DECEMBER 1968

AD 679270

SYSTEM AND SOFTWARE SIMULATOR

VOLUME II

This document has been approved for public release and sale; its distribution is unlimited

UNITED STATES ARMY
COMPUTER SYSTEMS SUPPORT
AND EVALUATION COMMAND
WASHINGTON, D.C. 20310

DEC 19 1968

CLEARINGHOUSE

3200.8 (Att 1 to Encl 1)
Mar 7, 66

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) C-E-I-R, Inc. 5272 River Road Washington, D. C.	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

SYSTEM AND SOFTWARE SIMULATOR VOLUME II

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Working Paper

5. AUTHOR(S) (First name, middle initial, last name)

Leo J. Cohen

6. REPORT DATE December 1968	7a. TOTAL NO. OF PAGES 445	7b. NO. OF REFS
8a. CONTRACT OR GRANT NO. DA 49-083 OSA-3306 b. PROJECT NO. c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	

10. DISTRIBUTION STATEMENT

Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES This documentation has been assembled and released by the sponsoring activity	12. SPONSORING MILITARY ACTIVITY USA Computer Systems Support and Evaluation Command Washington, D. C.

13. ABSTRACT

The System and Software Simulator (S3) is a digital event simulator written in FORTRAN IV and designed to perform simulations of computer systems hardware and software and of the workload being applied to the system.

This and the other three volumes constitute the complete documentation available on S3. Volume I describes the inputs, outputs, methods and capabilities of S3. Volume II contains the flowcharts, narrative description of the flowcharts, layouts and descriptions of the tables utilized by S3. Volume III contains descriptions of the assembly language used for preparation of input to S3, of the macro capability of the assembler, and of the modifications made to S3 to provide additional output data. Volume IV is the program documentation on the internal workings of the assembler. It consists of table descriptions, flow charts and narratives, and file descriptions.

These volumes are a collection of documentation delivered under two separate contracts. They have not been edited and as such are considered working papers.

DD FORM 1 NOV 1473

Security Classification

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Simulation						
Computer Systems						
Operating Systems						
Evaluation						
Timing						
Dynamic Event Simulation						

Security Classification

WORKING PAPER

The documentation on the System and Software Simulator (S3) contained in this and the other three volumes is considered a working paper and no claims are made as to its accuracy. There has been no attempt to edit the information. Discrepancies and inconsistencies are known to exist.

This information is being released as a service to interested parties and to satisfy the numerous requests for information on S3.

The documentation of S3 is contained in four volumes. Volumes I and II are contract end items delivered under contract number DA-49-083 OSA-3306 and contain the technical descriptions of S3. Volume I describes the inputs, outputs, methods and capabilities of S3. Volume II contains the flowcharts, narrative description of the flowcharts, layouts and descriptions of the tables utilized by S3.

Volumes III and IV contain the documentation delivered as contract end items under contract number DAAB09-68-C-0118. Volume III contains descriptions of the assembly language used for preparation of input to S3, of the macro capability of the assembler, and of the modifications made to S3 to provide additional output data. Volume IV is the program documentation on the internal workings of the assembler. It consists of table descriptions, flow charts and narratives, and file descriptions.

TABLE OF CONTENTS

Section I	Narrative Description of S^3 Flow Charts
Section II	S^3 Flow Charts
Section III	S^3 Tables and Table Descriptions

WORKING PAPER

SECTION I
NARRATIVE DESCRIPTION

of

S^3 FLOW CHARTS

PREPARED FOR

DEPARTMENT OF THE ARMY

UNDER

DEFENSE SUPPLY SERVICE
CONTRACT
#DA 49-083-OSA-3306

BY

C-E-I-R, Incorporated
5272 River Road
Washington, D. C. 20016

WORKING PAPER

INDEX

Description	Page
MAIN PROGRAM	1
SUBROUTINE SM1	2
SUBROUTINE ZERO	3
SUBROUTINES C1 to C35	5
SUBROUTINE SM2	7
SUBROUTINE ST1 QUEUE STATISTICS	9
SUBROUTINE ST2 MEMORY STATISTICS	11
SUBROUTINE ST3 FILE STATISTICS	13
SUBROUTINE ST4 DEVICE STATISTICS	15
SUBROUTINE ST5 CONTROL UNIT STATISTICS	17
SUBROUTINE ST6 CHANNEL STATISTICS	18
SUBROUTINE ST7 CPU STATISTICS	20
SUBROUTINE ST8 QUEUE TRANSACTION ANALYSIS	22
SUBROUTINE STAT WORKER ROUTINE STATISTICS	24
SUBROUTINE TDUMP	27
SUBROUTINE SNAP	28
SUBROUTINE S951	29
SUBROUTINE PCIS	31
SUBROUTINES DT1 to DT25	32
SUBROUTINE S1 TRANSFER	34
SUBROUTINE S2 TRANSFER ON PROBABILITY	35

INDEX

Description	Page
SUBROUTINE S3 READ	36
SUBROUTINE S4 WRITE	37
SUBROUTINE S5 FUNCTION	38
SUBROUTINE S6 END OF FILE	39
SUBROUTINE S7 SUBROUTINE	41
SUBROUTINE S8 EXIT	42
SUBROUTINE S9 LOOP	43
SUBROUTINE S10 MOVE	45
SUBROUTINE S11 MOVE AND EDIT	46
SUBROUTINE S12 COMPUTE	47
SUBROUTINE S13 MATH	48
SUBROUTINE S14 OPEN	50
SUBROUTINE S15 CLOSE	52
SUBROUTINE S16 TERMINATE	54
SUBROUTINE S17 PRINT	56
SUBROUTINE S18 CALL	57
SUBROUTINE S19 EXEC	58
SUBROUTINE S20 LOOP ON PROBABILITY	59
SUBROUTINE S31 MEMORY	62
SUBROUTINE S32 ALLOCATE	68
SUBROUTINE S33 DEALLOCATE	71

INDEX

Description	Page
SUBROUTINE S34 PACK	73
SUBROUTINE S35 EXAMINE-FIRST	74
SUBROUTINE S36 EXAMINE-NEXT	75
SUBROUTINE S37 EXAMINE-LAST	76
SUBROUTINE S38 PLACE	77
SUBROUTINE S39 SELECT	80
SUBROUTINE S40 BUFF	83
SUBROUTINE S41 SEEK	85
SUBROUTINE S42 IOREADY	87
SUBROUTINE S43 IOADVANCE	91
SUBROUTINE S44 IOTERM	94
SUBROUTINE S45 SET SWITCH	96
SUBROUTINE S46 RESET SWITCH	97
SUBROUTINE S47 TEST SWITCH	98
SUBROUTINE S48 INTERRUPT	99
SUBROUTINE S49 DISABLE	100
SUBROUTINE S50 ENABLE	101
SUBROUTINE S51 CLOCK	102
SUBROUTINE S52 RETURN	103
SUBROUTINE S53 ACTIVATE	106
SUBROUTINE S54 RECEIVE	108
SUBROUTINE S55 CYCLE	110

INDEX

Description	Page
SUBROUTINE S56 DESTROY	111
SUBROUTINE S57 PERIPHERAL	114
SUBROUTINE S59 MATCH	117
SUBROUTINE S61 CALLOUT	119
SUBROUTINE S62 PRIORITY	120
SUBROUTINE S101 ERROR HANDLER	122
SUBROUTINE S102 PUT IOT1 ON FEC	123
SUBROUTINE S103 PUT CLOCK	125
SUBROUTINE S104 FEC MANIPULATOR	126
SUBROUTINE S110 INTERRUPT HANDLER	133
SUBROUTINE S201 PUT FEC	135
SUBROUTINE S202 REORGANIZE FEC	136
SUBROUTINE S203 GENERATE TRANSACTION ITEM	137
SUBROUTINE S204 GENERATE TO TRANSACTION	139
SUBROUTINE S210 UNLOAD MEMORY STATISTICS	140
SUBROUTINE S211 SET MEMORY LOAD	141
SUBROUTINE S212 RANDOM NUMBER GENERATOR	142

MAIN PROGRAM

The MAIN program receives control after S3 is loaded into memory, MAIN then calls SM1 which initializes all of the tables in common. When SM1 is complete, it returns control to MAIN. MAIN then calls SM2 which executes the simulation model. When all statistic intervals have been completed, SM2 returns control to MAIN. MAIN then calls the two final statistic subroutines STAT1 and STAT. Upon completion of STAT the simulation is terminated.

SUBROUTINE SM1

SM1 receives control from MAIN. It begins by calling the card read routine C1-C34. These routines read the card type identified by the number after the "C". Each of these "C" routines places the data read into the appropriate table in common.

Once all of the hardware definition, configuration, and system parameter cards have been read, SM1 reads the worker routine cards, card type 80, and stores the information into the worker routine base table, file vector table, and the statement table. In addition, for each operating system worker routine received, the interrupt vector in the appropriate CPU item table is filled in.

If any errors in the worker routines are detected, the run is terminated without returning control to MAIN. If no errors are detected, control is returned to MAIN.

Subroutine ZERO

The ZERO Subroutine is used to initialize all of the tables for the simulator. The future events chain table is initialized to all zeroes. The CPU item table is initialized to all zeroes. The first word in each entry in the transaction item table is initialized to minus 1. All other words in each entry in the transaction item table are initialized to zero. The first word for each entry in the IO transaction item table is initialized to minus 1. The remainder of each entry is initialized to zero. The worker routine base table is initialized to all zeroes. The memory table is initialized to all zeroes. The page table is initialized to all zeroes. The load class table is initialized to all zeroes. The run class table is initialized to all zeroes. The queue table is initialized to all zeroes. The queue entry table is initialized to all zeroes. The file set table is initialized to all zeroes. The device set table is

initialized to all zeroes. The availability table is initialized to all zeroes. The channel table is initialized to all zeroes. The control unit table is initialized to all zeroes. The device class table is initialized to all zeroes. The general simulation table is first initialized to all zeroes. Within the general simulation table the Poisson Distribution table is initialized to the values described in the coding for the zero subroutine. In addition, the size of each table used in the simulator is entered into the general simulation table. The function table is initialized to all zeroes. The statement is initialized to all zeroes. Each entry in the switch table is initialized to minus 1. The to-from table is initialized to all zeroes. The mark time table is initialized to all zeroes. A RETURN is then executed transferring control back to the calling subroutine.

Subroutines C1 to C35

Subroutines C1 to C35 are used to read card types one to 35 into the simulator and to build the appropriate tables.

Since each of these subroutines follow the same logic pattern a single description will be given for all of them.

Each of these subroutines is called in sequence by SM1. The subroutine then reads one record from the input file. If the card type for the record read does not agree with the card type expected by the current subroutine and the expected card type is required, a diagnostic message is issued and the simulation terminated. If the card type for the record read does not agree with the card type expected but the expected card type is not required by the simulator, the input file is BACKSPACED and a RETURN is executed transferring control back to SM1 which will then call the next READ routine.

If the record read does contain the card type expected by the current subroutine error checking is performed and the data is moved into the appropriate table. Once a card type which cannot be processed by the current READ routine is encountered the input file is BACKSPACED and a RETURN is executed, transferring control back to SM1.

SUBROUTINE SM2

When SM2 receives control from MAIN, after SM1 has been executed, it sets the current CPU number to 1, and goes to the Cycle Handling subroutine. The cycle handler will remove the item at the top of the future events chain and begin the simulation run.

Once the simulation has begun, SM2 acts like a hardware CPU timing loop. As each statement accessed by SM2 is executed, control passes to the top of the SM2 loop. The loop consists of the following operations:

1) Test for error detected in the last statement.
2) Test for statistic time.
3) Access next instruction code (from NSI of COT).
4) Test for valid instruction code.
5) Access all operands.
6) Go to statement calling sequence.

Each statement which can be executed by the

simulator has a calling sequence in SM2. For a simple statement which does not generate an interrupt, or involve the future events chain, the calling sequence consists of a call to the "S" routine, which is numbered with the statement number to be executed. Following this call, is a transfer to the top of the SM2 loop. This transfer will be executed upon return from the "S" subroutine.

Statements which do generate an interrupt or require access to the future events chain will have a call to a "S" subroutine in order to handle the operation code, followed by one or more calls to second level subroutines numbered S1XX. The S1XX subroutines will handle the interrupt manipulation or accessing of the future events chain. Every statement calling sequence is ended by a transfer to the top of the SM2 loop.

SUBROUTINE ST1 QUEUE STATISTICS

The QUEUE STATISTICS subroutine examines the queue statistic switch in the general simulation table. If the switch is set off, a RETURN is executed immediately, transferring control back to the calling subroutine. If the queue statistics are to be printed, a page title and line title are printed immediately.

For each queue utilized by the current simulation, the queue load statistic is now closed out. This operation consists of subtracting the last queue reference time from the current simulator time and multiplying the result by the current number of entries in the queue. This value is then added to the total queue load and the last reference time is set equal to the current simulator time. Since the queue load field has a value in entry microseconds, the average number of entries is obtained by dividing the queue load by the current simulator time. The number of entries allowed in a queue is determined by subtracting the first entry

address from the last entry address and adding one. The percentage utilization of a queue is determined by dividing the average number of entries by the maximum number of entries and converting this to a percentage. The average wait time on a queue is calculated by dividing the queue load by the number of items placed on the queue.

These calculations are performed for each queue which has been utilized in the current simulation. When all queue statistics have been printed, a RETURN is executed, transferring control back to the calling subroutine.

SUBROUTINE ST2 MEMORY STATISTICS

The MEMORY STATISTICS subroutine accesses the memory statistic switch in the general simulation table. If this switch is off, a RETURN is executed immediately, transferring control back to the calling subroutine. If memory statistics are to be printed, the page title and line heading are printed immediately. A line of statistics is then printed for each memory used in the current simulation.

Before printing statistics for any memory, the memory load is first closed out. The memory load is closed out by subtracting the last memory reference time from the current simulator time, and multiplying this difference by the current number of pages busy. This value is then added to the previous value of the memory load to come up with the total memory load. At this point, the last memory reference time is reset to the current simulator time. The memory load field contains a value in page microseconds. Therefore, the

average number of pages busy is calculated by dividing the memory load by the current simulator time. The total number of pages in any given memory is calculated by subtracting the low page entry address from the high page entry address and adding 1. The percentage of utilization for a given memory is calculated by dividing the average number of pages busy by the total number of pages in the memory and converting this to a percentage.

When the memory statistics have been printed out for every memory utilized in a given simulation, a RETURN is executed, transferring control back to the calling subroutine.

SUBROUTINE ST3 FILE STATISTICS

The FILE STATISTICS subroutine examines the file statistics switch in the general simulation table. If this switch is off, a RETURN is executed immediately. If file statistics are to be printed, a page heading and line heading are printed immediately. For each file utilized in the current simulation, a line of statistics is printed.

If a file for which statistics are to be printed is currently seized, the total seized time field is now closed out. This operation is performed by subtracting the seized start time from the current simulator time and adding this to the total time seized. When this has been performed, the seized start time is set equal to the current simulator time.

The following conversions are performed, before printing out statistics for a single file: the seized time is converted to minutes; the percentage seized time is calculated by dividing the amount of time the

file would seize by the current time and multiplying the result by 100; the read/write time is converted to minutes, the percentage read/write time is calculated by dividing the read/write time by the current simulator time and multiplying the result by 100, the percentage of seize time used is calculated by dividing the total read/write time by the total time the file was seized and multiplying the result by 100. Once statistics have been printed out for each file utilized in this simulation, a RETURN is executed, transferring control back to the calling subroutine.

SUBROUTINE ST4 DEVICE STATISTICS

The DEVICE STATISTICS subroutine tests the device statistics switch in the general simulation table. If the switch is off, a RETURN is executed immediately. If device statistics are to be printed, the page heading and line heading are printed out immediately. There is one line of device statistics printed for each device which has been utilized during the current simulation.

Before printing out a line of device statistics, the following calculations are performed: the seek time is converted to minutes; the percentage seek time is calculated by dividing the total seek time by current simulator time and multiplying the result by 100; the penalty time is converted to minutes; the percentage penalty time is calculated by dividing the total penalty time by the current simulator time and multiplying the result by 100; the total device use time is converted to minutes; the percentage use time is calculated by dividing

the total use time by the current simulator time and multiplying the result by 100; the percentage of use time spent seeking on this device is calculated by dividing the total time seeking by the use time and multiplying the result by 100; the percentage of use time spent performing penalty operations for this device is calculated by dividing the total penalty time by the total use time and multiplying the results by 100. When the statistics have been printed out for each device utilized during this simulation, a RETURN is executed and control is transferred back to the calling subroutine.

SUBROUTINE ST5 CONTROL UNIT STATISTICS

The CONTROL UNIT STATISTICS subroutine accesses the control unit statistics switch in the general simulation table. If this switch is off a RETURN is executed immediately, transferring control back to the calling subroutine. If control unit statistics are to be printed, the page heading and line heading are printed immediately. There is one line of statistics printed for each control unit utilized during the current simulation.

Before a line of control unit statistics are printed, the percentage utilization is calculated by dividing the total time used by the current simulator time and multiplying the result by 100. In addition, the total time used by this control unit is converted to minutes from microseconds. Once statistics have been printed out for each control unit utilized in current simulation a RETURN is executed, transferring control back to the calling subroutine.

SUBROUTINE ST6 CHANNEL STATISTICS

The CHANNEL STATISTICS subroutine examines the channel statistics switch in the general simulation table. If the channel statistics switch is off, a return is executed immediately, transferring control back to the calling subroutine. If channel statistics are to be printed, the page and line headings are printed immediately. There is one line of channel statistics printed for each channel utilized in the current simulation.

