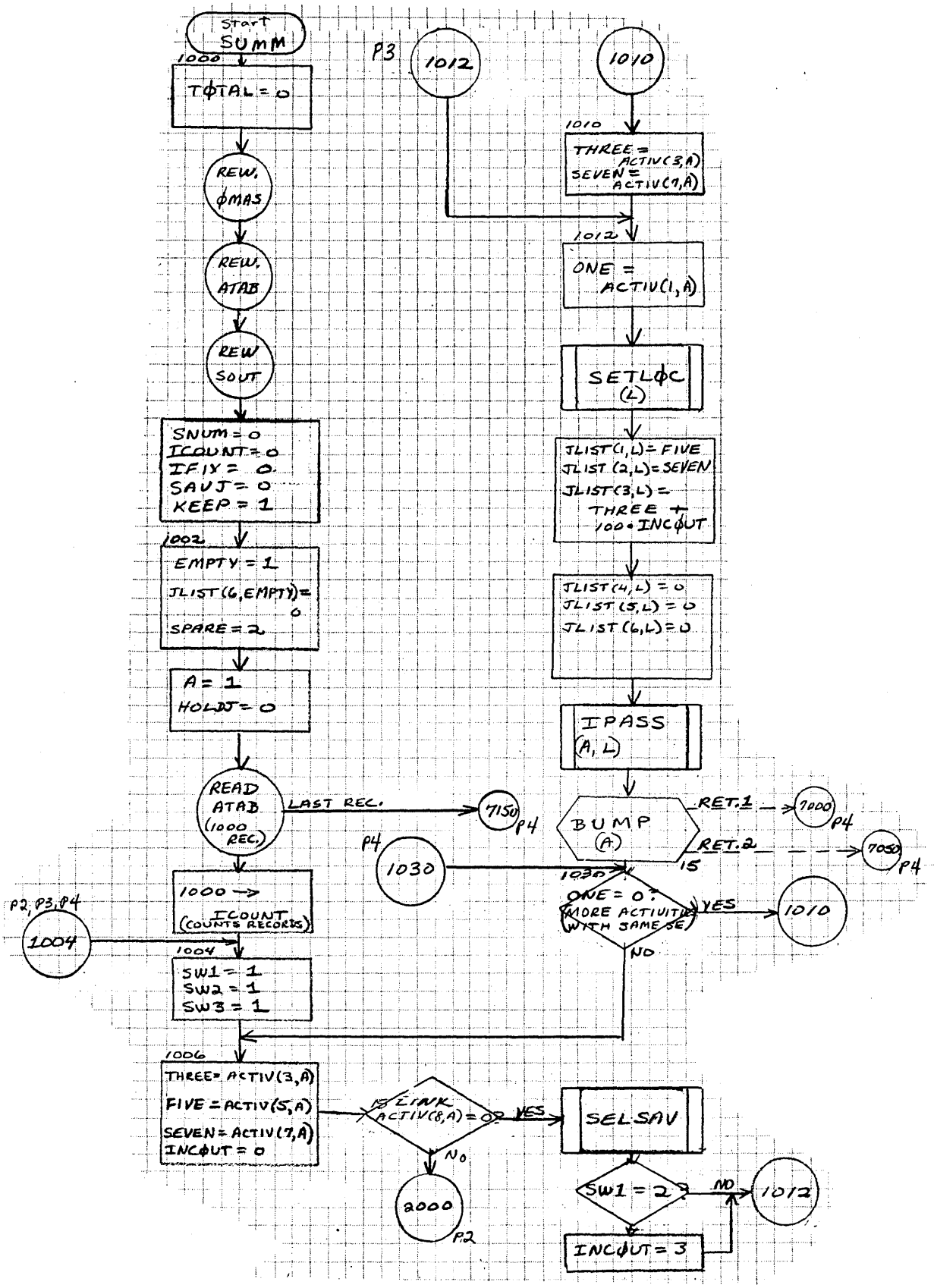


Figure 53. SUMM Subroutine Flowchart P1

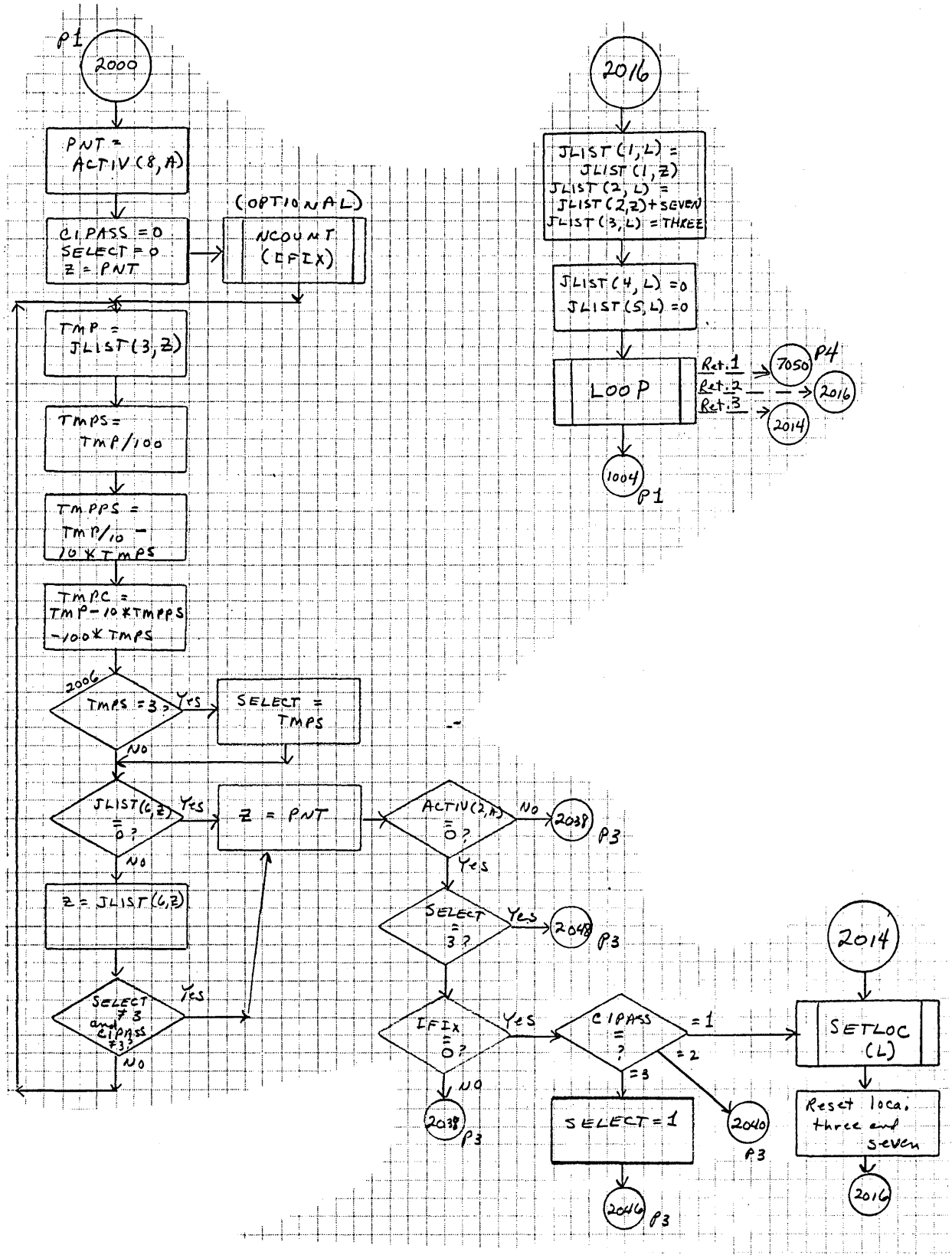


Figure 53. SUMM Subroutine Flowchart (cont'd) P2

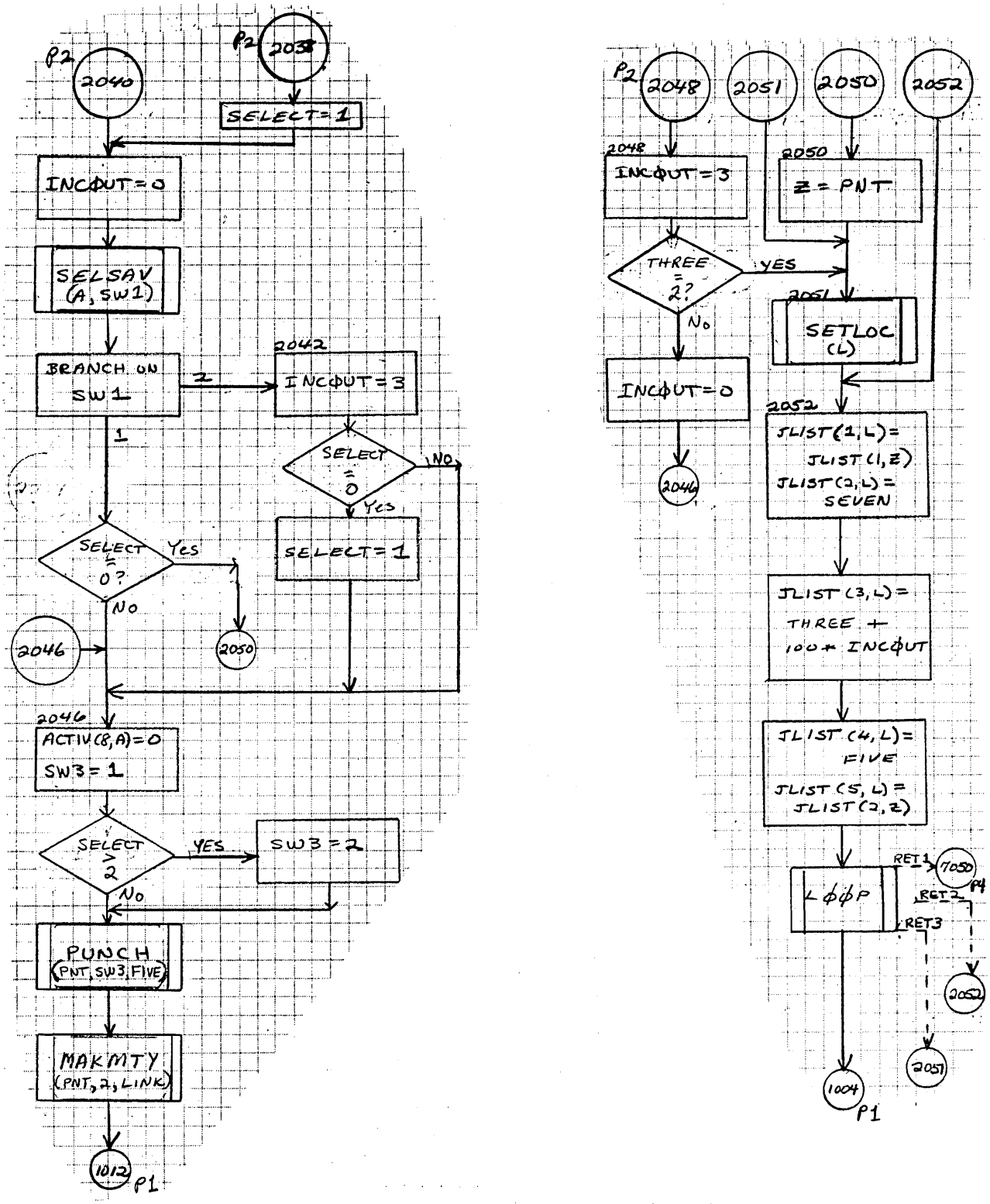


Figure 53. SUMM Subroutine Flowchart (cont'd) P3

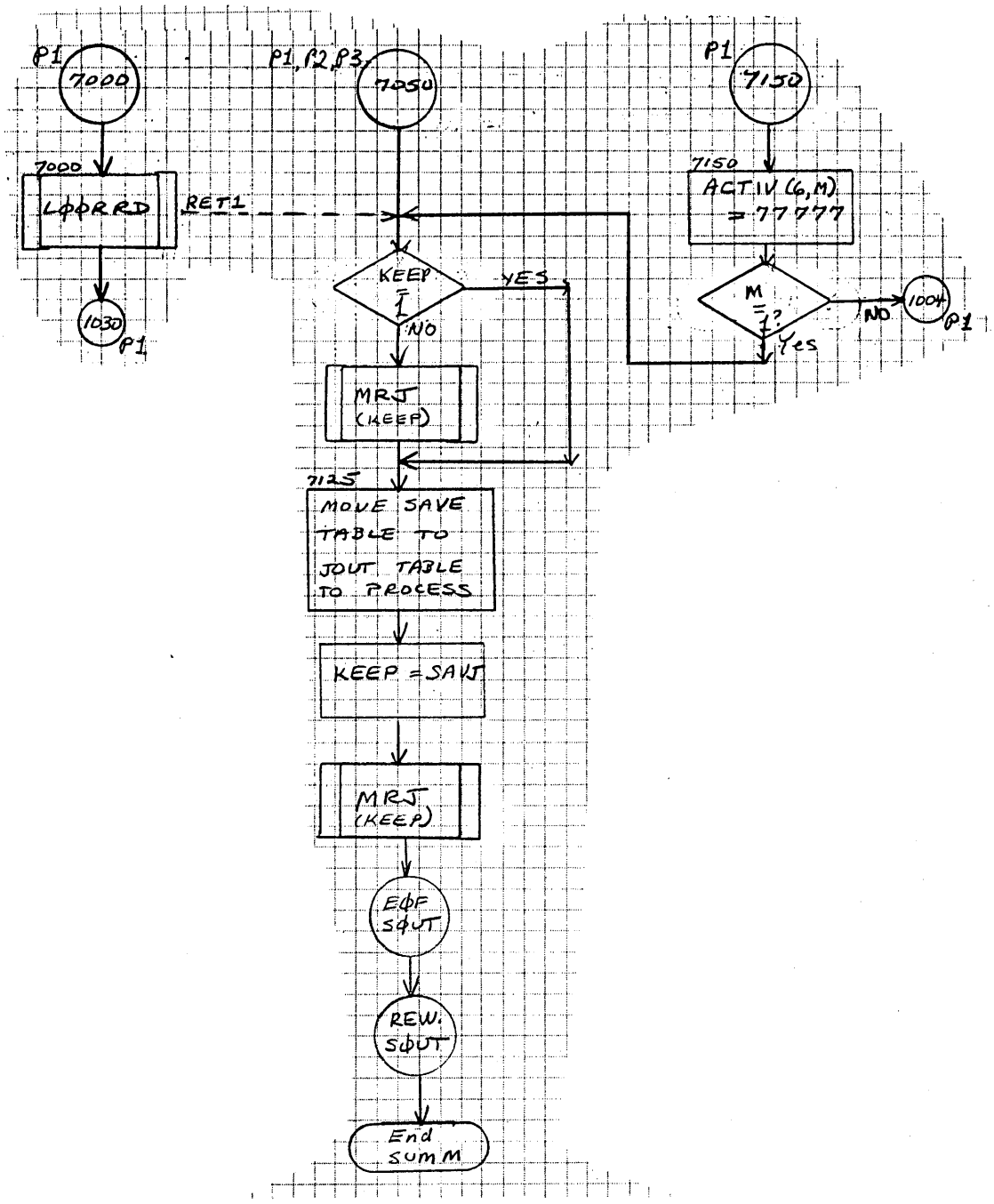


Figure 53. SUMM Subroutine Flowchart (cont'd) P4

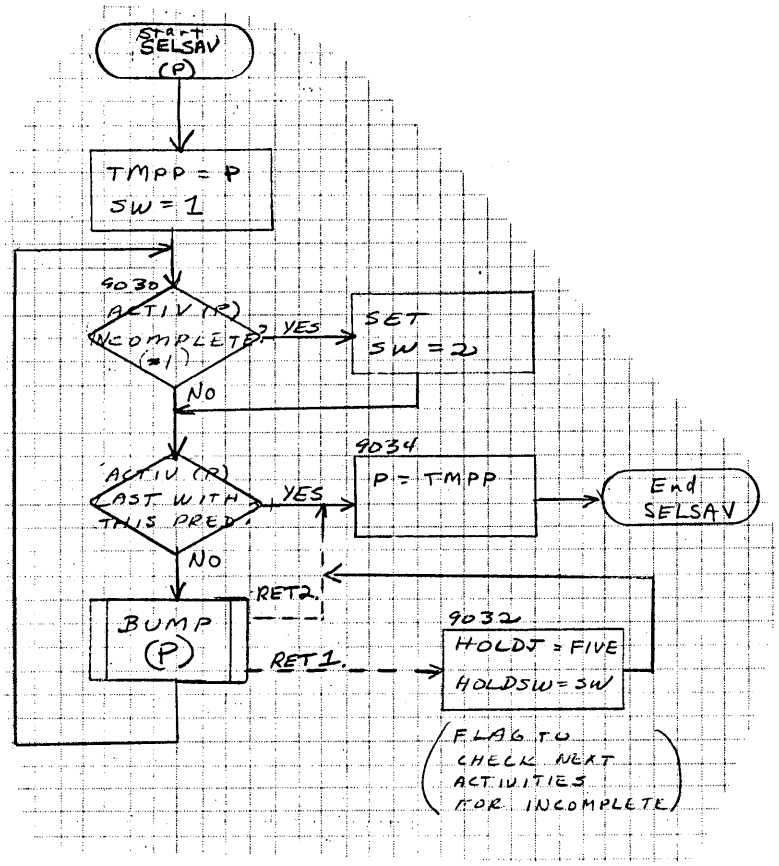


Figure 54. SELSAV Subroutine Flowchart

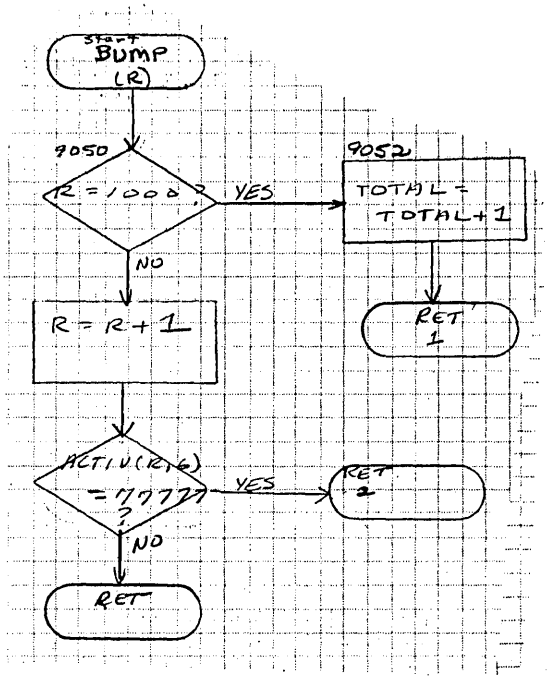


Figure 55. BUMP Subroutine Flowchart

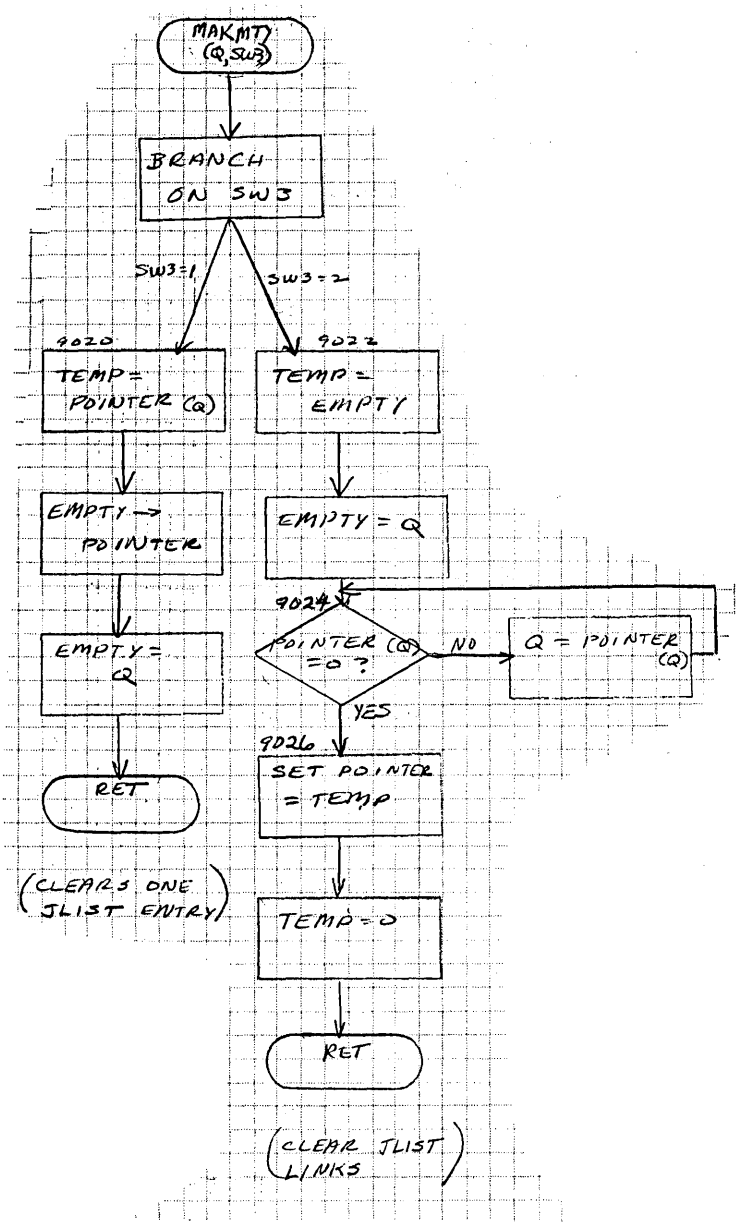


Figure 56. MAKMTY Subroutine Flowchart

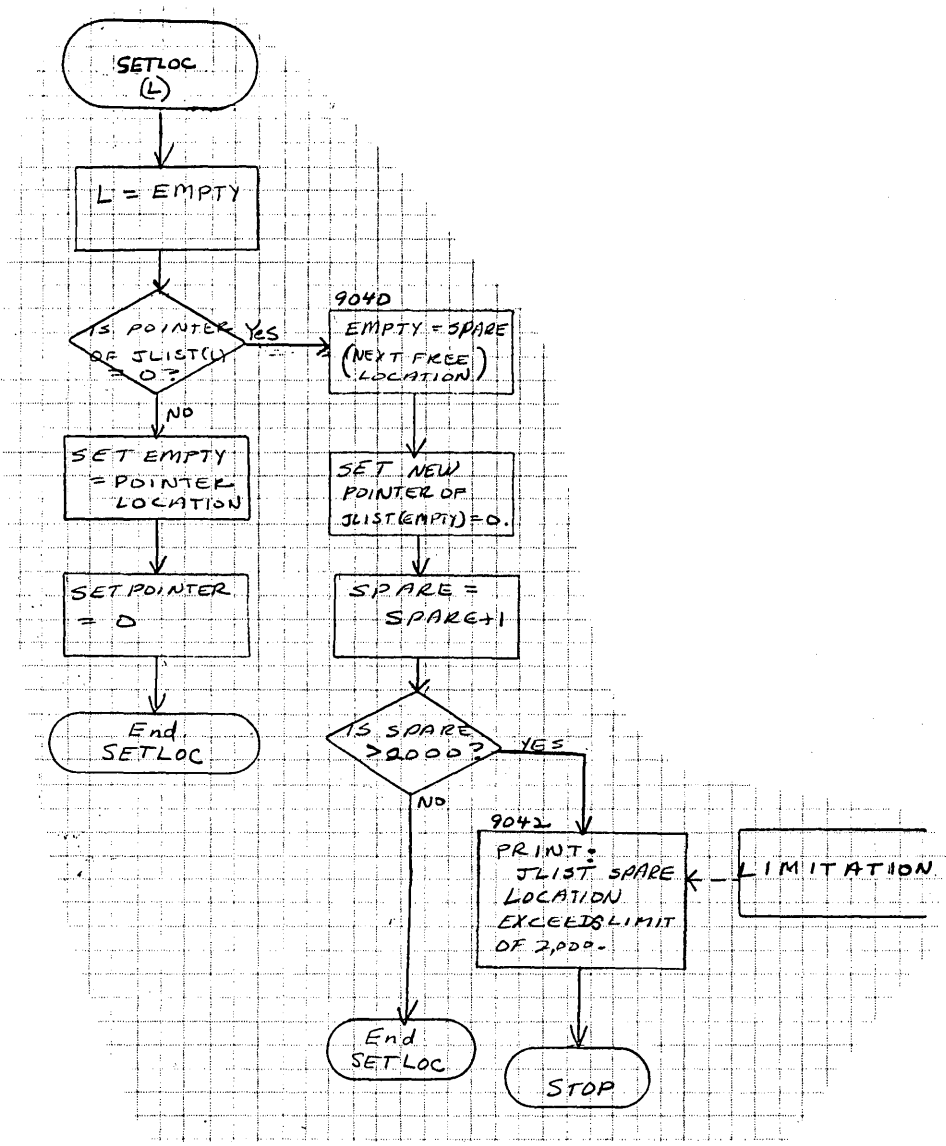


Figure 57. SETLOC Subroutine Flowchart

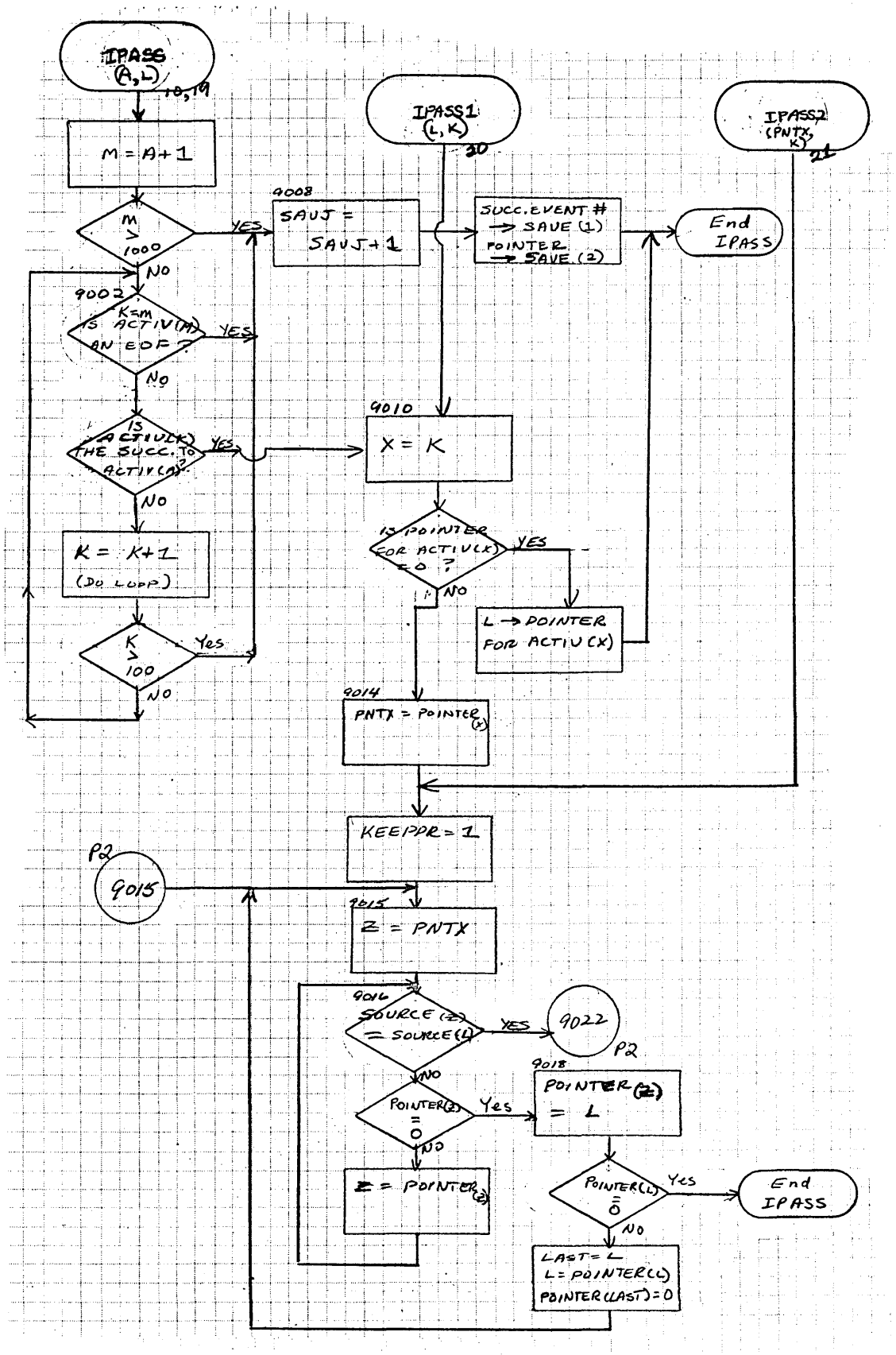


Figure 58. IPASS Subroutine Flowchart P1

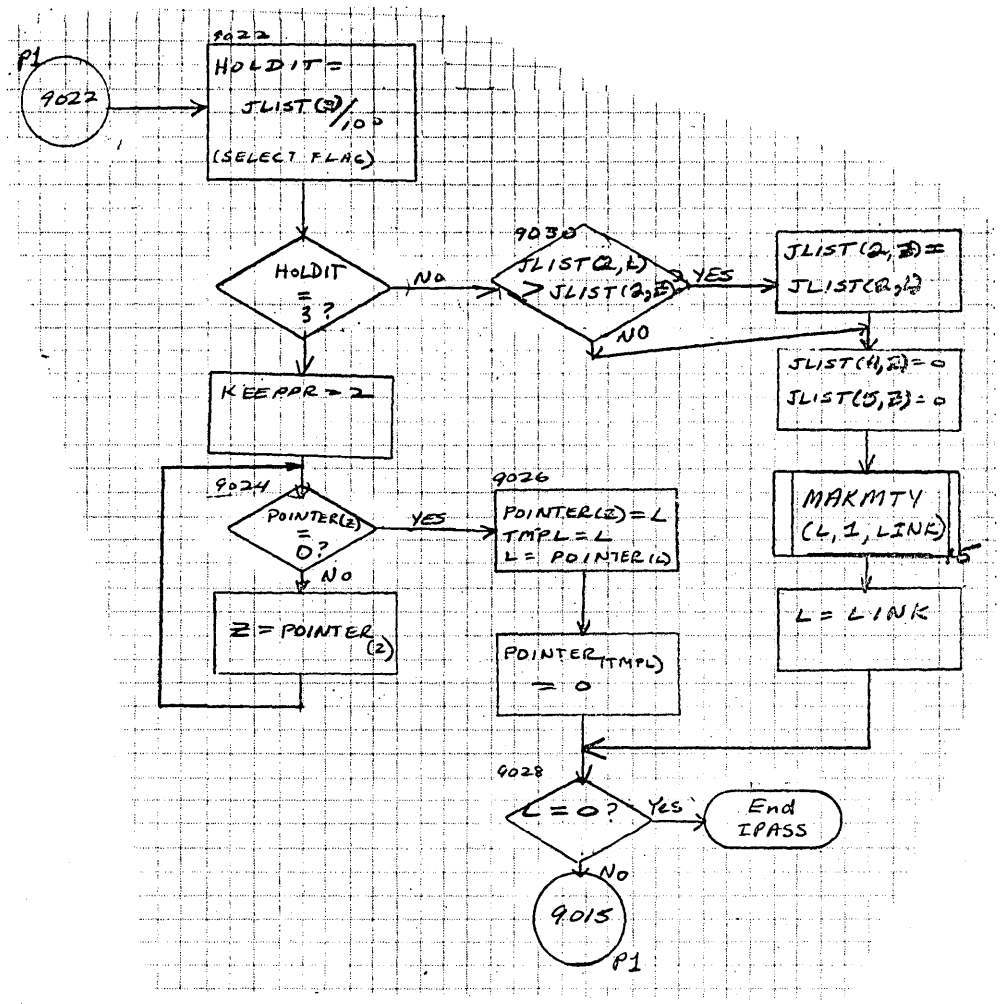


Figure 58. IPASS Subroutine Flowchart (cont'd) P2

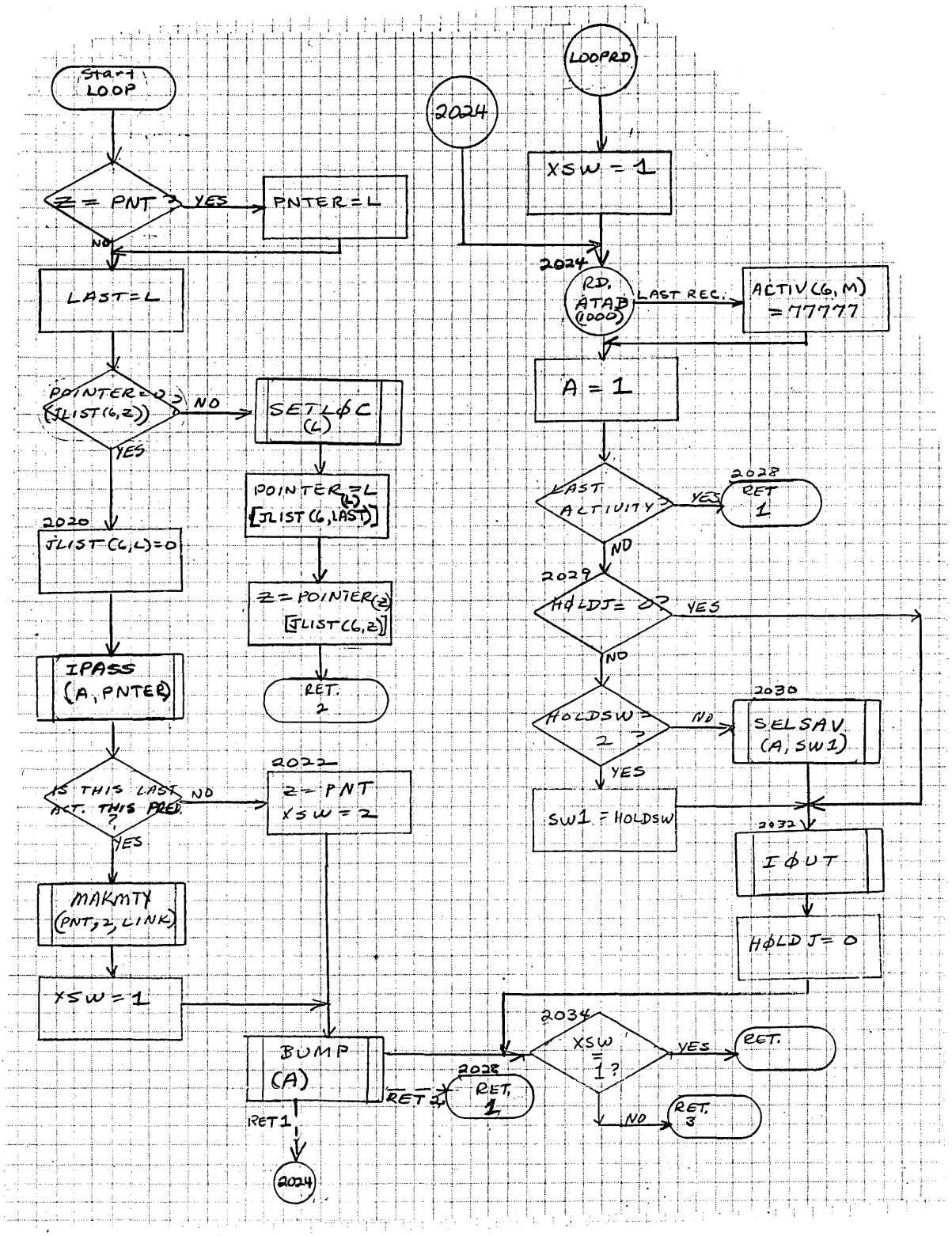


Figure 59. LOOP Subroutine Flowchart

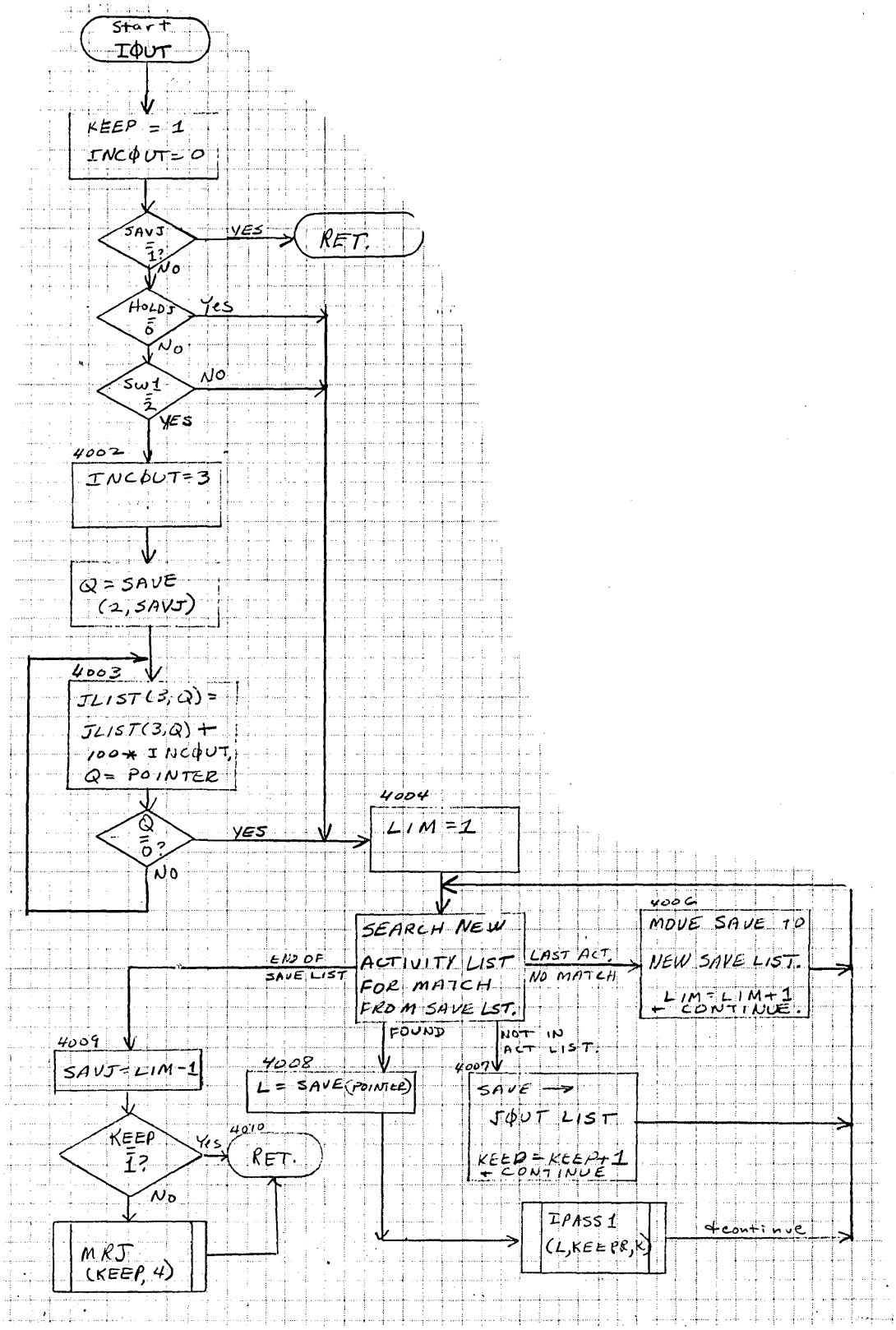


Figure 60. IOUT Subroutine Flowchart

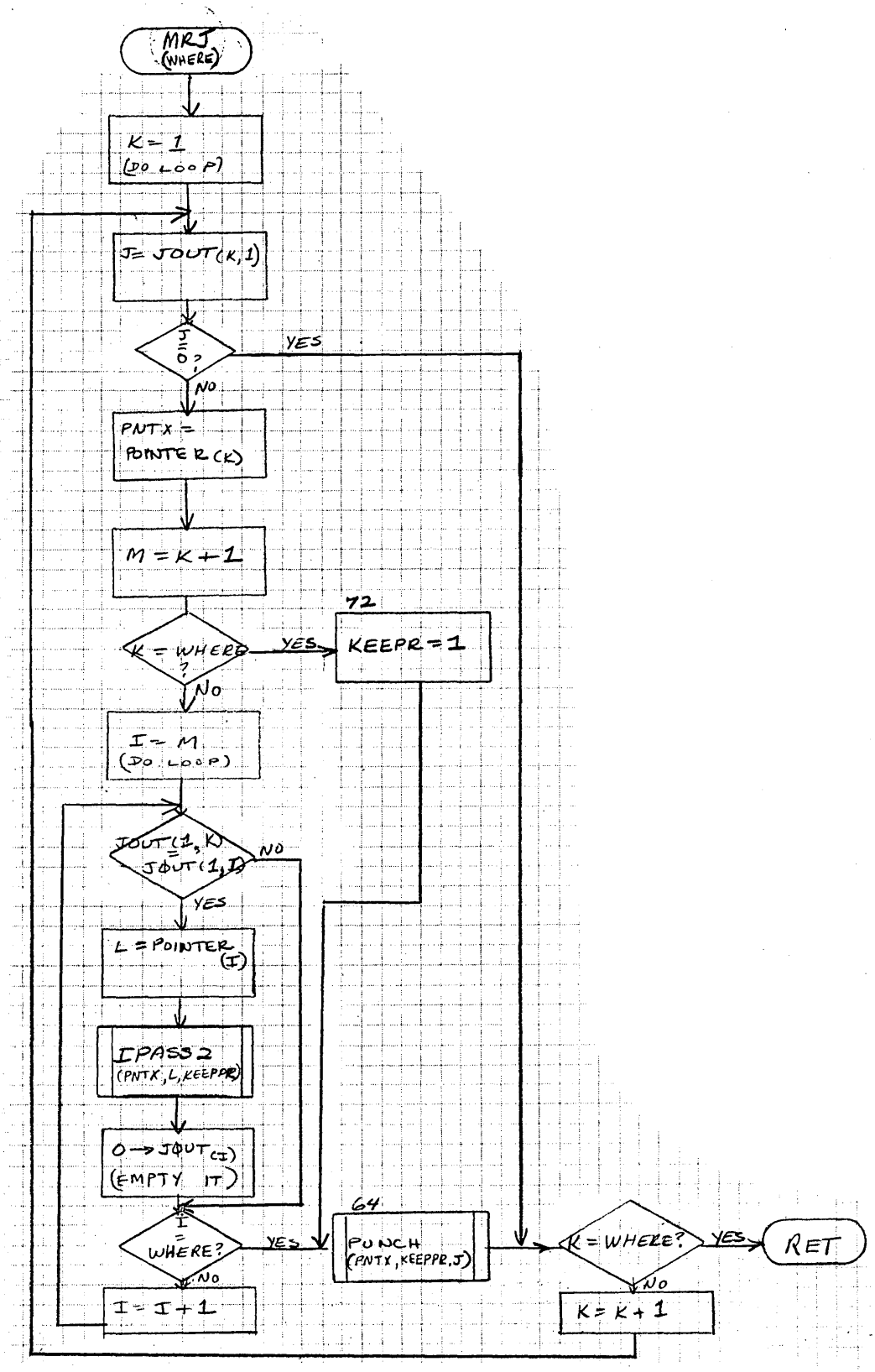


Figure 61. MRJ Subroutine Flowchart

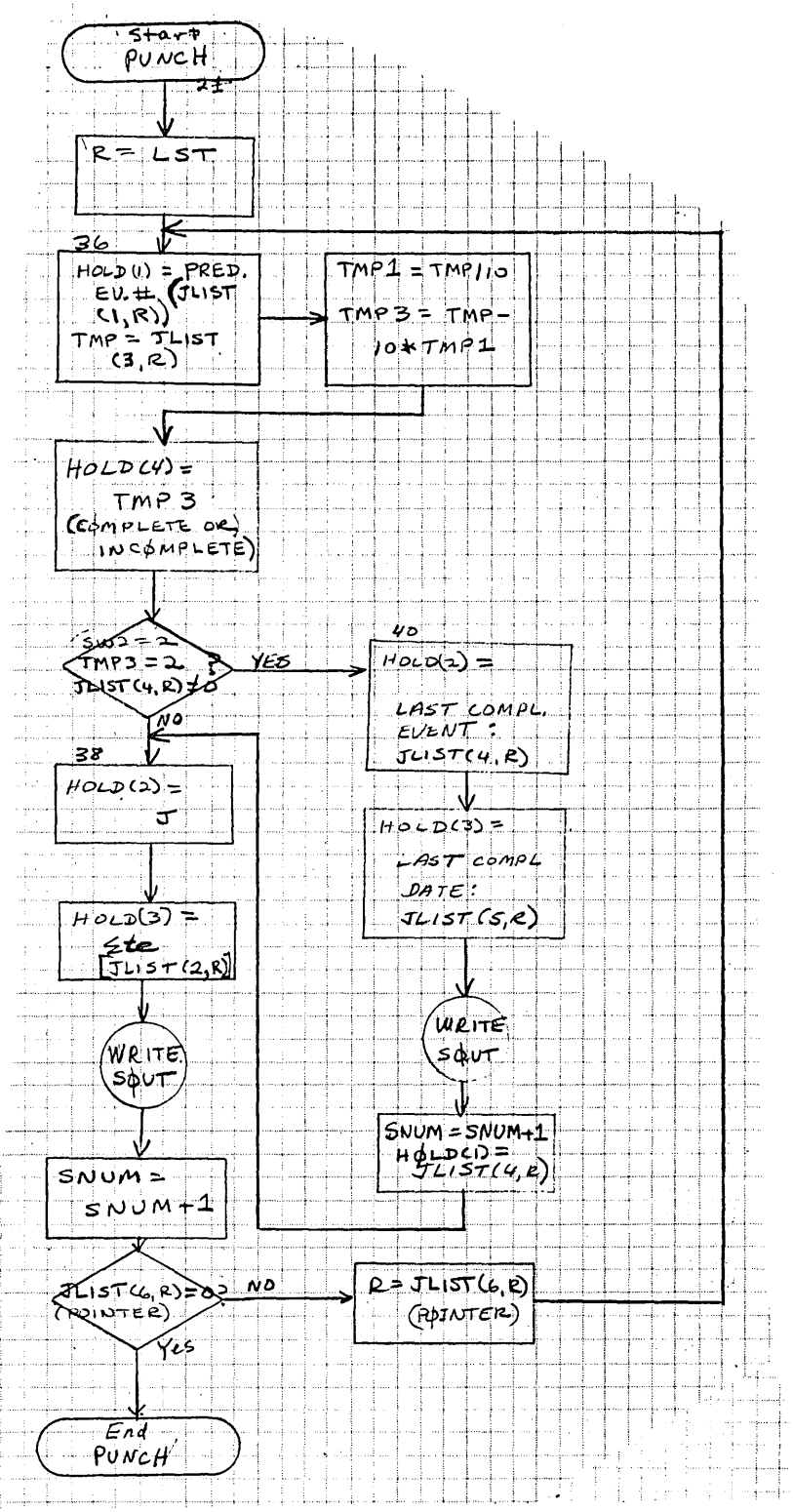


Figure 62. PUNCH Subroutine Flowchart

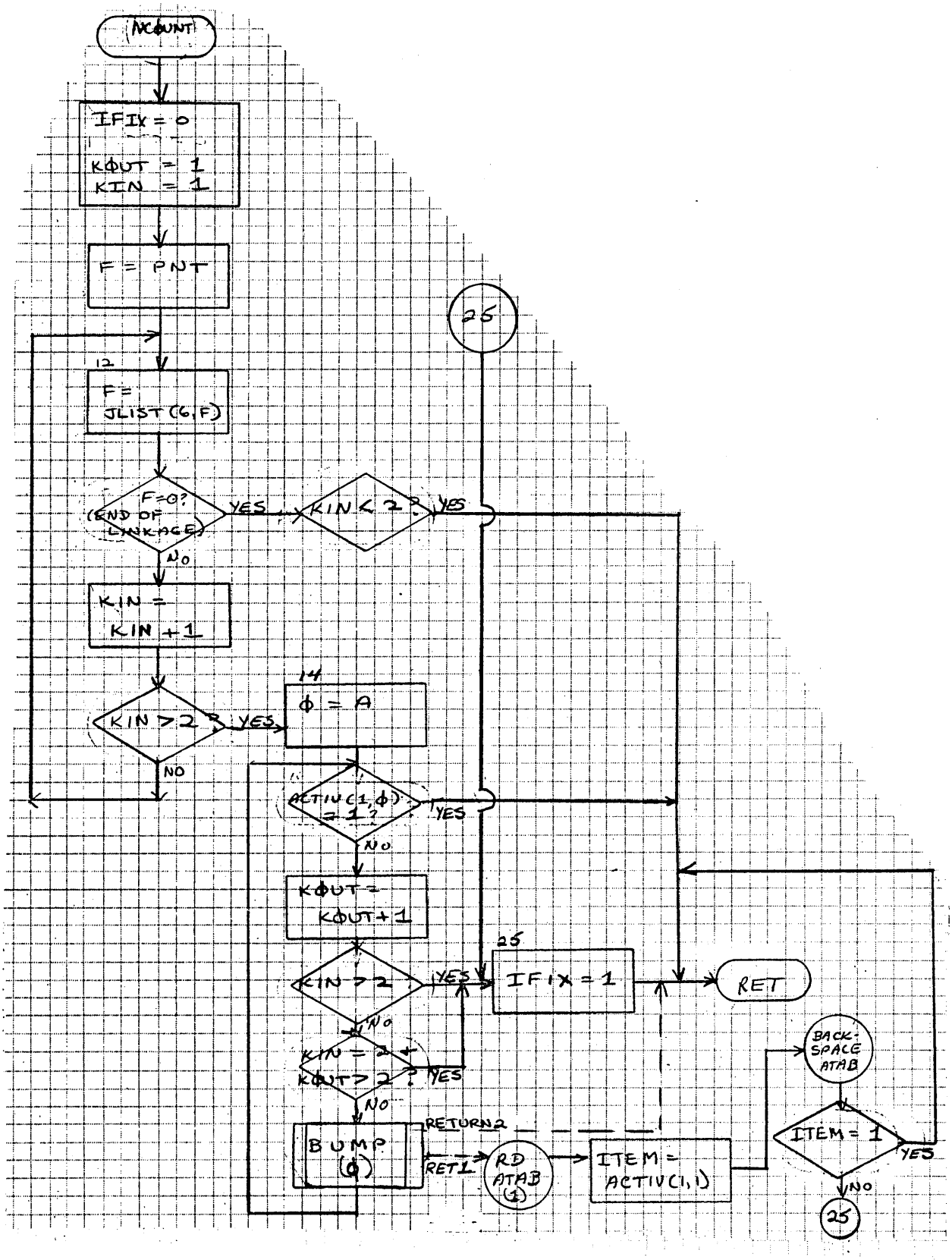


Figure 63. NCOUNT Subroutine Flowchart

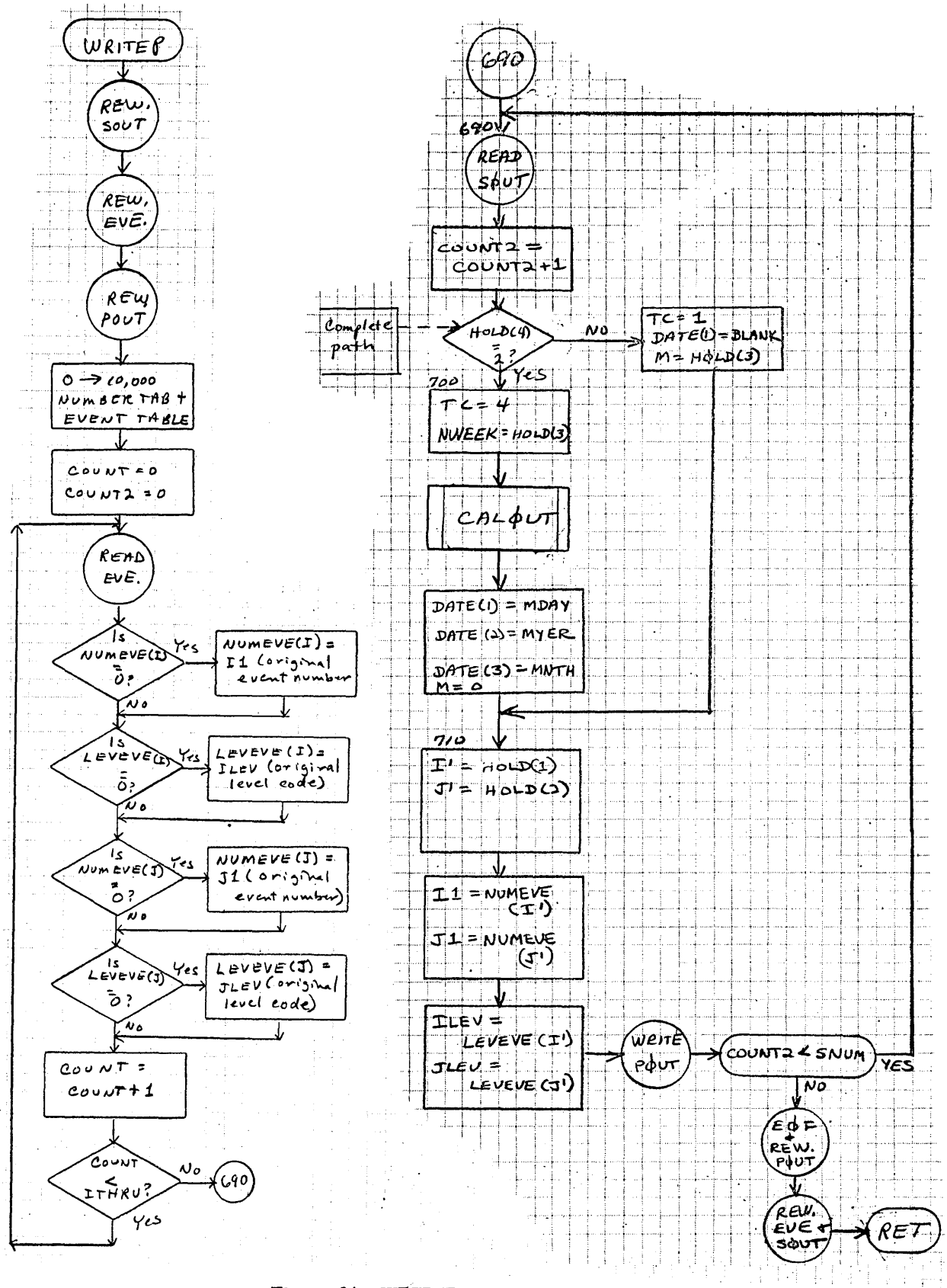


Figure 64. WRITEP Subroutine Flowchart

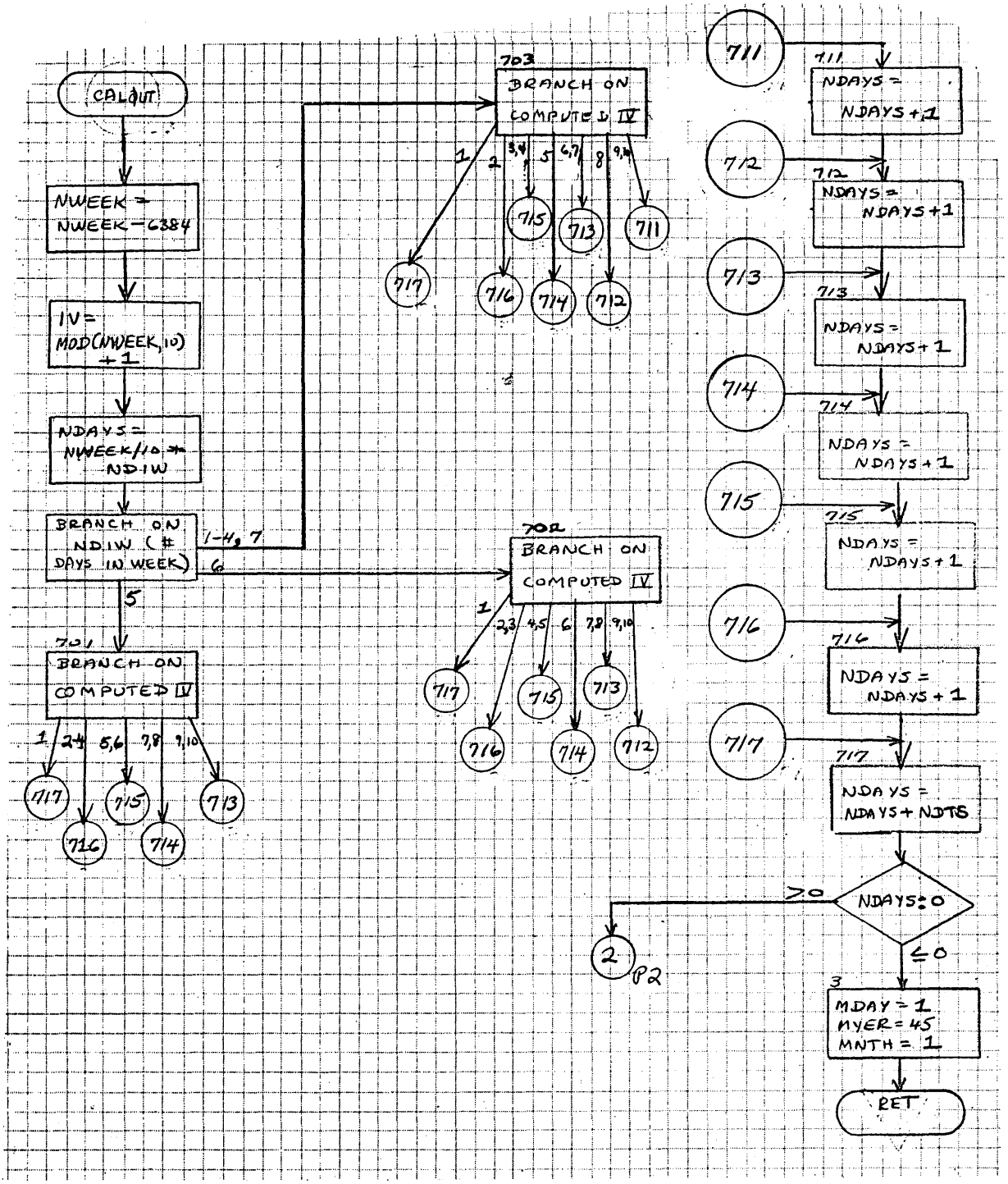


Figure 65. CALOUT Subroutine Flowchart P1

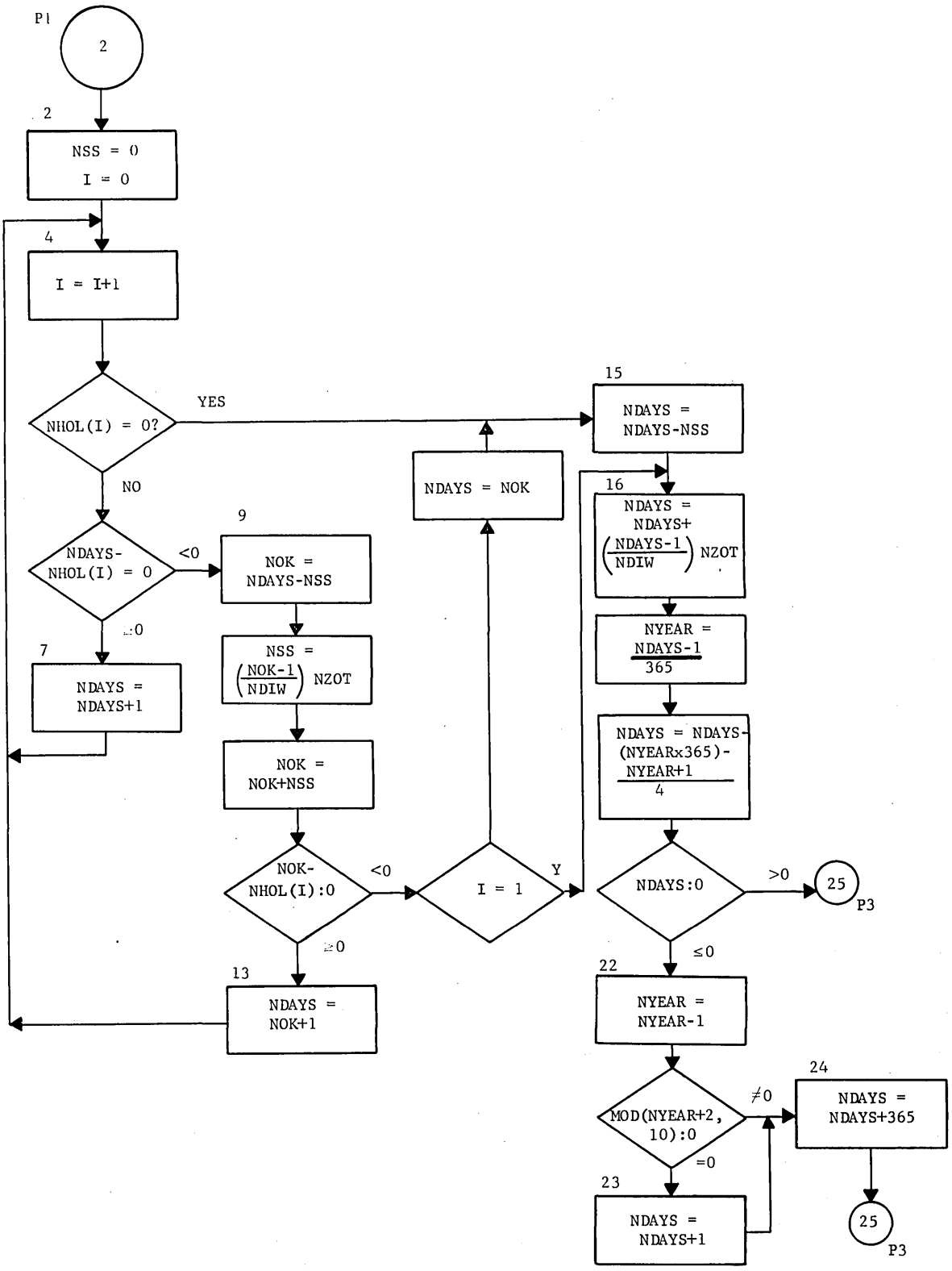


Figure 65. CALOUT Subroutine Flowchart (cont'd) P2

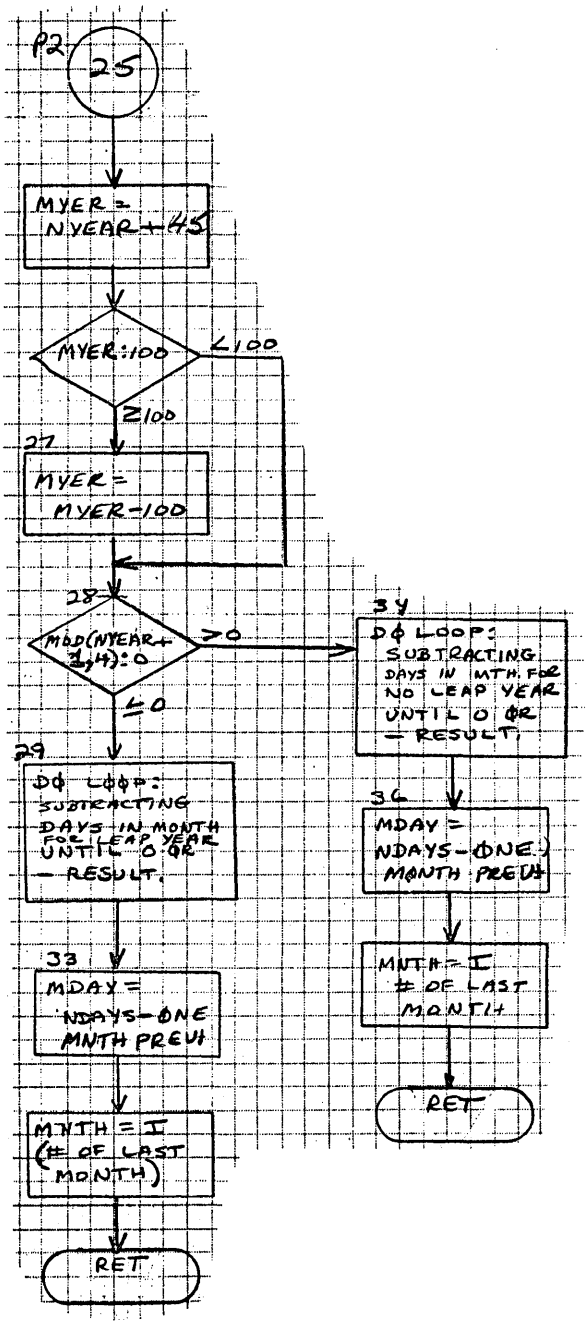


Figure 65. CALOUT Subroutine Flowchart (cont'd) P3

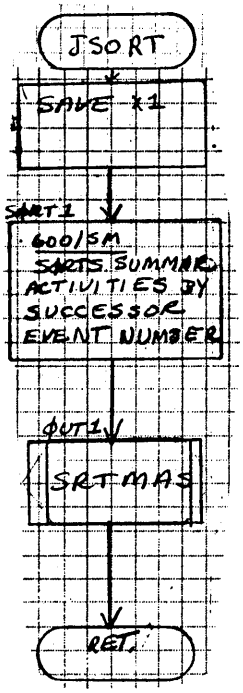


Figure 66. JSORT Subroutine Flowchart

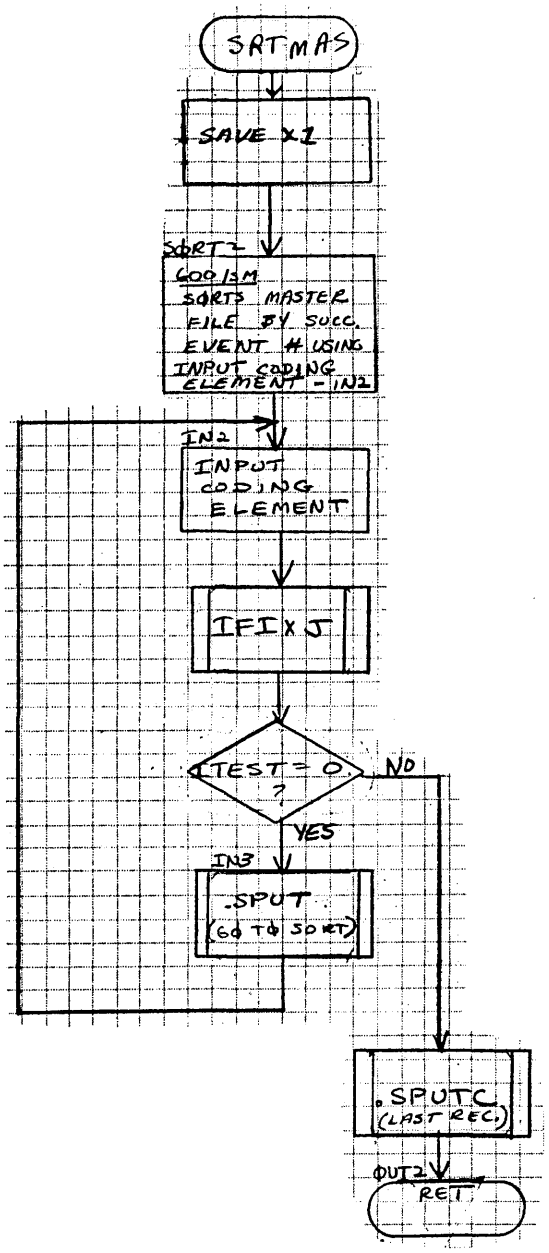


Figure 67 SRTMAS Subroutine Flowchart

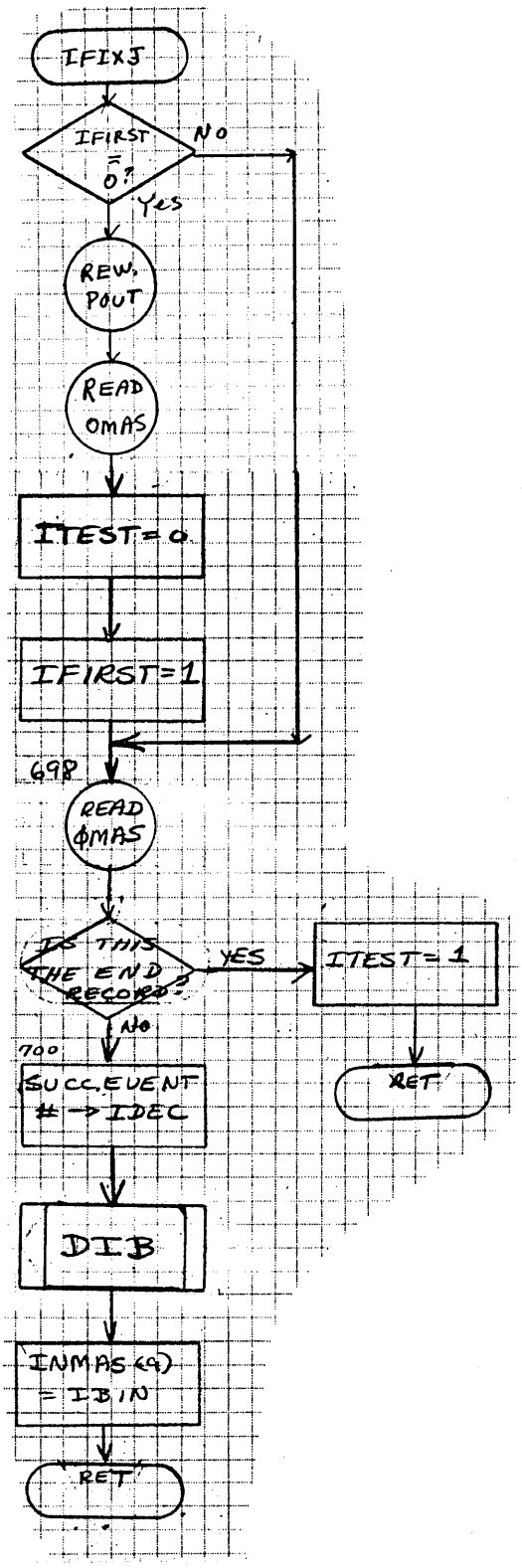


Figure 68. IFIXJ Subroutine Flowchart

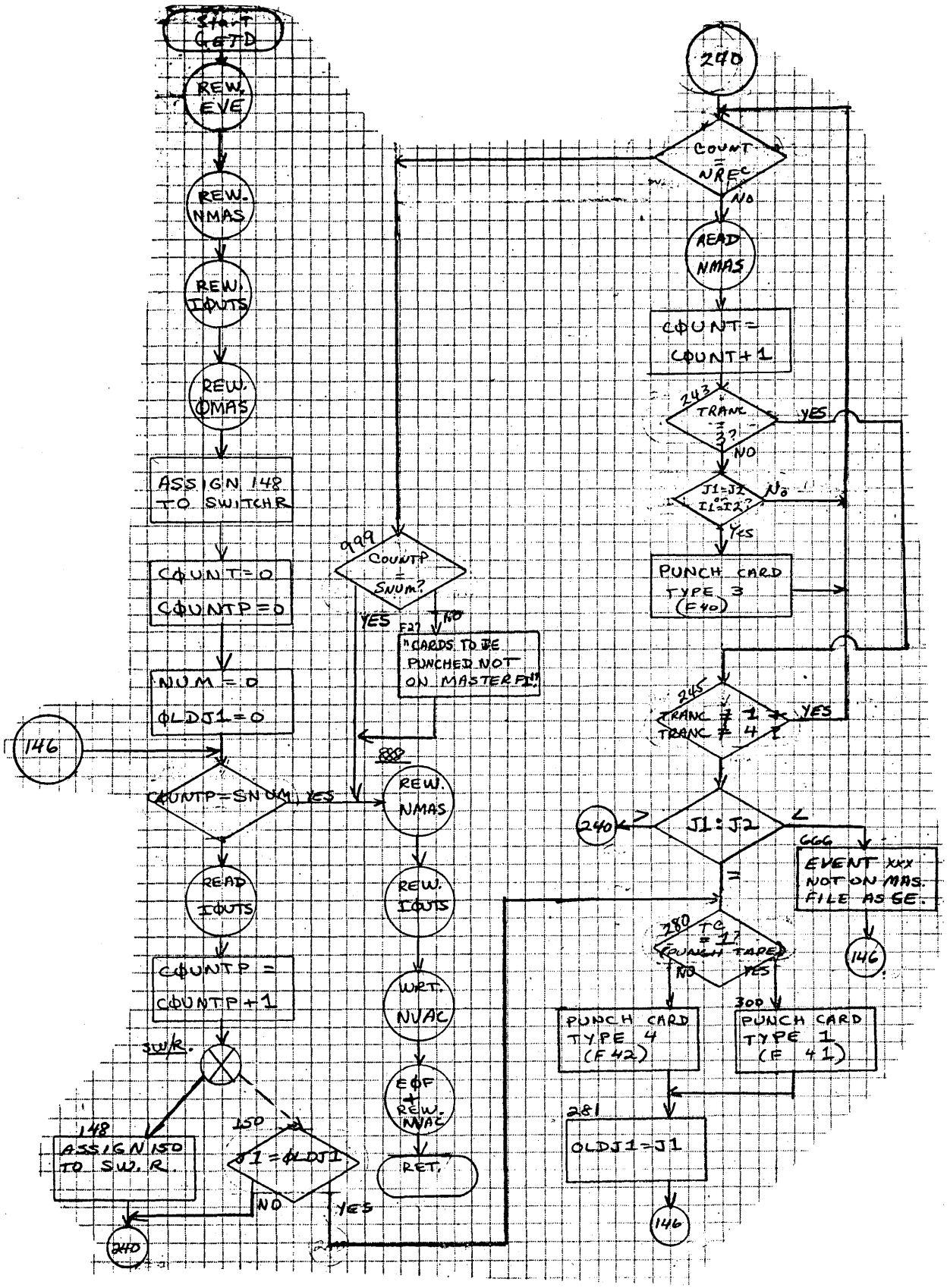


Figure 69. GETD Subroutine Flowchart

6. REPORT GENERATOR SECTION - REPGEN

The Report Generator Section (REPGEN) reads the report parameter cards, detects errors, and builds a table (ITAB) which the output section uses in setting up output reports. Flowcharts for the Report Generator Section are found at the end of this chapter.

The program begins in the subroutine PCARD by examining the first three columns (8 to 10) to determine the type of card; any misspelling beyond the first three columns is ignored. Subroutines used by PCARD are SCRMBL, CHKA, BCDBIN, and EXTRAC. A card must be one of the following types:

Event (EVE)
Activity (ACT)
Suppress (SUP)
Include (INC)
Exclude (EXC)
Title (TIT)
Break (BRE)
Print (PRI)
END

If any card fails to match one of the types specified above, an error message is printed: "Illegal Card Type--XXXXXXXX". (Note: Such a message may be prompted if the card type is keypunched in the wrong columns.) After printing this error message, the program reads another card and attempts to process it.

PARAMETER TABLE NDEX VALUES

When the type of card is acceptable, the program identifies the type of card by a number called the NDEX. These NDEX values are as follows:

Card Type	NDEX
EVE	1
ACT	2
SUP	3
INC	4
EXC	4
TIT	5
BRE	6
PRI	7
END	9

When the NDEX value is 9 (END card), the program's closing procedure tests to see whether a report is in progress and prints "END OF RUN" to conclude processing.

For other NDEX values, the program calls the subroutine SCRMBL to decipher the parameter field (columns 17 to 80) and build the BCD table PARAM. Fields separated by commas or periods on the card are blank filled and stored in the PARAM table in successive words; one word of 6 blanks separates each parameter. The ACODE (Activity CODE) parameter is treated in a special manner; the program expects to find format information following the word and allows three words (18 BCD characters corresponding to 18 columns of the activity code field) within the PARAM table to make allowance for it.

A sample card and its corresponding PARAM entries are shown below:

COL. 8 (Type)	COL. 16 (PARAMETERS)
ACTIVITY	ACODE 554000332211, PRED, SUCC

PARAM(1)	ACODE␣␣	
PARAM(2)	554000	18 DIGITS
PARAM(3)	332211	FOR ACODE
PARAM(4)	000000	FORMAT
PARAM(5)	␣␣␣␣␣␣	BLANKS MARK END OF ACODE PARAMETER
PARAM(6)	PRED␣␣	
PARAM(7)	␣␣␣␣␣␣	BLANKS MARK END OF PRED PARAMETER
PARAM(8)	SUCC␣␣	
PARAM(9)	␣␣␣␣␣␣	BLANKS MARK END OF SUCC PARAMETER
PARAM(10)	␣␣␣␣␣␣	2 WORDS OF BLANKS MARK
PARAM(11)	␣␣␣␣␣␣	END OF TABLE

Parameter fields are checked for the proper length, illegal parameters, and illegal form. Any errors are printed and processing passes to the next card read.

If no errors are detected in the card, the program starts to build the ITAB table. The construction of this table is described in the following section.

THE ITAB TABLE

The ITAB table passes full information to the Output program. It is constructed as described below.

ITAB(1)

The first word of the ITAB table contains the BCD number of the report in progress. The number of the report must appear on each card (columns 1 to 3) pertaining to that report, and any cards not matching the report number are printed as errors in card arrangement. When an activity or event type card is encountered, it signals the beginning of a new report and its report number becomes the new number in ITAB (1).

ITAB(2)

ITAB(2) contains the number of words (in binary) within the initial section of ITAB (parameter information from event, activity, suppress, title and print cards). Include and exclude cards are treated separately, and their parameter information is stored beginning in ITAB (40). If ITAB (40) is 0, there are no include/exclude cards for the report in progress. Break cards are also treated independently and their parameter information is stored beginning in ITAB (800). If ITAB(800) is 0, there are no break cards for the report in progress.

ITAB(3)

ITAB is a binary table with the one exception already noted above--ITAB(1), the report number, appears in BCD as follows: ###666. With ITAB(2) reserved for the table length, the parameter information actually begins in ITAB(3). Information from each card is stored in successive locations; the first entry for each card stipulates the number of parameter items which follow for the card. When the parameter list for a card is exhausted, the next location in ITAB represents the beginning of a new card and its parameters. Each report must begin with an activity or event card; therefore, ITAB(3) necessarily contains basic information about the kind of report. ITAB(3) is the binary representation for a four-digit decimal number of the form N00X, where N represents the number of sort parameters and X represents the NDEX number. N may be any number 0 to 8. If N is 0, there are no parameters given for the sort order of the report in progress, and the sort order for the previous report is assumed. If the first report contains no parameters for the sort order, master file sort order is used (Event order for event reports; successor, slack order for activity reports).

Sort Order Parameters

Parameters for the sort order of event and activity reports are identified through the ISORT table, assigned a number for the sort order, and stored in ITAB as a binary number.

ISORT Table	#
PRED	1
SUCC	2
ACODE	3
TE	4
TL	5
SLACK	6
EVENT	7
LEVEL	8

The following example illustrates the organization of ITAB for an activity report:

```
ITAB (3) = 3002*        3 parameters; NDEX = 2
ITAB (4) =    1            Sort order:  Pred
ITAB (5) =    2                         Succ
ITAB (6) =    4                         TE