A single channel in the channel table may consist of 1 or 2 entries. A selector channel utilizes a single entry and a multiplexor channel utilizes two entries. For a multiplexor channel, the first entry contains statistics on the multiplexor channel's use in burst mode. The second entry contains information for the channel's use in multiplexor mode. The following statistics are calculated before printing each line of channel statistics: the selector use time is converted

to minutes; the percentage of the available time used by a selector channel or by a multiplexor channel in burst mode is calculated by dividing the total time used by the current simulator time; the percentage of selector or burst mode utilization is calculated by dividing the number of characters transmitted by the total number of characters which could have been transmitted. If the channel for which statistics are being printed is a selector channel, the multiplexor channel statistics are set equal to zero and the percentage total utilization is set equal to the percentage selector utilization. For a multiplexor channel, the percentage multiplexor utilization is calculated by dividing the total number of characters transmitted by the total number of characters which could have been transmitted utilizing the maximum multiplexor transfer rate. The total percentage utilization for this channel is then calculated by adding the percentage burst mode utilization to the percentage multiplexor utilization. Once statistics have been printed for each channel utilized during the current simulation, a RETURN is executed transferring control back to the calling subroutine.

SUBROUTINE ST7 CPU STATISTICS

The CPU STATISTICS subroutine examines the CPU's statistics switch in the general simulation table. If the CPU statistics switch is off, a RETURN is executed immediately, transferring control back to the calling subroutine. If CPU statistics are to be printed, the page and line headings are printed immediately. A single line of CPU's statistics are printed for each CPU utilized during the current simulation.

Before a line of statistics is printed, the following calculations are performed: the operating system transaction item queue time, the operating system IO transaction queue time, the CPU cycle time and the operating system time are all converted to minutes; the total amount of CPU time available for worker routine transaction items is calculated by subtracting the total cycle time plus the total operating system time for the current CPU from the current simulator time and then converting the result

to minutes; the percentage utilization for a CPU is calculated by dividing the total time available for worker routines by the current simulator time and multiplying the result by 100.

It should be noted here that there may be a slight inaccuracy in the total amount of time allocated to worker routine transaction items. Under normal circumstances where the simulator time at the end of a statistics interval is very large compared to a single operating system transaction item's time, the fact that there may be one or more operating system transactions still in the system, whose total CPU time has not yet been accounted for, should make little difference to the statistics.

After a line of statistics has been printed out for each CPU utilized during the current simulation, a summary line is printed. The summary line contains the averages from each of the detail lines. Once the summary line has been printed, a return is executed and control is transferred back to the calling subroutine.

SUBROUTINE ST8 QUEUE TRANSACTION ANALYSIS

The QUEUE TRANSACTION ANALYSIS subroutine examines the queue transaction analysis switch in the general simulation table. If the switch is off, a return is executed immediately transferring control back to the calling subroutine.

If a queue transaction analysis is to be performed, the queue transaction file FORTRAN unit number 18 is rewound. During this rewind operation, the queue analysis table and the IO table are zeroed. The queue transaction file is read once for each queue used during the current simulation. For each of these queues page and line headings are printed.

The queue transaction file consists of blocks of 100 records, where each record contains the information about a single transaction's stay on some queue. A record is created each time a select statement is executed by any operating system. As the queue transaction file is read for each queue, an accumulation of the duration of

of time spent by transaction items and IO transaction items for a single worker routine is accumulated. The items accumulated are only for the queue currently being analyzed.

Once an end of file has been reached, all the entries in the analysis table are converted to seconds and a table is printed out for a single queue. This table will contain one line for each worker routine which had a transaction or IO transaction on the current queue being examined. After all of the queues have been analyzed, a RETURN is executed and control is transferred back to the calling subroutine.

SUBROUTINE STAT WORKER ROUTINE STATISTICS

The WORKER ROUTINE STATISTIC subroutine first initializes the worker routine statistics table. The worker routine statistics table is designed to contain minimum average and maximum values for all of the information about any worker routine in the system. The worker routine transaction termination file is written each time a transaction item is destroyed. The STAT subroutine reads back this file in the same order in which it was created and prints out a transaction termination list.

The transaction termination list consists of the following information: the statistical interval during which this transaction item was destroyed, (statistic intervals are numbered starting with zero); the transaction item identification number, consisting of the worker routine number plus a four digit transaction item sequence number; the time at which this transaction item was destroyed; the time that the

transaction was created; the total amount of CPU time utilized by this transaction; the total amount of time transaction items spent on queues; and the total amount of time which IO transaction items generated by this transaction spent on queues. In addition, there is a record of the total number of OP operations performed on up to 20 files for each transaction.

As the information from each transaction item termination is printed out, it is also accumulated into the worker routines statistics table. Once all transaction terminations have been printed, the worker routine statistic table is analyzed. There is one page of worker routine statistics printed for each worker routine which had a transaction terminate during the course of this simulation. The following statistics are printed from the worker routine statistics table: the number of transactions generated and the number of transactions called; the minimum, average, and maximum for each of the following values - turnaround time, CPU time, transaction item queue time, IO transaction item queue time. In addition, the minimum

average and maximum number of IO operations on each of the 20 files available to a worker routine are also printed. After all of the worker routine statistics have been printed, the message "END OF SIMULATION" is put out on the print file and the EXIT routine is called, terminating the simulation.

Subroutine TDUMP

The TDUMP subroutine calls each of the individual table dump subroutines and then returns control to the calling subroutine.

Subroutine SNAP

The SNAP subroutine first examines the SNAP subroutine control switch in the general simulation table. If this switch is off, a RETURN is executed immediately and control is transferred back to the calling subroutine.

If the SNAP control switch is on the SNAP counter in the general simulation table is incremented by one and a print out of the current simulator status is provided. The printout provided contains information about the current CPU number, the current operating transaction, the current available transaction, and the current IO transaction item. Once this printout has been provided, a RETURN is executed transferring control back to the calling subroutine.

Subroutine S951

Subroutine S951 first tests to determine if there is a valid current operating CPU. If no valid current operating CPU exists, a RETURN is executed immediately transferring control back to the calling subroutine. If a valid current operating CPU exists, the S951 subroutine next tests the trace control switch in the general simulation table. If the trace control switch is off, a RETURN is executed immediately transferring control back to the calling subroutine. If the trace control switch is on, a line of print is generated providing the current status within the simulator.

The information provided by subroutine S951 consists of the following items:

1. The current operating CPU number.
2. The current operating transaction number.
3. The current available transaction number.
4. The current IO transaction item number.

5. The current operation code.

6. The first operand.

7. The second operand.

8. The third operand.

9. The fourth operand.

10. The current simulator time.

11. The current future events chain code.

12. The current future events chain item.

13. The current subroutine number.

14. The current interrupt number.

15. The next sequential instruction for the COT.

16. The next sequential instruction for the AT.

17. The next sequential instruction code from the general simulation table.

18. The current error code.

19. The current program number.

20. The current random number.

Once the above information has been generated a RETURN instruction is executed transferring control back to the calling subroutine.

Subroutine POIS

The POIS subroutine first calls the random number generator. Upon return of control from the random number generator the random number will be found in the general simulation table. The range of the random number found will be from 0 to 99. The random number provided is then rounded down to the nearest multiple of 5 and then divided by 5. The number thus obtained is then added to the value 81 and an entry is extracted from the Poisson Distribution function found in the general simulation table. The value obtained from the Poisson Distribution table is then multiplied by the value supplied upon entry to the POIS subroutine. The result of this multiplication is then placed in the output parameter and a RETURN is executed, transferring control back to the calling subroutine.

Subroutines DT1 to DT25

Subroutines DT1 to DT25 are used to provide a printout of each of the tables from table 1 to table 25.

Since each of these subroutines follow the same logic pattern a single description will be given for all of them.

Each of these subroutines is called in sequence by TDUMP. The subroutine then tests a switch to determine if the user has indicated the current table printout is not of interest. If the switch is on a RETURN is executed transferring control back to TDUMP. If the switch if off indicating the user does want the printout of the current table the DR subroutine provides that printout from the information currently in the table by means of a FORMAT statement. In general, the DT subroutine does not print all possible entries from a table, but rather stops printing when there are no more entries in the table

being used by the current simulation. Once all possible entries for a given table have been printed a RETURN is executed transferring control back to TDUMP.

SUBROUTINE S1 TRANSFER

The TRANSFER subroutine picks up the current operating CPU from the general simulation table. It then accesses the current operating transaction for that CPU. The next sequential instruction is then replaced in the current operating transaction with the new transfer address. Upon completion of this operation control is returned to SM2.

SUBROUTINE S2 TRANSFER ON PROBABILITY

The TRANSFER ON PROBABILITY subroutine calls the random number generator, which will return a number within the range from 0 to 99. If the random number generated is less than the percentage probability for transferring, control is passed to the transfer subroutine which will update the next sequential instruction in the current operating transaction. If the random number generated is equal to, or greater than the percentage probability of transfer, control will be returned to SM2 without altering the next sequential instruction.

SUBROUTINE S3 READ

The READ subroutine accesses the current operating transaction within th· current operating CPU. The ordinal file number specified by the READ statement is then stored into this transaction as the current file request. The ordinal file is stored as a positive number, because the IO operation requested is an input operation. In addition, the interrupt code in the general simulation table is set to 2, in order to generate a Read/Write interrupt. Control is then returned to SM2. When SM2 receives control it will transfer control to Sil0, the Interrupt Handler.

SUBROUTINE S4 WRITE

The WRITE subroutine performs the exact same functions as the READ subroutine, with the exceptions that, the current file request is stored as a negative number in the current operating transaction, to indicate that the IO operation requested is an output operation. When control is returned to SM2, the interrupt subroutine S110 will be called just as for the READ statement.

SUBROUTINE S5 FUNCTION

The FUNCTION subroutine accesses the current operating transaction for the current operating CPU. This transaction item address is placed into the general simulation table as an input argument to S104, the Generate IO Transaction Item subroutine. Subroutine S104 is then called. When control is returned to the FUNCTION subroutine, the general simulation table will contain the address of the IO transaction item just generated. The FUNCTION subroutine then sets the IO transaction item type code equal to 5 to indicate a function IO transaction, and stores the Function Number from the FUNCTION statement into the IO transaction. The interrupt code in the general simulation table is then set to 3 to cause a function interrupt into the operating system. The next IO transaction word in the current operating CPU is then set to the address of the IO transaction item generated. Control is then returned to SM2, which will next call S110, the Interrupt Handler.

SUBROUTINE S6 END OF FILE

The END OF FILE subroutine accesses the current operating transaction for the current operating CPU. From part one of this transaction the address of the worker routine base is determined. The file vector table is then accessed, using the worker routine base, and the ordinal file number specified in the END OF FILE statement. The address of the real file associated with this ordinal file is determined from the file vector table. The total number of IO operations performed for this ordinal file is then determined by looking into the appropriate entry of the transaction item part two. If the ordinal file number specified in the END OF FILE statement is greater than 7, then part two transaction items may be chained together and it will be necessary to look down this chain to find the correct entry.

If the total number of IO operations performed for this ordinal file is less than the total number of blocks in this real file, then a RETURN is executed with-

out altering the next sequential instruction. However, if all of the blocks in this real file have been read, or written, the next sequential instruction is set to the transfer location specified in the END OF FILE statement. At this point, a RETURN is executed returning control to SM2.

SUBROUTINE S7 SUBROUTINE

The SUBROUTINE subroutine accesses the current operating transaction, for the current operating CPU. It then examines the subroutines next sequential instruction addresses in the transaction item looking for an empty space. If no empty space can be found, then an error code of 1 is set and a return to SM2 is executed. This error code specifies more than three nested subroutines have been attempted. If an empty slot has been found in one of the three locations supplied, then the next sequential instruction is stored into this location. The NSI in the transaction item is then set to the transfer location specified by the SUBROUTINE statement, and a return is executed to SM2.

SUBROUTINE S8 EXIT

The EXIT subroutine accesses the current operating transaction for the current operating CPU. It then examines the three subroutines next sequential instruction addresses in this transaction item, looking for a return address. If no return address is found, the subroutine error code is set to 1, and a return to SM2 is executed. This error code specifies that an EXIT statement was encountered without being called by a SUBROUTINE statement. If a return address is found, the return address is set into the next sequential instruction location and the return address location is set to zero. When these operations have been completed, control is returned to SM2.

SUBROUTINE S9 LOOP

The LOOP subroutine accesses the current operating transaction in the current operating CPU. It then compares the next sequential instruction location for this transaction with the three loop address locations in the same transaction. If an equal condition is encountered, this means that the loop segment is currently active and that one or more loops have previously been executed. The loop count associated with this loop address is then tested for zero to determine if the appropriate number of loops have been completed. If the loop count is zero, then the loop address is also set to zero, and a return is executed, transferring control to SM2 without altering the next sequential instruction. If the loop count is not zero, this means that additional loops must be performed, therefore, the loop counter is decremented by 1, and the next sequential instruction is set to the transfer location specified by the LOOP statement.

In the case where the loop is not active, the

transaction item is examined for an empty loop address location. If no empty loop address location can be found this means that a fourth loop statement has been encountered, without the completion of any previous loop. This is an error condition, the error code is set to 1, and the return is executed, transferring control to SM2. If an empty loop address location can be found, the next sequential instruction address is stored into the loop address, the next sequential location is set to the transfer location specified in the loop statement, and the loop count is decremented by 1. A return is then executed, transferring control back to SM2.

SUBROUTINE S1O MOVE

The amount of time required to simulate a MOVE operation is calculated by dividing the number of characters to be moved by the number of characters per Logical Data Unit, in order to determine the number of Logical Data Units to be moved. The number of Logical Data Units is then multiplied by the time to move a single Logical Data Unit, and this gives the time to complete the entire MOVE operation. This time is then stored as the ADVANCE TIME in the time entry of the current operating transaction for the current operating CPU. A return is then executed, transferring control back to SM2. When SM2 receives control, it will call the Place Transaction Item on the Future Events Chain subroutine.

SUBROUTINE S11 MOVE AND EDIT

The time required to execute a MOVE AND EDIT operation is calculated by multiplying the number of characters to be moved and edited by the amount of time to move and edit a single character for the current CPU. This time is then stored in the ADVANCE TIME entry of the time section of the current operating transaction. A return is then executed, transferring control back to SM2, which will then call the Place Transaction Item on the Future Events Chain subroutine.

SUBROUTINE S12 COMPUTE

The time required to simulate a COMPUTE operation is determined by multiplying the number of instructions specified in the compute statement by a factor drawn at random from a Poisson distribution, and then multiplying the result of this operation by the Gibson Mix Time, the average time to execute a single operation in this computer. This time is then stored as ADVANCE TIME, in the time entry of the current operating transaction, for the current operating CPU. A return is then executed, transferring control back to SM2, which will then call the Place Transaction Item on the Future Events Chain subroutine.

SUBROUTINE S13 MATH

The MATH subroutine accesses the current operating transaction for the current operating CPU. From the current operating transaction it determines the address of the worker routine base. From the worker routine base, it accesses the program type. The program type is used to determine which is the preferred arithmetic set to be used for this MATH statement. If the program type is commercial, then decimal arithmetic times will be used if available. If decimal arithmetic times are not available, then fixed point, or floating point will be used in that sequence. If the program type is scientific, then floating point arithmetic times will be used if available. If floating point times are not available, then fixed point, or decimal times will be used in that sequence.

The total amount of time to be simulated for this MATH statement is determined by multiplying the

number of add, multiply and divide operations times the appropriate add, multiply and divide times for a single operation, using the correct type of arithmetic for the current CPU. This time is then stored in the time entry of the current operating transaction as the ADVANCE time. A RETURN is then executed, and control is transferred to SM2. When SM2 receives control, it will call the Place Transaction Item on the Future Events Chain subroutine. If a MATH statement is executed with no decimal, fixed point, or floating point arithmetic times supplied for the current CPU, the error code is set to 1, and a RETURN is executed, transferring control back to SM2.

SUBROUTINE S14 OPEN

The OPEN subroutine accesses the current operating transaction for the current operating CPU. It then examines the total number of IO operations performed for the ordinal files specified in the OPEN statement. If the total number of IO operations field is negative, the file is closed. If the file is not closed, an error condition exists, since a worker routine is attempting to open a file which is already open. The error code is set to 101 and the error handling subroutine S101 is called. Since the error code is greater than 100, the simulation is not terminated, but continues from here. If the file is closed, or after the error message has been printed, the total number of IO operations field is set to positive, indicating the file is now open. Next the code specifying whether this is an OPEN-INPUT, or an OPEN-OUTPUT operation is examined. If this code is invalid, the error code is set to 1 in the general simulation table and a return is executed transferring control back to SM2. If this is an OPEN-OUTPUT

operation, the ordinal file number from the OPEN statement is set into the current file request as a negative number, indicating an output operation. If this is an OPEN-INPUT statement, then the ordinal file number from the OPEN statement is stored into the current file request as a positive number indicating an input operation. At this point, the generate IO transaction item subroutine S104 is called, after the worker routine number for this transaction is set into the general simulation table. Upon return from this subroutine the general simulation table contains the address of the generated IO transaction item. The IO transaction type is then set to 1 indicating an open operation. The interrupt code in the general simulation table is set to 8 indicating an OPEN/CLOSE interrupt, and the IO transaction item address is set into the CPU item as the next IO transaction item. At this point a RETURN is executed, transferring control back to SM2, where the Interrupt Handling subroutine S110 will be called.

SUBROUTINE S15 CLOSE

The CLOSE subroutine accesses the total number of IO operations performed for the ordinal file specified in the CLOSE statement. It then sets the total number of IO operations performed negative to indicate that this file is closed. The ordinal file number from the CLOSE statement is then stored into the transaction item as the current file request. The generate IO transaction item subroutine S104 is then called, after storing the transaction item number into the general simulation table as a calling argument. When control is returned to the CLOSE subroutine, the address of the generated IO transaction item is found in the general simulation table. The CLOSE subroutine then sets the IO transaction type equal to 2 indicating a close operation and stores the rewind code into the IO transaction item, specifying whether or not the file is to be rewound. The interrupt code in the general simulation table is then set equal to 8, indicating an Open/Close interrupt is to be generated and

the address of the IO transaction item is stored into the current CPU item as the next IO transaction item. A RETURN is then executed, transferring control back to SM2. SM2 will then call subroutine S110 the Interrupt Handling subroutine.

SUBROUTINE S16 TERMINATE

The TERMINATE subroutine sets the interrupt code in the general simulation table equal to 6, in order to cause a Program Termination interrupt. If the TERMINATE statement has a zero argument, indicating that no program is to be called when this TERMINATE statement is executed, a RETURN is now executed and control is passed to SM2. If a program is to be called out when this terminate is executed, a test is made to see that the AT for the current operating CPU is equal to zero. If the AT is not equal to zero, the error code is set equal to 1 and the RETURN is executed, transferring control back to SM2. If the AT is equal to zero, S53 the ACTIVATE subroutine is called. The ACTIVATE subroutine will generate a transaction for the worker routine specified in the TERMINATE statement and place the transaction word into the AT of the current operating CPU. The TERMINATE subroutine then passes the AT transaction word plus the current CPU number to subroutine S201, which will place

that transaction item at the top of the future events chain. A RETURN is then executed which will transfer control back to SM2. When SM2 gets control it will call subroutine S110 the Interrupt Handling Subroutine.

Subroutine S17 PRINT

The PRINT subroutine first stores its subroutine number into the general simulation table. The ordinal file number for which the current request is to be executed is then accessed from the statement table. This file request is set negative since the current IO operation is an output operation. The file request thus obtained is stored into the current operating transaction as the current file request. The interrupt code in the general simulation table is set equal to 2, and the number of space operations to be performed after this PRINT is obtained from the statement table and stored into the current operating transaction. The available transaction is then set equal to the current operating transaction, and the current operating transaction is set equal to zero. A RETURN is then executed transferring control back to the calling subroutine.

Subroutine S18 CALL

The CALL subroutine first stores its subroutine number into the general simulation table. The called program number is then accessed from the statement table and stored into the current operating transaction. If the current operating transaction is to be delayed while the called program is executed the interrupt number in the general simulation table is set equal to 9 and a RETURN is executed transferring control back to the calling subroutine. If the current operating transaction is not to be delayed while the called program executes, the interrupt number in the general simulation table is set equal to 10 before the RETURN is executed transferring control back to the calling subroutine. If the delay code in the statement table is in error an error code of 1 is set and a RETURN is executed.

Subroutine S19 EXEC

The EXEC subroutine stores its subroutine number into the general simulation table. The first operand is then accessed from the general simulation table and stored as the current interrupt number into the general simulation table. A RETURN is then executed transferring control back to the calling subroutine.

Subroutine S20 LOOP ON PROBABILITY

The LOOP ON PROBABILITY subroutine accesses the current operating transaction in the current operating CPU. It then compares the next sequential instruction location for this transaction with the three LOOP address locations in the same transaction. If an equal condition is encountered, this means that the LOOP segment is currently active and that one or more loops have previously been executed. The loop count associated with this loop address is then tested for zero to determine if the appropriate number of loops have been completed. If the loop count is zero, then the loop address is also set to zero, and a RETURN is executed, transferring control to SM2 without altering the next sequential instruction. If the loop count is not zero, this means that additional loops may be performed, therefore, the loop counter is decremented by one. A call is then made to the random number generator which will place a new random number

into the general simulation table. A test is then made between the probability of falling out of the LOOP ON PROBABILITY and the random number provided. If the random number provided is less than the probability of falling out of the LOOP ON PROBABILITY instruction the current loop address and loop counter values are zeroed and a RETURN is executed transferring control back to the calling subroutine. If the random number provided is greater than the probability of falling out of the loop instruction th next sequential instruction counter is set to the transfe location specified by the loop on probability statement.

In the case where the loop is not active, the transaction item is examined for an empty loop address location. If no empty loop address location can be found this means that a fourth loop statement has been encountered, without the completion of any previous loop. This is an error condition. The error code is set to 1, and a RETURN is executed, transferring control to SM2. If an empty loop address location

can be found the next sequential instruction address is stored into the loop address, and the loop count is decremented by one. At this point, the loop subroutine acts as if the loop segment had been currently active.

SUBROUTINE S31 MEMORY

The MEMORY subroutine begins by zeroing all internal counters. It then examines the queue parameter of the MEMORY statement to see if the transaction item for which the memory test is being made is on a queue, of transaction items in the current operating CPU. If the transaction item is on a queue, the MEMORY subroutine accesses the transaction word from the queue entry table. If the transaction item is not on a queue, it must be the available transaction for the current CPU. From the transaction item, the MEMORY subroutine accesses the worker routine base address. From the worker routine base, the memory subroutine sets AIP equal to the number of instruction pages required for this transaction item. It then sets ADP equal to the number of data pages and AIOP equal to the number of IO pages. If the IO allocation switch in the general simulation table is less than zero, the memory subroutine sets the number of IO pages required (AIP) equal to 0. If the worker routine

for which memory is being allocated is reentrant, and there is an active transaction item associated with that worker routine, the number of instruction pages (AIP) is also set to 0.

At this point, the MEMORY subroutine tests the memory allocation code stored in the general simulation table. If the memory allocation code is equal to 1, the number of contiguous pages will be equal to the number of instruction pages required rounded up to the nearest integer. In addition, the number of noncontiguous pages required will be equal to the sum of the number of data pages and the number of IO pages rounded up to the nearest integer. If the memory allocation code is equal to 2, the number of contiguous pages will be equal to the sum of the number of data pages and the number of IO pages required, rounded up to the nearest integer. In addition, the number of noncontiguous pages will be equal to the number of instruction pages rounded up. If the memory allocation code is equal to 3, there will be two pieces of contiguous memory required. The instruction

pages required will be allocated as one contiguous section and the sum of the data pages and the IO pages required will be allocated as a separate, but also, contiguous area. At this point, the memory subroutine will set the largest contiguous pages required equal to the larger of these two areas, and the secondary contiguous storage required equal to the lessor of these two areas. If the memory allocation code is equal to 4, all of the memory requirements will be considered noncontiguous. Therefore, the noncontiguous pages required will be set equal to the sum of the IO pages, the data pages and the instruction pages rounded up to the nearest integer value. If the memory allocation code is equal to 5, all memory requirements will be considered to be contiguous.

The memory subroutine now sets a pointer to the lowest page of the low memory specified and the highest page of the high memory specified. A scan is then made of all of the pages within this range. The result of this scan is a pointer to the address of the largest contiguous

area of storage available, a pointer to the next largest area of contiguous storage available, the size of the next largest contiguous storage available, the size of the next largest contiguous storage area and a count of the total number of noncontiguous pages left. At this point a comparison is made between the number of pages available and the number of pages required.

If the largest contiguous storage area required is greater than the largest contiguous storage area available, then a test to determine if a PACK operation would provide sufficient memory is made. If the largest contiguous area required can be satisfied, then the remaining pages of the largest contiguous storage available is compared to the next largest contiguous storage available. If the remaining piece is larger than the next contiguous storage area available, then the next largest contiguous storage available is added to the noncontiguous storage available and the remainder is set equal to the second largest contiguous storage available. The next step is to examine if the second largest contiguous storage area required can

be supplied out of the second largest contiguous storage available. If this storage cannot be supplied, then the test to determine if a PACK operation would provide sufficient memory is made. If the second largest contiguous storage area can be satisfied, then the remaining test is to see if there is sufficient noncontiguous storage available to satisfy those requirements. If this test can be satisfied, then the memory test is considered to be successful and the appropriate exit line address is set into the general simulation table. In addition, most of the information calculated by the MEMORY subroutine is stored into the general simulation table for use by the ALLOCATE subroutine. At this point a RETURN is executed and control is transferred back to SM2.

In the event that a test to determine if a PACK operation would supply sufficient memory is required, all of the available pages are added into one counter and all of the required pages are added into a second counter. If the total number of required pages is less than, or equal to the total number of available pages, then a PACK

would supply sufficient memory. In this event, the appropriate exit line is set into the general simulation table, a RETURN is executed, and control is passed back to SM2. If the test for a PACK operation is unsuccessful, then the no memory exit line is set into the general simulation table and a RETURN is executed, also transferring control back to SM2.

SUBROUTINE S32 ALLOCATE

The ALLOCATE subroutine examines the parameters placed in the general simulation table by the MEMORY subroutine, and determines if the available transaction in the current operating CPU is the same transaction for which the memory statement was executed. If this test is not satisfied, the error code is set=1 and a RETURN is executed. The ALLOCATE subroutine next tests to determine if a PACK was required, but not executed. If this is true the error code is set=2 and a RETURN is executed. If no error conditions exist, the ALLOCATE subroutine calls S210, the Reset Memory Load subroutine. When control is returned to the ALLOCATE subroutine a test is made to determine if a PACK has been performed after the MEMORY statement was executed, but before this ALLOCATE statement. If a PACK was performed, then the pack code in the general simulation table is set equal to 0 and a scan is made to determine the lowest page which is empty in the memory range specified. Once this scan has been

completed, all memory requirements are allocated starting at the first free page as a single contiguous area. At this point a RETURN is executed and control is passed back to SM2.

If a PACK was not performed, then memory will be allocated into the areas determined by the MEMORY subroutine. When allocating storage for a transaction item, the transaction word for that transaction item is stored into the page table. The exception to this rule is the allocation of instruction storage for a reentrant worker routine. In this case, the worker routine base address is placed into the page table as a negative integer, rather than the transaction word. This allows deallocation of data and IO storage without deallocating the instruction storage. The allocation is performed by starting at the address of the largest contiguous storage area available, as passed by the MEMORY subroutine and allocating the largest number of pages required. The subroutine next starts at the address of the second largest contiguous area of storage available, and allocates the second largest

number of pages required. The allocate subroutine next sets a pointer to the lowest page in the memory range specified. It then allocates all noncontiguous storage required from any free page available. If any of the three allocating loops detects a page which is not empty when it should be, the error code is set equal to 3 and a RETURN is executed.

Upon completion of the ALLOCATE subroutine, the fields set in the general simulation table by the MEMORY subroutine are cleared, and the next sequential instruction counter is set to zero. At this point the run class pointer in the transaction item is set equal to the current CPU number, and a 1 is added to the number of transaction items with memory allocated counters in the worker routine base. A RETURN is then executed transferring control back to SM2.

SUBROUTINE S33 DEALLOCATE

The DEALLOCATE subroutine first calls the Reset Memory Load subroutine S210. Upon receiving control back, the DEALLOCATE subroutine scans the memory page table for pages containing a transaction word equal to the available transaction in the current operating CPU. Each of these pages is set equal to 0. If the transaction for which storage is being deallocated is not reentrant, a 1 is subtracted from the count of transaction items with memory allocated in the worker routine base, the Set Memory Load subroutine S211 is called, the next sequential instruction is set to equal 0, and a RETURN is executed, transferring control back to SM2.

If the transaction for which storage is being deallocated is reentrant, a test is made to see if the number of transactions with memory allocated is greater than 1. If it is greater than 1, the action taken is the same as for a nonreentrant transaction item. If the number of active transaction items is equal to 1, a new

scan is made through the memory page table in order to deallocate all instruction pages for the transaction item. These pages are identified as having the negative worker routine base address in the page entry. At this point a 1 is subtracted from the number of transaction items with memory allocated in the worker routine base, the Set Memory Load subroutine S211 is called, the next sequential instruction counter in the general simulation table is set equal to zero, and a RETURN is executed.

SUBROUTINE S34 PACK

The PACK subroutine calls the Reset Memory Load subroutine S210. Upon return of control to the PACK subroutine it sets a pointer to the low page and the high page in the memory range specified. The PACK subroutine then proceeds to move all occupied pages to the low end of the range and 0 all pages at the high end of the memory range. Upon completion of this operation the PACK subroutine calls the Load Memory subroutine S211, resets the next sequential instruction counter in the general simulation table, and if a PACK was required it sets the pack performed code into the general simulation table. At this point a RETURN is executed and control is transferred back to SM2.

SUBROUTINE S35 EXAMINE-FIRST

The EXAMINE-FIRST subroutine accesses the queue number from the general simulation table. It then resets the queue pointer to the queue start address, sets the queue increment equal to 1 and the queue limit equal to the queue end in the general simulation table. A test is then performed to see if the first entry in the queue is equal to zero. If the first entry is 0, the next sequential instruction counter in the general simulation table is set equal to 0 and a RETURN is executed. If the first entry in the queue is not equal to 0, a test is made to determine if the queue entry is capable of running in the current CPU. If the current entry is capable of running the next sequential instruction code in the general simulation table is set equal to 1, and a return is executed. If the current entry in the queue is not capable of running in the current CPU, then the EXAMINE-FIRST subroutine will go back to look at the next item on the queue.

SUBROUTINE S36 EXAMINE-NEXT

The EXAMINE-NEXT subroutine accesses the queue to be examined from the general simulation table. If the queue pointer in the queue table is set equal to the queue limit, as set by the examine first or examine last subroutines, the NSI pointer in the general simulation table is set equal to zero and a RETURN is executed. Otherwise, the queue increment as set by the EXAMINE-FIRST, or EXAMINE-LAST subroutine is added to the current queue pointer and the current queue entry is examined for zero. If the current queue entry is equal to zero, the next sequential instruction counter in the general simulation table is set equal to zero and a RETURN is executed. If there is a current entry in the queue, the current entry is tested to see if it can run in the current CPU. If the current entry can run, the next sequential instruction counter is set equal to 1 and a RETURN is executed. Otherwise, the no find exit line is set in the general simulation table and a RETURN is executed.

SUBROUTINE S37 EXAMINE-LAST

The EXAMINE-LAST subroutine performs the same functions as the EXAMINE-FIRST subroutine, with the following exceptions.

1) The queue pointer is initialized to the queue end rather than the queue start.

2) The queue increment is set equal to minus 1 rather than plus 1.

3) The queue limit is set equal to the queue start rather than the queue end.

SUBROUTINE S38 PLACE

The PLACE subroutine first determines the queue number from the PLACE statement in the general simulation table. If the queue specified is full, the error code is set equal to 4, and a RETURN is executed. If the queue is not full, a test is made to see that the rule that only AT's may be placed on AT queues, and only IOT's may be placed on IOT queues, has not been violated. If this rule has been violated, an error code of 1 is set and a return is executed. A test is then made to see if a zero item is being placed. If a zero item is being placed, the error code is set equal to a 2, and a RETURN is executed. If the item being placed is a negative AT, the PLACE is ignored and the AT is set equal to 0. This eliminates cycle transaction words from piling up on queues. Next a test is made to determine the queuing method for the queue specified in the place statement.

If the queueing method for the queue selected is priority, the item being placed is inserted in t

queue just ahead of the first transaction with lower priority. If the queueing method specified is first in, first out, the item being placed is put after the last entry currently on the queue. If the queueing method for the queue selected by the place statement is last in, first out, the item being placed goes at the top of the queue.

Once the item has been placed on the queue the appropriate word in the CPU item is set equal to zero. The current number of entries on the selected queue is incremented by 1, and the previous load is calculated. The queue load is calculated by multiplying the elapsed time by the number of entries on the queue. If the current number of entries on the queue is greater than the maximum number at any previous time, the maximum number of entries field in the queue table is set equal to the current number of entries. At this point, the last reference time in the queue table is set equal to the current clock time. A 1 is also added to the total number of place operations on this queue. In

addition, the queue start time for the AT or IOT placed on the selected queue is now set equal to the current clock time. The queue pointer is then reset to the queue start and a RETURN is executed, transferring control back to SM2.

SUBROUTINE S39 SELECT

The SELECT subroutine accesses the queue number from the SELECT statement in the general simulation table. It then tests the entry in the queue entry table pointed to by the queue pointer for a zero. If the entry is zero, the error code is set equal to 1 and a RETURN is executed. If the current queue item is not zero, the queue type is then compared with the item specified for selection by the SELECT statement. If the queue type is wrong, the error code is set equal to 2, and a RETURN is executed. If the queue type is correct, the SELECT subroutine next tests the appropriate slot in the CPU item to insure that no item is overlaid by the select statement. If an item would be overlaid by this SELECT, the error code is set equal to 3, and a RETURN is executed. The transaction item for the IO transaction word is then accessed, and a test is made to determine if the run class is equal to zero. If the run class is not equal to zero, a test is made to determine if the current CPU is capable

of running this transaction. If the current CPU may not run this transaction the error code is set equal to 4, and a RETURN is executed. If the run class pointer was equal to zero, a test is made to determine if the current CPU is able to load the transaction being examined. If the current CPU may not load the transaction under examination, the error code is also set to 4, and a RETURN is executed.

If the current CPU is able to either run or load the transaction under examination, the transaction word is extracted from the queue and placed into the appropriate word of the CPU item. The queue is then compacted to eliminate the space created by the extracted transaction word. The queue statistics are then updated in the same way as for the PLACE subroutine. Each time a SELECT statement is executed, a new record is generated by the select subroutine and placed into a table local to this subroutine. When 100 such records have been accumulated a block of records is written out to Fortran unit number 18. These records are examined by the STAT1 routine and queue statistics are developed from this

information at the end of a simulation run. Upon completion of the PLACE operation the queue pointer for the selected queue is set equal to the queue start location. A RETURN is then executed and control is passed back to SM2.