ITAB (7)   next card begins here.
```

* Item carried in binary must be converted to decimal to obtain this number.

For an event report, ITAB might appear:

```
ITAB (3) = 3001*                    3 parameters; NDEX = 1
ITAB (4) =    8                       Sort order:  Level
ITAB (5) =    7                                    Event
ITAB (6) =    4                                    TE
ITAB (7) - next card begins here
```

PARAMETER CARD TYPES

There are 7 report parameter card types which are described below.

Suppress Card

The suppress type card is optional. Parameters are examined and must match one of the items of the SUPP data table. Matching items are assigned a number which is stored in ITAB for output interpretation. Parameters failing to correspond with the SUPP table are printed as errors. The SUPP table is as follows:

DESC (last 10 columns of description)	1
CHANGE (change codes)	2
ACODE (activity code, with format)	3
SDATE (scheduled date)	4
MCSLK (most critical slack)	5

The first entry within ITAB for a suppress card is of the form N00X, where N represents the number of suppress parameters and X is the NDEX number 3. N may be any number 1 to 5. If the parameter list is empty (N = 0), an error is detected and printed.

For a suppress card, ITAB might appear:

```
ITAB  (N)              4003*      4 parameters; NDEX 3
ITAB  (N + 1) =           1          Suppress:  DESC
ITAB  (N + 2) =           4                     SDATE
ITAB  (N + 3) =           2                     CHANGE
ITAB  (N + 4) =           5                     MCSLK
ITAB  (N + 5)  Next card begins here.
```

Title Card

The title card is used to change the title for the report. The parameter field is scanned and leading blanks are ignored, since the output section enters the title. The program permits 36 characters (including intervening blanks) of title. The form for the first ITAB entry for a title card is N00X. N, the number of alphanumeric words in the title,is 1 if the title parameter field is all blanks; otherwise, N is 6. X is the NDEX 5. The card

```
TITLE                    COMPUTER INSTALLATION
```

* Item carried in binary must be converted to decimal to obtain this number.

would appear:

```
ITAB (N)     =     6005*      (6 words, NDEX 5)
ITAB (N + 1) =     COMPUT
ITAB (N + 2) =     ER␣␣INS
ITAB (N + 3) =     TALLAT
ITAB (N + 4) =     ION␣␣␣␣
ITAB (N + 5) =     ␣␣␣␣␣␣␣
ITAB (N + 6) =     ␣␣␣␣␣␣␣
ITAB (N + 7) =     Next card begins here.
```

The card:

```
TITLE    ␣␣␣␣ .....␣
```

with blank parameter field signals the program to print no title and appears in ITAB:

```
ITAB (N)     =     1005*      1 word; NDEX 5
ITAB (N + 1) =     ␣␣␣␣␣␣␣
ITAB (N + 2) =     Next card begins here
```

Print Card

The print card must be the last card of each report sequence. This card signals the end of a report. If the program fails to find a print card, an error is indicated. If the print card is not the last card of the sequence, cards following the print card are considered out of order and printed as an error condition. The form of the first word of the ITAB entry for a print card is N00X, where N represents the number of parameters and X is the NDEX 7. N may be either 1, 2, or 3. If the parameter list of the print card does not contain a number, the program assumes that one copy of the report is requested; a binary one is stored in the second word of the ITAB print entry. If a binary number of 8 digits or less appears in the parameter list, it is considered the number of copies requested by the user; this number is stored in the second word of the ITAB entry. Two other acceptable print parameters form the PRINTP table, and items matching the table entry are assigned a corresponding number as shown below:

PRINTP	#
COMPL	1
SHORT	2

Examples of print cards and corresponding ITAB entries are shown below:

A.

```
PRINT              ␣␣␣␣␣
ITAB (N)     =     1007*      1 Parameter; NDEX 7
ITAB (N + 1) =     1          1 copy
END of initial section of ITAB
```

* Item carried in binary must be converted to decimal to obtain this number.

B.

```
PRINT                    5, COMPL

ITAB (N)      =          2007 *     2 Parameters; NDEX 7
ITAB (N + 1)  =             5          5 copies
ITAB (N + 2)  =             1          COMPL
END of initial section of ITAB
```

C.

```
PRINT                    SHORT, COMPL

ITAB (N)      =          3007 *     3 Parameters; NDEX 7
ITAB (N + 1)  =             1          1 copy
ITAB (N + 2)  =             2          SHORT
ITAB (N + 3)  =             1          COMPL
END of initial section of ITAB
```

Include/Exclude Card

Include and exclude cards are treated together and stored in memory together, beginning at ITAB(40). The program accepts up to 25 include/exclude (I/E) cards. The form of the first word of ITAB for each new card is NWIX, where X is a number 1 to 6 which is formed by scanning the parameter field for a comparison parameter enclosed by periods (or commas). The comparison parameter, Y, is equated to corresponding numbers through the COMPAR table below:

COMPAR	Y
.EQ.	1
.GT.	2
.LT.	3

An index value held in location IE is added to this number. For exclude cards, IE = 0. For include cards, IE = 3. Thus, X takes on the following values and interpretations:

X		COMPAR		IE	MEANING
1	=	1	+	0	EXCLUDE .EQ.
2	=	2	+	0	EXCLUDE .GT.
3	=	3	+	0	EXCLUDE .LT.
4	=	1	+	3	INCLUDE .EQ.
5	=	2	+	3	INCLUDE .GT.
6	=	3	+	3	INCLUDE .LT.

I is a number 3 to 7 representing the item of the report which is the one for inclusion or exclusion. Each include/exclude card names its base, and the program uses a selected portion of the ISORT table to determine its validity. A portion of the ISORT table and corresponding numbers are shown below:

ISORT	#
ACODE	3
TE	4

* Item carried in binary must be converted to decimal to obtain this number.

ISORT	#
TL	5
SLACK	6
EVENT	7

W is a number 1 to 3 representing the number of words in memory required to store each numeric or alphabetic value to be compared with the report item I above. W is 1 (1 binary word) for all comparisons except ACODE. ACODE is kept in BCD and may require 1, 2, or 3 words.

N is a number 1 to 27 representing the number of comparative values listed on the card. N must be 1 for comparisons .GT. or .LT.. A list of comparative values, permissible for .EQ. comparisons, is limited by card size. Twenty-seven is a maximum for values of one digit; that is, 27 values and 26 commas within the parameter field.

Values which will not fit on a single card may be put on a second card, which is identical with the first card, except that its list of values in ascending order is a continuation of those on the earlier card.

A sample exclude card and ITAB entry are shown below:

```
EXCLUDE SLACK .GT. 10

ITAB(40)                1162*  ---> EXCLUDE .GT.
                               ---> SLACK
                               ---> 1 WD. FOR EACH COMPARE
                               ---> 1 COMPARE
ITAB(41)                  12       OCTAL REPRESENTATION OF
                                       DECIMAL #10.
ITAB(42)   Next I/E card begins here.  If none, set to zero to signal the
           last entry.
```

When ACODE items are to be compared on the I/E cards, the ACODE format must be defined. Trailing spaces are converted to 0 for comparison purposes.

For example:

```
INCLUDE ACODE 00111011.EQ.ABCOO,IJK22

ITAB (N)          2134  ---> INCLUDE .EQ.
                        ---> ACODE (signals next 3 words for format)
                        ---> 1 WD. for each compare
                        ---> 2 comparisons