SUBROUTINE S40 BUFF

The BUFF subroutine accesses the AT out of the current operating CPU. From this AT it extracts the current file request. The current file request is an ordinal file number placed into the transaction item by the read/write/print subroutines. If the ordinal file number is positive, the IO operation requested is an input operation. If the ordinal file number is negative the IO operation will be an output operation. The BUFF subroutine next tests to see if the ordinal file being examined is open. This test is by passed for operating system transactions since they do not have to open their files. If this test is failed, the error code is set equal to 101, which will not terminate the simulation, an S101 the Error Handler is called. Next the buffer count for this ordinal file is tested for zero. If the buffer count is equal to zero, the next sequential instruction counter in the general simulation table is set equal to 2, and a RETURN is executed, unless the transaction being examined is an operating system transaction. If the

buffer count is equal to zero for an operating system transaction, an IO transaction item is generated and the next sequential instruction counter is set equal to 3 before a RETURN is executed.

If the buffer count for the ordinal file being examined is not equal to 0, the record count is reduced by 1. If the record count is now equal to 0, the buffer count is also reduced by 1. If the record count is not equal to zero now the next sequential instruction counter is set equal to zero and a RETURN is executed. If the buffer count has just been reduced by 1 an IO transaction item is generated by a call to S204, and the record count is reset to the number of records contained by a single buffer. A test is then performed to determine if the IO transaction item generated was the result of a print operation. If it is, the number of print lines to be skipped after this IO operation is performed, are stored into the IO transaction item. In any event, the next sequential instruction counter is now set to 1, and a RETURN is executed.

SUBROUTINE S41 SEEK

The SEEK subroutine accesses the IO transaction word from the current operating CPU. From the IO transaction item it obtains the device number and from the device table it obtains the address of the device class entry associated with this device. If the device type is not random access, the next sequential instruction counter is set equal to 1 and a RETURN is executed. In this event, a SEEK is not required.

If the device is a random access device, then the to-from table address in the device set table is tested for zero. If the to-from table address is zero this means that there is only one file on this device and there will be no seek time involved. In this case the next sequential instruction counter is also set equal to 1 and a RETURN is executed. If the to-from table address is not equal to zero, then the to-from table is accessed using the current real file number as the "TO" file and

the last file accessed from the device table as the "FROM" file. If the to-from entry accessed is negative, this means that a search operation must be performed and no SEEK will be required. In this event the next sequential instruction counter is set equal to 1, and a RETURN is executed. If the to-from table entry accessed is positive, the IO transaction item type is set equal to 3, indicating a SEEK operation will be performed and the IOADVANCE subroutine S43 is called. In this event the next sequential instruction counter is set equal to zero and a RETURN is executed. If this exit line is taken the IO transaction word from the current operating CPU has been set equal to 0.

SUBROUTINE S42 IOREADY

The IOREADY subroutine first teststo determine if the IO transaction word required is in the current CPU item or on a queue. Once the IO transaction word has been obtained, a test is made to see if the IO transaction item is a seek continue or some other type of transaction item. For a seek continue IO transaction item, a transfer is made immediately to the test availability table for channel control unit pair, portion of the IOREADY subroutine. For IO transaction items other than seek continue, a test is made to determine if the IO transaction item has been correctly initialized. If the IO transaction item has been correctly initialized, a test is made for a function IOT, which is treated differently from others. Next, a test is made to determine if the file required by this IO transaction item is available. If the file is not either free, or seized by this transaction item, then the file is considered to be busy and the IOREADY test fails. If the IOREADY test fails, the next sequential instruction counter is set

equal to zero, and a RETURN is executed. If the file is available, the next test determines whether or not the required device is available. A device is considered to be free if it has no seizing transaction word, or if the current seizing transaction item is the same as the transaction item which generated the current IO transaction item. If the device required is not available, then the busy exit line from the IOREADY statement is taken. If both the file and the device are available, a test is made to determine if a channel-control pair from the availability table can be found. Since more than one channel or control unit may be used to access a single device, there may be many pairs of channel-control unit paths in the availability table for this device. Associated with a channel-control pair in the availability table is a direction entry, which indicates whether the channel-control unit pair may be used for an input, an output or both types of operations.

In order for a channel to be considered free,

the device transfer rate for the operation to be initiated must not exceed the maximum channel rate on the channel selected. For a multiplexor channel this test is complicated by the fact that the channel may operate in either multiplexor or burst mode.

If the channel being tested for busy is currently operating in the burst mode, it is automatically considered to be busy. If the channel being tested is operating in the multiplexor mode and the device transfer rate for the new IO operation plus the current multiplexor transfer rate would exceed the maximum multiplexor transfer rate, the channel is considered busy. For a selector channel the busy test is very simple in that the channel is either busy or free, since only a single IO operation may be performed at one time.

If the IOREADY subroutine examines all of the pairs of channels and control units available for accessing a single device without finding a channel control unit pair, then the IOREADY test is considered

to have failed and the next sequential instruction counter is set equal to zero and a RETURN is executed. If all of the requirements can be satisfied, that is a file, device, channel, and control unit are all available, then the next sequentail instruction counter is set equal to 1, indicating that the IO operation may be performed. Before a return is executed, however, the device number, control number and channel number for the IO operation are stored in the IO transaction item. This information will be used by the IOADVANCE subroutine.

SUBROUTINE S43 IOADVANCE

The ICADVANCE subroutine begins by storing its subroutine number into the general simulation table. It then zeroes out a number of time fields which will be used to determine the device, control, channel unit times required for this IO operation. After determining that the IO transaction item has been properly initialized, a test is made to see whether the IO transaction item is a function transaction. For a function IO operation, the device, control unit, and channel times required can be found in the function table.

If the current IO transaction is not for a function, the appropriate file is seized. At this point the device set table is examined to determine if there is a to-from table for this device. If a to-from table exists, then the to-from entry is accessed and A is set equal to the to-from time. Next a test is made to determine whether there is a penalty time associated with this device. If there is, B is set

equal to the penalty time. Next, the start + stop time and the device time for the current IO operation are determined.

The read/write time for the current IO operation is determined by dividing the buffer length for the current file by the character-per-second transfer rate for the current device, and multiplying the result by 1,000,000 to give a transfer time in microseconds. The device required for the current IO operation is then seized.

For a print IO transaction item, the from time is then calculated by multiplying the number of lines to be skipped by the time required to skip one line.

For an open input IO transaction only, all times for this IO transaction item will be multiplied by the number of buffers required in order to simulate the filling of all available buffers at open time. For an open output transaction item, all times are set equal to zero, so that no IO facilities will be required to perform this operation.

For a close IO transaction item, any file or device which was permanently seized by a peripheral statement will now be set to temporary seize for the duration of this IO operation. If the device required for this operation is a tape and rewind was specified in the close statement, then the rewind time will now be added to the device time.

For a seek IO transaction, all times except the to-from times are set equal to zero. For a seek-continue IO operation, the to-from time is set equal from zero and all other times are left as they are.

At this point, the channel control and device times are summed from the individual times above and IO statistics are calculated. The appropriate channel is then seized and a RETURN is executed, transferring control back to SM2.

SUBROUTINE S44 IOTERM

The IOTERM subroutine accesses the queue number from the current statement in the general simulation table. If the queue type is not equal to 1, the error code is set equal to 1 and a RETURN is executed. The queue entry table is then accessed to pick up the address of the transaction item to be examined. The address of the IO transaction item to be examined is accessed from the IOT slot in the current operating CPU. If the number of future event chains items in the IO transaction item is not equal to zero, a match cannot occur. In this event, the next sequential instruction counter in T20 is set equal to zero and a RETURN is executed. If the IO transaction item was not generated by the current transaction item, a match is also impossible.

The file section of the transaction item under examination is next checked for any open file with a buffer count of zero. If a buffer count of

zero exists, this means that the current IO requirements of this transaction item have not yet been satisfied. In this event, the NSI in T20 is again set to zero and a RETURN is executed. If the buffer counts for all open files are other than zero, all IO requirements have been fulfilled, and this transaction may be selected off a deferred queue. In this event the next sequential instruction counter in T20 is set equal to 1 and a RETURN is executed.

SUBROUTINE S45 SET SWITCH

The SET SWITCH subroutine accesses the switch number from the current statement in T20. If the switch number accessed is less than 21, the switch to be set is one local to the current CPU. If the switch number accessed is greater than 200, or greater than 20 and less than 101, the switch number is invalid. In this event an error code is set, and a RETURN is executed. If the switch number accessed is greater than 101, the switch to be set is a global switch, accessible by all CPU's.

Once the switch number has been validated and the appropriate switch type determined, an entry table 23 is set equal to 1 indicating that the switch is on. A RETURN is then executed, transferring control back to SM2.

SUBROUTINE S46 RESET SWITCH

The RESET SWITCH subroutine performs the same tests as the SET SWITCH subroutine. Once the validity of the switch number and the type of switch to be set have been determined, the appropriate entry in T23 is set equal to a-1, indicating that the switch is off. When this operation has been completed, a RETURN is executed and control is transferred back to SM2.

SUBROUTINE S47 TEST SWITCH

The TEST SWITCH subroutine performs the same preliminary tests as the RESET SWITCH and SET SWITCH subroutines. Once the validity of the switch number and the type of switch to be tested have been determined, the appropriate entry in T23 is examined. If the switch counter to be examined is greater than zero, indicating that the switch is on, the next sequential instruction counter in T20 is set equal to zero and a RETURN is executed. If the switch counter being examined is less than zero, indicating that the switch is off, the next sequential instruction counter is set equal to one and a RETURN is executed.

SUBROUTINE S48 INTERRUPT

The INTERRUPT subroutine accesses the number of the CPU to be interrupted and the interrupt number to be generated from the current statement in T20. A test is then performed to insure that the interrupt number is valid and that an interrupt address has been provided for the CPU which is to be interrupted. If either of these tests fails, the error code in T20 is set equal to 101 and a RETURN is executed. If the interrupt to be generated is valid, the FEC code is set equal to the CPU number plus 100, and the FEC item is set equal to the interrupt number. The FEC manipulator time is then set equal to the current clock, and subroutine S201 is then called. This places the newly generated FEC code and item at the top of the future events chain. A RETURN is then executed transferring control back to SM2.

SUBROUTINE S49 DISABLE

The DISABLE subroutine accesses the number of the interrupt to be disabled from the current statement in T20. If the interrupt number is equal to 99, all interrupts are to be disabled. If a single interrupt is to be disabled a test is made to insure that the interrupt specified has been defined for the current CPU. If the interrupt is undefined, the error code is set equal to one and a RETURN is executed. If the interrupt is defined, the interrupt is disabled by making the interrupt address negative, and a RETURN is executed.

If all interrupts for the current CPU are to be disabled, all positive interrupt addresses for the current CPU are made negative. When this operation has been completed, a RETURN is executed.

SUBROUTINE S50 ENABLE

The ENABLE subroutine performs exactly the same functions as the DISABLE subroutine, S49, with the exception that the interrupt addresses are set positive, rather than negative.

SUBROUTINE S51 CLOCK

The CLOCK operand is accessed from the current statement in T20. The four digit integer of the form DDEE expressed in microseconds is then converted to a real number. This is then added to the current simulator time and stored in the clock item time for the current CPU. A RETURN is then executed transferring control back to SM2. When SM2 receives control, a call to S103, the Place Clock Item on the Future Events Chain subroutine, is performed.

SUBROUTINE S52 RETURN

The RETURN subroutine is used to transfer control from the current operating transaction, which is normally an operating system transaction, to the available transaction for the current CPU. The requirements for the successful execution of this subroutine are an available transaction and no IO transaction in the current CPU.

The RETURN subroutine accesses the current CPU from the general simulation table. If the IO transaction word for the current CPU is not equal to zero, an error code of one is set and a RETURN is executed. If the available transaction word for the current CPU is equal to zero, an error code of two is set and a RETURN is executed. If the available transaction is negative, indicating a cycle transaction, all further tests for the available transaction are ignored. If the available transaction is not a cycle transaction, the worker routine base for the available transaction

is accessed. The program type from the worker routine base is then stored in the general simulation table. If the program type of the available transaction is equal to one, indicating an operating system transaction, all further tests on the available transaction is not for an operating system, the run class pointer is tested for zero. If it is zero, an error code of three is set into T20 and a RETURN is executed. This indicates that the available transaction has not yet been loaded and may not be returned to. If the run class pointer is not equal to zero, a test is made to insure that the current CPU is in the available transaction's run class. If the condition is not met, an error code of four is set in T20 and a RETURN is executed.

Once all of the required tests for the current CPU item and the available transaction have been made, the current operating transaction is closed out. Since the current operating transaction should be an operating system transaction, the total CPU time, IOT queue time, and TI queue time are added into the appropriate fields

in the current CPU item. The COT slot for the current CPU item is then set equal to the AT slot. If the new COT is a cycle transaction item, the cycle start time for the current CPU is set equal to the current time and a RETURN is executed. Otherwise, the NSI code in T20 is set equal to zero and the AT and IOT slots for the current CPU item are set from the current operating transaction item's hold areas. At this point all entries in the transaction item table occupied by the old COT are freed by setting the first word of each entry equal to -1. A return is then executed, transferring control back to SM2.

SUBROUTINE S53 ACTIVATE

The ACTIVATE subroutine accesses the current operating CPU from the general simulation table and tests to insure that the available transaction word for the current CPU is equal to zero. If this test is not satisfied, an error code of one is set into T20 and a RETURN is executed. The first operand of the ACTIVATE statement is then accessed from the current statement in T20. The worker routine base table is then examined to insure that there is an entry for the worker routine to be activated. If no entry exists in the worker routine base, an error code of two is set in the general simulation table. and a RETURN is executed. If a worker routine base does exist, subroutine S203, the Create Transaction Item subroutine, is called. Upon return of this control from S203, the address of the transaction item created will be found in the general simulation table. The creation time for the new transaction item is then set

equal to the current clock time, and the total number of transaction items called counter in the worker routine base is incremented by one. At this point, the available transaction word in the current CPU is set equal to the newly created transaction item address. The next sequential instruction counter in the general simulation table is then set equal to zero and a RETURN is executed, transferring control back to SM2.

SUBROUTINE S54 RECEIVE

A RECEIVE subroutine accesses the current operating CPU from the general simulation table, it then tests to insure that the available transaction word for the current CPU item is equal to zero. If this test is not successful, an error code of one is set and a RETURN is executed. If the AT is equal to zero, the future events chain is searched for an item which has generated a receive interrupt for the current CPU but has not yet been removed from the chain. If no such item exists, the error code is set equal to two and a RETURN is executed. If the item is found, a pointer is set to the address of the item, and subroutine S202, the Reorganize FEC subroutine is called. Upon return of control from S202, a test is made to see whether the newly received transaction item was generated or called. For a called transaction item, the transaction item's address is stored as the available transaction for the current CPU and a RETURN is executed, transferring control back to SM2.

For a generated transaction item, the creation time is set equal to the current clock time and the time off the future events chain field is set equal to zero. Then the total number of generated transaction items field in the worker routine base is incremented by one. If the generation limit for the newly received transaction item has been reached, the available transaction word in the current CPU item is set equal to the newly received transaction item, and a RETURN is executed. If the creation limit has not been reached, a new transaction item is generated by a call to S203, the FEC time for the newly created transaction item is set equal to the current clock time plus the spread, either fixed or poisson, and the new transaction item is placed on the future events chain by a call to S201. At this point, the available transaction word in the current CPU item is set equal to the received transaction, and a RETURN is executed, transferring control back to SM2.

SUBROUTINE S55 CYCLE

The CYCLE subroutine accesses the current operating CPU from the general simulation table. A test is then made to insure that the AT is equal to zero. If the AT is not equal to zero, an error code of one is set, and a RETURN is executed. The IOT field is then tested for zero; if the IOT is not equal to zero, the error code is set equal to two, and a RETURN is executed. If both of these tests are executed successfully, the current operating transaction field is set negative and the start cycle time for the current CPU is set equal to the current clock time. The total CPU time, total IOT queue time, and total TI queue time fields from the current operating transaction are then added into the appropriate fields in the current CPU item. At this point all entries in the transaction item table for the current COT are set free by initializing the first word in each entry to -1. The next sequentail instruction code is then set equal to zero and a RETURN is executed, transferring control back to SM2.

SUBROUTINE S56 DESTROY

The DESTROY subroutine accesses the first operand from the current statement in the general simulation table. I the operand is equal to one, the available transaction will be destroyed and if the operand is equal to two, the IO transaction item will be destroyed. If the operand is invalid, an error code of one will be set and a RETURN will be executed.

The destroy available transaction routine tests to insure that all files have been closed and moves the total number of IO operations performed by the available transaction into table ITX. If any file in the transaction item being destroyed has not been closed, an error code of two is set and a RETURN is executed. If all files have been closed, a record is written out onto FORTRAN unit 15, containing all the statistics and the total number of IO operations on each file for the transaction being destroyed. Each entry in the transaction item table used by the

transaction being destroyed is then set free by initializing the first word for each entry to -1. The available transaction word in the current CPU item is set equal to zero, and a RETURN is executed, transferring control back to SM2.

The destroy IO transaction item routine examines the IO transaction item to see whether the number of FEC items is equal to zero. If the number of FEC items is not equal to zero, there are additional pointers to this transaction item still on the future events chain. In this case, the IO transaction word in the current CPU item is set equal to zero, the IO transaction item is not destroyed, and a RETURN is executed. If the number of items on the FEC field is equal to zero, this means that the IO transaction item entry in the IO transaction item table may be eliminated. In this event, the total IO queue time is added to the total IOT queue time in the transaction item which generated this IOTI. In addition, the number of IO transaction items outstanding field in the trans-

action item is reduced by one. The entry in the IO transaction item table is then freed by initializing the first word to -1, the IO transaction word in the current CPU item is set equal to zero, and a RETURN is executed, transferring control back to SM2.

SUBROUTINE S57 PERIPHERAL

The PERIPHERAL subroutine accesses the first operand from the current statement in the general simulation table. If this operand is equal to zero, the transaction item for which the peripheral statement will be executed must be the available transaction in the current CPU item. If the first operand is not equal to zero, it must be the address of a queue, from which the transaction item will be accessed. In this case, the queue type is tested to insure that it contains only transaction items. If the queue type specifies that it contains IO transaction items, the error code is set equal to one, and a RETURN is executed. The current entry as specified by the queue pointer in the current queue is then accessed and used as the transaction item for which the PERIPHERAL statement will be executed.

From the transaction item specified by the PERIPHERAL statement, the worker routine base and file

vector tables are accessed. If the number of files used by this transaction item is equal to zero, the NSI counter in the general simulation table is set equal to one, and a RETURN is executed. This indicates that all IO is available.

For transaction items which perform IO operations there will be one entry in the file vector table for each file referenced. Associated with each entry in the file vector table is a seizing code, which specifies whether the device, file, or nothing should be seized. The PERIPHERAL subroutine examines each entry and if the device is to be seized, seizes it. If the file is to be seized, it seizes the file. If all requirements can be satisfied, the NSI code is set equal to 1, and a RETURN is executed. If all of the seizing requirements cannot be satisfied, the NSI code is set equal to zero, and all facilities which have been seized for this transaction item are freed. At this point a RETURN is executed, transferring control back to SM2. Notice that if a transaction

item attempts to seize a device which is currently busy that has not been permanently assigned to another transaction item, the seizing code is stored into the eighth word of the device set table. In this case the device will be permanently seized when the current IO operation on the device has finished.

SUBROUTINE S59 MATCH

The MATCH subroutine accesses the available transaction for the current CPU and the first operand from the current statement in the general simulation table. If the queue number specified in the match statement is equal to zero, an error code of one is set, and a RETURN is executed. Otherwise, the current queue entry as specified by the queue pointer in the selected queue is accessed. If the queue specified does not contain transaction items, an error code of 2 is set and a RETURN is executed. If the queue type is correct, the called worker routine number from the transaction item on the queue is compared to the worker routine number for the available transaction. If these two worker routine numbers are not equal, the NSI code is set equal to 1, and a RETURN is executed, indicating that no match has been found. If the two worker routine numbers are equal, the calling transaction item number field from the available transaction item is compared to the address of

the transaction item on the queue. If these two transaction item addresses are not equal, the next sequential instruction code is set equal to one and a RETURN is executed. If the two transaction item addresses are equal, the next sequential instruction code is set equal to zero, indicating that a match has been found, and a RETURN is executed, transferring control back to SM2.

SUBROUTINE S61 CALLOUT

The CALLOUT subroutine accesses the current CPU from the general simulation table. It picks up the first operand from the current statement and the available transaction for the current CPU item.

The program number to be called is then obtained from the available transaction and passed to the ACTIVATE subroutine, S53. Upon return of control from the ACTIVATE subroutine the new available transaction will be the newly called transaction item. The original available transaction is then restored and the PLACE subroutine, S38, is called. The PLACE subroutine puts the original transaction on the queue specified by the CALLOUT statement. When control is returned from the PLACE subroutine, the new transaction item is made the available transaction and the calling transaction item number is stored into the newly created available transaction. At this point, a RETURN is executed and control is transferred back to SM2.

SUBROUTINE S62 PRIORITY

The PRIORITY subroutine accesses the first two operands from the current statement in the general simulation table. If the queueing method for both queues specified is not priority, the error code is set equal to 2 and a RETURN is executed. If the queueing method for both queues is priority, the first entry in each of the two queues is accessed. If either or both of the two queues selected contains IO transaction items then the transaction item which created the IO transaction item is accessed. At this point a comparison is made between the priorities of two transaction items. If the item from the first queue has a priority of the item from the second queue, the next sequential instruction code is set equal to zero and a RETURN is executed. If the item from the second queue has the higher priority, then the next sequential instruction code is set equal to 1, and a RETURN is executed. If both of the queues are empty,

the next sequential instruction code is set equal to 2, and a RETURN is executed. Notice that if only one of the two queues is empty, the item from the other queue is specified as having the higher priority.

SUBROUTINE S101 ERROR HANDLER

The ERROR HANDLER subroutine examines the error code in the general simulation table. If the error code found is less than 100, the subroutine number and the error code are printed out with the message "SIMULATION TERMINATED". At this point any dump switch which has been turned off, is turned on and all tables are printed out before terminating the simulation.

If the error code found is greater than 100, the current subroutine number and the error code are printed out with the message "SIMULATION CONTINUING". In this event, a RETURN is executed which will transfer control back to SM2 without printing out any of the tables.

SUBROUTINE S102 PUT IOTI ON FEC

The PUT IOTI ON FEC subroutine accesses the IO transaction item from the current operating CPU. The current CPU number is then stored into that IO transaction item, and the current FEC item is set equal to the IO transaction word. If this is a SEEK IOTI, the number of FEC items is set equal to 1, the FEC code is set equal to 31, and the PUT FEC iteration counter is set equal to 1, bypassing the test for equal end times.

The PUT IOTI subroutine may place as many as 3 occurrences of the IO transaction item on the future events chain. This depends on the number of equalities found between the channel control and device end time. If all three end times are equal, only one item is placed on the future events chain. The FEC code specifies which facilities are to be set free when an IO termination occurs. If the FEC code is equal to 21, this item specifies a channel control

and device end. If the FEC code is equal to 22, this item specifies a channel and control end. If the FEC code is equal to 23, this item specifies a control and device end. If the FEC code is equal to 24, this item specifies a channel and device end. If the FEC code is equal to 25, this item specifies a channel end only. If the FEC code is equal to 26, this item specifies a control end only. If the FEC code is equal to 27, this item specifies a device end only.

Once the number of items to be placed on the future events chain has been determined, a separate call is made to subroutine S201, the PUT FEC subroutine, for each item to go on the future events chain. When this operation has been completed, the IO transaction word in the current CPU item is set equal to zero and a RETURN is executed, transferring control back to the calling subroutine.

SUBROUTINE S103 PUT CLOCK

The PUT CLOCK subroutine sets the current FEC code equal to 99, indicating a clock item, and sets the current FEC item equal to the current CPU number. The PUT FEC time field in the general simulation table is set equal to the next clock time from the current operating CPU. A test is then performed to determine whether there is a current clock item for the current CPU on the future events chain, it is removed before the new clock item is placed on the future events chain. The new clock item is placed on the future events chain by a call to subroutine S201, the PUT FEC subroutine. At this point the clock item switch is turned on, indicating that there is a clock item on the future events chain for the current CPU, and a RETURN is executed transferring control back to SM2.

SUBROUTINE S104 FEC MANIPULATOR

The FEC MANIPULATOR is used to place the current operating transaction for the current CPU onto the future events chain. Each time a transaction item is placed on the future events chain, the first item on the future events chain for which the interrupt is enabled, is removed from the chain.

The FEC MANIPULATOR sets the FEC code equal to the CPU number and the FEC item equal to the current operating transaction word. For the current operating transaction, the time off FEC field is calculated by adding the advance time and the current clock time. The time off FEC field in the general simulation table is then set equal to the time off FEC field in the transaction item, and the current CPU number is stored into the current operating transaction. Subroutine S201, the PUT FEC subroutine, is then called, placing the current operating transaction onto the future events chain. Upon return of control from S201, the COT slot for the current CPU item is set equal to zero.

A pointer is then set to the top of the future events chain and each item is examined. If the FEC code is less than zero, the current FEC transaction must be a generated or called transaction item. For these items, if the FEC code is less than -5, a receiver interrupt has already been generated and the item on the future events chain is ignored. Therefore, the pointer is advanced by one, and the next item is examined. If the FEC code is in the range from -1 to -5, a test is made to determine whether the receiver interrupt has been disabled. If the receiver interrupt is disabled, once again the item is ignored and the FEC pointer is advanced by one. If the receiver interrupt is enabled, 100 is subtracted from the FEC code, indicating that an interrupt has been generated, and the interrupt code in the general simulation table is set equal to 5. At this point the current operating CPU number is set equal to the CPU number specified for the receipt of the current FEC item. If the time off FEC field for the current FEC item is greater than

the current clock time, the current clock time is advanced. Control is then passed to the routine which interrupts the COT for the current CPU.

If the FEC code is in the range from 1 to 5, then the current FEC item is a normal transaction item which has been advancing time. In this event the current CPU in the general simulation table is set equal to the current FEC code and the current operating transaction for that CPU is set equal to the FEC itme. At this point the current clock time is set equal to the time off FEC for the current FEC item, and the advance time is added to the total CPU time for that transaction item. The time off FEC and the advance time fields for the current FEC item are then set equal to zero, and the FEC pointer is set so that a call to subroutine S202 may remove this item and code from the future events chain. A RETURN is then executed, transferring control back to SM2.

If the FEC code is in the range from 21 to 31, the current FEC item is an IO transaction item,

In this case the current CPU number is accessed from the IO transaction item. If this IOTI is for a seek operation, then the seek interrupt is tested. If the seek interrupt is disabled, then the next item on the FEC is examined. If the seek interrupt is not disabled, then the interrupt code in the general simulation table is set equal to 7. If this is a normal IOTI, the IO termination interrupt is tested. If the IO termination interrupt is not disabled, the interrupt code is set equal to 1. The FEC pointer is then set to the current FEC item and a call to subroutine S202 removes that item from the future events chain. The current clock time in T20 is then advanced to the appropriate end time in the IO transaction item. The IOTW is stored as the next IO transaction item in the current operating CPU. At this point, control is passed to the routine which interrupts the current operating transaction for the current operating CPU.

If the FEC code is equal to 99 the current FEC item is a clock transaction. In this event the

current CPU number is set equal to the FEC item. If the clock interrupt is disabled, this item on the future events chain is ignored and the pointer is advanced by 1. If the interrupt is enabled, the clock item switch in the current CPU item is set equal to zero, indicating that there is no clock item on the future events chain, and the interrupt code in T20 is set equal to 4. The current simulator time is then advanced to the clock item time, and the FEC pointer is set to point to the current FEC item, so that a call to subroutine S202 may remove it from the future events chain. At this point control is passed to the routine which will interrupt the current operating transaction for the current operating CPU.

If the FEC code is in the range from 101 to 105, the current FEC item is a generated interrupt. In this case, the current CPU number is set equal to the FEC code less 100, and the interrupt number is set equal to the FEC item. If the interrupt specified is disabled,

this FEC item is ignored, and the FEC pointer is advanced 1. If this interrupt is enabled, the FEC pointer is stored and a call to subroutine S202 removes the current item from the future events chain. Control is then passed to the routine which interrupts the current operating transaction for the current operating CPU.

The routine to interrupt the current operating transaction for the new current operating CPU first examines the CPU to determine if it is cycling. If the CPU is cycling, the current simulated time less the start cycle time is added to the total cycle time for that CPU. At this point a RETURN is executed and control is transferred to SM2. If the current CPU is not cycling, the future events chain is searched for a transaction item for the current CPU. If no transaction item is found on the future events chain, the error code is set equal to 3 and a RETURN is executed.

Once this transaction has been found, the time to go is calculated by subtracting the time to go from

the original advance time. The elapsed time is then added to the total CPU time used by this transaction item and the advanced time is reset using the new time-to-go figure. This transaction item is then restored to its position as COT for the current operating CPU. The FEC item pointer is then stored so that a call to subroutine S202 may remove the COT entry from the future events chain. A RETURN is then executed transferring control back to SM2.

SUBROUTINE S110 INTERRUPT HANDLER

The INTERRUPT HANDLER examines the current operating transaction for the current operating CPU. If the COT is not negative, the AT and IOT for the current CPU are stored into the COT's transaction item. At this point the current COT is made the AT and the next IOT is moved into the IOT slot for the current CPU. If the original COT was negative, indicating a cycle transaction item, the storing of the AT and IOT is ignored. The interrupt number is then accessed from the general simulation table and a test is made to insure that the interrupt is not disabled. If the interrupt is disabled, the error code is set equal to 6 and a RETURN is executed.

If the interrupt number is equal to 7, indicating an end of seek interrupt, the IO transaction item type is altered from a seek to a seek continue. At this point, for a seek or any IO termination interrupt, the device, channel or control unit which is to be set free is released. Interrupts which

are not caused by IO activity bypass this releasing of facilities and go directly to the routine which creates an operating system transaction item.

In order to create an operating system transaction item, the operating system worker routine number is accessed from the CPU to be interrupted and stored into the general simulation table for use by subroutine S203. A call to subroutine S203 then creates the operating system transaction item. Upon return of control from S203 the interrupt number field is zeroed and the current program type is set equal to 1. The current operating transaction for the CPU to be interrupted is then set equal to the newly generated transaction item. The interrupt address which was calculated for the test to insure that the current interrupt was not disabled is now set into the new COT. This enables the new operating system transaction to begin executing instructions at the right location. In addition, the creation time for the new COT is set equal to the current simulated time. At this point a RETURN is executed which will transfer control back to SM2.

SUBROUTINE S201 PUT FEC

The PUT FEC subroutine examines the last item on the future events chain to determine if the chain is full. If the chain is full, an error code of 1 is set and a direct call to subroutine S101, the error handler, is made. If the future events chain is not full, the pointer is set to the start of the future events chain, and the time each item is due to come off the chain is compared to the time the new item is due to come off the chain. As soon as an item is found which is due to come off the chain after the new iten, that item and all subsequent items are moved down by 1 to provide a space for the new item. At this point, the new FEC code and the new FEC item are placed on to the future events chain and the FEC item count is incremented by 1. A RETURN is then executed and control is transferred back to the calling subroutine.

SUBROUTINE S202 REORGANIZE FEC

The REORGANIZE FEC subroutine accesses a pointer to the item to be removed and the FEC counter from the general simulation table. The item after the item to be removed and all subsequent items are then shifted down from the chain by one position, overlaying the item to be removed. Notice that in this subroutine the future events chain has been EQUIVALENCED to an integer array. This allows both the FEC code and the FEC item for a single entry on the future events chain to be moved with a single instruction. When this operation has been completed, the last entry in the future events chain is set equal to zero and the FEC item count is reduced by one. A RETURN is then executed transferring control back to the calling subroutine.

SUBROUTINE S203 GENERATE TRANSACTION ITEM

The GENERATE TRANSACTION ITEM subroutine accesses the worker routine number for the transaction to be generated from the general simulation table. If the transaction type is less than 3 indicating that the transaction will be for an Operating System or Primary Worker, a pointer is set to the high end of the transaction item table. For other types of transaction items the pointer is set to the start of the transaction item table. The transaction item table is then searched for a free entry, which is indicated by a -1 in the first word of the entry. If there is no free entry in the transaction item table, the error code is set equal to 1, and the error handler subroutine, S101, is called.

Once a blank entry has been found, it's address is stored back into T20 and will be used as the address for the transaction item until it is destroyed. The part 1 transaction item record will then be initialized, using the information stored

in the worker routine base. Once the part 1 record has been initialized, the transaction item table pointer is incremented by 1 and the search for the next entry takes place. The address of the time portion of the transaction item is then stored into the transaction item part 1 record, and the time section is initialized to all zeroes. After the time section has been initialized, the transaction item table pointer is again incremented by 1 and a search is made for the file portion of the transaction item. The address of this part 2 transaction item entry is also stored into the transaction item part 1. If this worker routine has more that 7 files associated with it, there will be additional part 2 transaction item entries in the transaction item table. These entries are chained together using the last word of each entry to store the address of the next entry. The last entry in this chain will contain a zero in the last word. When the transaction item has been generated, a RETURN is executed transferring control back to the calling subroutine.

SUBROUTINE S204 GENERATE IO TRANSACTION

The GENERATE IOTI subroutine accesses the transaction item for which this IOTI is to be generated from the general simulation table. It then searches the IO transaction item table looking for a blank entry, which is indicated by a -1 in the first word of an entry. If the IOTI table is full, the error code is set equal to 1 and the error handler subroutine S101 is called.

Once a blank entry has been found in the IOTI table, the transaction item which created this IOTI and the current CPU are stored in the new IOTI. If this IOTI was created as a result of an OPEN, CLOSE, READ, or WRITE statement, the real and ordinal file numbers are also stored in the IOTI. If this IOTI was created as the result of a function statement, the real and ordinal file numbers in the IOTI are set equal to zero. All other entries in this IOTI are initialized to zero. The number of IOTI's outstanding in the transaction item creating this IOTI is then incremented by 1. At this point, a RETURN is executed and control is transferred back to the calling subroutine.

SUBROUTINE S210 UNLOAD MEMORY STATISTICS

The UNLOAD MEMORY STATISTICS subroutine updates the memory load field for each memory being used in the current simulation. The memory load field in the memory table contains an entry which is defined as page microseconds. This field is updated by taking the current simulator time and subtracting the last memory reference time. This value is then multiplied by the current number of pages busy in this memory and the result is added to the memory load field. At this point, the last reference time for this memory is set equal to the current simulator time. When this operation has been performed for all memories used in the current simulation, a RETURN is executed and control is transferred back to the calling subroutine.

SUBROUTINE S211 SET MEMORY LOAD

The SET MEMORY LOAD subroutine counts the number of pages busy in each of the memories used by the current simulation. The result of this accumulation is stored in the number of pages busy field for each of these memories. When this operation has been completed, a RETURN is executed and control is transferred back to the calling subroutine.