ITAB (N + 1)  =   001110
ITAB (N + 2)  =   110000        ACODE format
ITAB (N + 3)  =   000000
*ITAB (N + 4)  =   ABCOO␣       Comparison 1
ITAB (N + 5)  =   IKJ22␣        Comparison 2
ITAB (N + 6)  =   Next I/E card begins here.  If none, zero.
```

* Note that ACODE comparisons are left in BCD and blank filled.

The ACODE format is checked for each card to assure that the comparative values are of the proper length.

Break Card

Break cards are treated separately and stored in core, beginning in ITAB(820). The program accepts up to 50 break cards; however, if the option to break on any change in the major sort field is used, the program accepts only 1 card. Like the I/E cards, the break cards depend upon the comparison of values to determine when to break the report and begin a new page. There are several notable differences, however:

1. Break card values all refer to the major sort key rather than to another item defined on the card itself.

2. Break cards permit only .EQ. and .GT. comparisons from the COMPAR table.

3. Comparative values must agree in length with the major sort key, warranting particular care for the ACODE format.

4. Only one comparative value should be used per break card, and the cards must be in ascending order in the deck.

5. The parameter field for break cards may be absolutely blank, signaling a break on any change in the major sort key.

6. The break parameter field may contain comparative information plus a title--indicating a subtitle to be used at the break indicated on the report.

7. The break card may contain only a subtitle to be printed for any break in the major sort key.

8. Break cards using comparison values must all be .EQ. or all .GT., to maintain the logic of the continuing major sort key comparisons.

Two program pecularities deserve mention:

(1) In preparing break cards, the programmer should make sure that the title information does not look like a comparison. The title ".AB. JOINT DEVELOPMENT" appears as an illegal comparison to the program.

(2) Titles used on cards with comparison values must contain at least one alphabetic before a comma or period appears, or at least one alphabetic within the first 12 digits. The program checks for a list of values which are illegal on break cards. The appearance of one alphabetic character in one of the positions noted assures the program that this is a title. ACODE values placed in an illegal list elude this check if they contain alphabetics, and appear as a title.

Break card storage is similar to that of previous cards. The first entry for each card is of the form TNSX, where X is 0, 1, or 2. 1 and 2 represent the comparative parameters from the COMPAR table listed below. 0 represents the absence of a comparative value and signals the program to break on any change in major sort field.

COMPAR	#
.EQ.	1
.GT.	2

S is the major sort key taken from the initial activity or event card or from the previous sort order if no sort order is named for this report.

N is the number of words required to store the comparative value. It is 1 for all values except those used when the major sort is ACODE. In this case, N may be 1, 2, or 3, depending on the length of the major sort key.

T is the number of title words: 6 for title following data in ITAB; 0 for no title.

The following examples illustrate possible break cards and their ITAB entry:

1. BREAK ␣␣␣␣␣␣␣␣

*ITAB(820) = 0080 → Break on any change
→ Major sort by level
→ No comparisons
→ No title

ITAB(821) Zero since the program permits only one Break card when the break is on any change in major sort field.

2. BREAK BEARING PRODUCTION

ITAB(820) = 6080 → Break on any change
→ Major sort by level
→ No comparisons
→ 6 words of title

ITAB(821) = BEARIN
ITAB(822) = G␣PROD
ITAB(823) = UCTION
ITAB(824) = ␣␣␣␣␣␣
ITAB(825) = ␣␣␣␣␣␣
ITAB(826) = ␣␣␣␣␣␣
} Title

ITAB(827) Zero to terminate the break table.

3. ACTIVITY ACODE 001111111000222000, PRED
BREAK .EQ. AAABBBB, ELECTRONIC ASSEMBLY

*ITAB(820) 6231 → Break on .EQ.
→ Major sort ACODE
→ Words to hold comparative value
→ Title words

ITAB(821) = AAABBB
ITAB(822) = B␣␣␣␣␣
} Comparative Value

ITAB(823) = ELECTR
ITAB(824) = ONIC A
ITAB(825) = SSEMBL
ITAB(826) = Y␣␣␣␣␣
ITAB(827) = ␣␣␣␣␣␣
ITAB(828) = ␣␣␣␣␣␣
} Title

ITAB(829) Next break card begins here; this is zero, if there are no more break cards.

* Note that the ACODE format does not appear in the break table here. The program checks location ACODE to decide whether values on the break card match the 1's, indicating the major sort field of the activity code.

There may be 25 I/E cards and 50 break cards when the comparative feature is used, or only 1 break card when the break is made on any change in the major sort field. There may be only 1 activity or event card, 1 suppress card, 1 title card, and 1 print card for each report. Additional cards for any of these limits cause an error comment.

Any error comment within a report card printout prevents the printing of the report. When cards for a report are in good order, processing passes to the SETOUT section.

After completion of the report by the SETOUT section, control is returned to the REPGEN section to process the next group of report parameter cards.

END Card

This continues until an END card terminates the output reports.

REPGEN INPUT AND OUTPUT FORMATS

Input to the Report Generator Section is the report parameter cards which are formatted as follows:

Columns 1-3--Report number

Columns 8-15--Card type (See Figure 70.)

Columns 17-80--Parameters (See Figure 70.)

Output is the ITAB table in core memory.

Card Type (col. 8-15)	Acceptable Parameters (col. 17-80)	Interpretation
EVENT	EVENT, LEVEL, SLACK, TE, TL	Report sequence--any or all in order of most major to most minor.
ACTIVITY	SUCC, PRED, SLACK, TE, TL, ACODE $Y_1, Y_2 \ldots\ldots Y_n$	Same as above; $Y_n = 0 - 9$ $n < 19$.
SUPPRESS	DESC, CHANGE, SDATE, ACODE Y_1, $Y_2 \ldots . Y_n$, MCSLK	Fields to be omitted in each line. $Y_n = 0$ or 1 $n < 19$.
INCLUDE/ EXCLUDE	One card for each line below: EVENT .LT. or .GT. W or .EQ. (List W) SLACK .LT. or .GT. A or .EQ. (List A) TE. LT. or .GT. B or .EQ. (List B) TL .LT. or .GT. B or .EQ. (List B) ACODE $Y_1, Y_2 \ldots\ldots Y_n$ .LT. or .GT. Z or .EQ. (List Z)	Include or Exclude whole lines if condition exists. A = signed decimal number (1 to 7 digits) B = unsigned decimal number (1 to 8 digits) W = 1 to 8 digits Z = alphanumeric characters List = A, A, A, A,etc., in ascending order. $n < 19$. ACODE and EVENT may not occur on same report.
TITLE	36 alphanumeric characters	Report title replaces last used.
BREAK	~~bb~~ ——————→	Break on any change in the major sort field and use no subtitle.
	36 alphanumeric characters with one card for each value listed below	Break on any change in major sort field and use subtitle.
	.EQ. V, 36 characters (more than one of these cards is reasonable)	Break when major field equals V, and use 36 characters as subtitle.
	.GT. V, 36 characters (more than one of these cards is reasonable)	Break when major field is greater than V, and use 36 characters as subtitle.
	Note: .EQ. and .GT. comparisons may not be mixed within a given report.	
PRINT	$D_1 D_2$, SHORT, COMPL	$D_1 D_2$ = Number of times print is desired. SHORT if fast activity option is desired. COMPL if completed events and activities to be omitted.
END		End of all report parameters.

Figure 70. Report Parameter Card Layout

REPORT GENERATOR SECTION FLOWCHARTS

The following pages contain the flowcharts for the Report Generator Section listed in the following order:

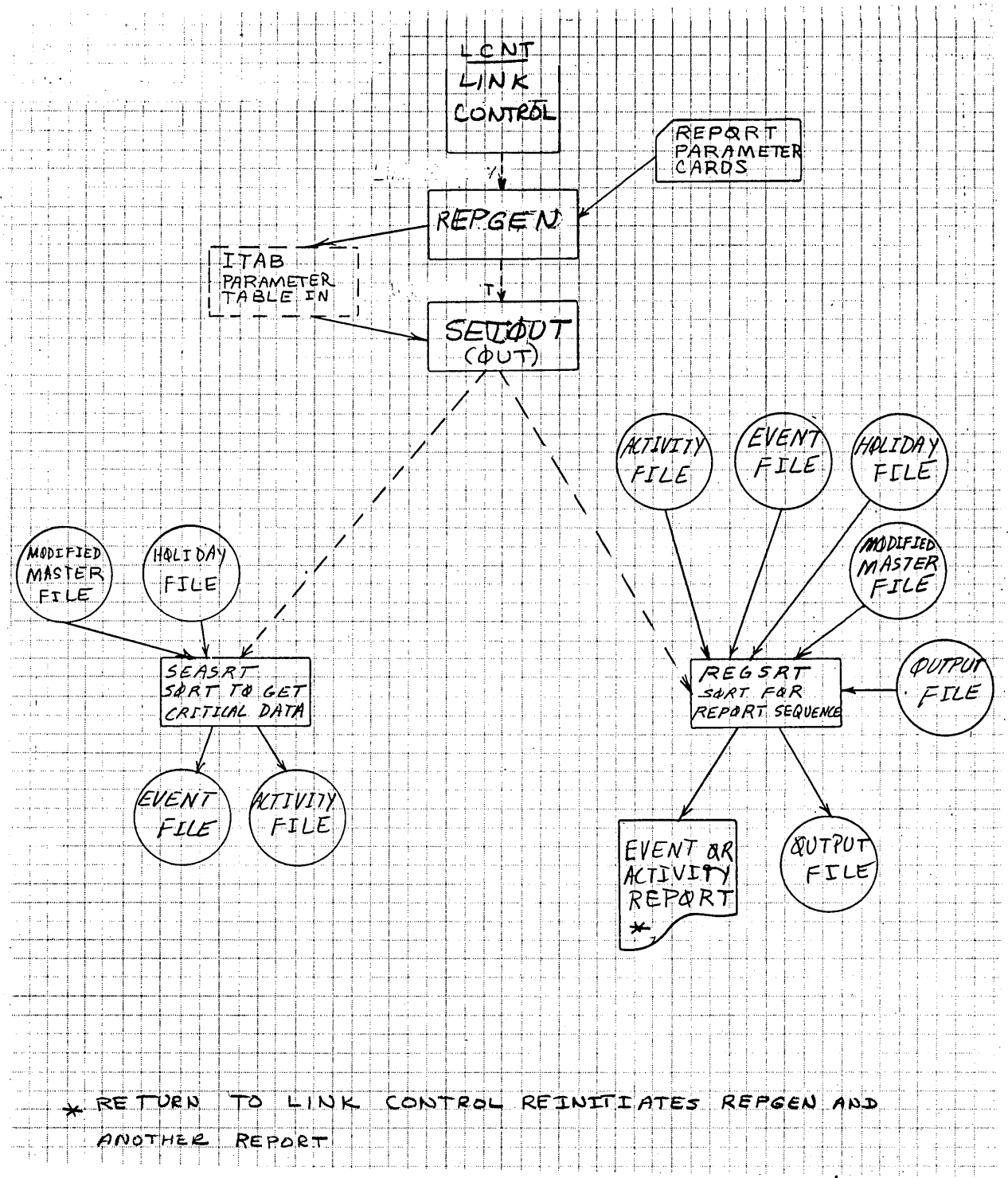


Figure 71. REPGEN and SETOUT Linkage and File Relationships

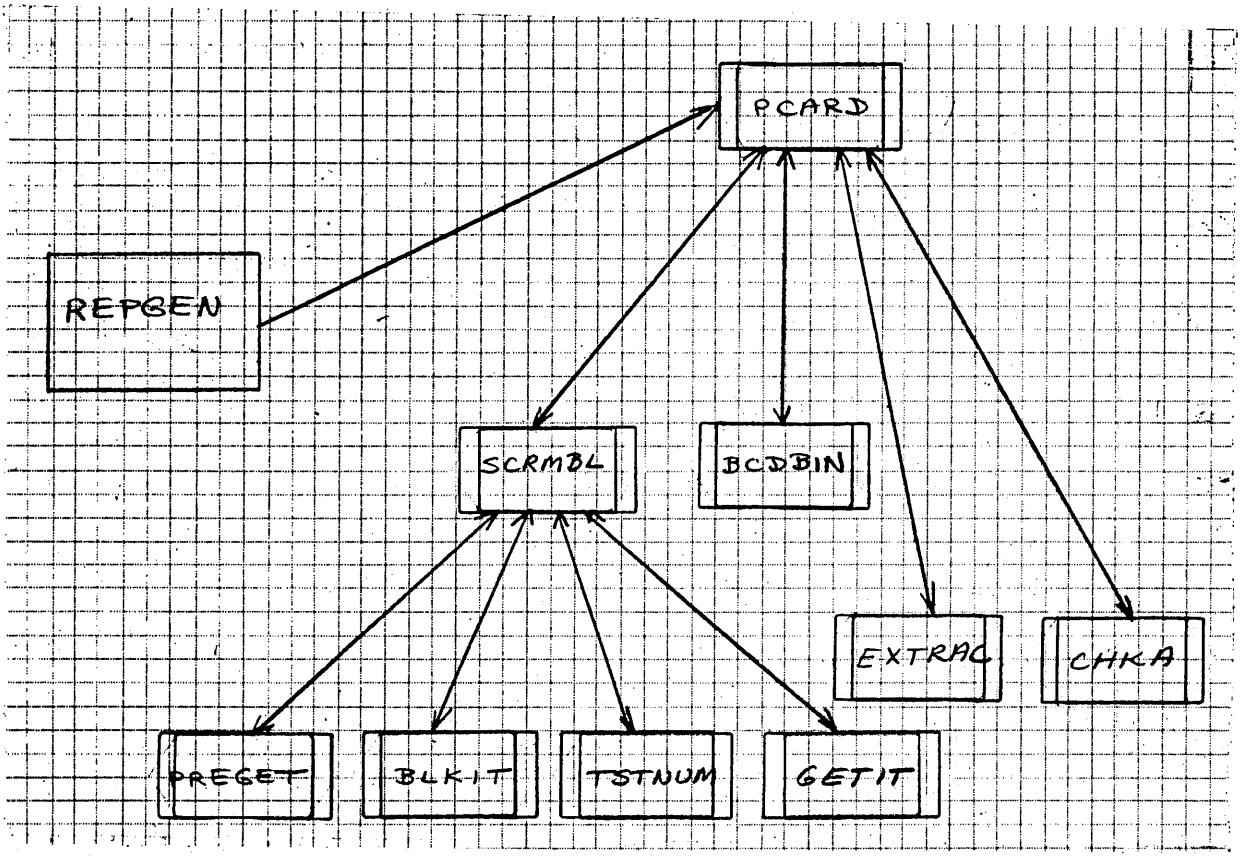


Figure 72. REPGEN Section Flowchart

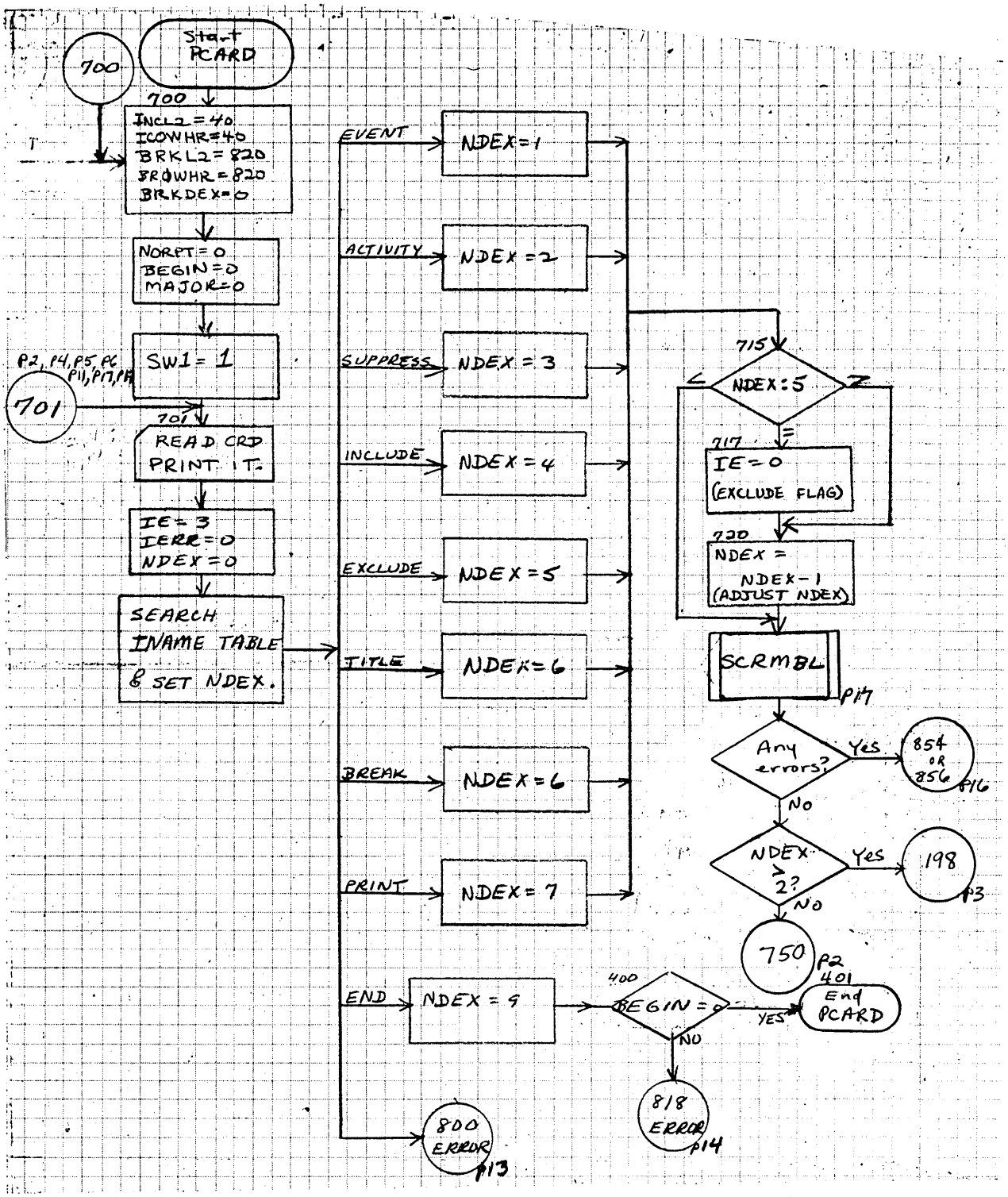


Figure 73. PCARD Subroutine Flowchart P1

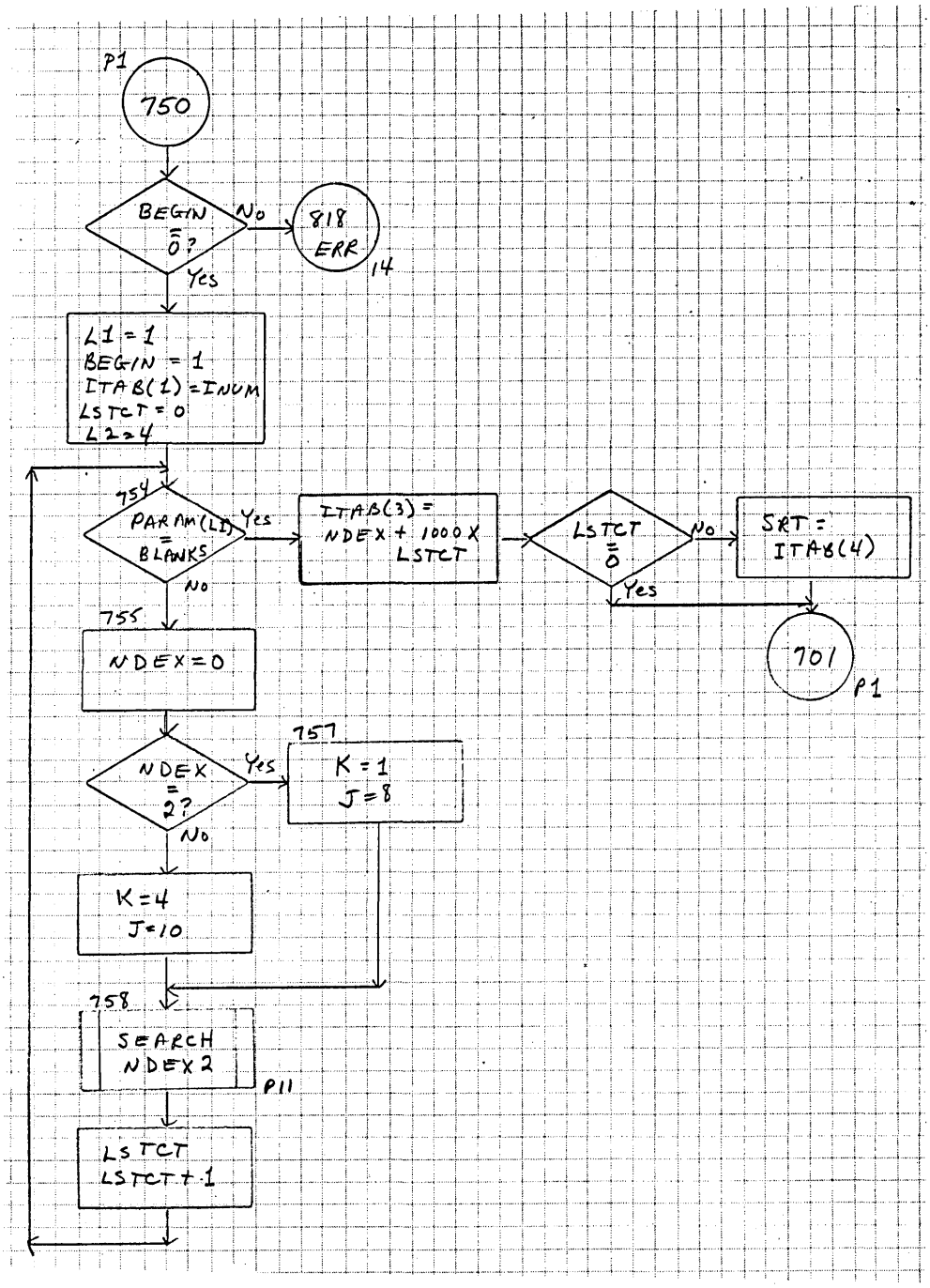


Figure 73. PCARD Subroutine Flowchart (cont'd) P2

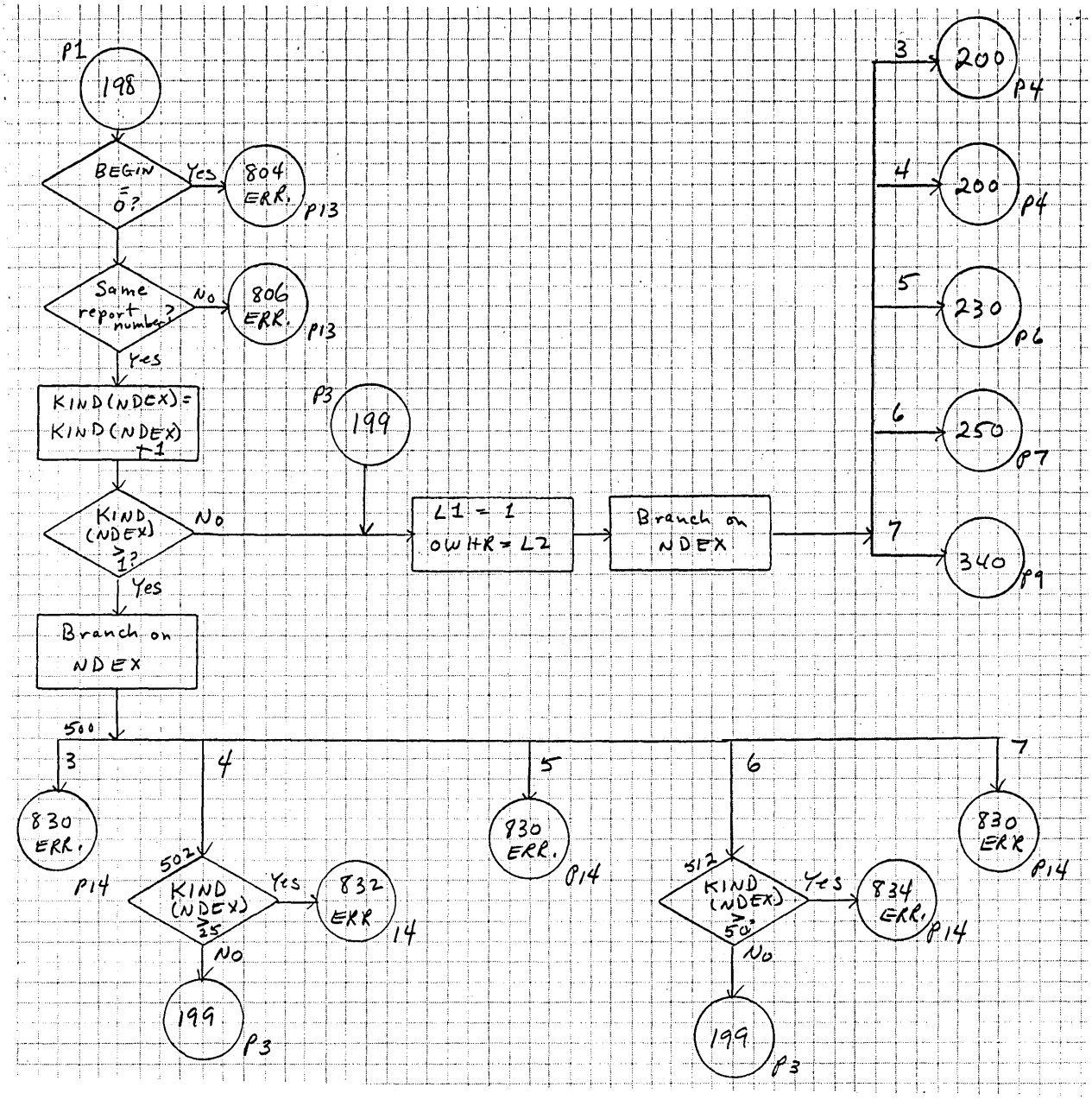


Figure 73. PCARD Subroutine Flowchart (cont'd) P3

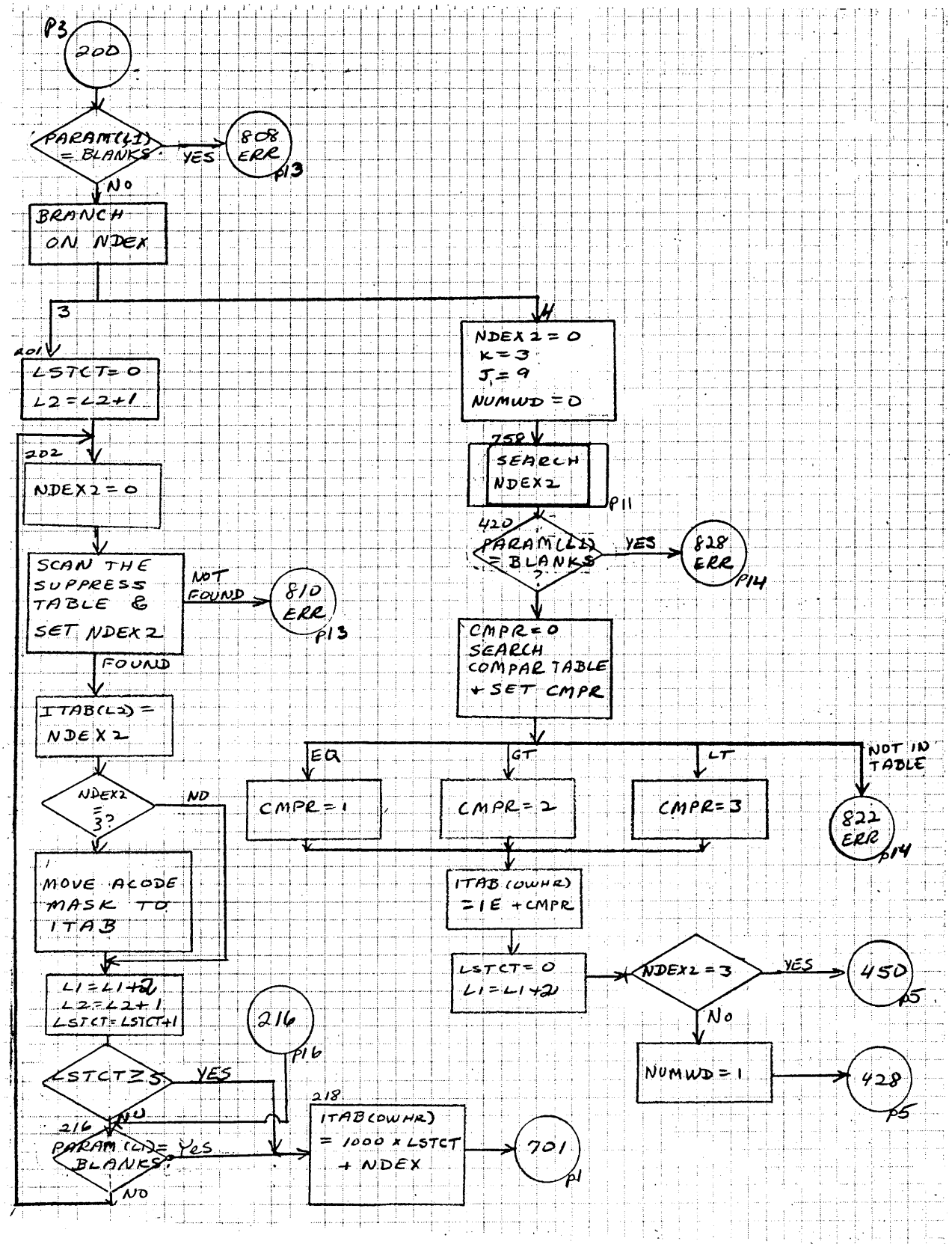


Figure 73. PCARD Subroutine Flowchart (cont'd) P4

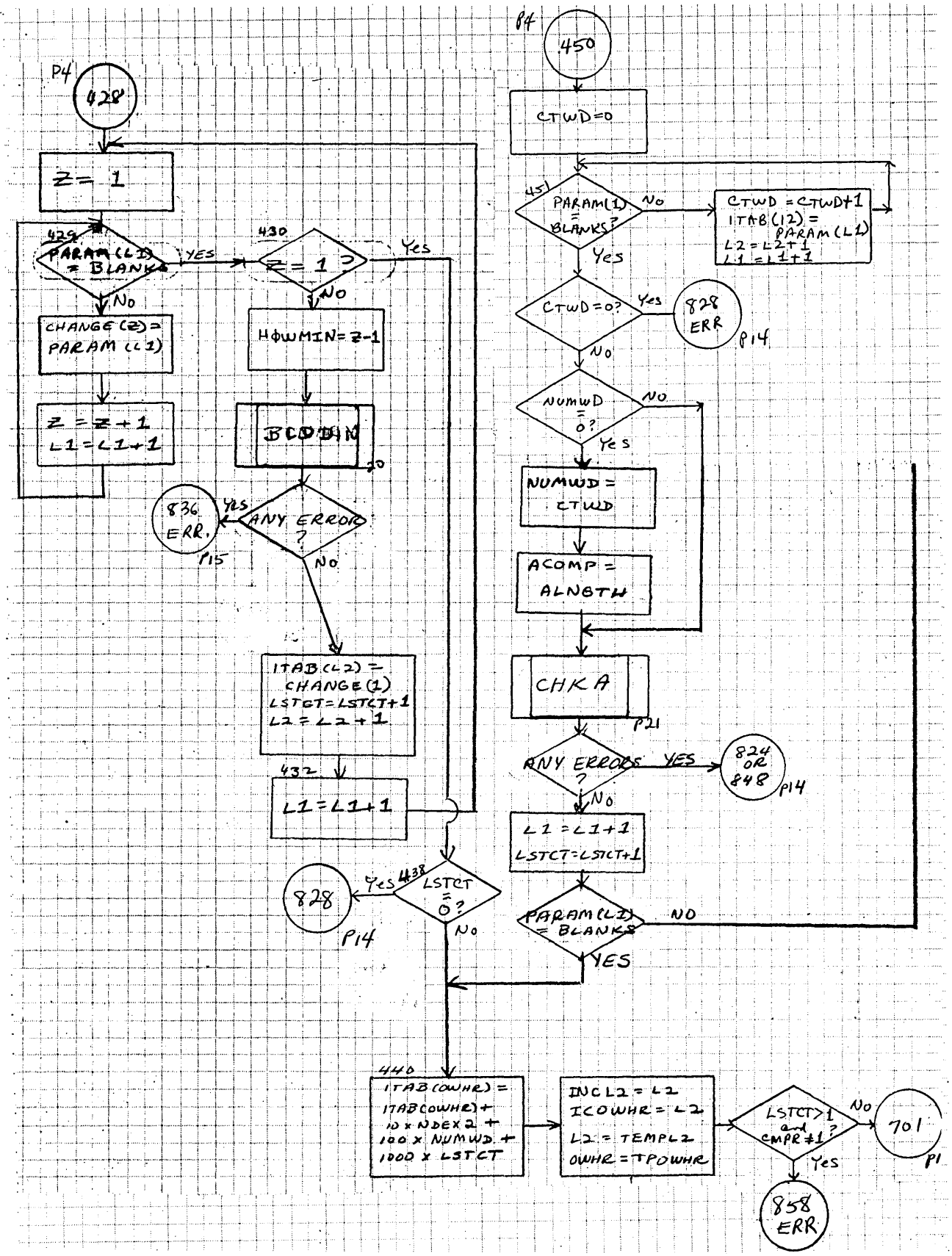


Figure 73. PCARD Subroutine Flowchart (cont'd) P5

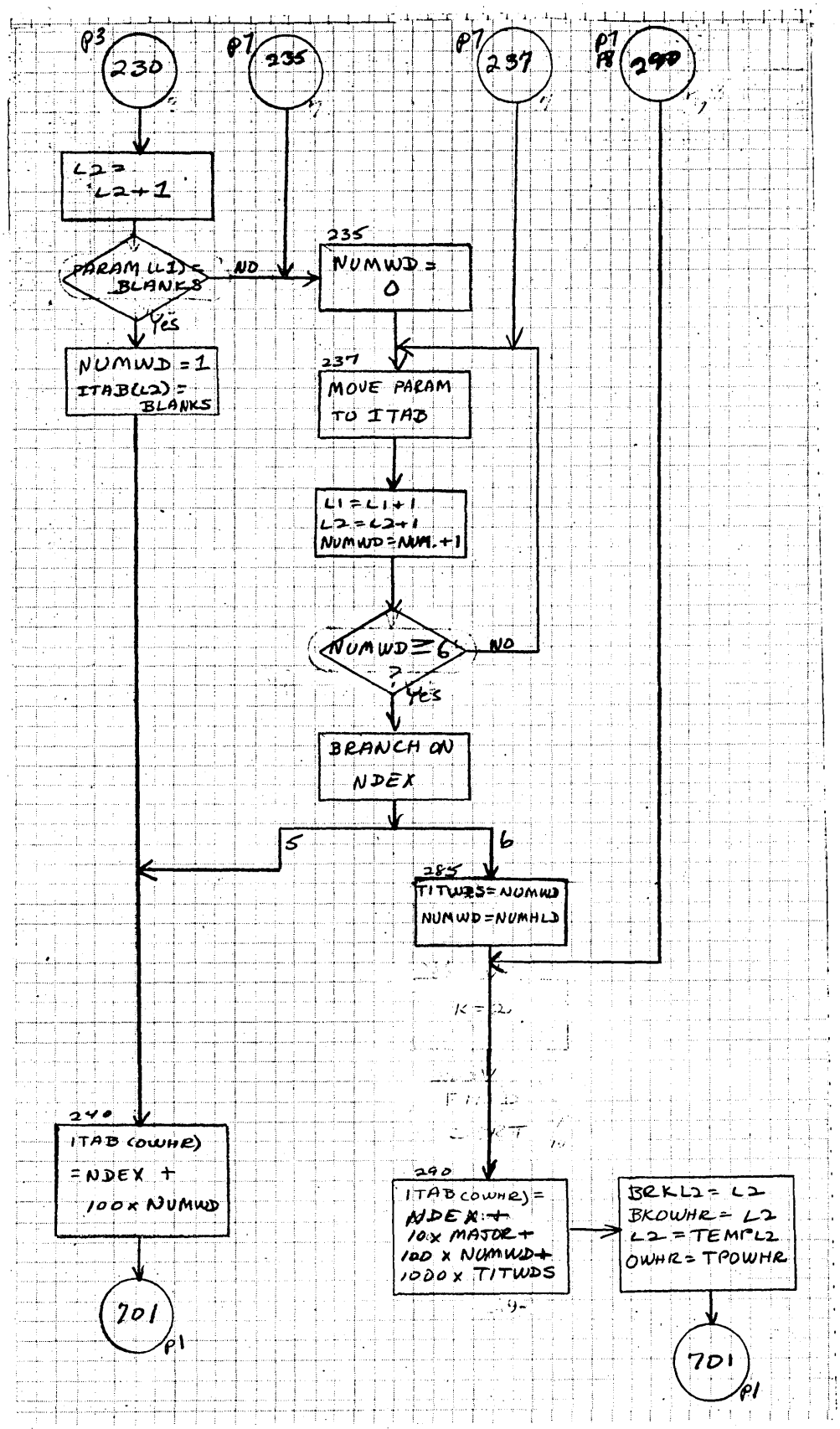


Figure 73. PCARD Subroutine Flowchart (cont'd) P6

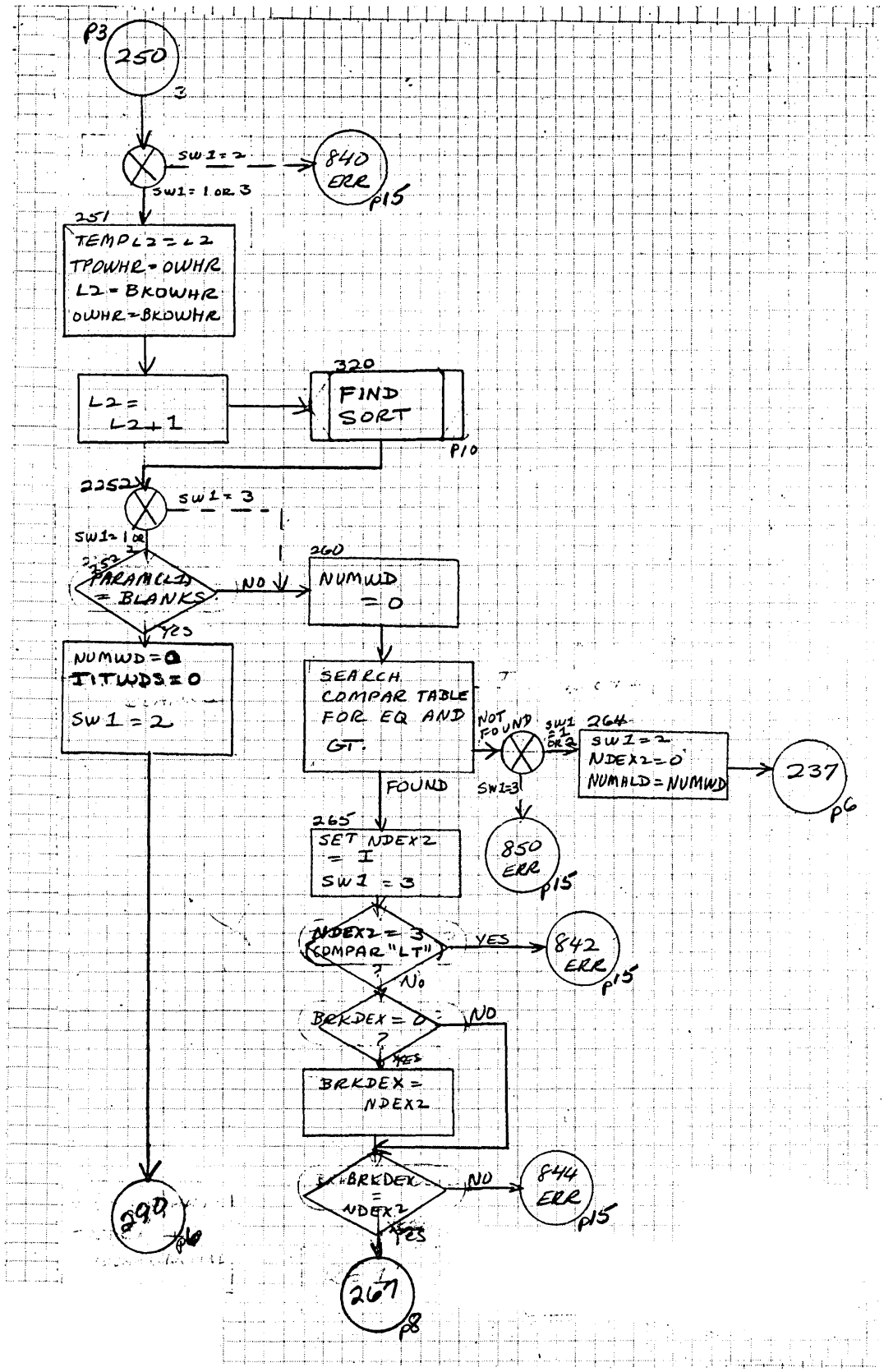


Figure 73. PCARD Subroutine Flowchart (cont'd) P7

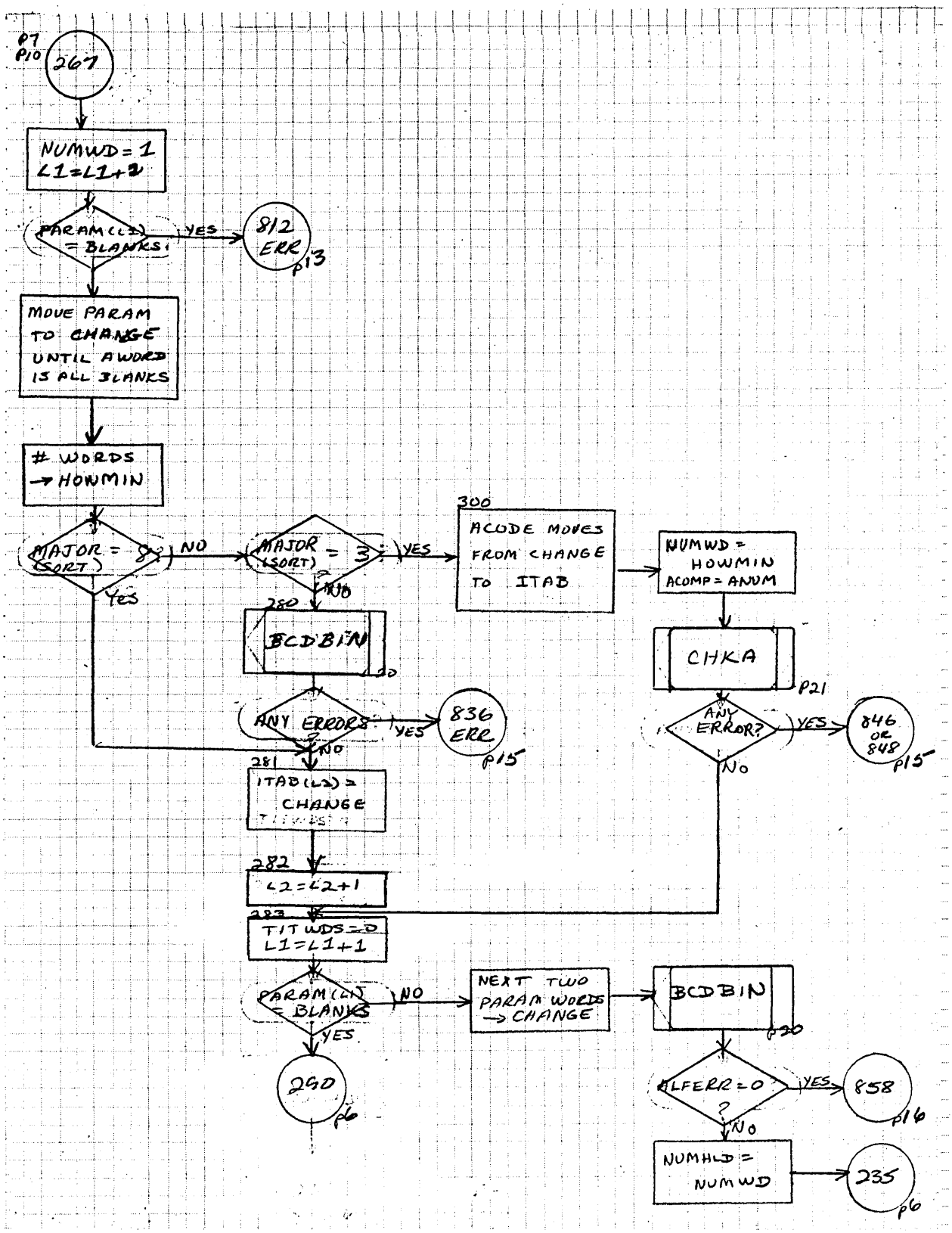


Figure 73. PCARD Subroutine Flowchart (cont'd) P8

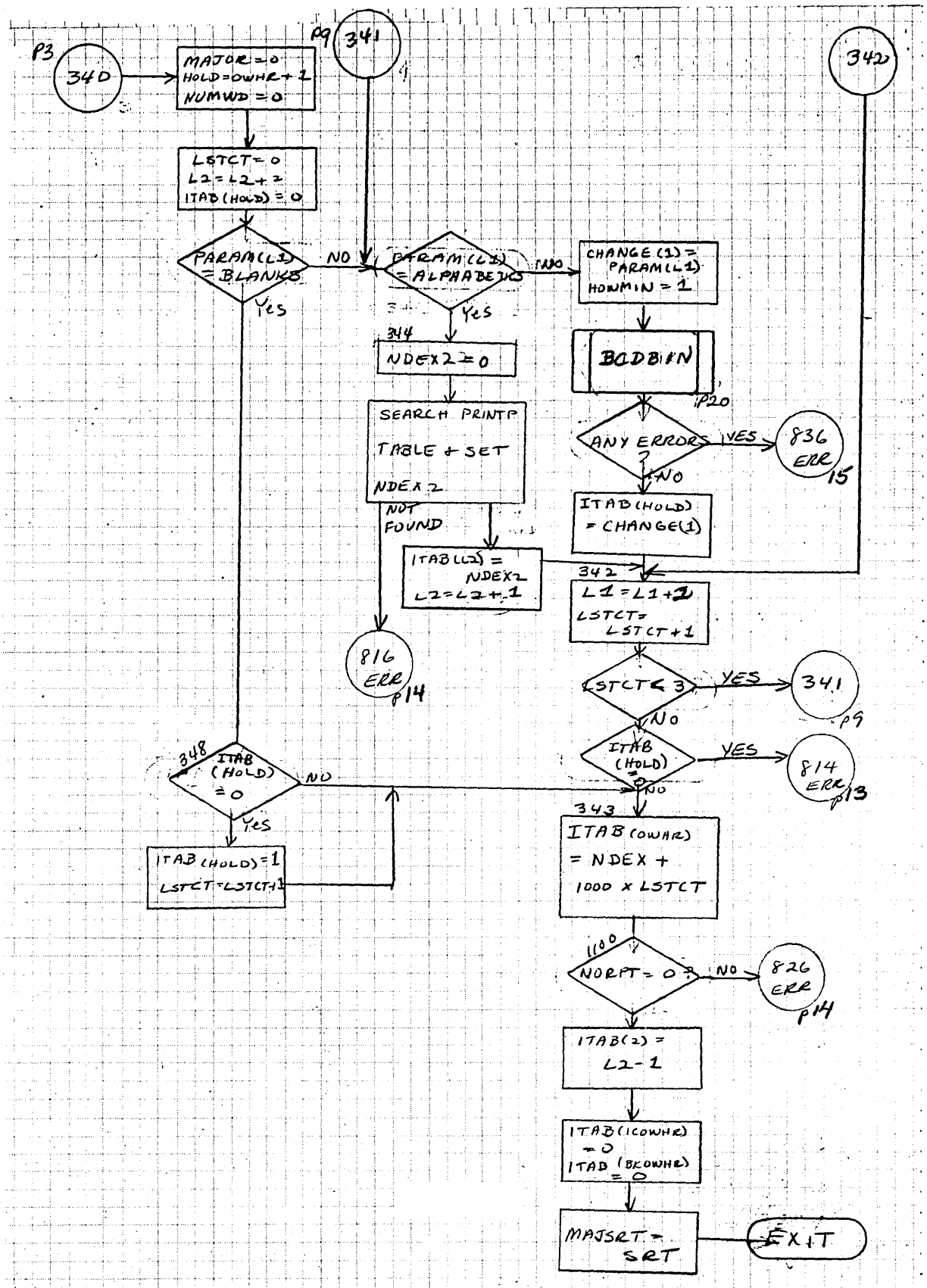


Figure 73. PCARD Subroutine Flowchart (cont'd) P9

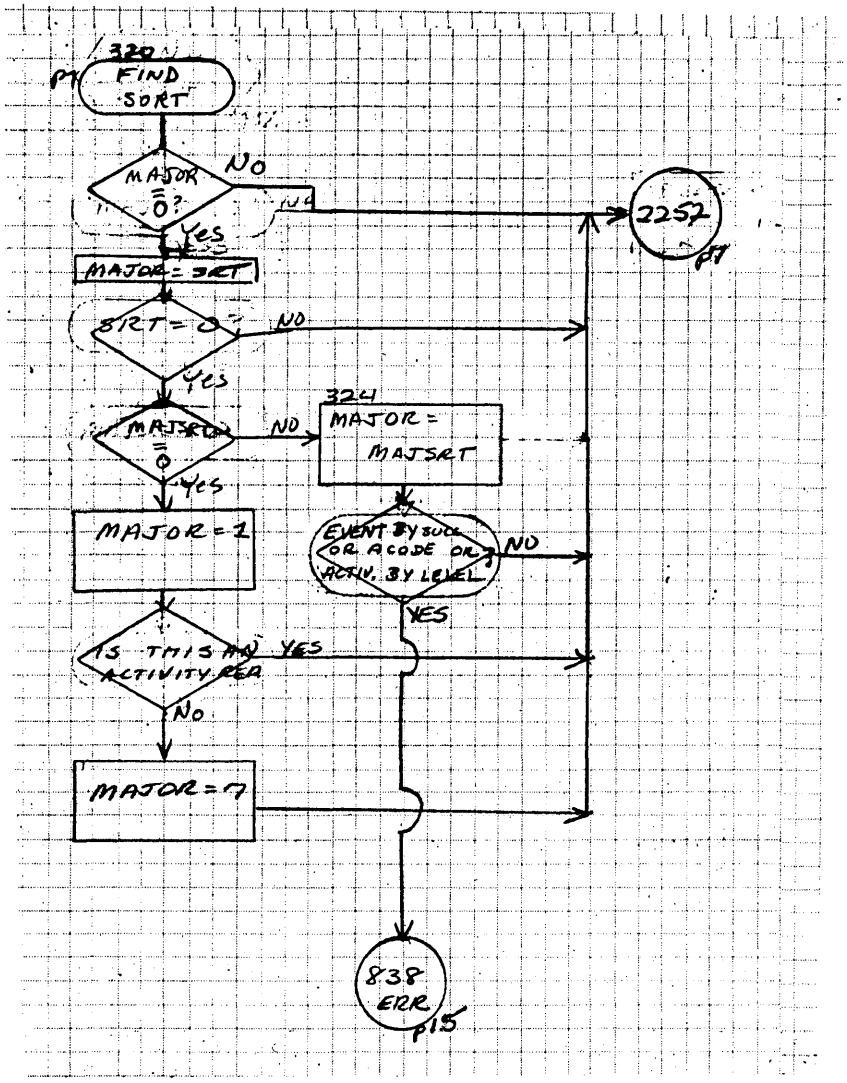


Figure 73. PCARD Subroutine Flowchart (cont'd) P10

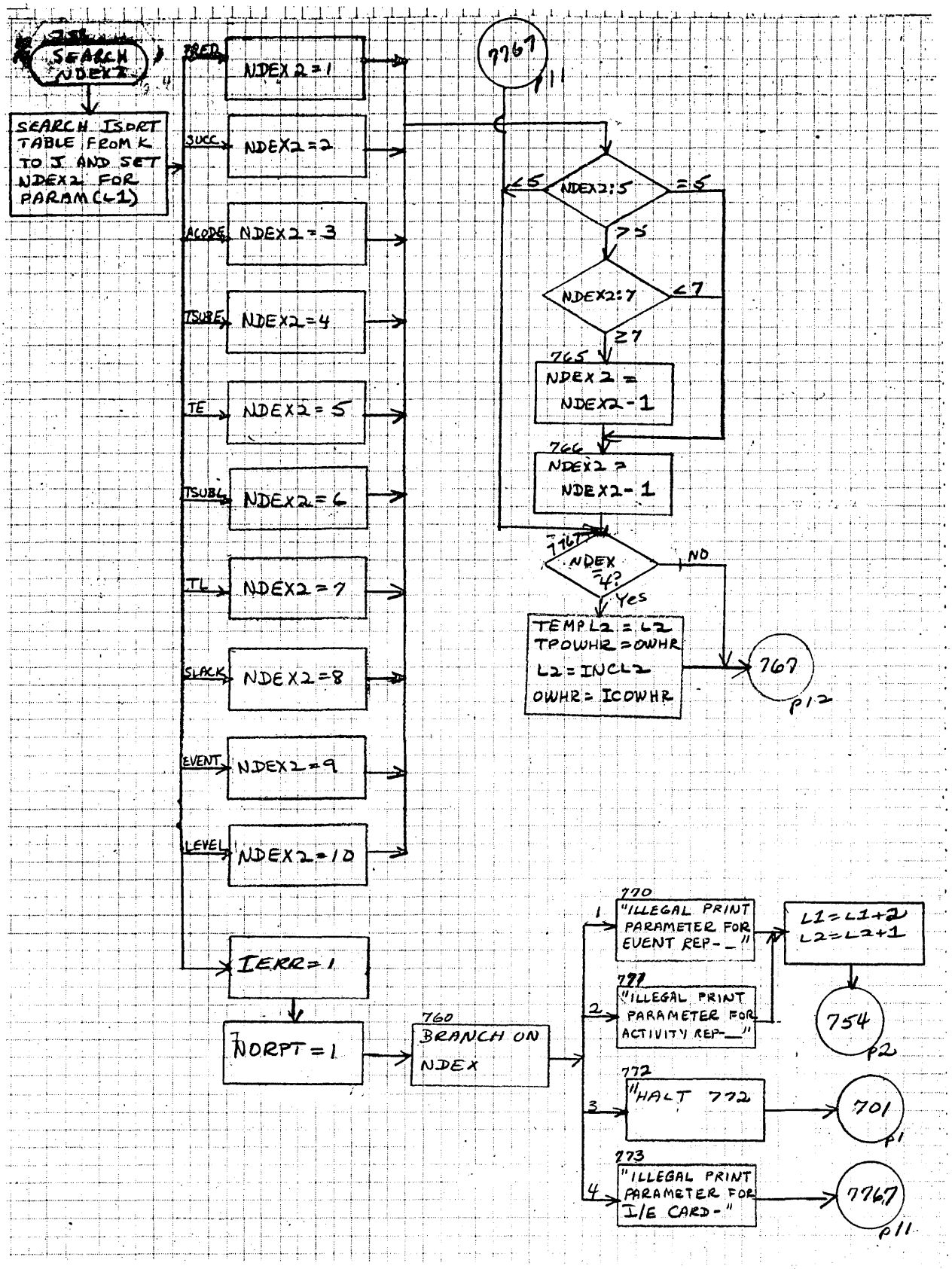


Figure 73. PCARD Subroutine Flowchart (cont'd) P11

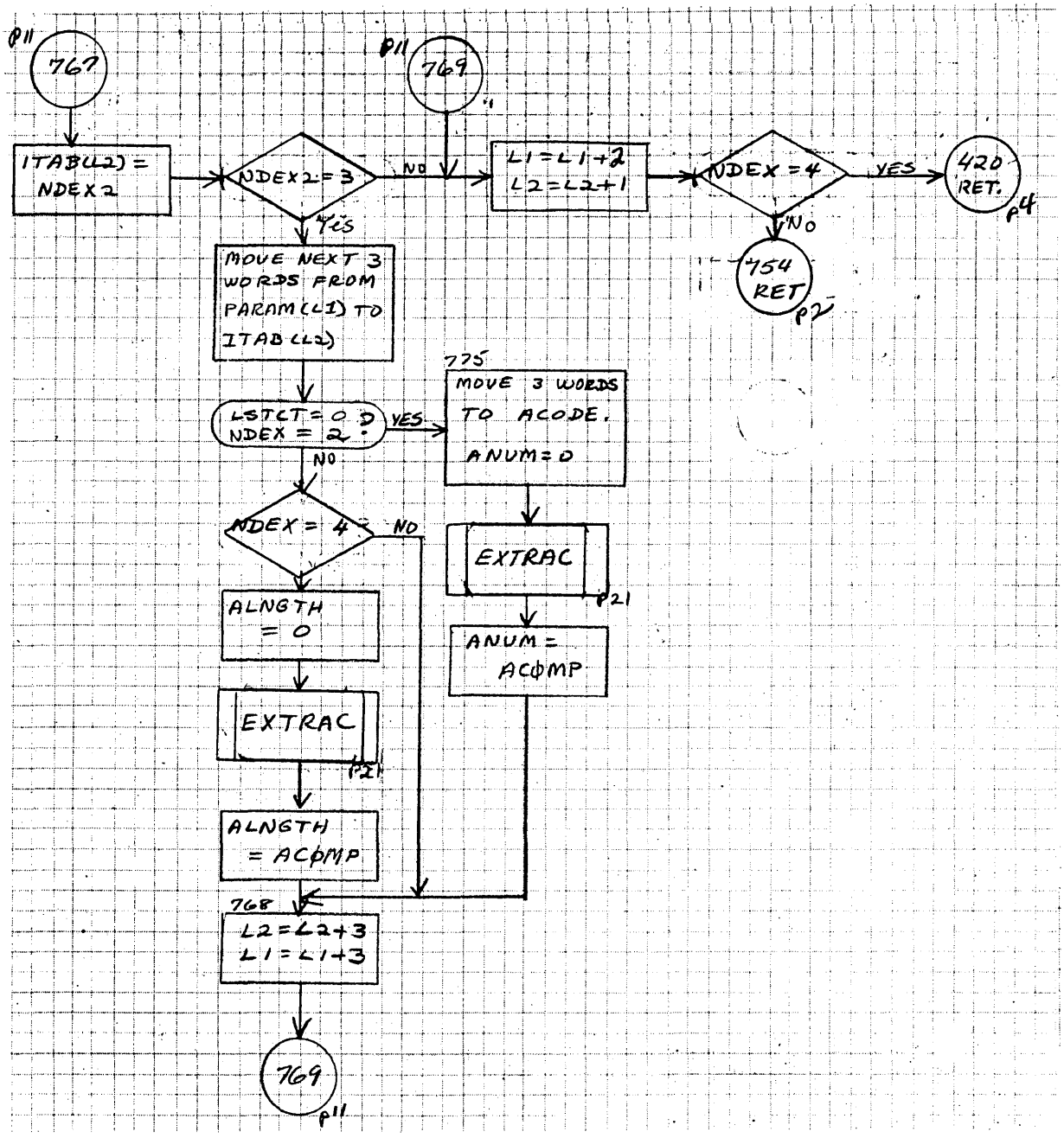


Figure 73. PCARD Subroutine Flowchart (cont'd) P12

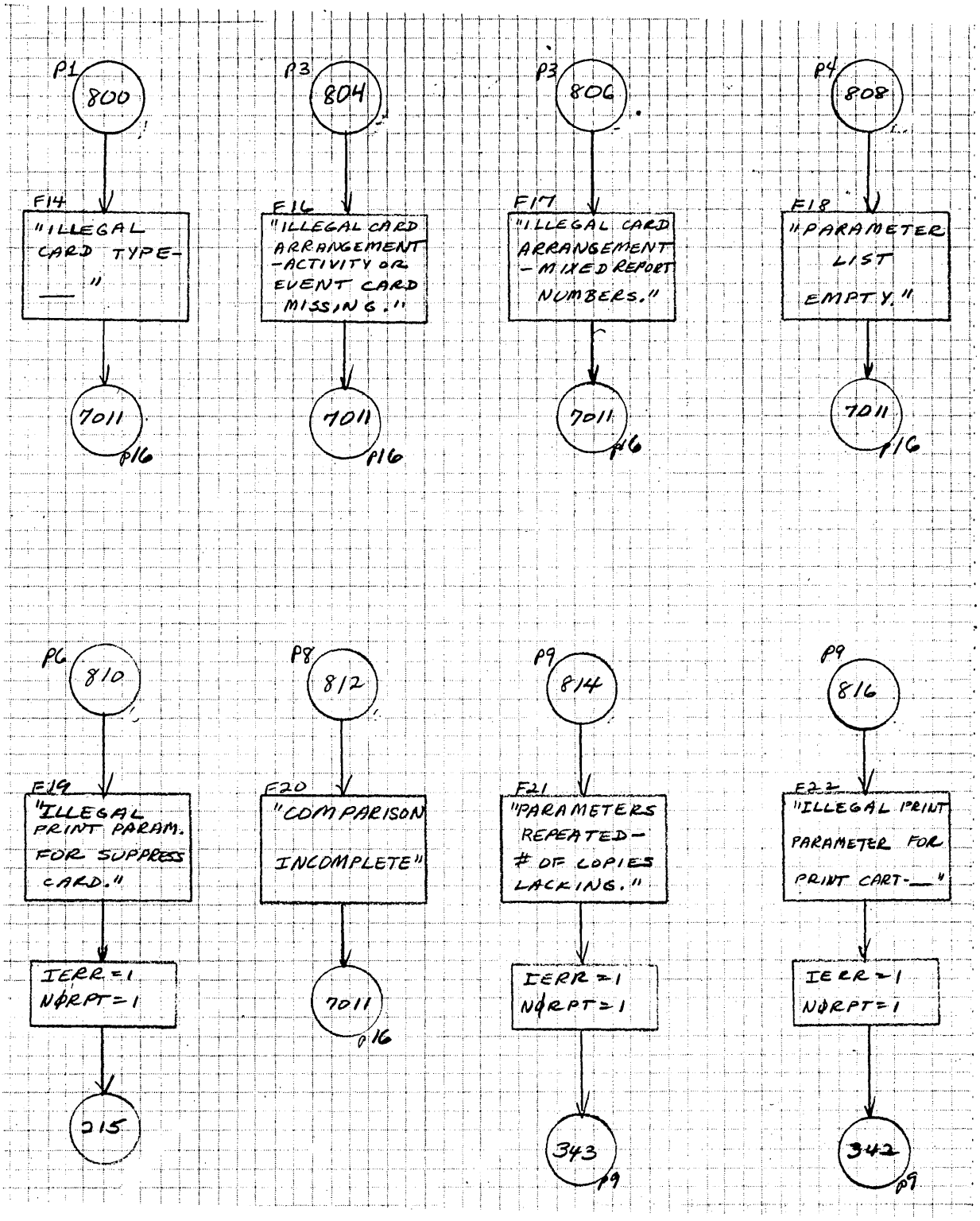


Figure 73. PCARD Subroutine Flowchart (cont'd) P13

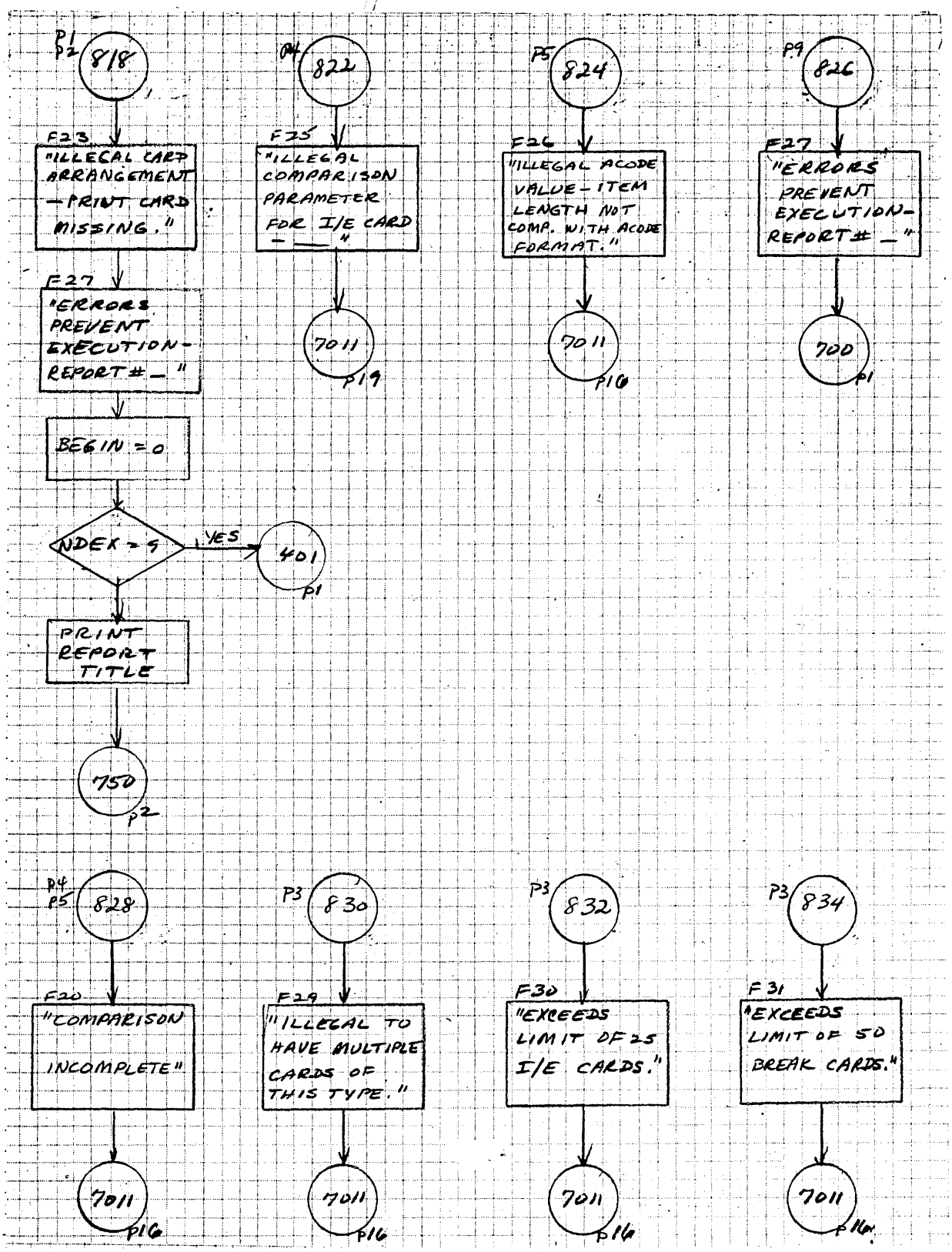


Figure 73. PCARD Subroutine Flowchart (cont'd) P14

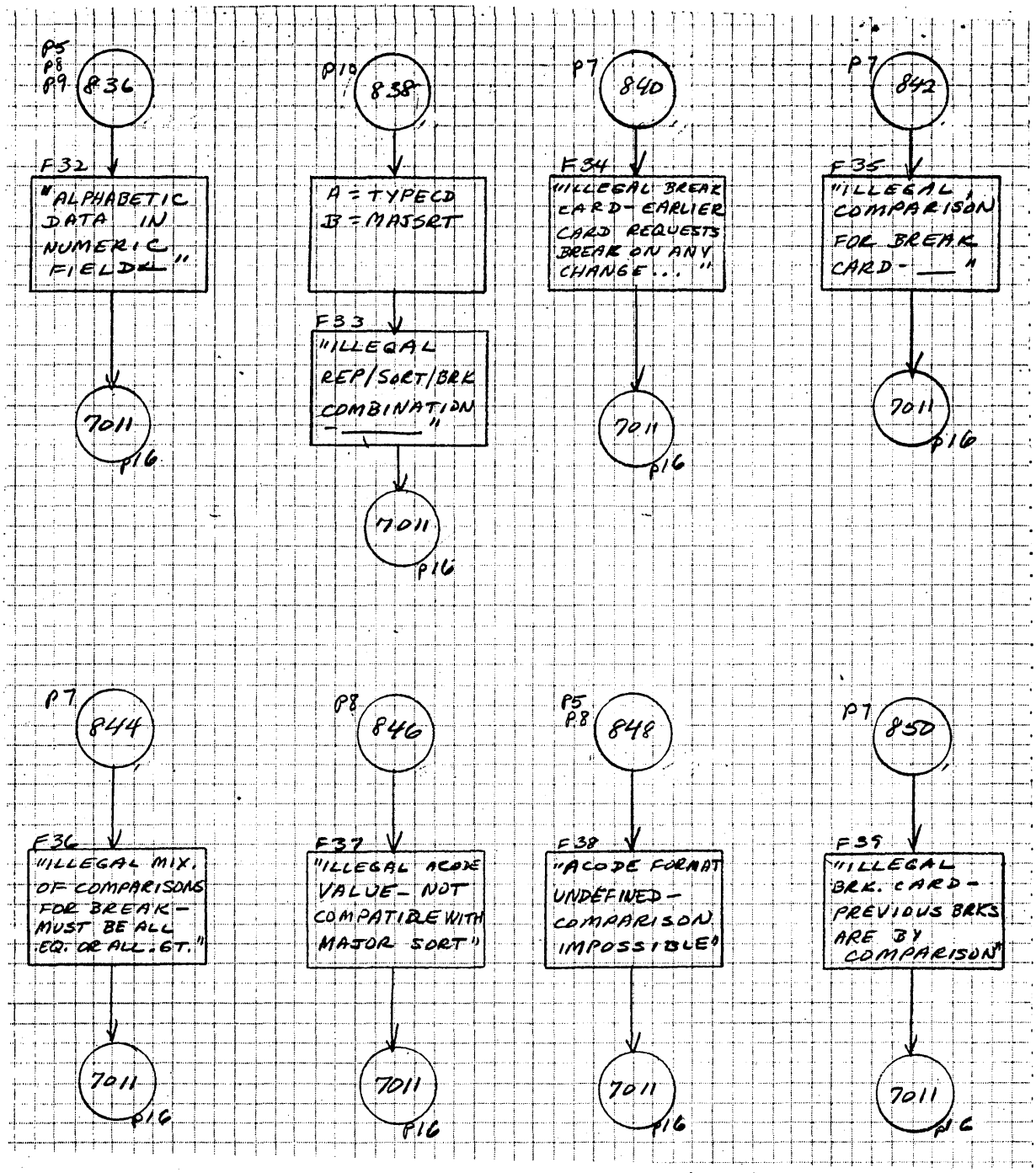


Figure 73. PCARD Subroutine Flowchart (cont'd) P15

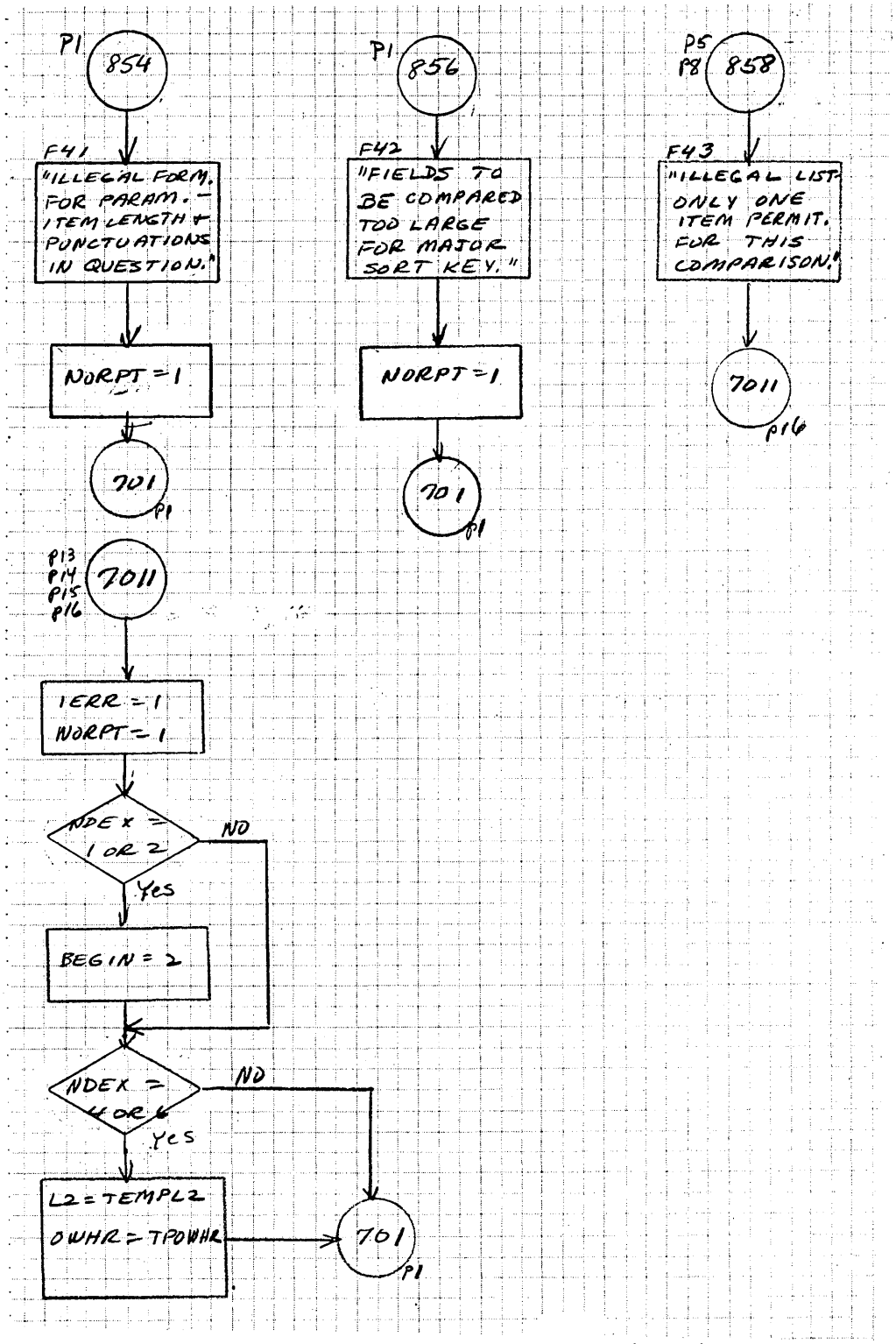


Figure 73. PCARD Subroutine Flowchart (cont'd) P16

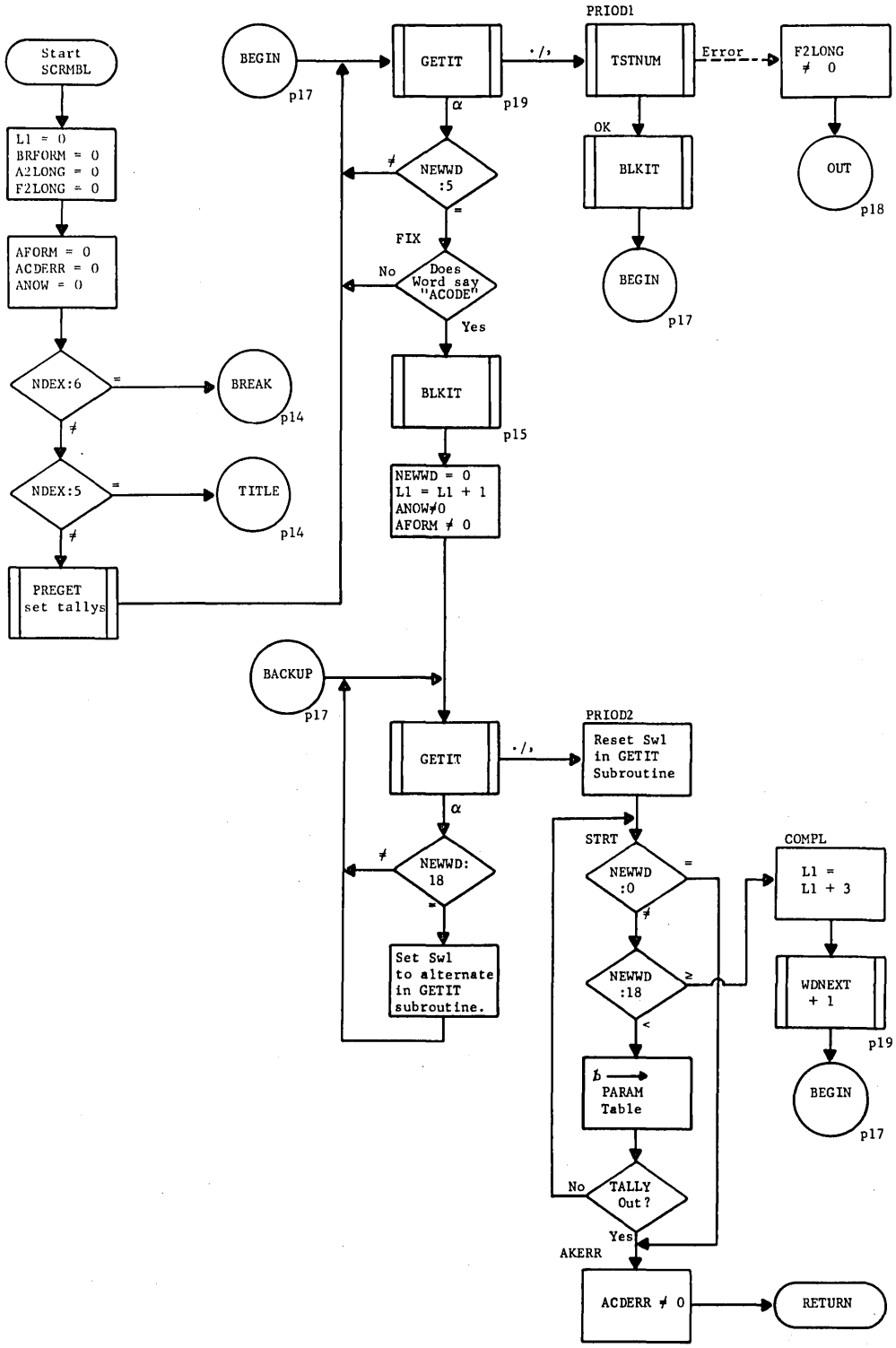


Figure 74. SCRMBL Subroutine Flowchart P17

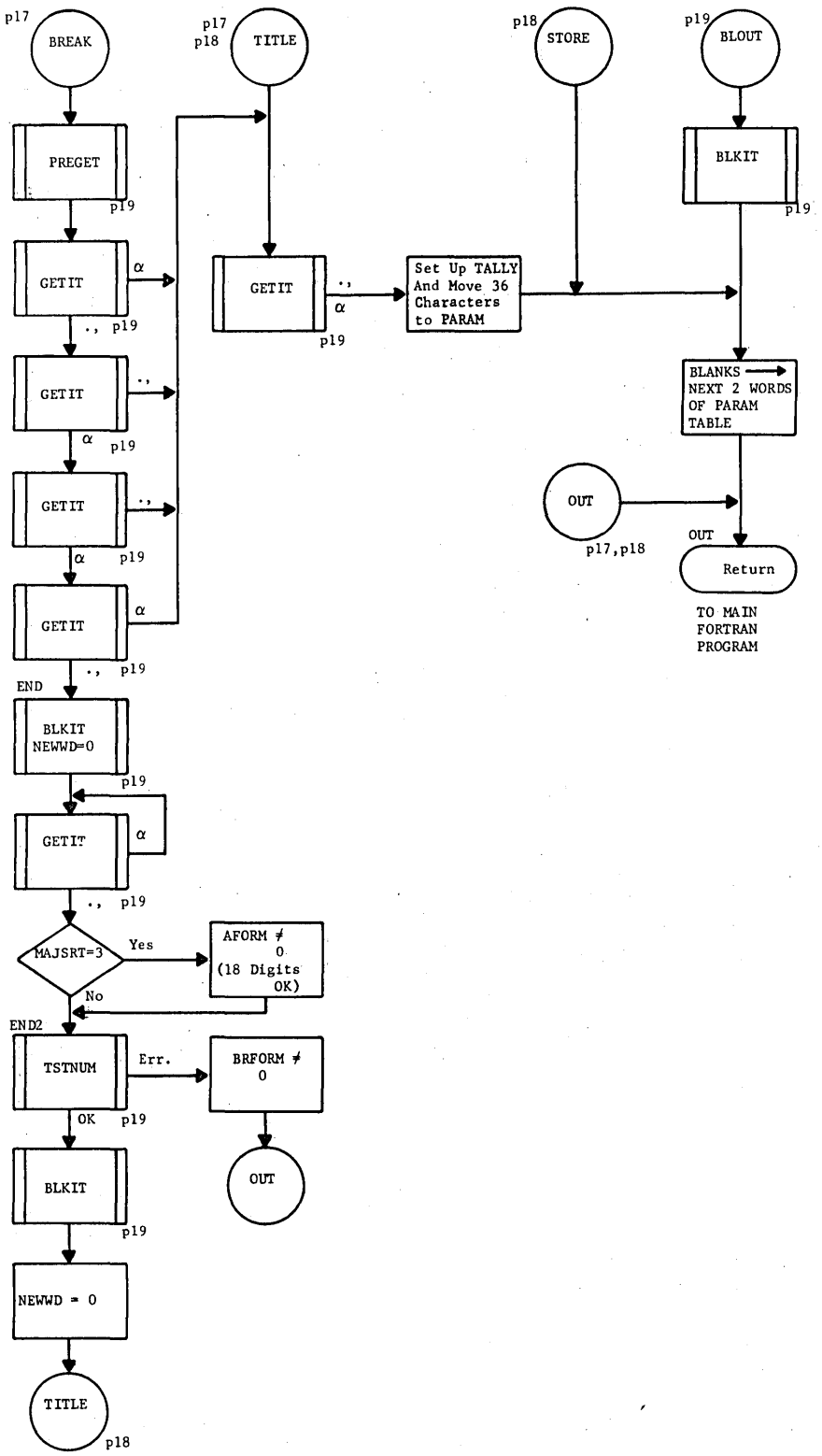


Figure 74. SCRMBL Subroutine Flowchart (cont'd) P18

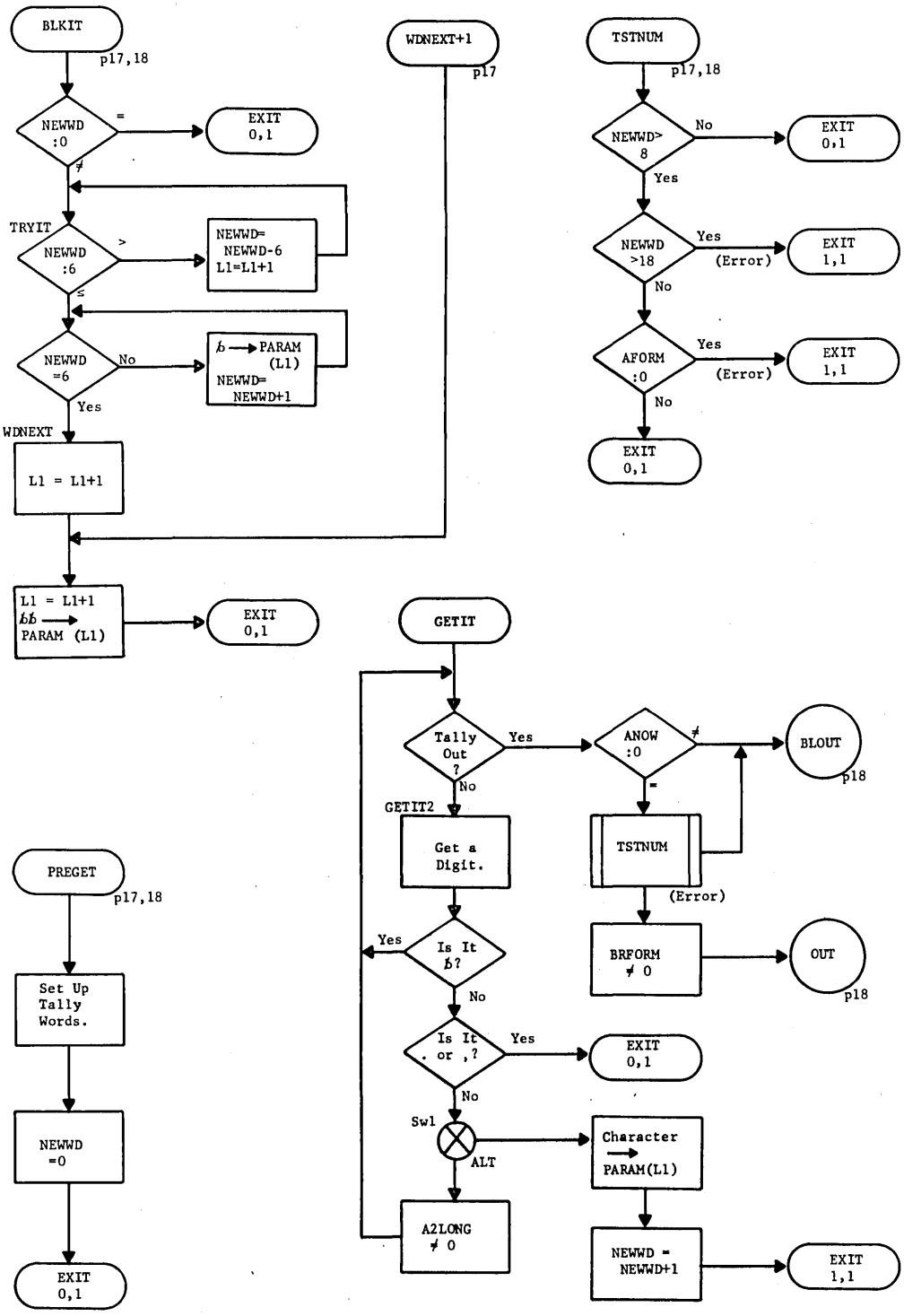


Figure 74. SCRMBL Subroutine Flowchart (cont'd) P19

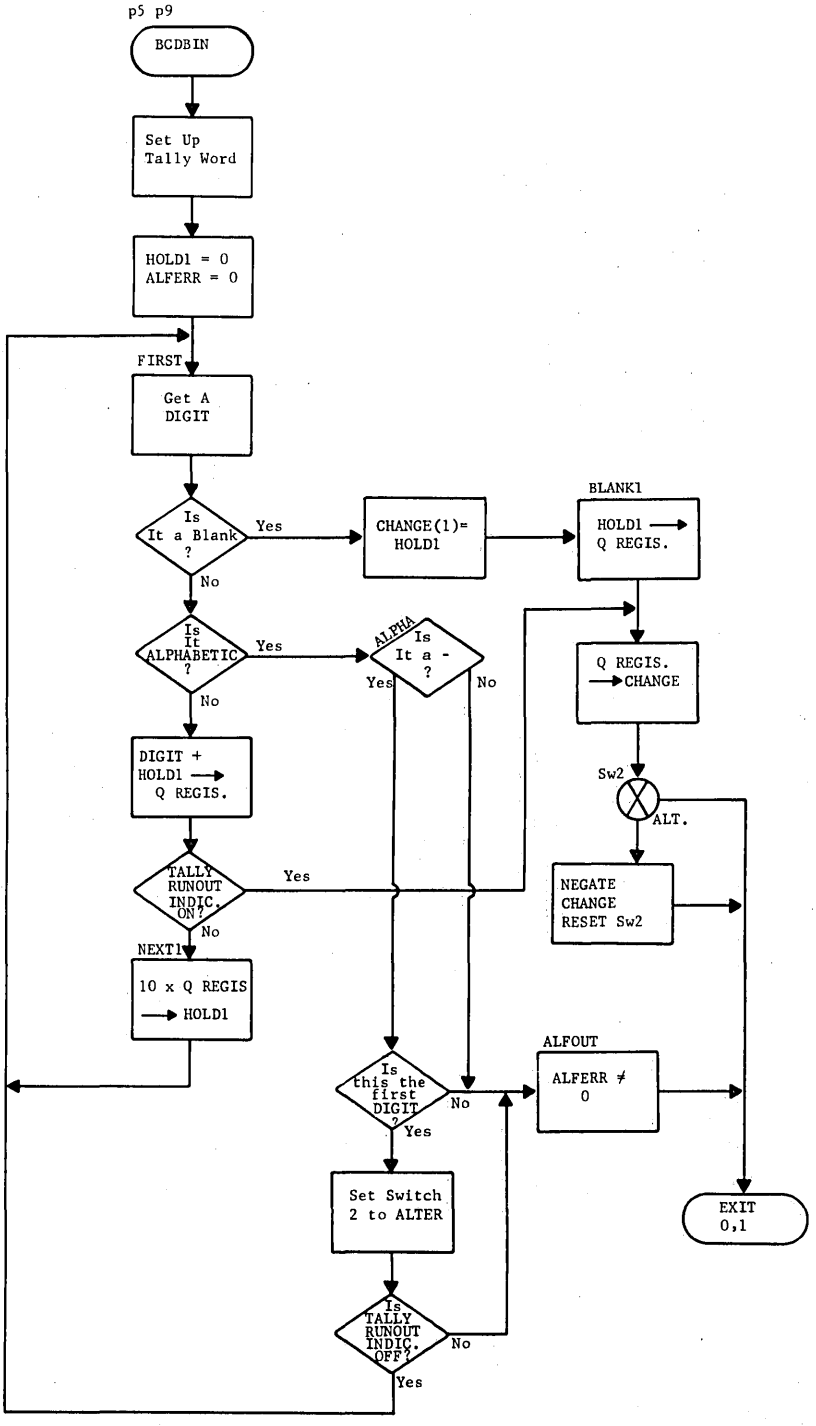


Figure 75. BCDBIN Subroutine Flowchart P20

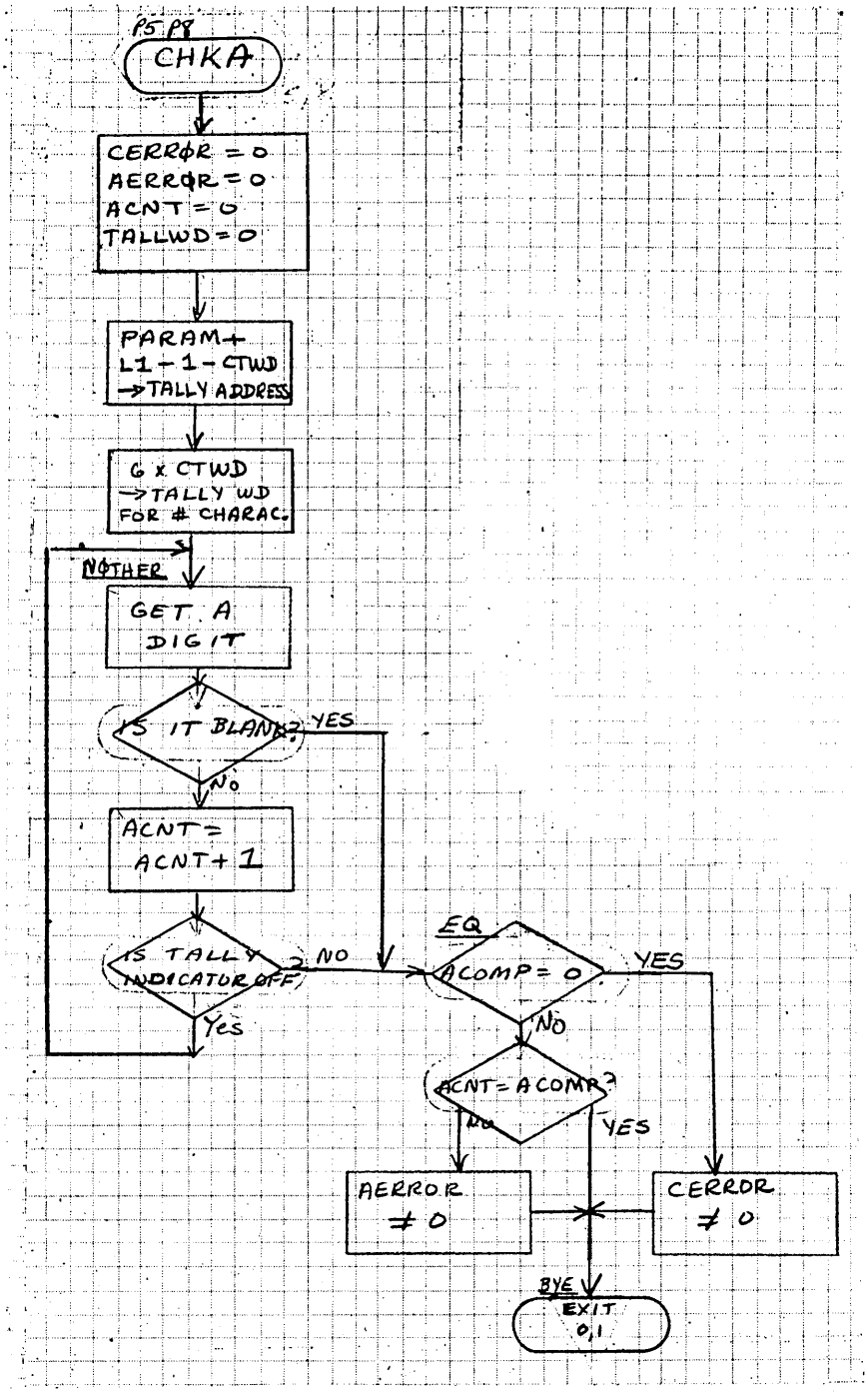


Figure 76. CHKA Subroutine Flowchart P21

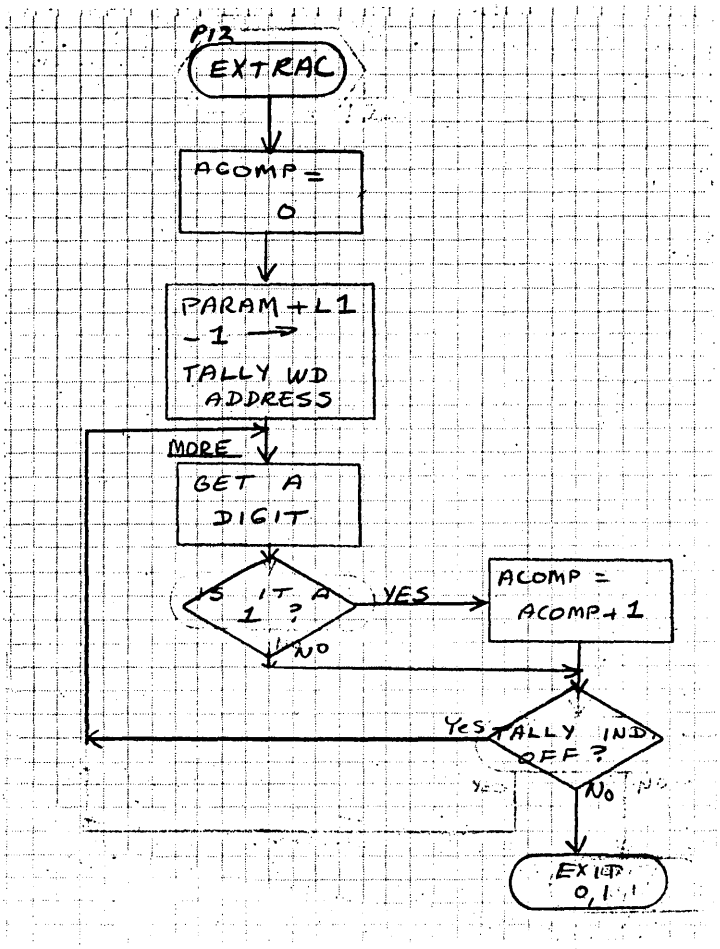


Figure 77. EXTRAC Subroutine Flowchart P22

7. OUTPUT REPORT SECTION - SETOUT

The Output Report Section, SETOUT, prints activity reports and event reports, using parametric data that has been extracted from the report parameter cards.

Some values required for the reports must be calculated. Some require conversion and formatting. The critical data--Critical Predecessor, Level Code of Critical Predecessor, Event Slack, and End Event Expected Date--is determined in the SETOUT Section. Flowcharts for SETOUT Section are found at the end of this chapter. Figure 71 at the end of chapter 6 shows both REPGEN and SETOUT linkage and file relationships.

SETOUT INPUT

Input to SETOUT is a table in core and a file to be read. The table (ITAB) contains data from the output parameter cards. This data determines the characteristics of the report. The table ITAB has been built by subroutine PCARD in link REPGEN. The input file is a modification of the master file. (See Modified Master File format, Chapter 4.)

SETOUT OUTPUT

Various combinations of three reports--event report, standard activity report (2 lines of data per activity), and short form activity report (1 line per activity)-- are produced as output on the printer. A file is also written as output for each run. This file is saved as possible input for another report of the same type, so that the user can take advantage of preordering. The activity file format and the event file format are shown on the following pages.

ACTIVITY FILE RECORD FORMAT

WORD	CONTENTS	FORMAT
1	Expected Date	0———17 18———26 27———35 BCD Month, Binary Day, Binary Yr.
2	Latest Date	Same as Word 1
3	Activity Slack	Weeks and tenths times ten
4	t_e	Same as Word 3
5	Pred. Event #	Binary
6	Expected Time	Weeks and tenths times ten + 6384
7	Pred. Ev. Level Code, Suc. Ev. Level Code, Crit. Pred. Level Code, Pred. Ev. Int. Flag, Suc. Ev. Int. Flag, Crit. Pred. Int. Flag	6 BCD Characters
8	Event Slack	Same as Word 3
9	Succ. Event #	Binary
10	Latest Time	Same as Word 6
11	Short Path Flag	BCD, Character #5
12	Crit. Pred. #	Binary
13	Change Code	Binary
14	Suc. Event Exp. Date	Same as Word 1
15	Scheduled Date	BCD MMDDYY where MM=month; DD=day; YY=year
16-21	Activity Description	36 BCD Characters
22-25	Activity Code	22 - 4 BCD Characters 23 - 4 BCD Characters 24 - 4 BCD Characters 25 - 6 BCD Characters

In addition:

Bit 35 of words 22 and 23 flag Current Activities and Actual Dates, respectively.
Character 5 of word 24 represents the "Run Date Option"

EVENT FILE RECORD FORMAT

WORD	CONTENTS	FORMAT
1	Expected Date	0——— 17 18 ——— 26 27 ——— 35 BCD Month, Binary Day, Binary Yr.
2	Latest Time	Weeks and tenths times ten + 6384
3	Critical Pred. #	Binary
4	Scheduled Date	BCD MMDDYY where MM=month; DD=day; YY=year
5	Level Code of Crit. Pred; Lev. Code of Event; Event Interface Flag; Crit. Pred. IFL; Change Code; Short Path Flag	6 BCD Characters in respective areas excepting "Change Code," which is binary
6	Actual Date	Same as word 4
7	Expected Time	Same as word 2
8	Event Number	Binary
9	Slack	Weeks in tenths times ten
10	Latest Date	Same as word 1
11-16	Event Description	36 BCD characters

PREPARE OUTPUT--LINK SETOUT

Link SETOUT is organized around two sort links. The first sort link determines the critical data needed for event reports and standard activity reports. There is also some data calculation in the first sort link which must be accomplished even if critical data is not required; that is, in the case of the short activity report. When the short activity report is used, these calculations are executed in the second sort link. The basic function of the second sort link is to order the file according to the reporting sequence required.

The four subroutines in the SETOUT link are: OUT, CALOUT, MINT, and DIB.

Subroutine OUT

The Subroutine OUT is a preface routine executed once at the beginning of each report. It processes portions of ITAB and determines which link is to be called next.

OUT calls the link SEALNK when the first event or standard activity report is to be processed. SEALNK is not called for a short activity report, even if it is the first report of a run.

OUT calls the link REGLNK if SEALNK has been executed once, or if the current report is a short activity report.

Subroutine CALOUT

The subroutine CALOUT converts date to time. The output consists of 3 words: binary day (MDAY), BCD month (MNTH), and binary year (MYER). Input is the binary number of weeks (NWEEK).

Subroutine MINT

MINT is a function which converts BCD to binary. Its use is shown by the example: A=MINT (B, N) where A = binary variable; B = BCD variable; N = number of characters (1 to 8) starting with character position 0. See Figure 8 for a flowchart of MINT.

Subroutine DIB

The subroutine DIB converts BCD to binary for event numbers and is faster than MINT. Input is two words of eight BCD characters in IDEC (1) and IDEC (2), starting in character position 0. Output is IBIN. See Figure 50 for a flowchart of DIB.

SORT LINK--SEALNK

Link SEALNK sorts the modified master file, to determine the critical data: critical predecessor, level code of critical predecessor, event slack, and end event expected date. It does this by sorting with a sequence using the successor event number as the major key and activity slack as the minor key. Two files are created as output: an event file and an activity file. These are used as input to REGLNK.

Subroutine SEASRT

Subroutine SEASRT provides the macro parameters and linkage necessary for the sort, and calls the sort. The GE-625/635 Sort/Merge System is used.

Subroutine SEICE

Subroutine SEICE is the sort input coding element. It processes the input file to prepare it for the sort. For example, it converts the event number to a binary value and calculates the activity slack. It is executed once for each record of input.

Subroutine SEOCE

Subroutine SEOCE is the sort output coding element. It rearranges the data, finds the critical data and writes the event and activity files.

The sort key sequence is set up to provide groups of records for SEOCE. All records in a group share the same successor event. The order within the group is from the smallest to largest slack. Thus, the critical path data resides in the first record of the group. There is one event record constructed for each group, and there is one activity record for each record of the group.

Subroutine LEVCDE

Subroutine LEVCDE searches a table of level codes to find the correct event level code for the event file and the level code of the critical predecessor for the activity file. The table correlates level codes with event numbers. It is constructed in the subroutine SEICE. The format of the level code table SEAS, which is called LC (501), in COMMON is ten octal digits for a binary number and one BCD character for the level code.

Subroutine SEOF

Subroutine SEOF writes the end-of-file record and mark on the output tapes and rewinds the tapes.

SORT LINK--REGLNK

Link REGLNK sorts event or activity records in report order and prints the event or activity report.

Subroutine REGCNT

Subroutine REGCNT determines which files should be used as input to and output from the sort. Input can be a modified master file, an activity file, or an event file. Files are rotated so that only four are needed, aside from the three required by the sort.

Subroutine SORMAC

Subroutine SORMAC constructs the table specifying the sort key field description. The sort system is designed to derive this table from the macro parameters. This table may vary from one report to another, but the macro calls remain constant; therefore, the subroutine constructs the table dynamically. The location of the table is given to the sort system in the subroutines REGST and REGSRT.

Subroutine REGST

Subroutine REGST calls the sort for an event report. It provides parametric data and linkage for coding elements.