Subroutine S212 RANDOM NUMBER GENERATOR

The RANDOM NUMBER GENERATOR subroutine first stores its subroutine number into the general simulation table. It then zeroes the output field in the general simulation table. The current random number is then multiplied by the random number seed. The result of this operation is set positive, and then converted from a binary fraction to a decimal fraction. The result of this operation is multiplied by 100 which will provide a random number in the range from zero to 99. The value obtained by this operation is stored into the general simulation table and a RETURN is executed transferring control back to the calling subroutine.

WORKING PAPER

SECTION II

S^3 FLOW CHARTS

PREPARED FOR

DEPARTMENT OF THE ARMY

UNDER

DEFENSE SUPPLY SERVICE
CONTRACT
#DA 49-083-OSA-3306

BY

C-E-I-R, Incorporated
5272 River Road
Washington, D. C. 20016

WORKING PAPER

INDEX

Description	Page
S-3 SIMULATOR	1
MAIN PROGRAM	2
SM1 SIMULATOR INITIALIZATION	3
ZERO	16
C1 READ CARD TYPE 1	18
C2 READ CARD TYPE 2	19
C3 READ CARD TYPE 3	20
C4 READ CARD TYPE 4	21
C5 READ CARD TYPE 5	22
C6 READ CARD TYPE 6	23
C7 READ CARD TYPE 7	24
C8 READ CARD TYPE 8	25
C9 READ CARD TYPE 9	26
C10 READ CARD TYPE 10	27
C21 READ CARD TYPE 21	28
C22 READ CARD TYPE 22	29
C23 READ CARD TYPE 23	30
C24 READ CARD TYPES 24 AND 25	31
C26 READ CARD TYPE 26	33
C27 READ CARD TYPE 27	34
C28 READ CARD TYPE 28	35

INDEX

Description	Page
C29 READ CARD TYPE 29	36
C31 READ CARD TYPE 31	37
C32 READ CARD TYPES 32 AND 33	38
C34 READ CARD TYPE 34	39
C35 READ CARD TYPES 35 AND 36	40
SM2 SIMULATION EXEC	41
ST1 QUEUE STATISTICS	55
ST2 MEMORY STATISTICS	56
ST3 FILE STATISTICS	57
ST4 DEVICE STATISTICS	58
ST5 CONTROL UNIT STATISTICS	59
ST6 CHANNEL STATISTICS	60
ST7 CPU STATISTICS	62
ST8 QUEUE TRANSACTION ANALYSIS	64
STAT TRANSATION ITEM ANALYSIS	66
T DUMP	68
SNAP	69
TRACE - SS51	70
POIS POISSON TABLE LOOKUP	71
DT 1	72
DT 2	73

INDEX

Description	Page
DT 3	74
DT 4	75
DT 5	76
DT 6	77
DT 7	78
DT 8	79
DT 9 AND DT 10	80
DT 12	81
DT 13	82
DT 14	83
DT 15	84
DT 16	85
DT 17	86
DT 18	87
DT 19	88
DT 20	89
DT 21	90
DT 22	91
DT 23	92
DT 24	93
DT 25	94

INDEX

Description	Page
S1 TRANSFER	95
S2 TRANSFER ON PROBABILITY	96
S3 READ	97
S4 WRITE	98
S5 FUNCTION	99
S6-FOF	100
S7 SUBROUTINE	101
S8 EXIT	102
S9 LOOP	103
S10 MOVE	104
S11 MOVE-EDIT	105
S12 COMPUTE	106
S13 MATH	107
S14 OPEN	108
S15 CLOSED	110
S16 TERM	112
S17 PRINT	114
S18 CALL	115
S19 EXEC	116
S31 MEMORY	117
S32 ALLOCATE	123
S33 DEALLOCATE	128

INDEX

Description	Page
S34 PACK	129
S35 EXAMINE FIRST	131
S36 EXAMINE NEXT	133
S37 EXAMINE LAST	134
S38 PLACE	135
S39 SELECT	138
S40 BUFF	140
S41 SEEK	141
S42 IO READY	142
S43 IO ADVANCE	149
S44 IO TERM	157
S45 SET	159
S46 RESET SW	160
S47 TEST SW	161
S48 INTERRUPT	162
S49 DISABLE	163
S50 ENABLE	165
S51 CLOCK	166
S52 RETURN	167
S53 ACTIVATE	169
S54 RECEIVE	170
S55 CYCLE	172

INDEX

Description	Page
S56 DESTROY	173
S57 PERIPHERAL	174
S59 MATCH	177
S61 CALLOUT	178
S62 PRIORITY	179
S101 ERROR HANDLER	181
S102 PUT IOTT ON FEC	182
S103 PUT CLOCK ITEM ON FEC	186
S104 PLACE IT ON FEC	187
S110 INTERRUPT HANDLER	194
S201 PLACE ITEM ON FUTURE EVENTS CHAIN	201
S202 REORGANIZE FUTURE EVENTS CHAIN	204
S203 GENERATE TI	205
S204 GENERATE IOTI	207
S210 CALCULATE MEMORY LOAD	208
S211 SET MEMORY PAGES BUSY	209
S212 RANDOM NUMBER GENERATOR	210
CVNT	211

December 1, 1967

S-3 Simulator

The Implicit Coding Convention for S-3

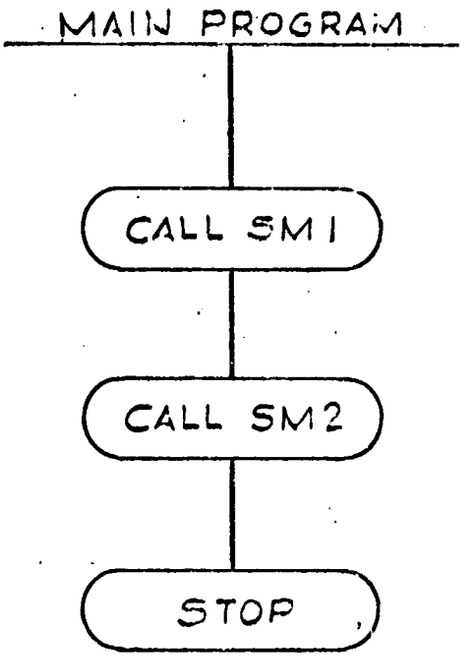
At the start of every subroutine in the S-3 simulator is an implicit statement which specifies the type of constants and arrays used in the subroutine.

Any variable or array which begins with the letters (A-H) contains a real single precision value and will occupy one word in the System 360.

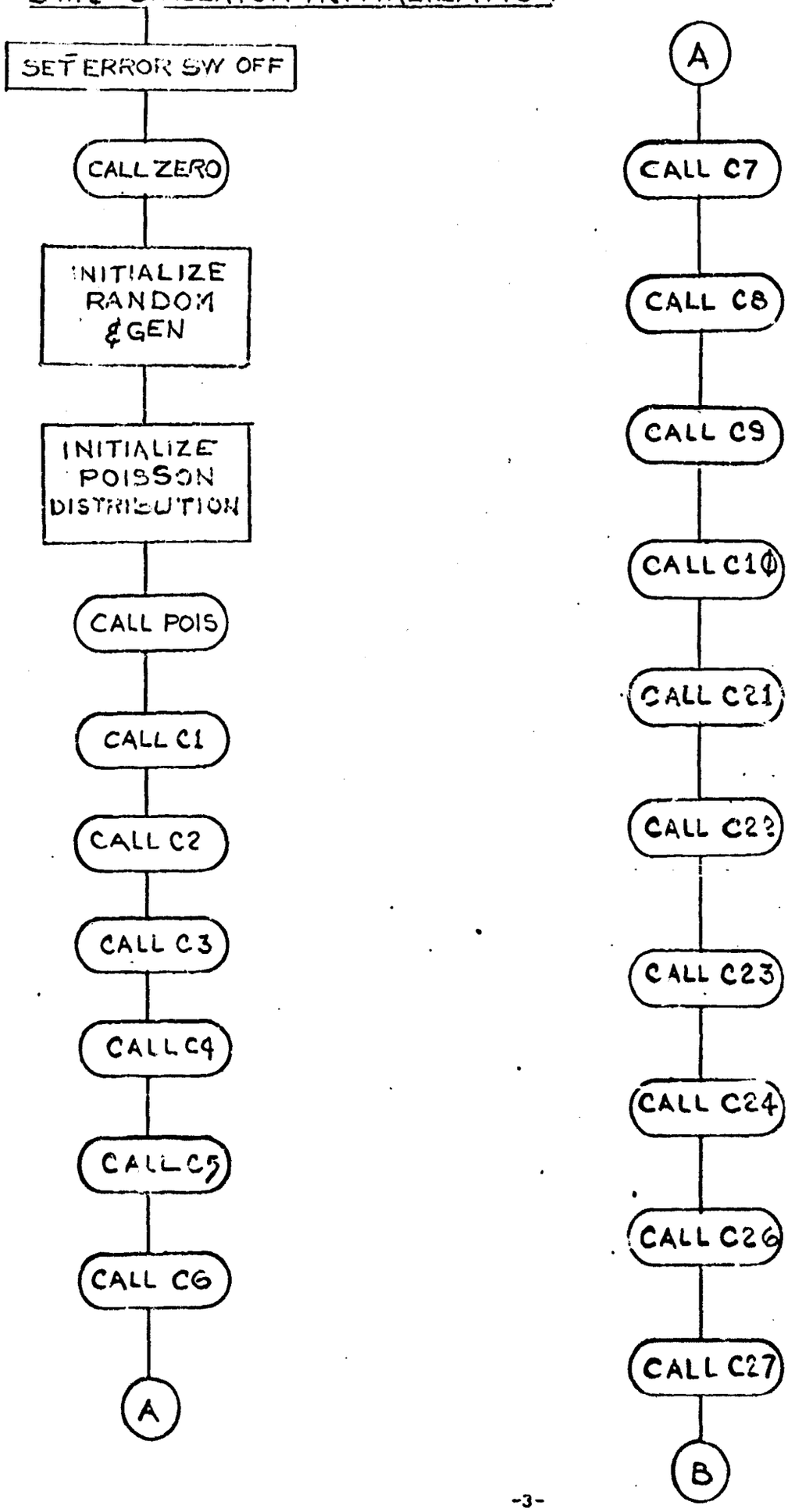
Any variable or array which begins with the letters (I-N) contains an integer value and will occupy one word in the System 360.

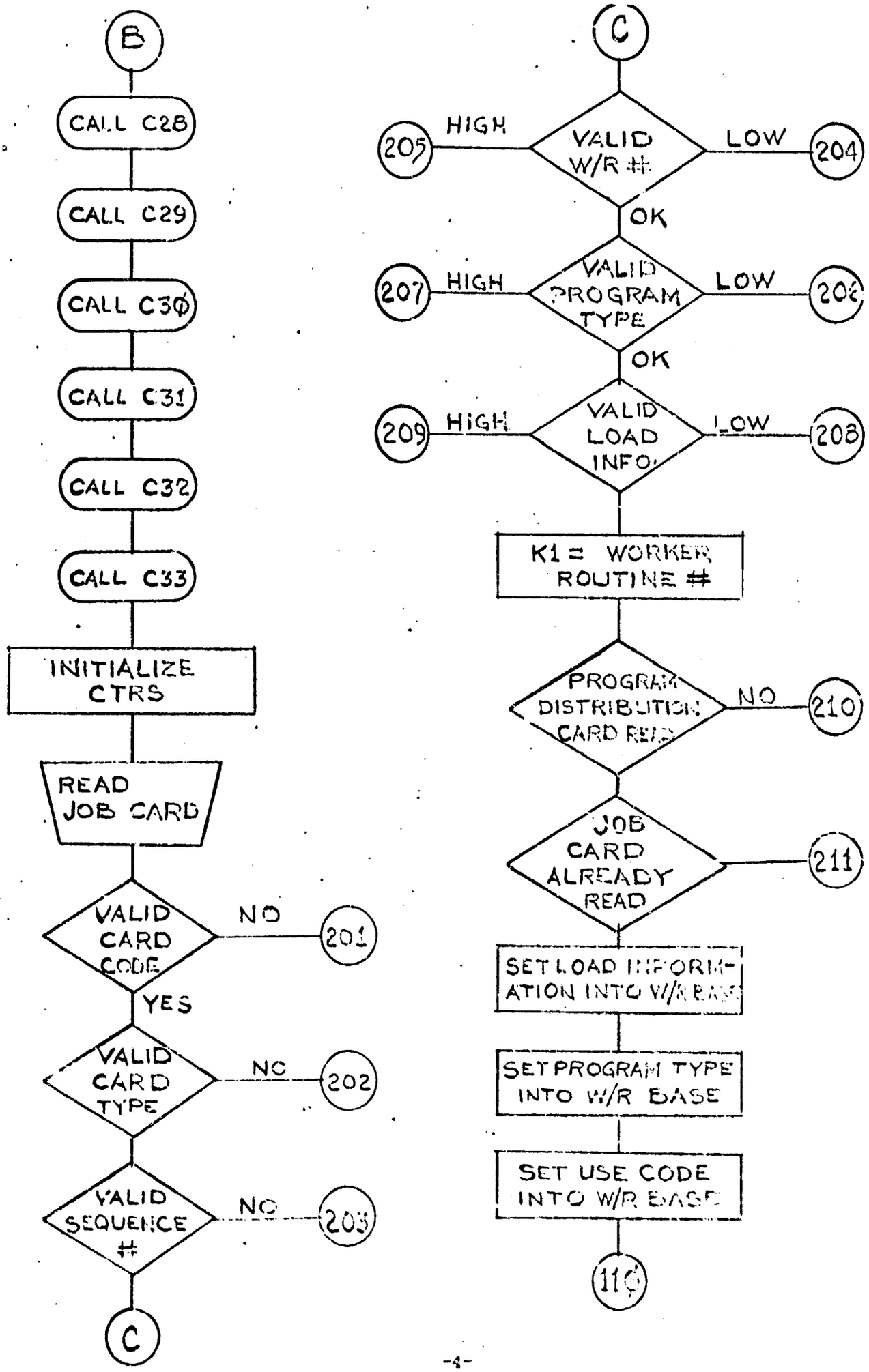
Any variable or array which begins with the letters (P-Y) contains an integer value and will occupy one half-word in the System 360.

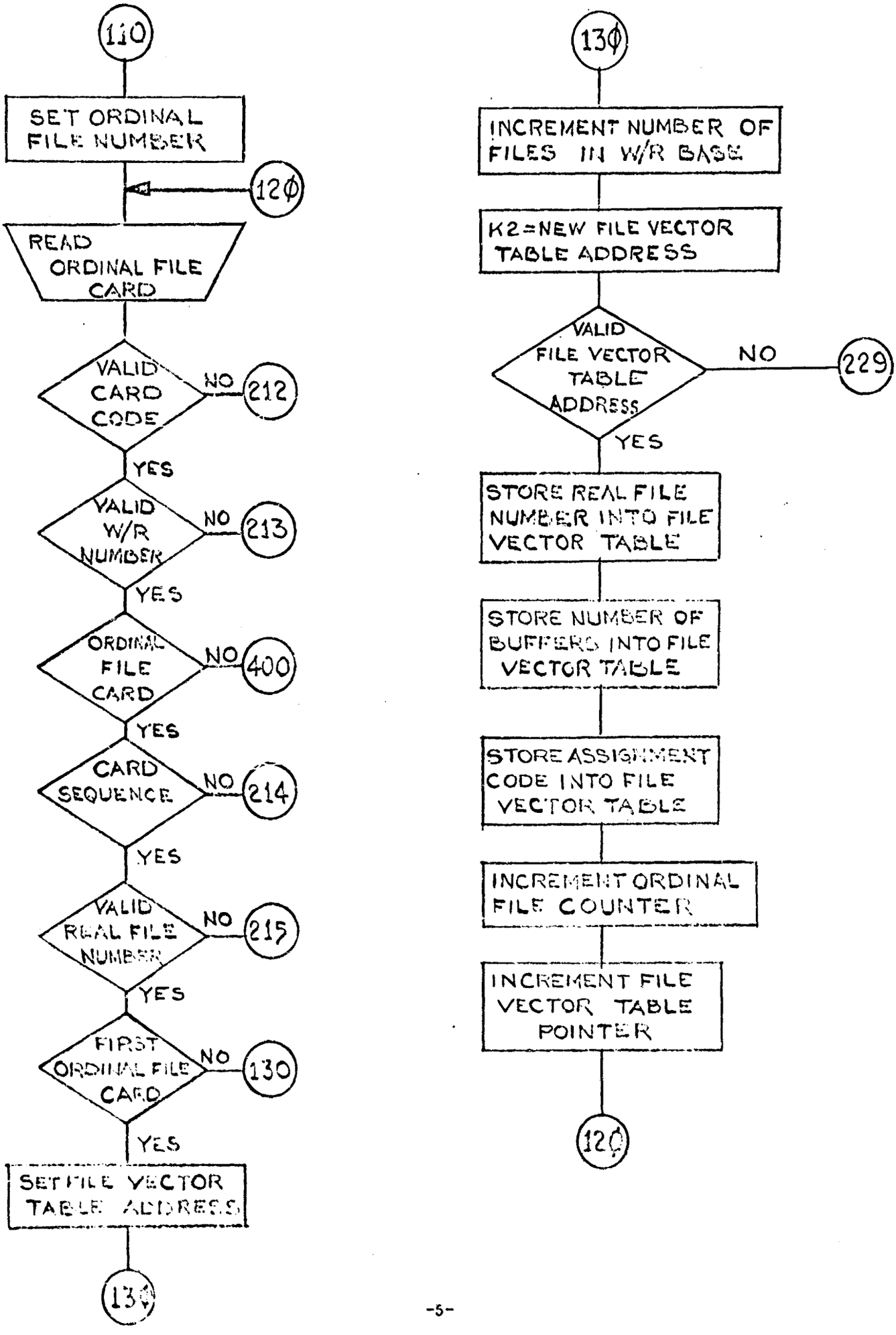
Any variable or array which begins with the letter (Z) contains a real double-precision value and will occupy two words (one double word) in the System 360.