Subroutine REGSRT

Subroutine REGSRT calls the sort for activity reports. It provides parametric data and linkage for user coding elements.

Subroutine RECON

The subroutine RECON reconstructs the file as it is fed in as input to the sort. The reconstruction is executed only when a modified master file is the input for a short activity report. This occurs when SEALNK has not been executed during the entire run, signifying that no event or full-length activity report has been requested. When reconstruction is not executed, a record is read and left unchanged.

Subroutine REPST

The subroutine REPST is executed prior to the output of each report; that is, it is executed once at the beginning of the subroutine SEOCE.

Subroutine PRC

Subroutine PRC controls execution of the subroutines involved in SEOCE, the subroutine which prints the reports and writes the output tapes. PRC calls the following subroutines: PAB, PBB, PEB, PSP, HDG, and BREAK. It uses these routines to prepare the data. PRC executes the printer write and the tape write.

Subroutine HDG

Subroutine HDG controls the setup of the print line images for the report heading. It uses subroutines PRDT, ISO, BID, and SUP.

Subroutine PRDT

Subroutine PRDT moves heading characters to the print line image from the data area. Its calling sequence is set by HDG. The parameters specify source, destination, and number of characters. JC is the variable set to the number of characters to be moved. JD is the variable set to the character position of the first character in the data array (DTA) to be moved. JP is the variable set to the first print position of the print line image to which data is to be moved. LP is the choice of print line image arrays. The sequence of the parameters is (JC, JD, JP, LP).

Subroutine SUP

Subroutine SUP implements the suppress function for a given print line image of a main heading, column heading, or data line. It is called by the subroutines HDG, PAB, PBB, PEB, and PSB. It uses variables that have been set from report parameters.

Subroutines PAB, PBB

Subroutines PAB and PBB control the setup of the print line images for the standard (long form) activity report. Two lines are necessary for each activity in this report. PAB handles the first line and PBB handles the second line. These subroutines control the line setup by using subroutines PRNT, NUM, SUP, and BID.

Subroutine PSB

Subroutine PSB controls the setup of the print line image for the short form activity report. It also uses the subroutines PRNT, NUM, SUP, and BID.

Subroutine PEB

Subroutine PEB controls the setup of the print line image for the Event Report. It also uses the subroutines PRNT, NUM, SUP, and BID.

Subroutine PRNT

Subroutine PRNT moves characters to the print line image from a data area. These are data characters as distinquished from the heading characters moved by the subroutine PRDT. PRNT is called by the subroutines PAB, PBB, PEB, and PSB. The parameters serve the following purposes:

NC specifies the number of characters, PD, the data position, and PP, the position in the print line image. A sample sequence is:

```
CALL PRNT (NC, PD, PP)
```

Subroutine BID

Subroutine BID converts one word from binary to BCD. Input is IBIV, and output is IDVC. It is used by the subroutines HDG, PAB, PBB, PSB, and PEB.

Subroutine ISO

Subroutine ISO determines the location of an embedded name; that is, where a name is if it may be anywhere within a given number of words. Input is the size of the data area (NS). Outputs are the number of leading blanks (NL) and trailing blanks (NT). ISO is used by HDG.

Subroutine NUM

Subroutine NUM converts an event number from binary to the form dd-ddd-ddd. Input is NU. Outputs are K2 (dd), K3 (-), K4 (ddd), K5 (-), and K6 (ddd). NUM is used by the subroutines PAB, PBB, PEB and PSB.

Subroutine BREAK

Subroutine BREAK determines whether the print page should be restored on the current record (for the purpose of separating reports). Inputs are the current record (PREC) and the table ITAB. Output is the logical variable BRK. If BRK is true, a break should occur; if BRK is false, a break should not occur.

Subroutine INEX

Subroutine INEX determines whether the line for the current record should be excluded from the report. Its inputs are the current record (PREC) and the table (ITAB). Output is the logical variable EXCL. If EXCL is true, a line should be excluded; if false, the line should be included.

Subroutine SETIP

Subroutine SETIP extracts characters of the activity code which correspond to the major sort field. Input is GLOP and output is SVI. SETIP is used by the subroutines BREAK and INEX.

Subroutine BCDIT

Subroutine BCDIT performs logical comparison of the arguments. BCDIT is a function which is used by the subroutines BREAK and INEX.

Subroutine LEDZ

Subroutine LEDZ converts leading blanks to zeroes for three BCD digits, which are produced by the subroutines BID and NUM and used by the subroutines PAB, PEB, and PSB.

OUTPUT REPORTS SECTION FLOWCHARTS

The following pages contain the flowcharts for the Output Reports Section listed in the following order:

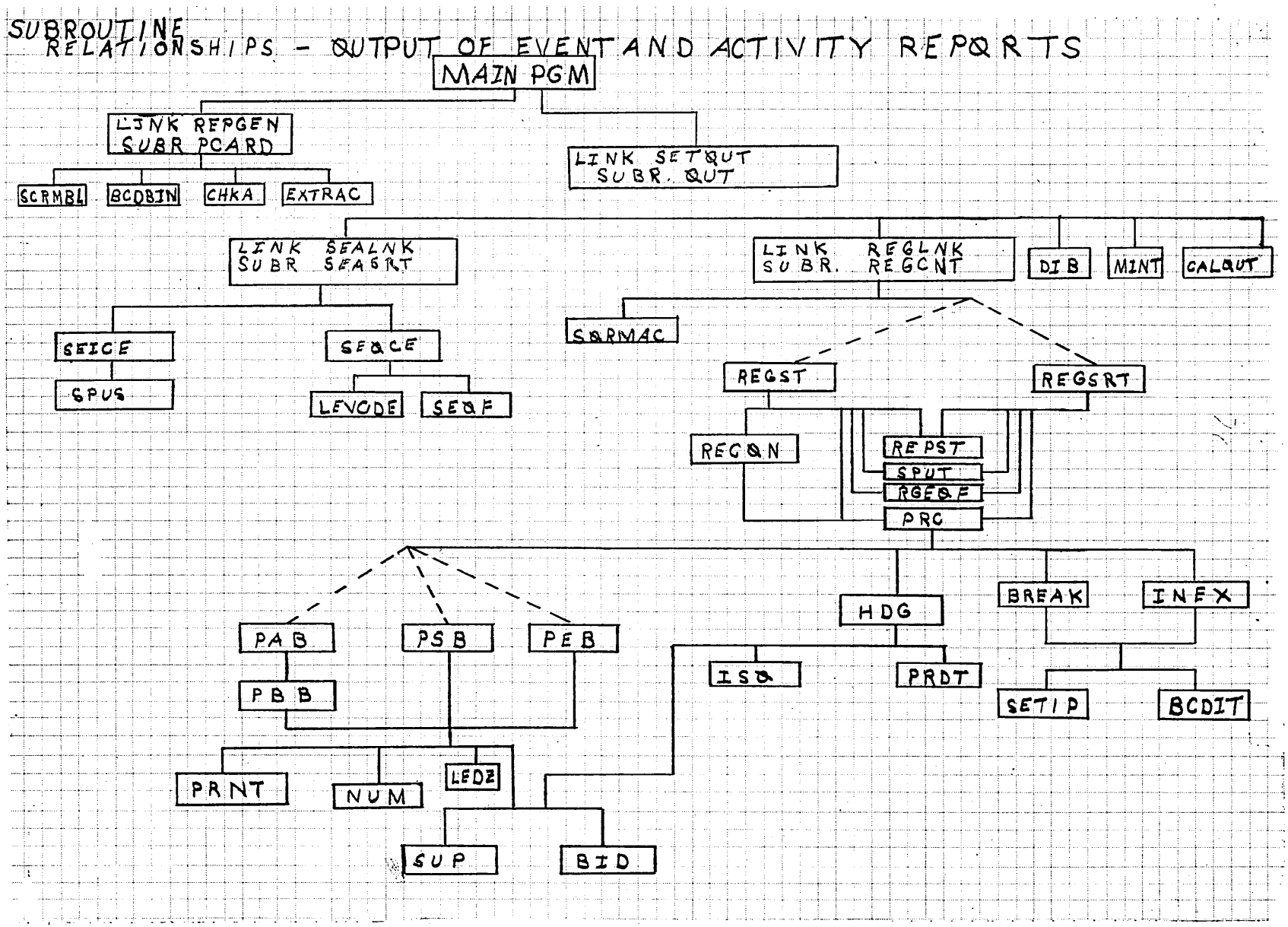


Figure 78. SETOUT Section Subroutine Relationships

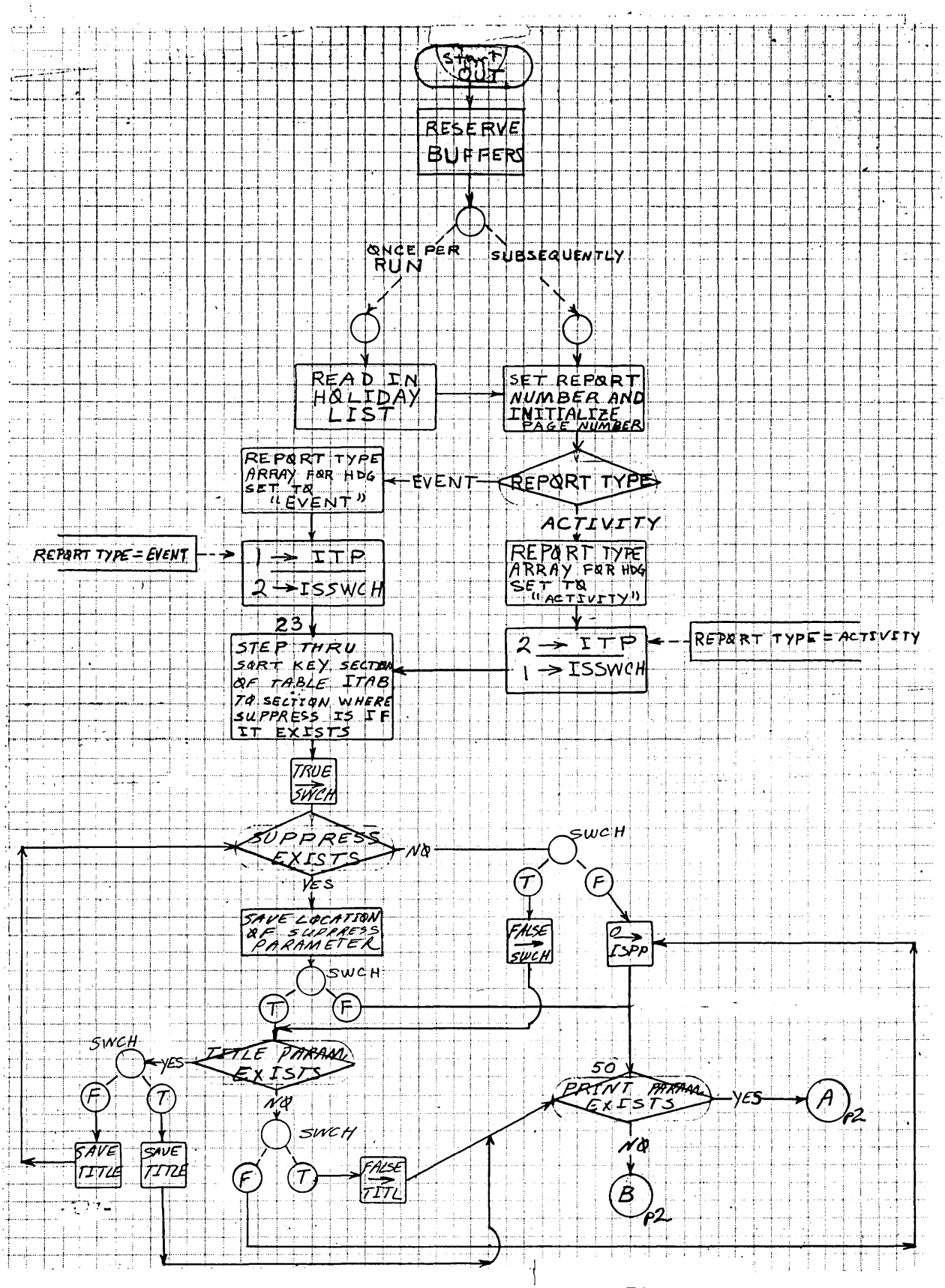


Figure 79. OUT Subroutine Flowchart P1

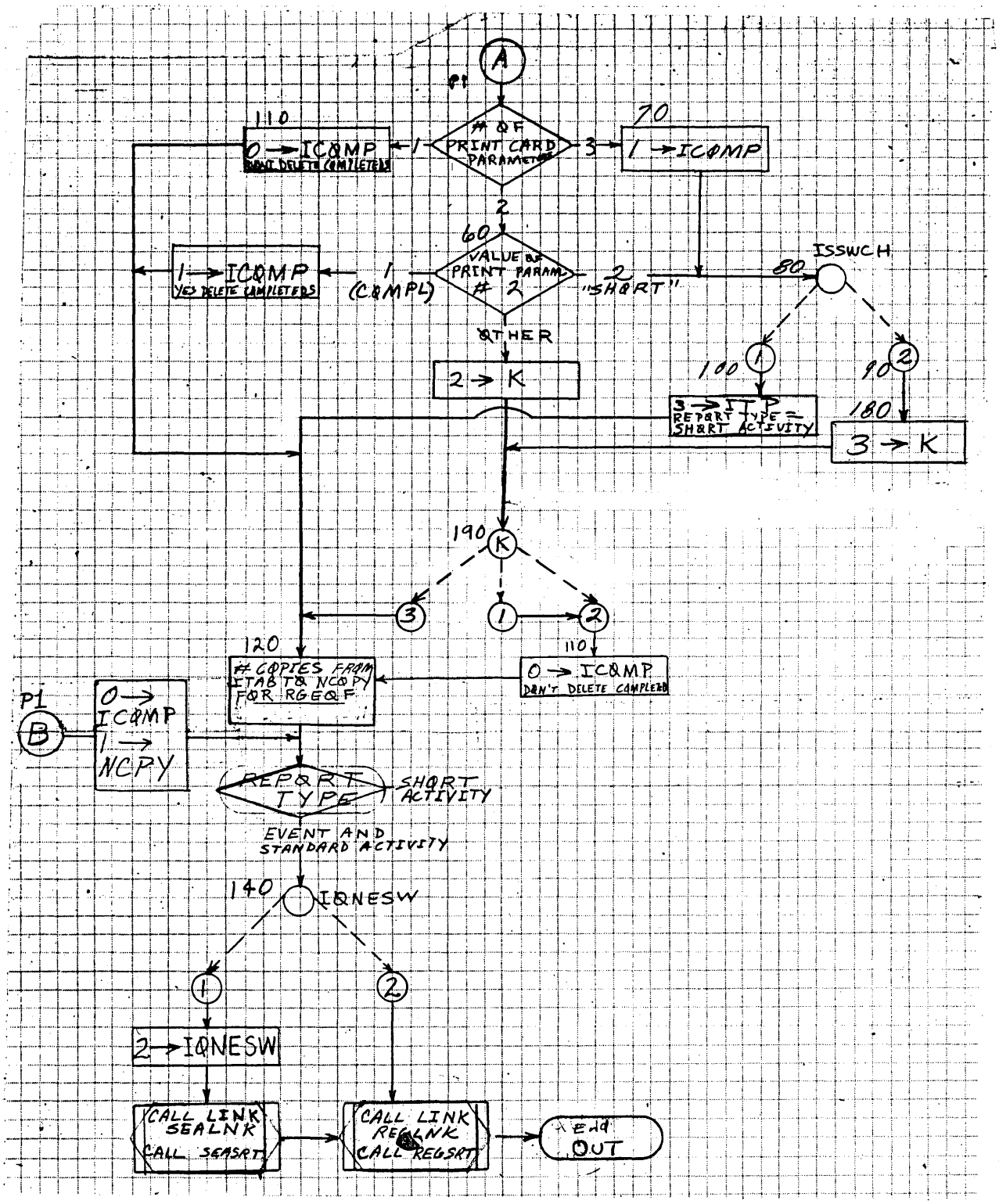


Figure 79. OUT Subroutine Flowchart (cont'd) P2

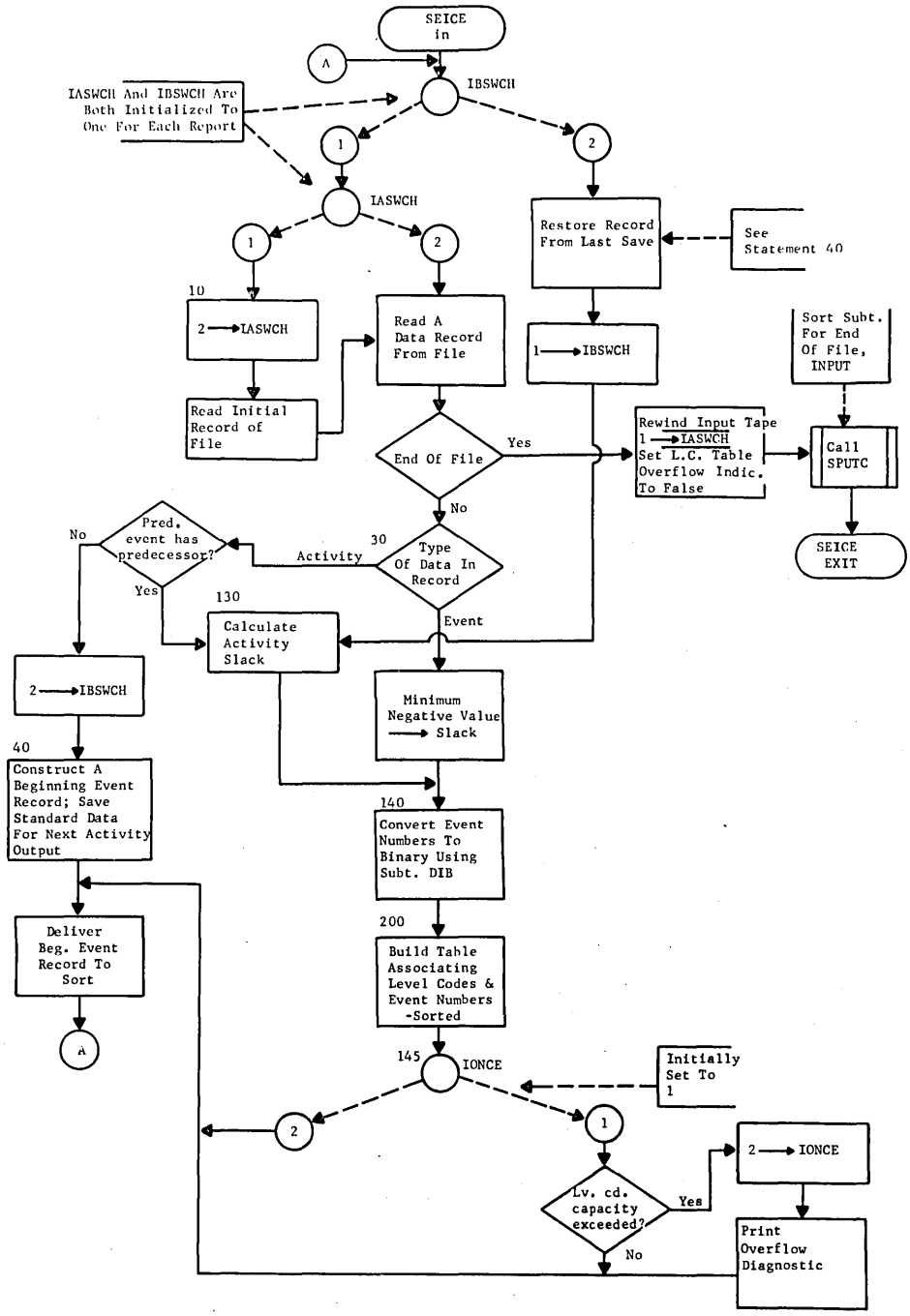


Figure 80. SEICE Subroutine Flowchart

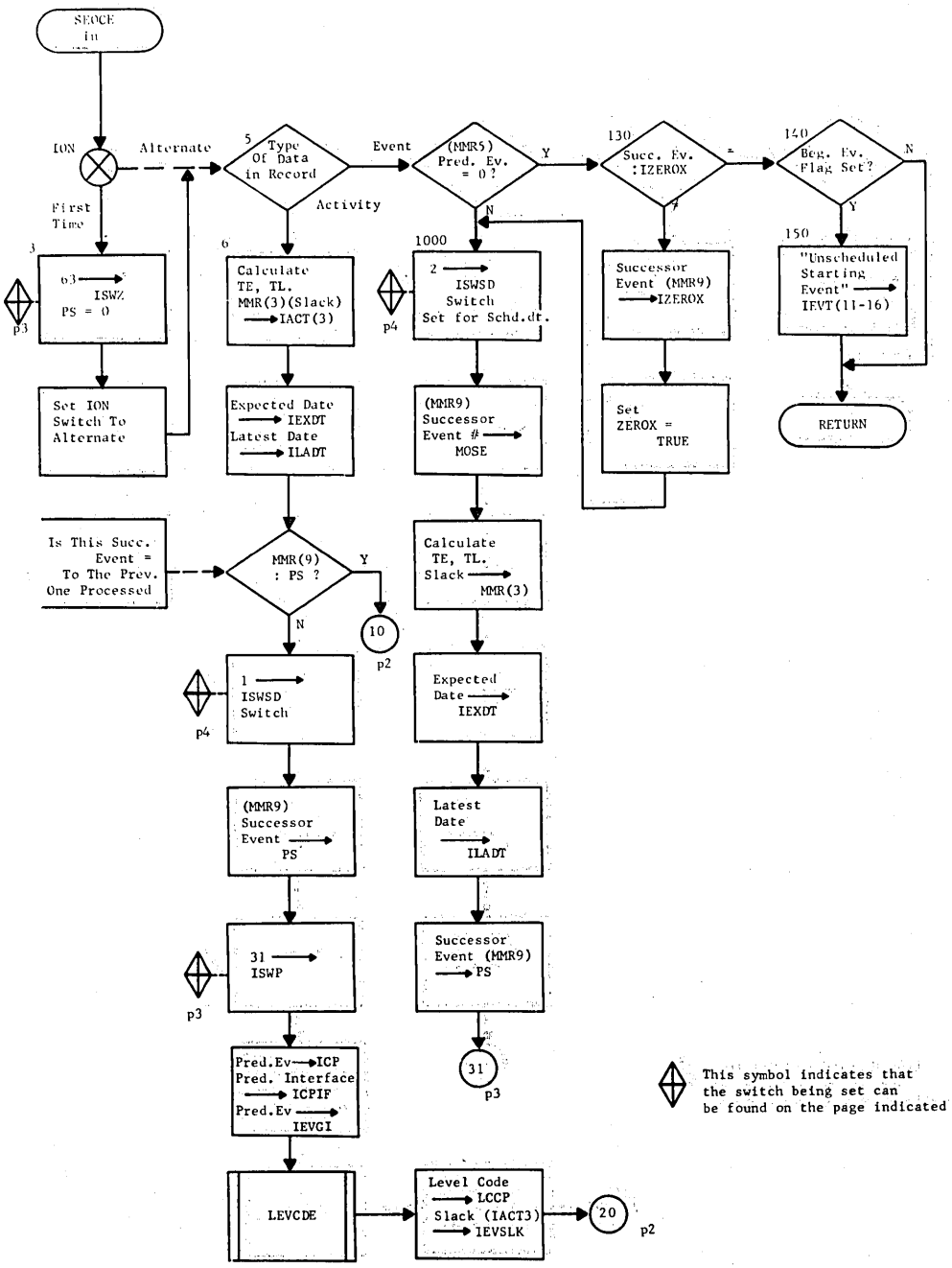


Figure 81. SEOCE Subroutine Flowchart P1

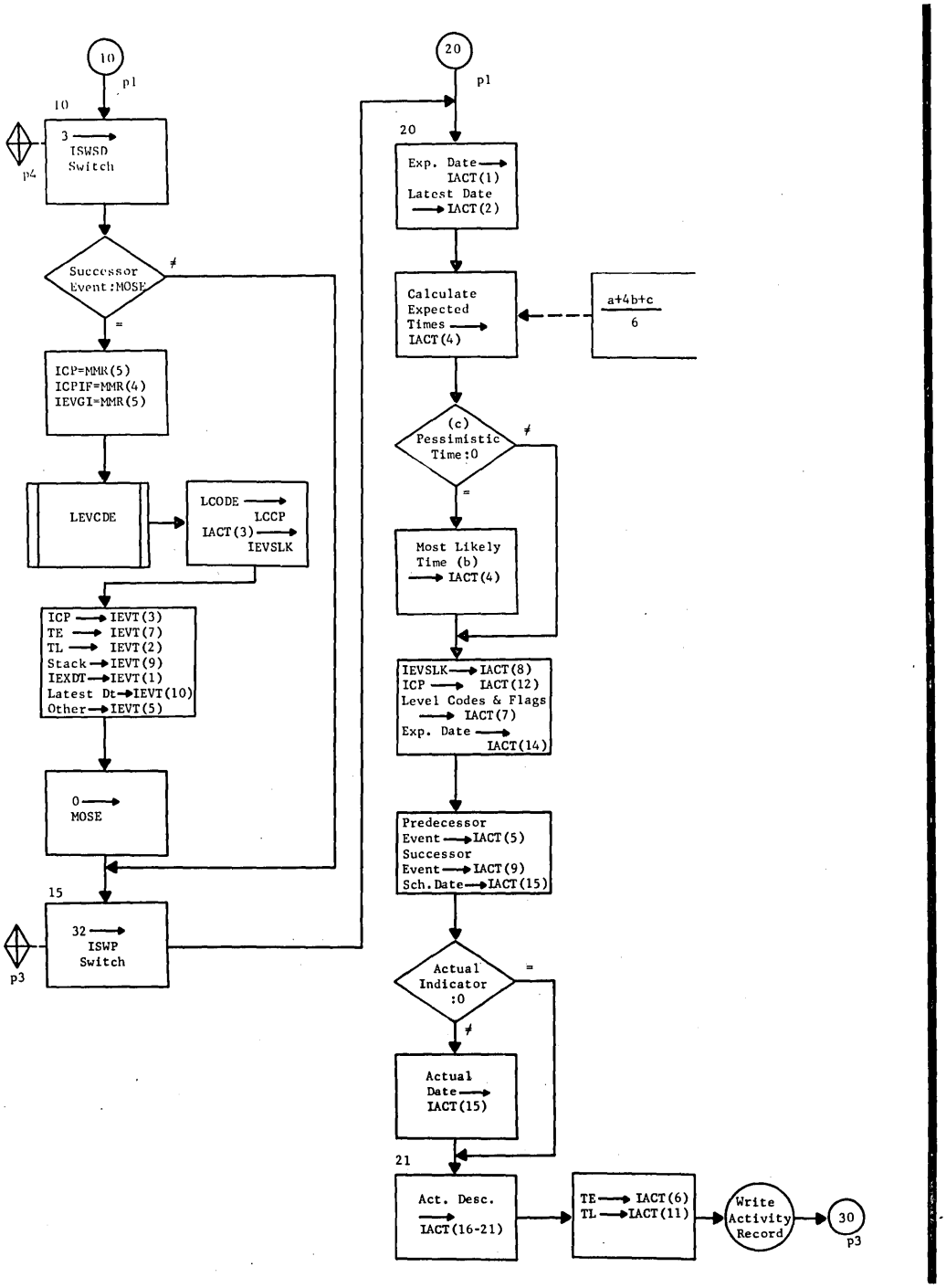


Figure 81. SEOCE Subroutine Flowchart (cont.) P2

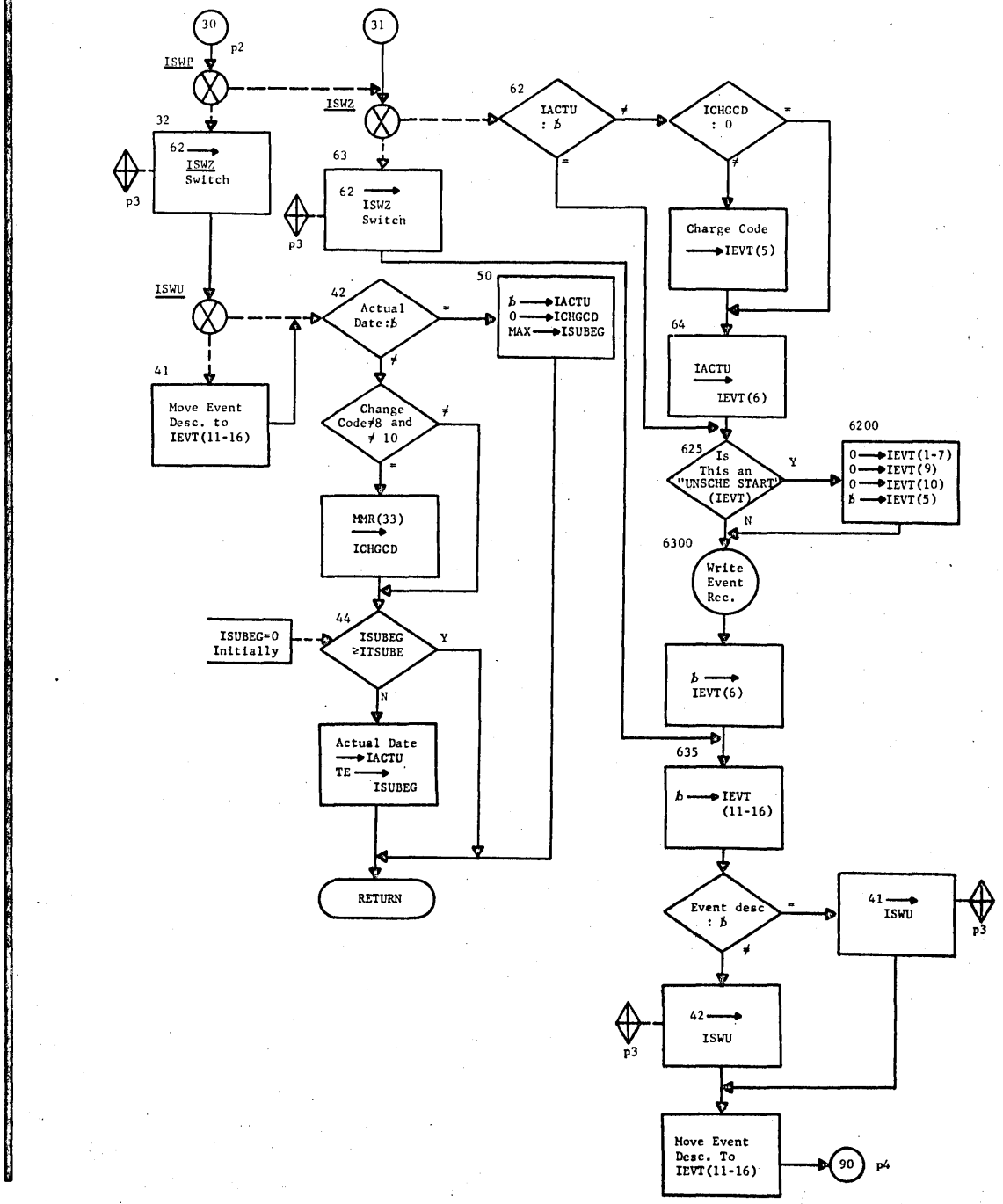


Figure 81. SEOCE Subroutine Flowchart (cont.) P3

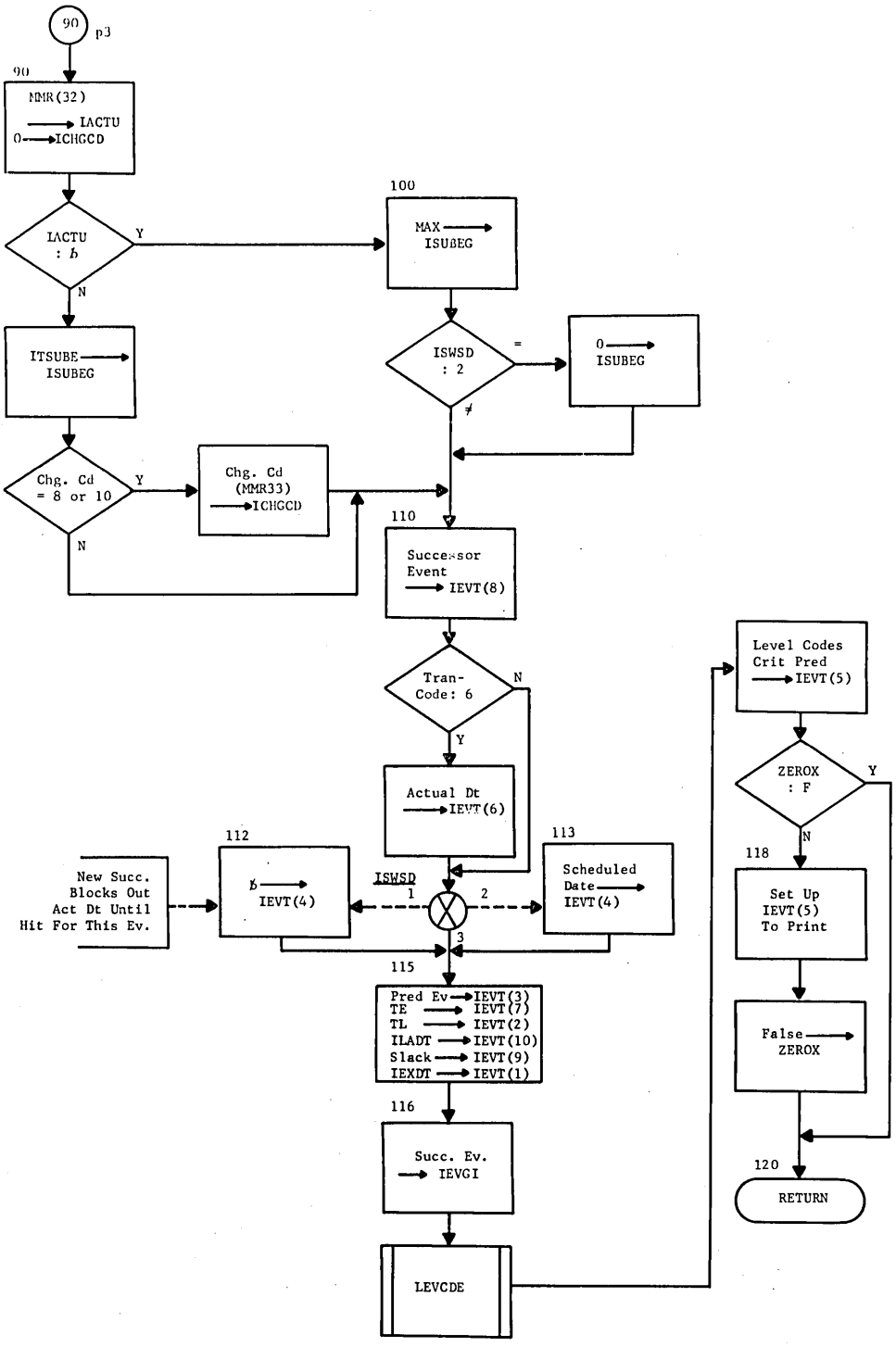


Figure 81. SEOCE Subroutine Flowchart (cont.) P4

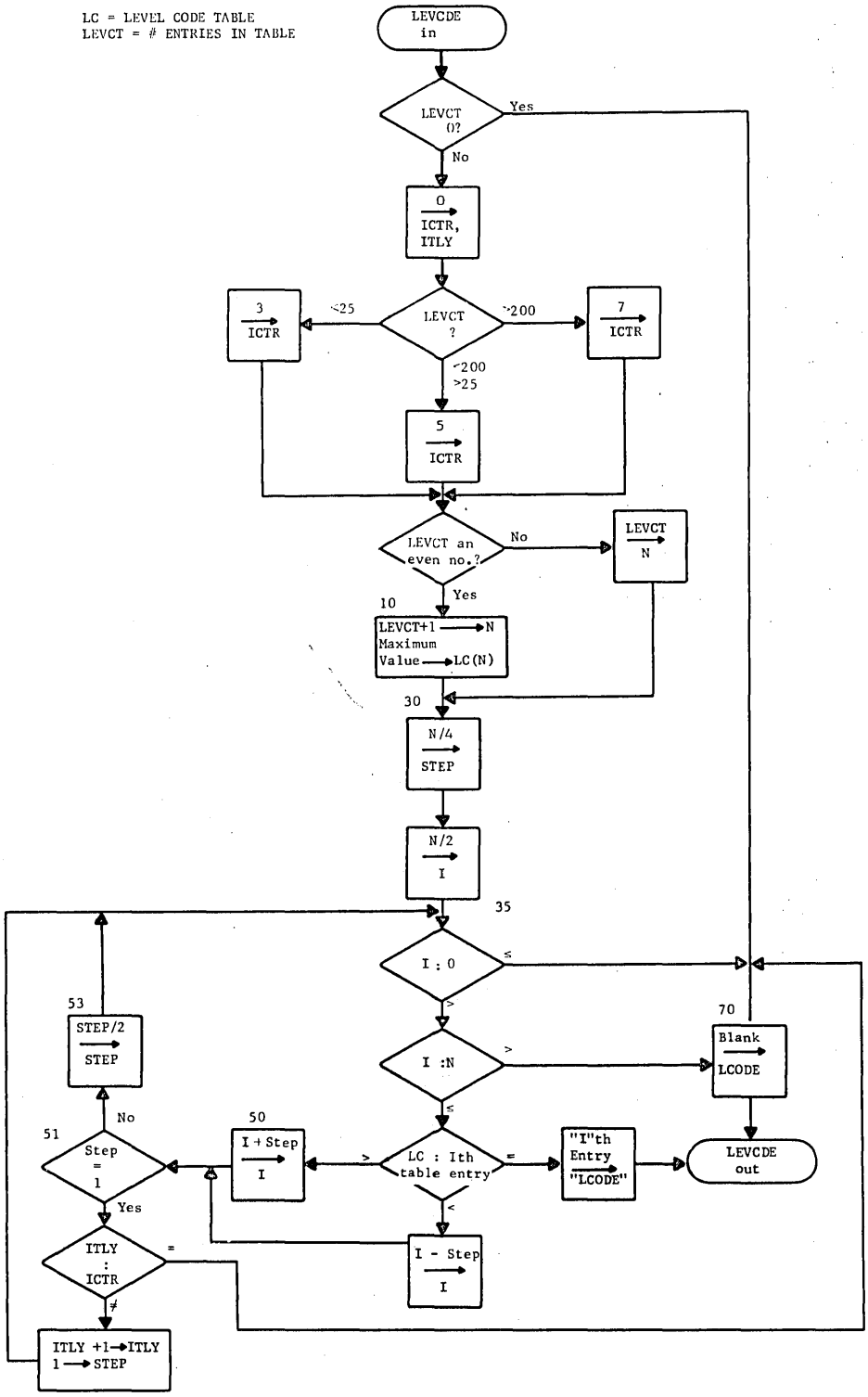


Figure 82. LEVCDE Subroutine Flowchart

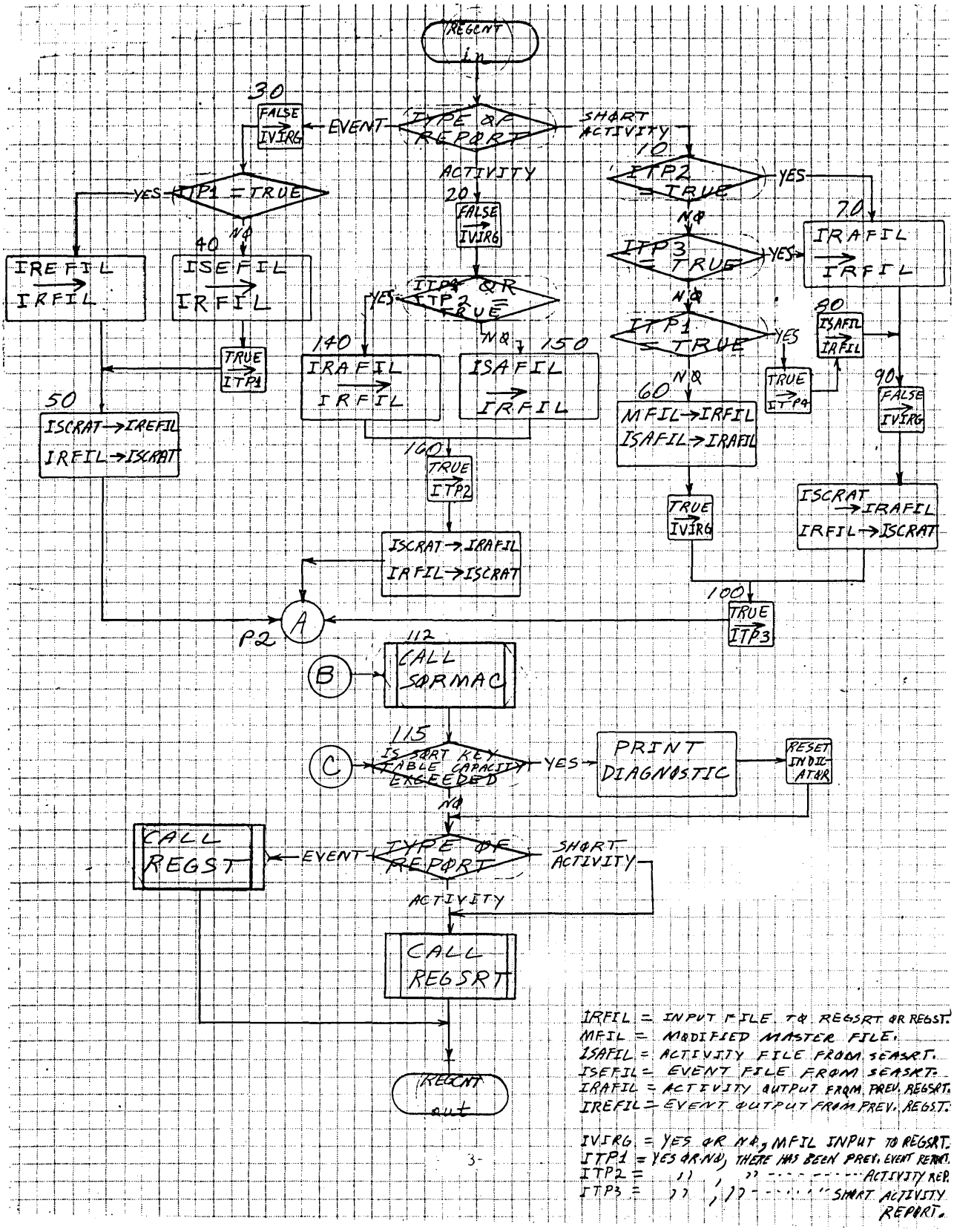


Figure 83. REGCNT Subroutine Flowchart P1

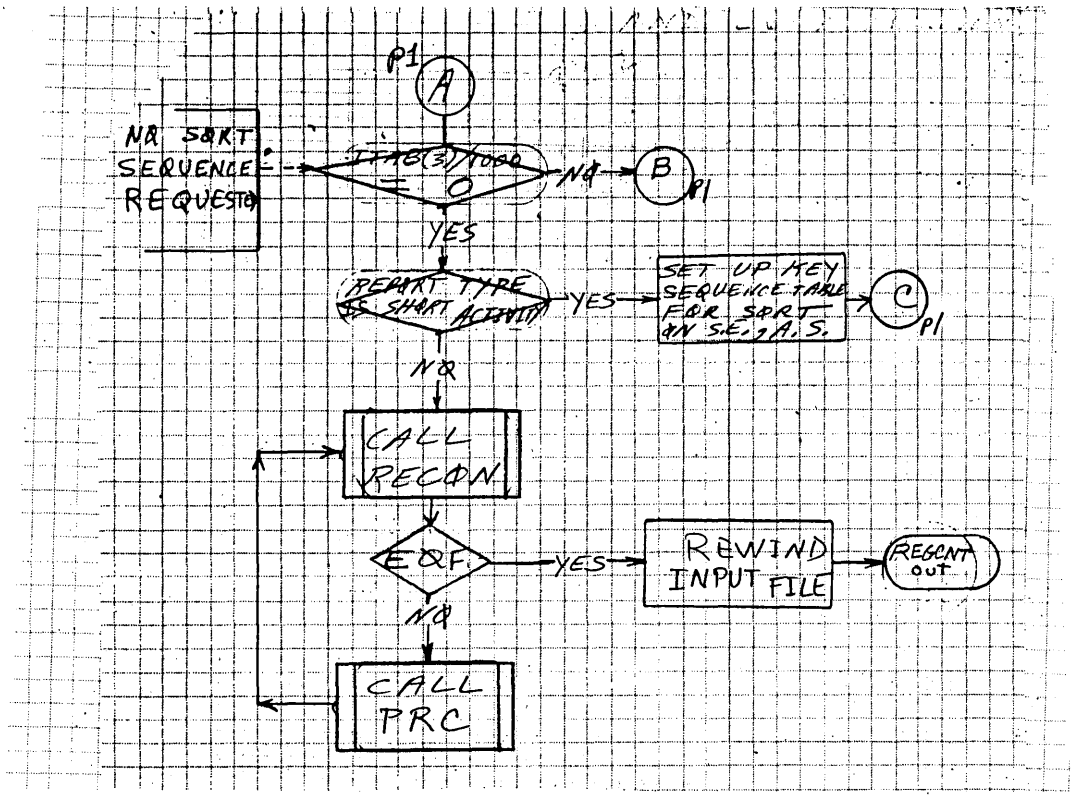


Figure 83. REGCNT Subroutine Flowchart (cont'd) P2

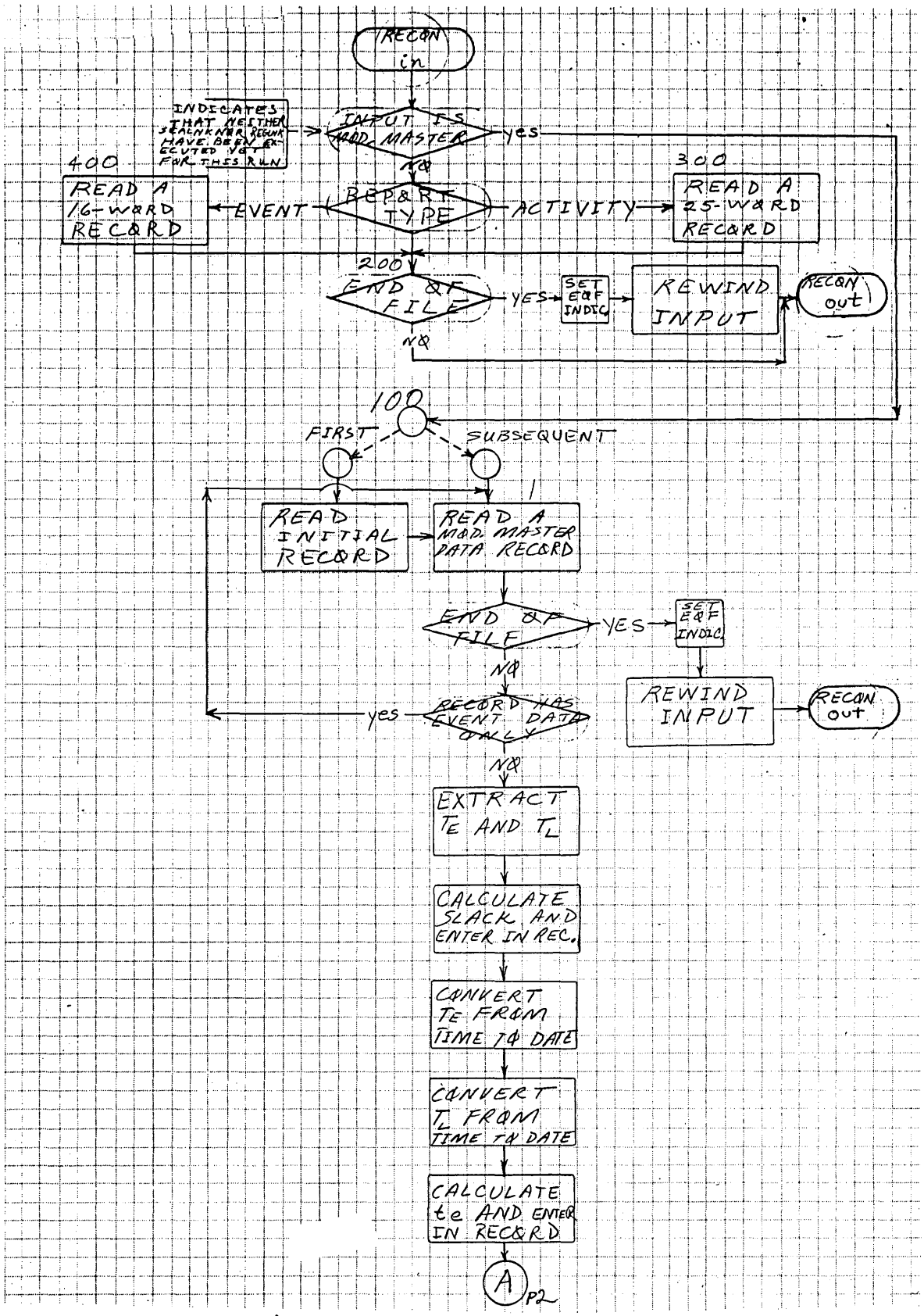


Figure 84. RECON Subroutine Flowchart P1

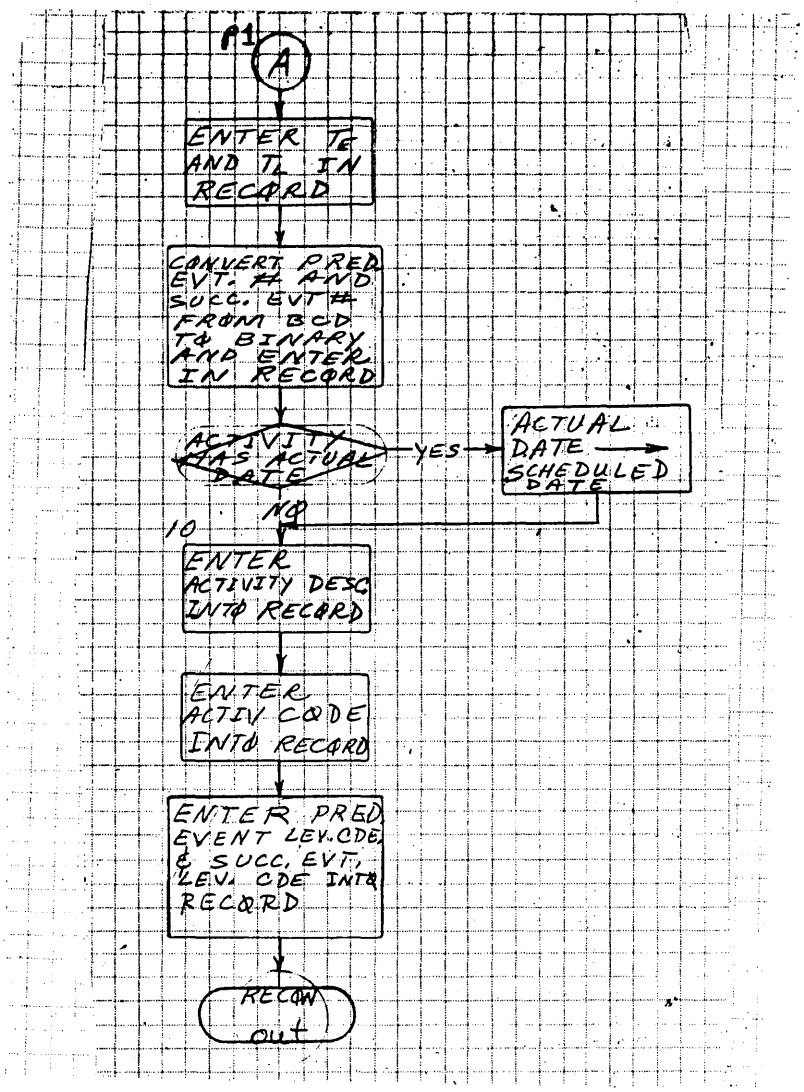


Figure 84. RECON Subroutine Flowchart (cont'd) P2

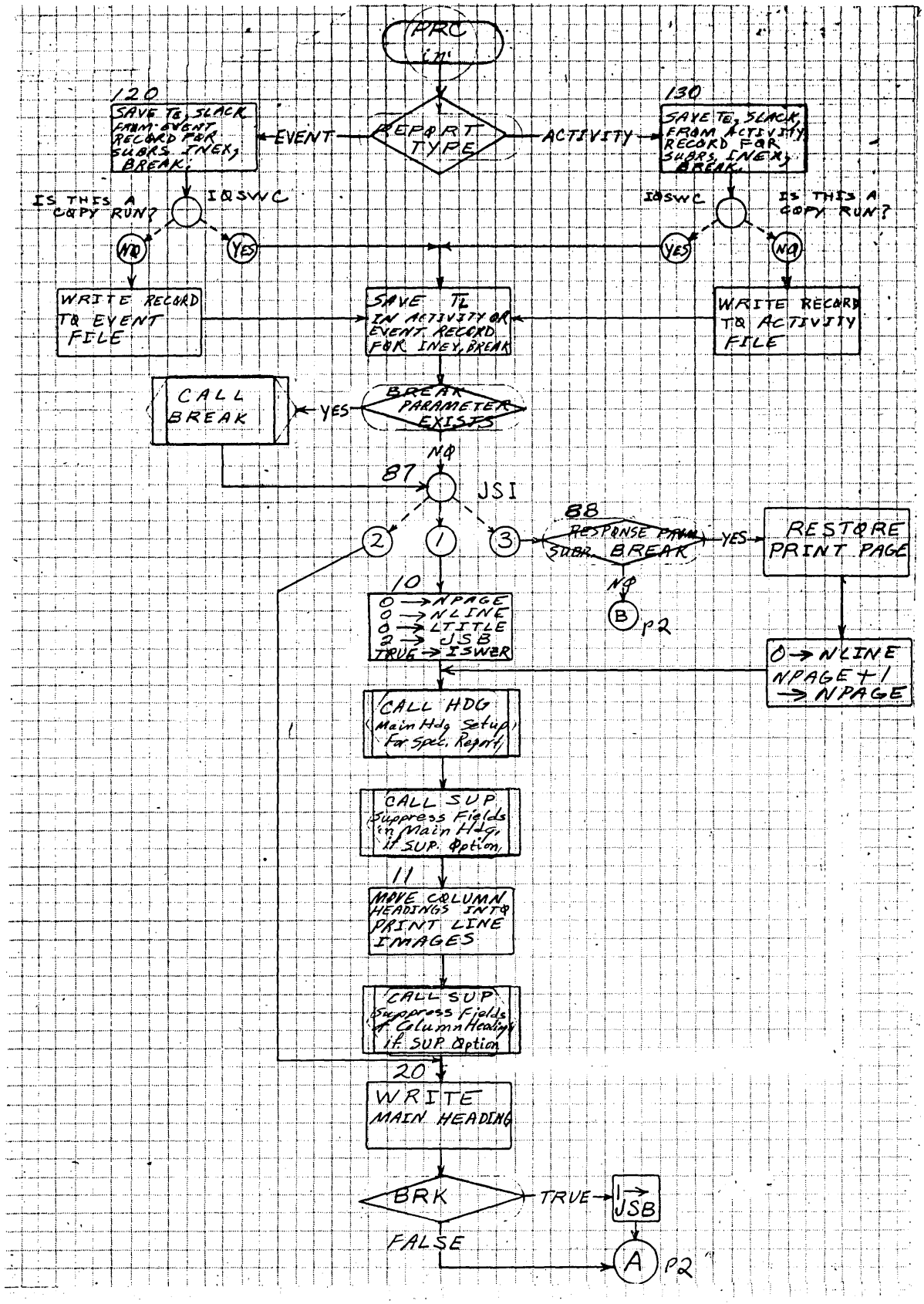


Figure 85. PRC Subroutine Flowchart P1

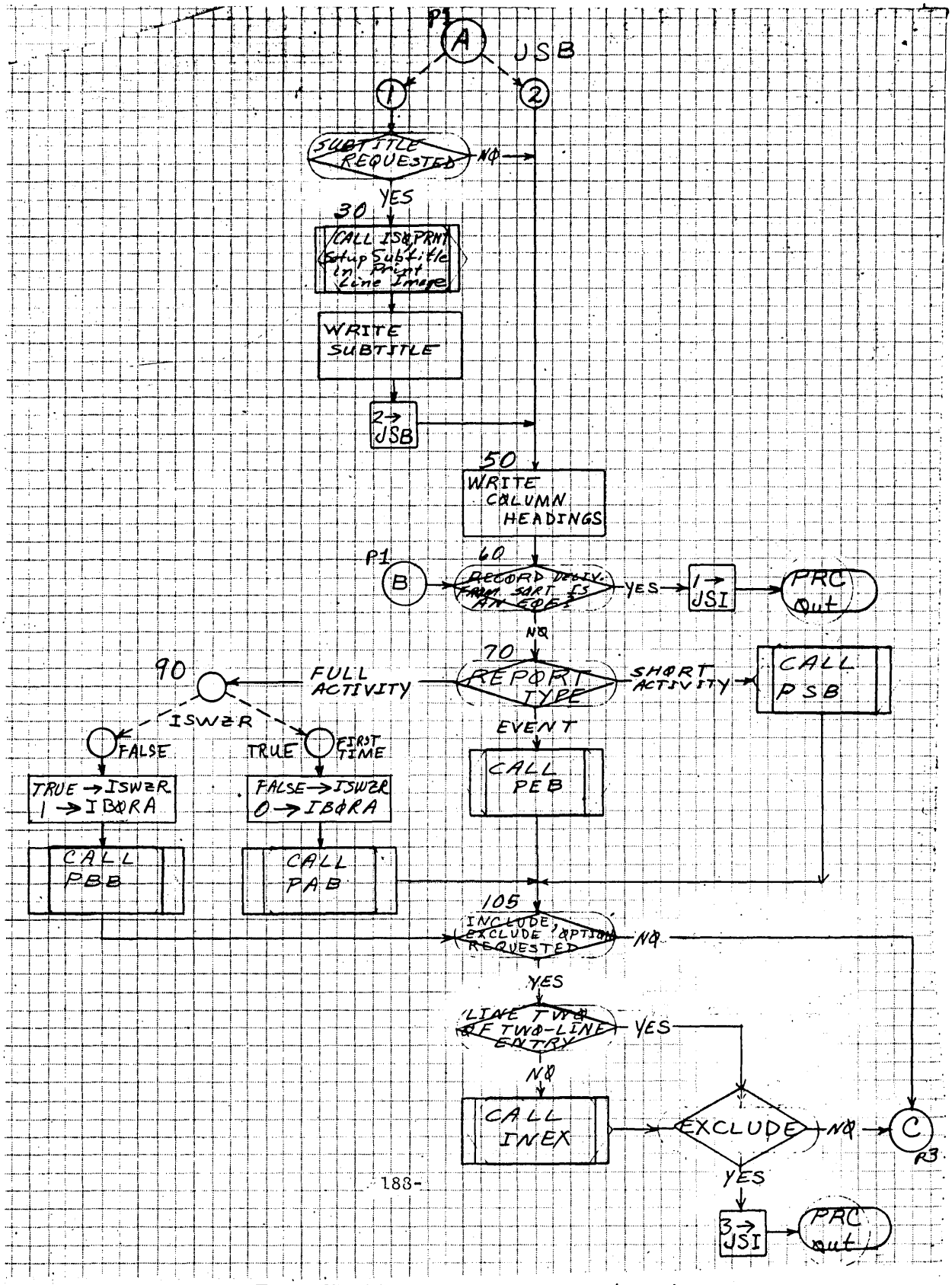


Figure 85. PRC Subroutine Flowchart (cont'd) P2

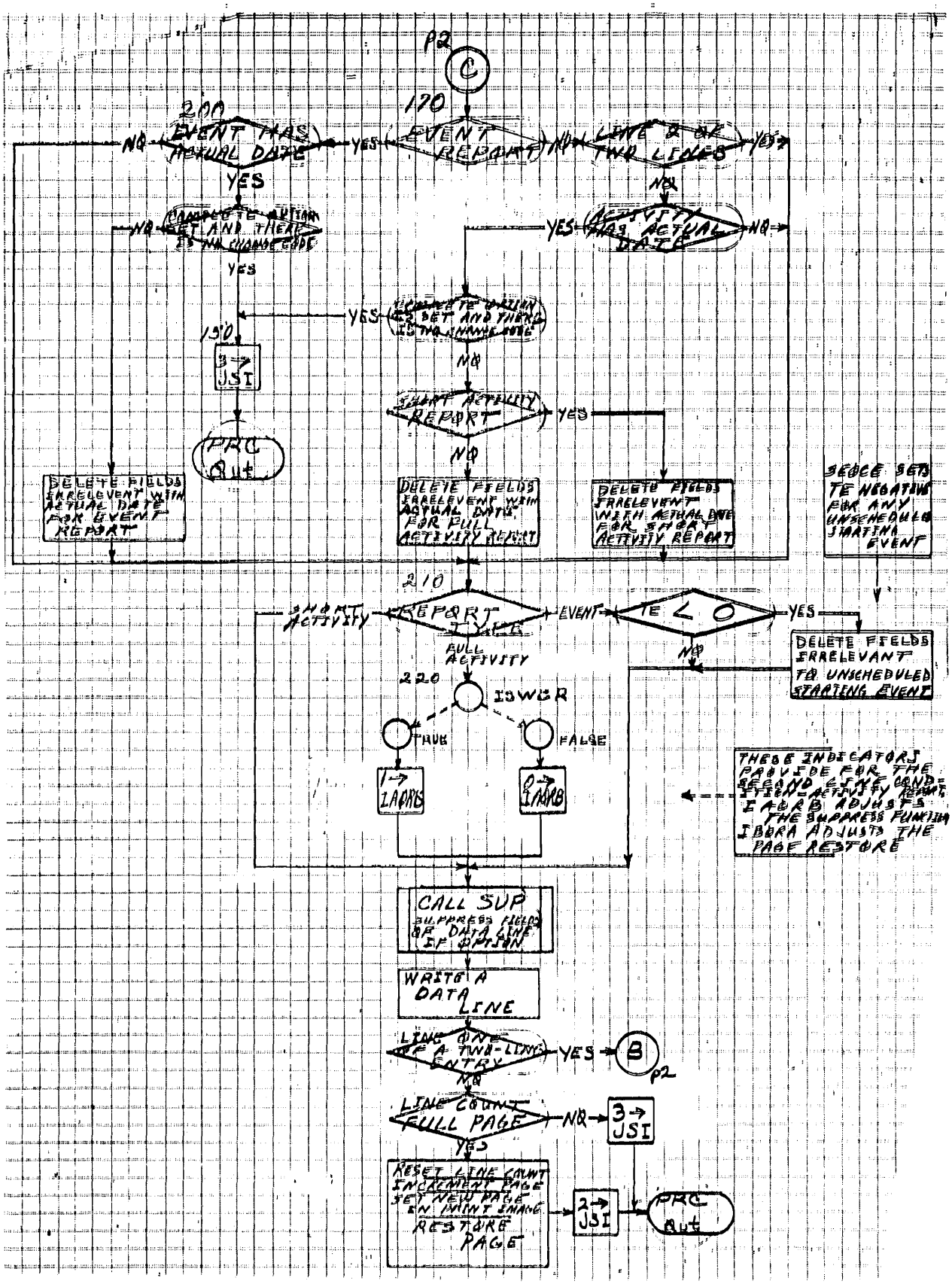


Figure 86. PRC Subroutine Flowchart (cont'd) P3

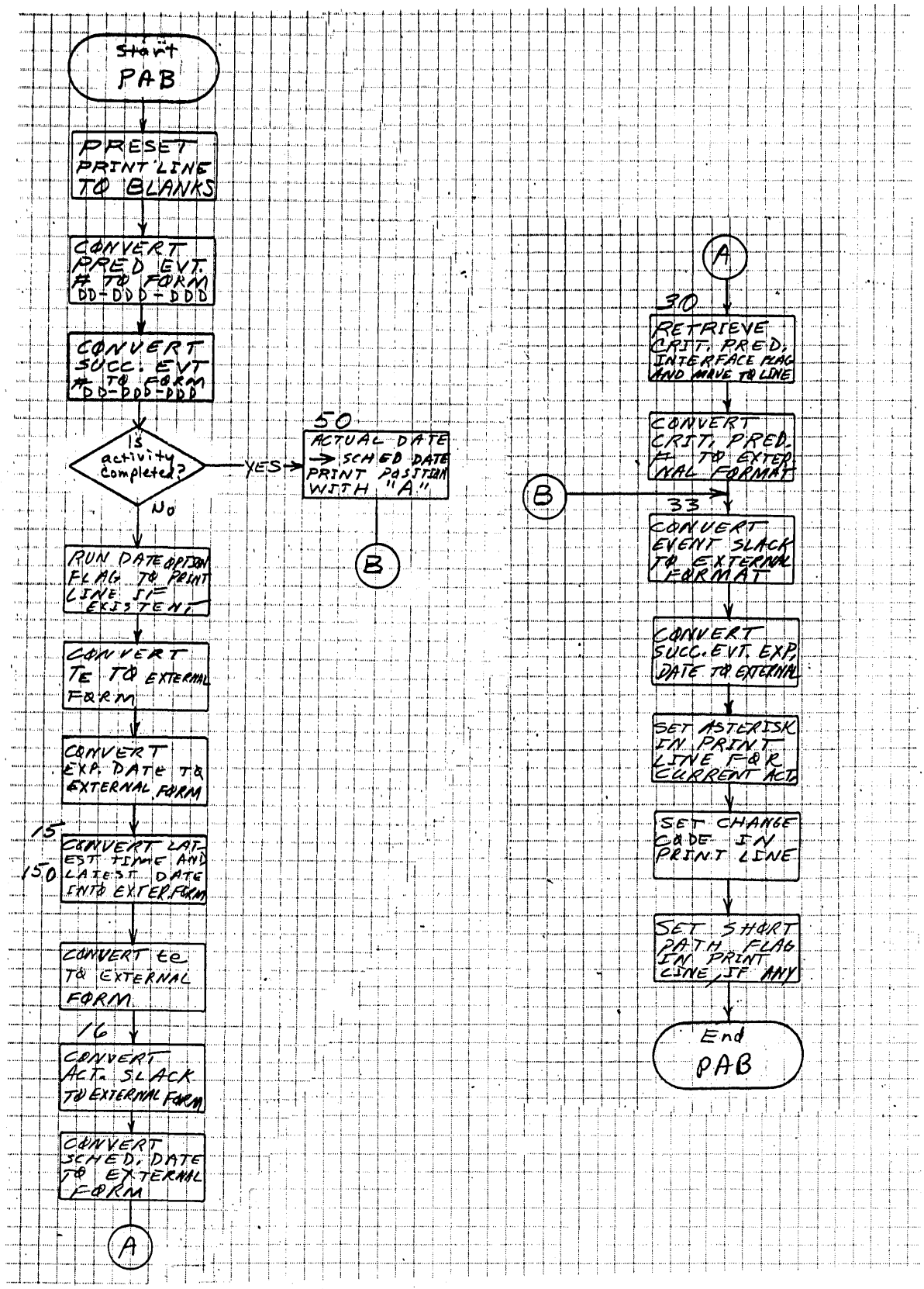


Figure 86. PAB Subroutine Flowchart

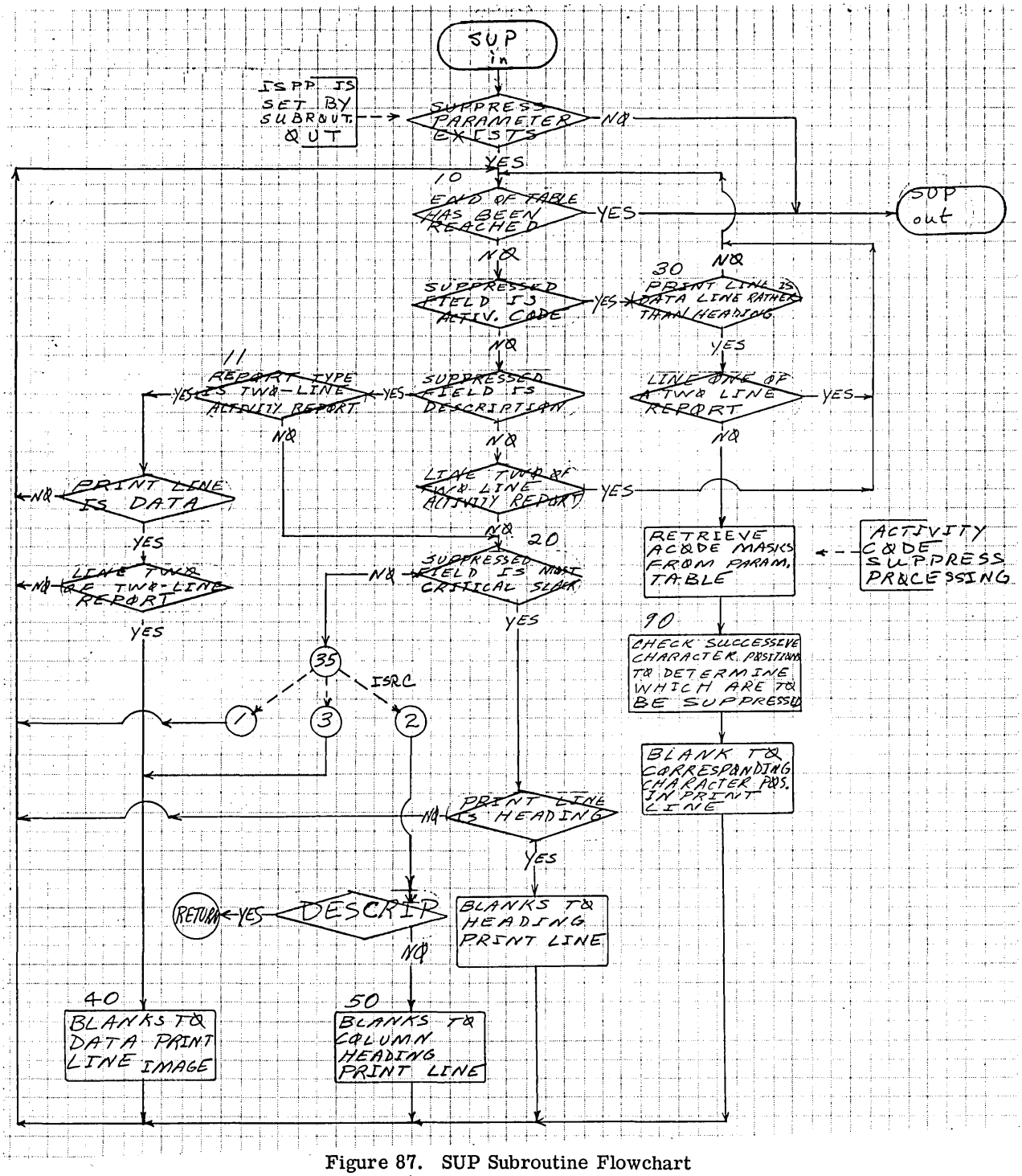
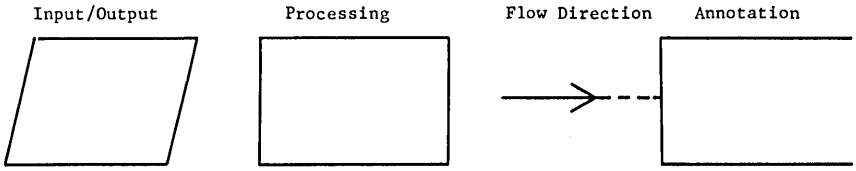


Figure 87. SUP Subroutine Flowchart

APPENDIX

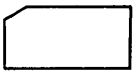
ASA STANDARD FLOWCHART SYMBOLS

Basic Symbols

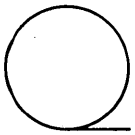


Specialized Input/Output Symbols

Punched Card

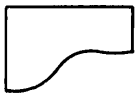