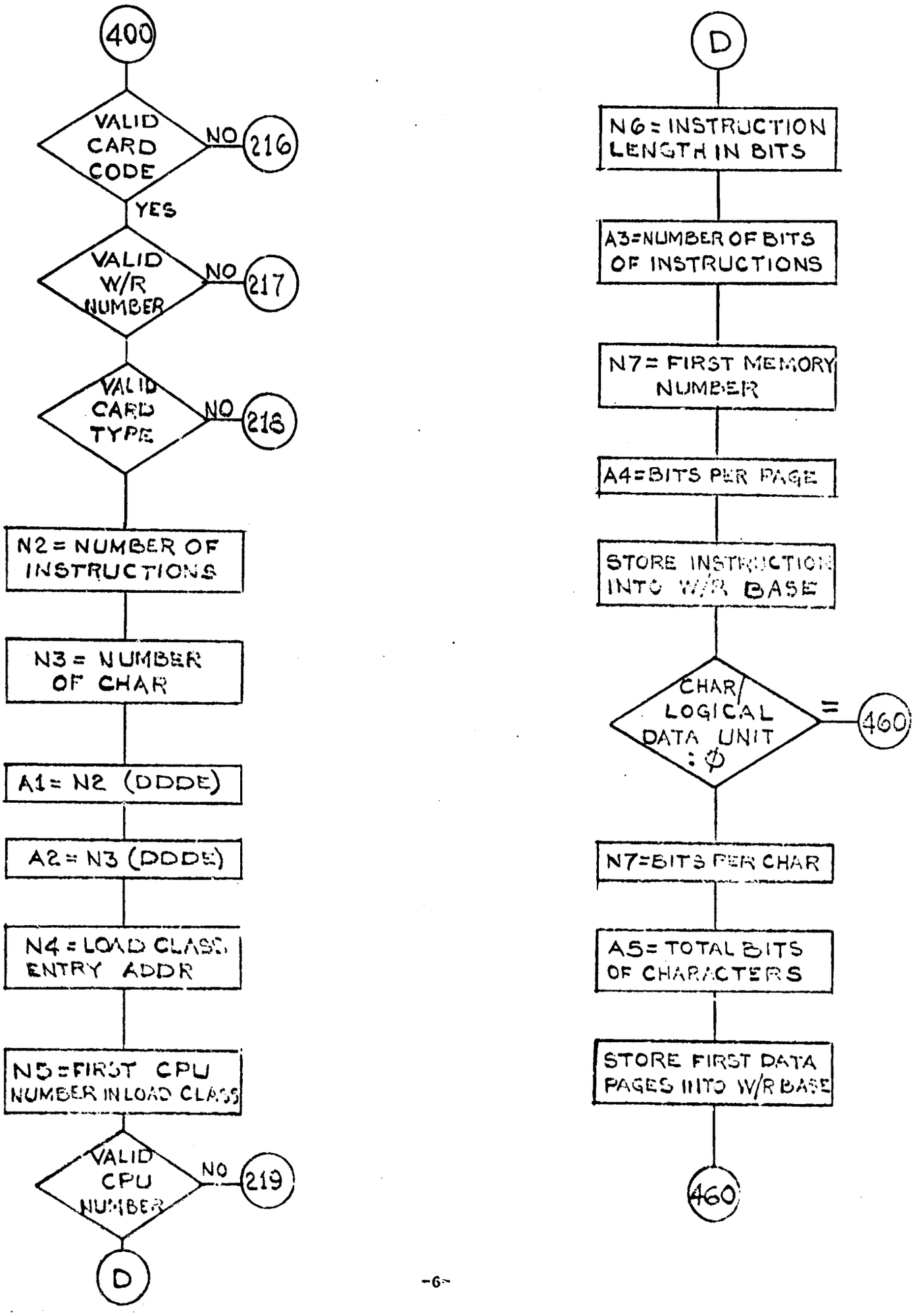

MAIN PROGRAM
CALL SM1
CALL SM2
STOP

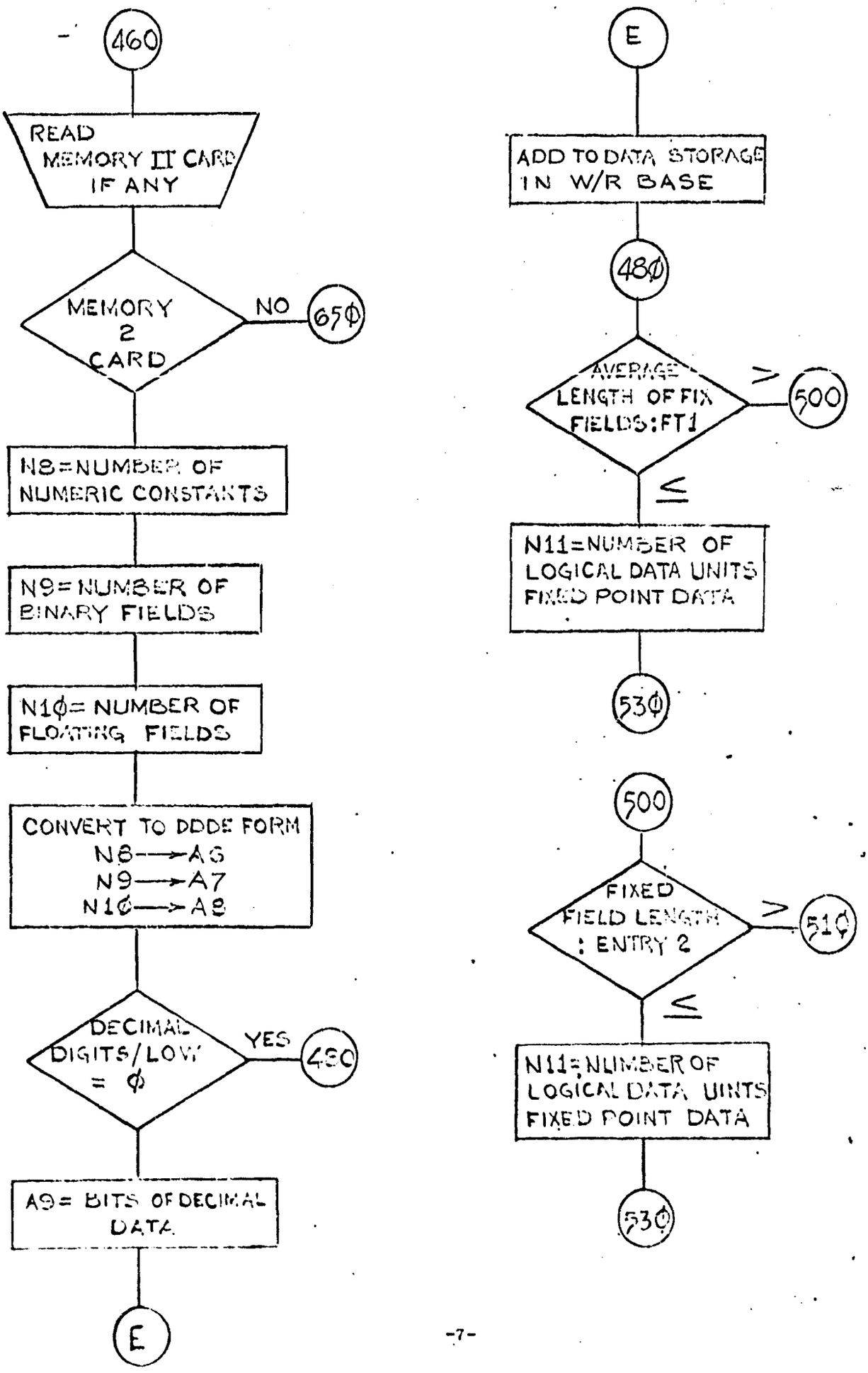
SM1 SIMULATOR INITIALIZATION

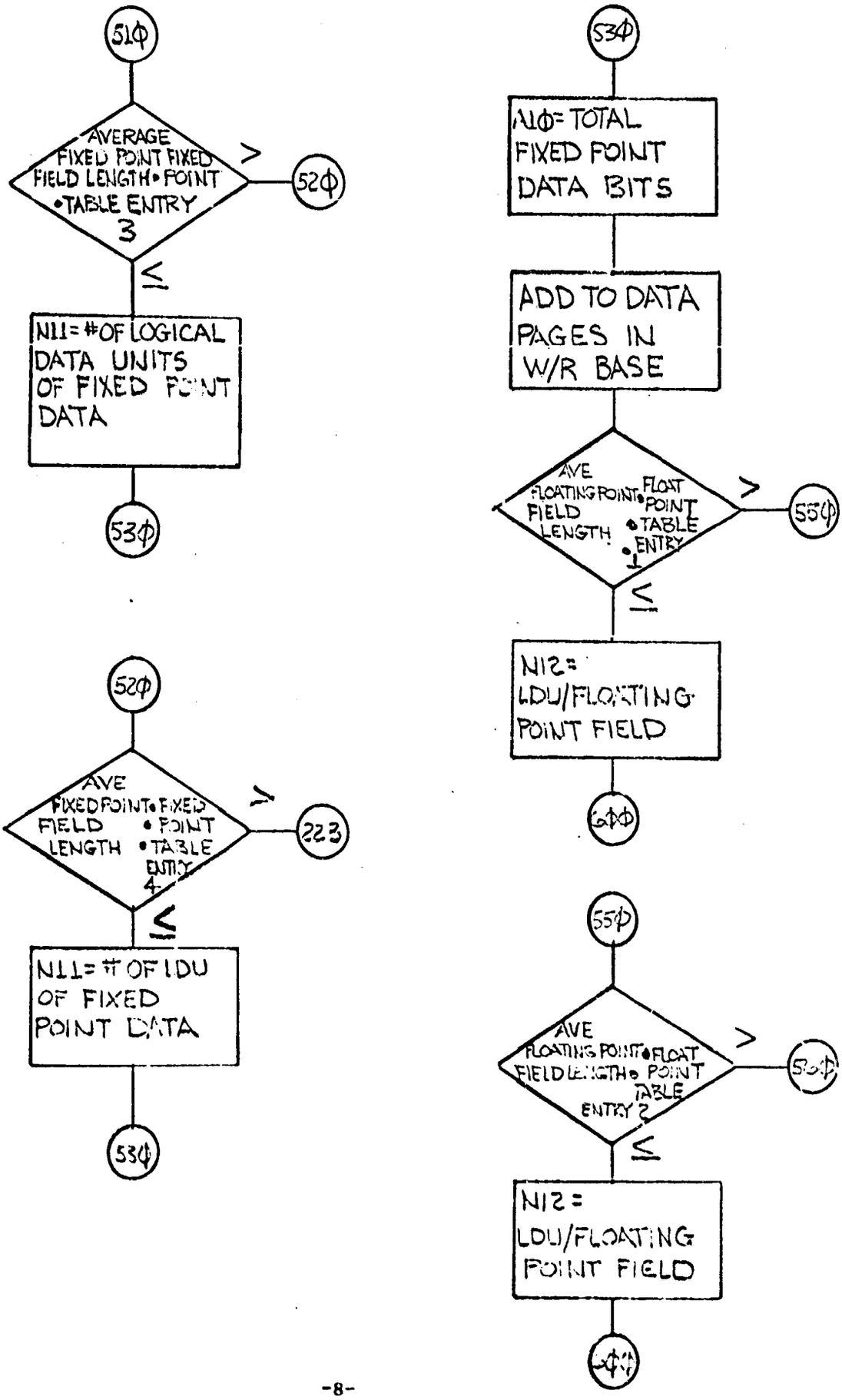


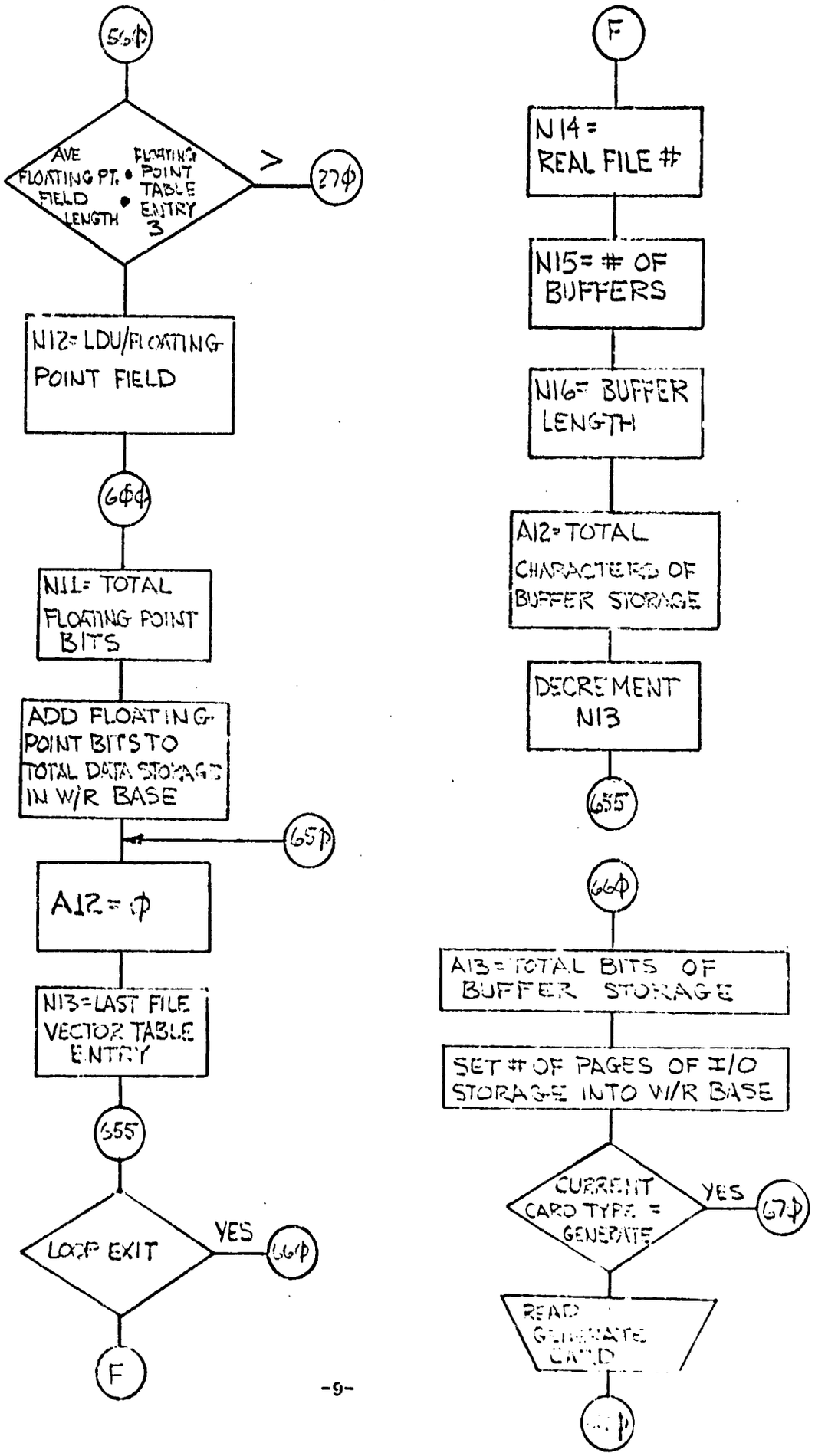

B
CALL C28
CALL C29
CALL C30
CALL C31
CALL C32
CALL C33
INITIALIZE CTRS
READ JOB CARD
VALID CARD CODE
NO
201
YES
VALID CARD TYPE
NO
202
VALID SEQUENCE #
NO
203
C
C
VALID W/R #
HIGH
205
LOW
204
OK
VALID PROGRAM TYPE
HIGH
207
LOW
206
OK
VALID LOAD INFO.
HIGH
209
LOW
208
K1 = WORKER ROUTINE #
PROGRAM DISTRIBUTION CARD READ
NO
210
JOB CARD ALREADY READ
211
SET LOAD INFORMATION INTO W/R BASE
SET PROGRAM TYPE INTO W/R BASE
SET USE CODE INTO W/R BASE
110