Magnetic Tape

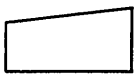


Punched Tape

Document

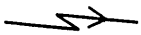


Manual Input

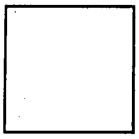


Display

Communication Link

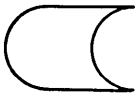


Auxiliary Operation

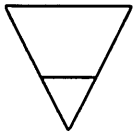


Specialized Processing Symbols

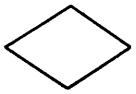
On-line Storage



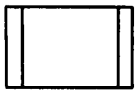
Off-line Storage



Decision



Predefined Process

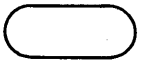


Manual Operation

Additional Symbols

Connector

Terminal



KEY-WORD INDEX

The key-word index is formed by permuting titles, paragraph names, descriptive phrases, and figure names, putting each key word in the index position in the center of the page. The rest of the phrase appears on either side of the key word.

A phrase which is too long to be printed to the left of the key word is truncated at the left margin. The beginning of the phrase is placed on the right side of the page, preceded by an asterisk.

DOCUMENT REVIEW SHEET

TITLE: GE-625/635 PERT/TIME

CPB #: 1192A

FROM:

Name: _____

Position: _____

Address: _____

_____

Comments concerning this publication are solicited for use in improving future editions. Please provide any recommended additions, deletions, corrections, or other information you deem necessary for improving this manual. The following space is provided for your comments.

COMMENTS: _____

Please cut along this line

NO POSTAGE NECESSARY IF MAILED IN U.S.A.
Fold on two lines shown on reverse
side, staple, and mail.

STAPLE STAPLE

FOLD

FIRST CLASS
PERMIT, No. 4332
PHOENIX, ARIZONA

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY

GENERAL ELECTRIC COMPANY
COMPUTER EQUIPMENT DEPARTMENT
13430 NORTH BLACK CANYON HIGHWAY
PHOENIX, ARIZONA - 85029

ATTENTION: DOCUMENTATION STANDARDS AND PUBLICATIONS B-90

FOLD

Progress Is Our Most Important Product

GENERAL ELECTRIC

INFORMATION SYSTEMS DIVISION

LITHO U.S.A.