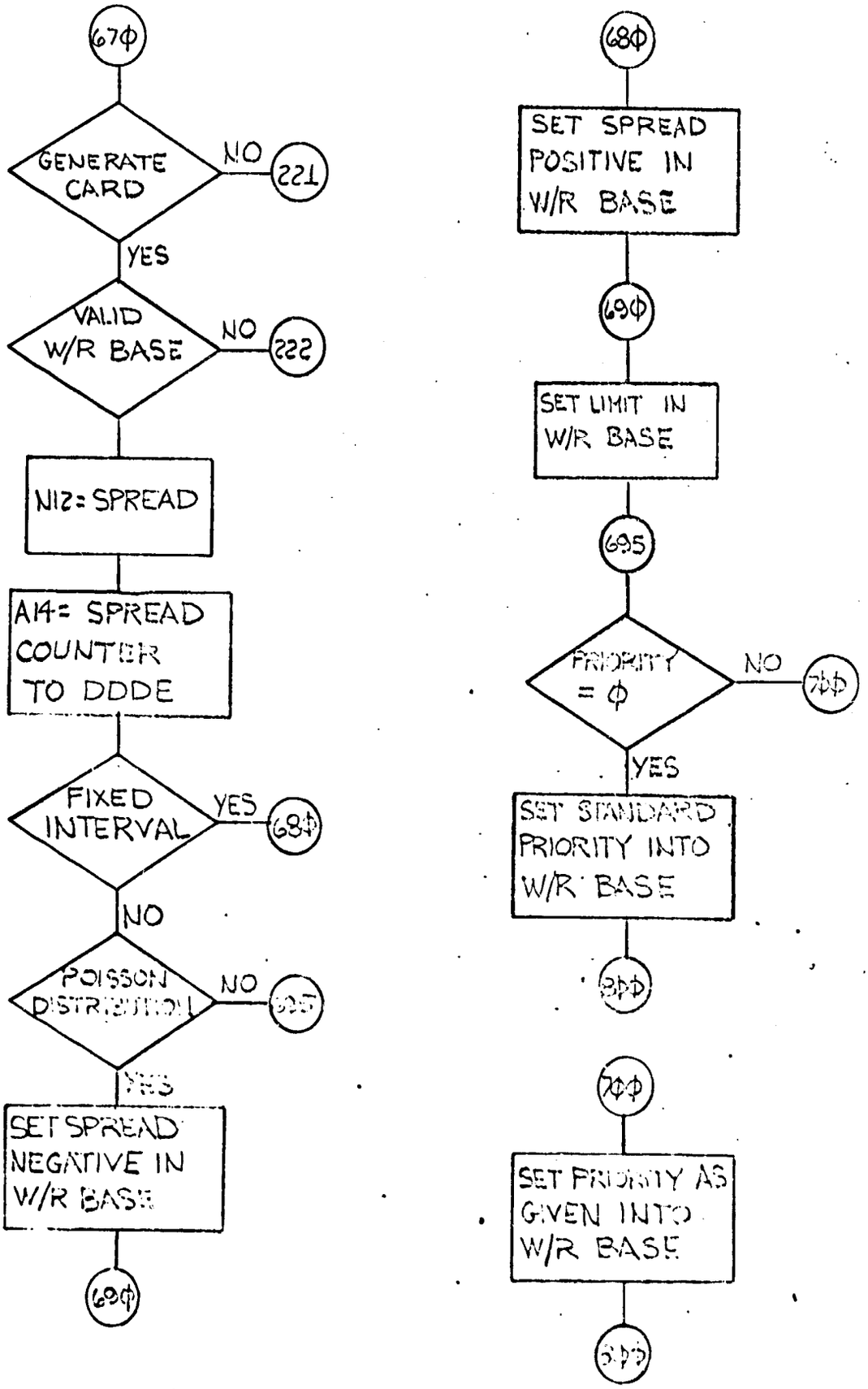

110
SET ORDINAL FILE NUMBER
120
READ ORDINAL FILE CARD
VALID CARD CODE
NO
212
YES
VALID W/R NUMBER
NO
213
YES
ORDINAL FILE CARD
NO
400
YES
CARD SEQUENCE
NO
214
YES
VALID REAL FILE NUMBER
NO
215
YES
FIRST ORDINAL FILE CARD
NO
130
YES
SET FILE VECTOR TABLE ADDRESS
130
INCREMENT NUMBER OF FILES IN W/R BASE
K2=NEW FILE VECTOR TABLE ADDRESS
VALID FILE VECTOR TABLE ADDRESS
NO
229
YES
STORE REAL FILE NUMBER INTO FILE VECTOR TABLE
STORE NUMBER OF BUFFERS INTO FILE VECTOR TABLE
STORE ASSIGNMENT CODE INTO FILE VECTOR TABLE
INCREMENT ORDINAL FILE COUNTER
INCREMENT FILE VECTOR TABLE POINTER
120

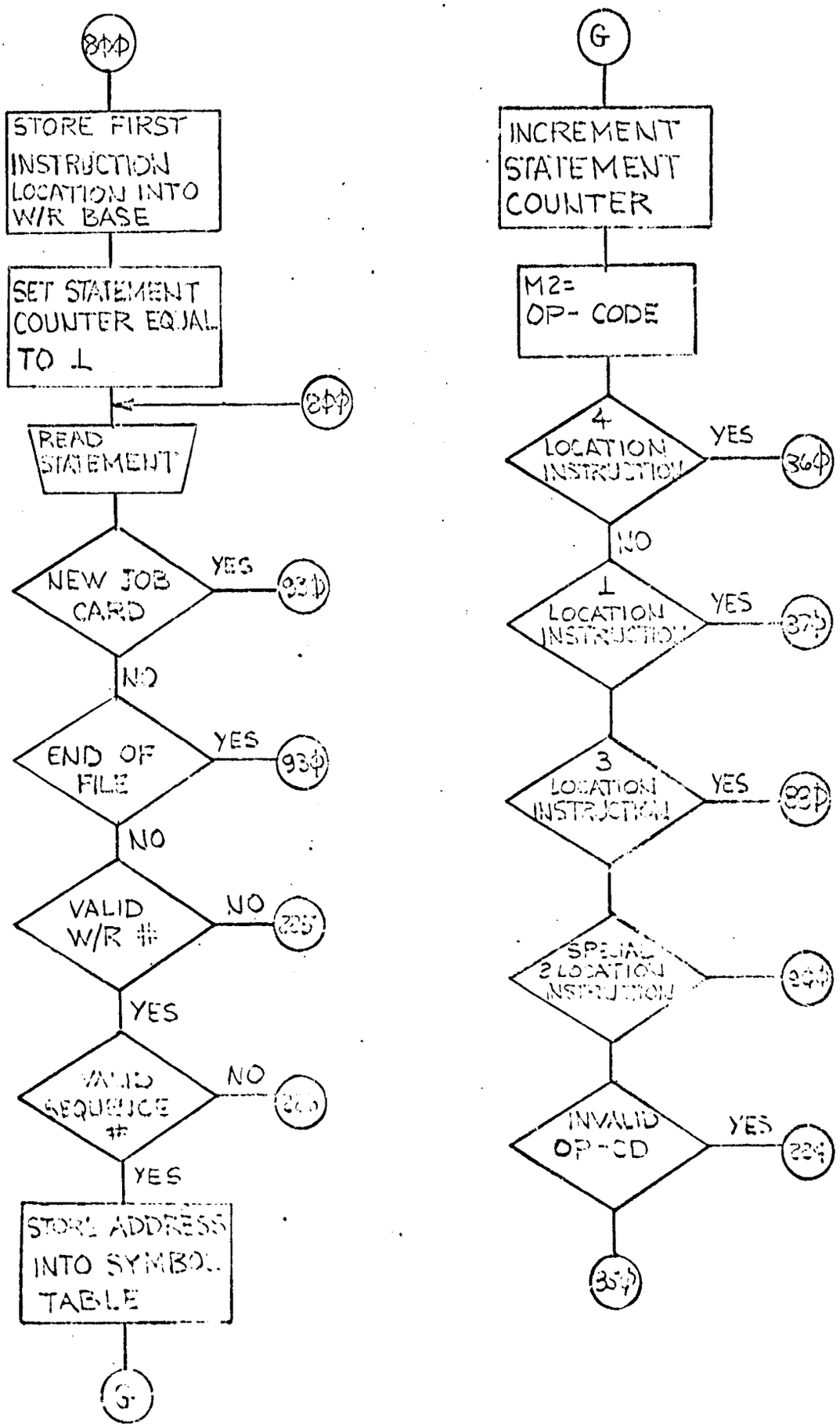

400
VALID CARD CODE
NO
216
YES
VALID W/R NUMBER
NO
217
VALID CARD TYPE
NO
218
N2 = NUMBER OF INSTRUCTIONS
N3 = NUMBER OF CHAR
A1 = N2 (DDDE)
A2 = N3 (DDDE)
N4 = LOAD CLASS ENTRY ADDR
N5 = FIRST CPU NUMBER IN LOAD CLASS
VALID CPU NUMBER
NO
219
D
D
N6 = INSTRUCTION LENGTH IN BITS
A3 = NUMBER OF BITS OF INSTRUCTIONS
N7 = FIRST MEMORY NUMBER
A4 = BITS PER PAGE
STORE INSTRUCTION INTO W/R BASE
CHAR/ LOGICAL DATA UNIT : Ø
=
460
N7 = BITS PER CHAR
A5 = TOTAL BITS OF CHARACTERS
STORE FIRST DATA PAGES INTO W/R BASE
460

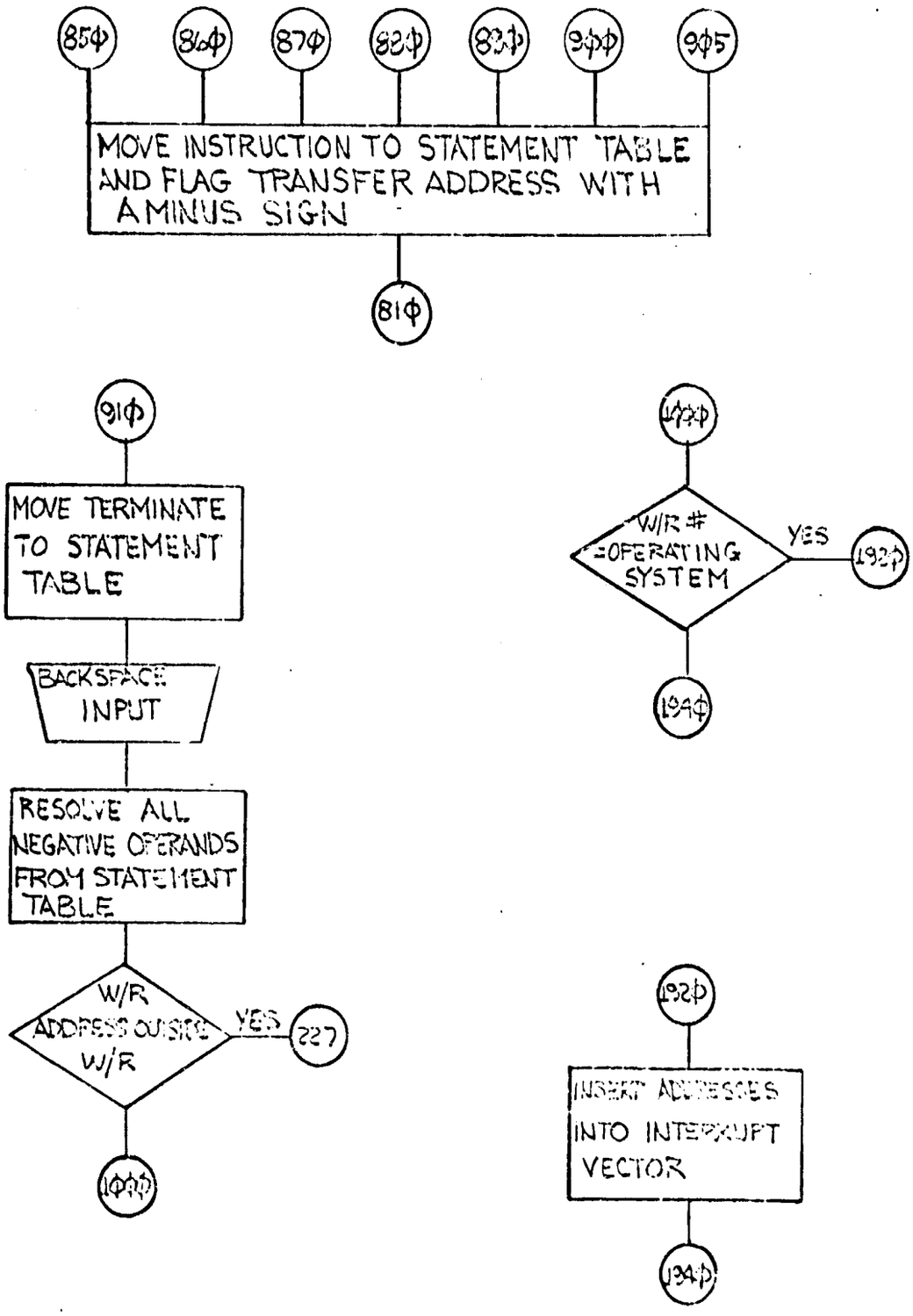

460
READ MEMORY II CARD IF ANY
MEMORY 2 CARD
NO
650
N8=NUMBER OF NUMERIC CONSTANTS
N9=NUMBER OF BINARY FIELDS
N10= NUMBER OF FLOATING FIELDS
CONVERT TO DDDE FORM
N8 → A6
N9 → A7
N10 → A8
DECIMAL DIGITS/LOW = 0
YES
490
A9= BITS OF DECIMAL DATA
E
E
ADD TO DATA STORAGE IN W/R BASE
480
AVERAGE LENGTH OF FIX FIELDS:FT1
>
500
≤
N11=NUMBER OF LOGICAL DATA UNITS FIXED POINT DATA
530
500
FIXED FIELD LENGTH : ENTRY 2
>
510
≤
N11=NUMBER OF LOGICAL DATA UINTS FIXED POINT DATA
530

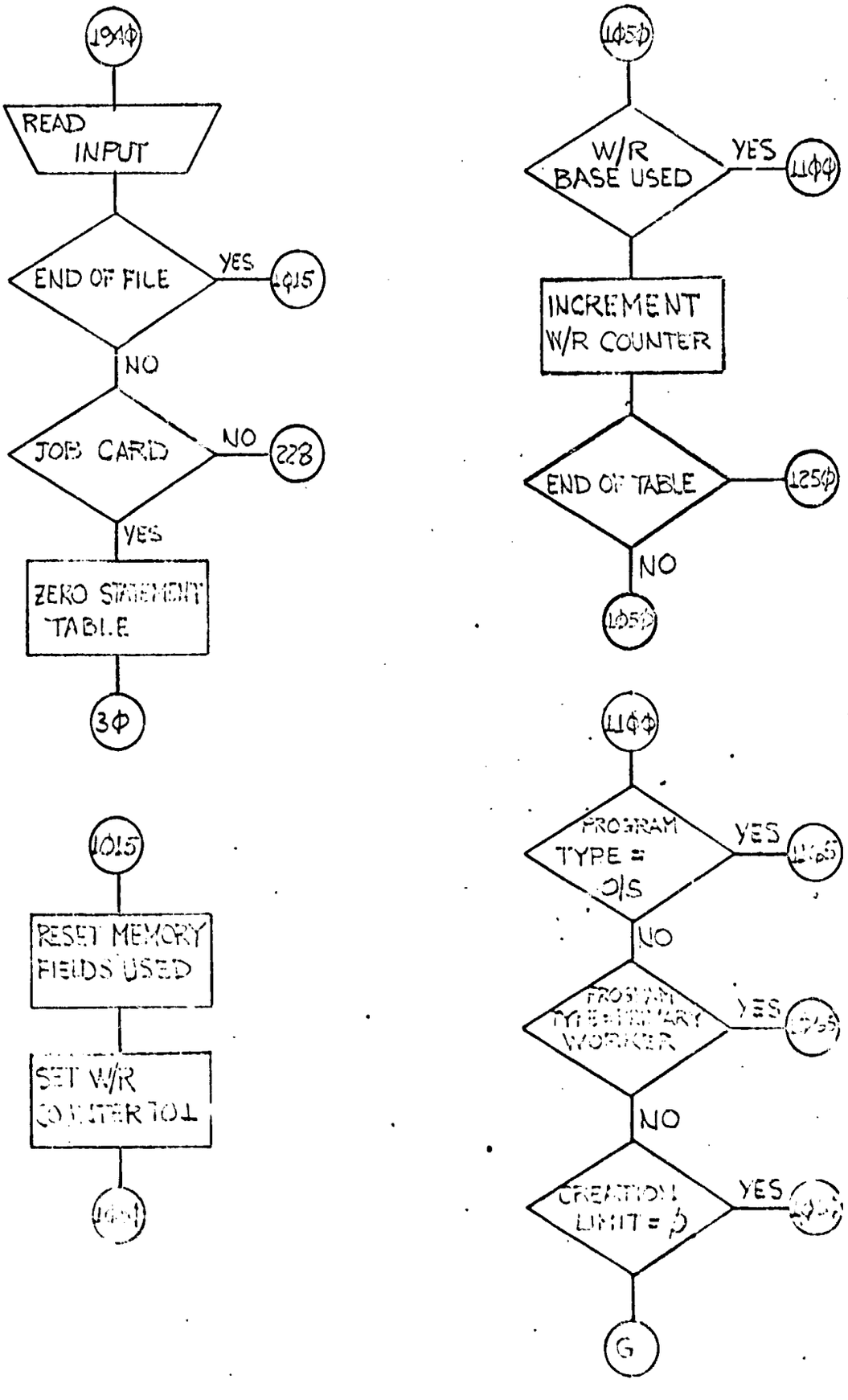

510
AVERAGE FIXED POINT FIELD LENGTH • FIXED POINT • TABLE ENTRY 3
>
520
≤
N11 = # OF LOGICAL DATA UNITS OF FIXED POINT DATA
530
534
N10 = TOTAL FIXED POINT DATA BITS
ADD TO DATA PAGES IN W/R BASE
AVE FLOATING POINT FIELD LENGTH • FLOAT POINT • TABLE ENTRY 1
>
550
≤
N12 = LDU/FLOATING POINT FIELD
540
520
AVE FIXED POINT FIELD LENGTH • FIXED POINT • TABLE ENTRY 4
>
223
≤
N11 = # OF LDU OF FIXED POINT DATA
530
550
AVE FLOATING POINT FIELD LENGTH • FLOAT POINT TABLE ENTRY 2
>
560
≤
N12 = LDU/FLOATING POINT FIELD
570

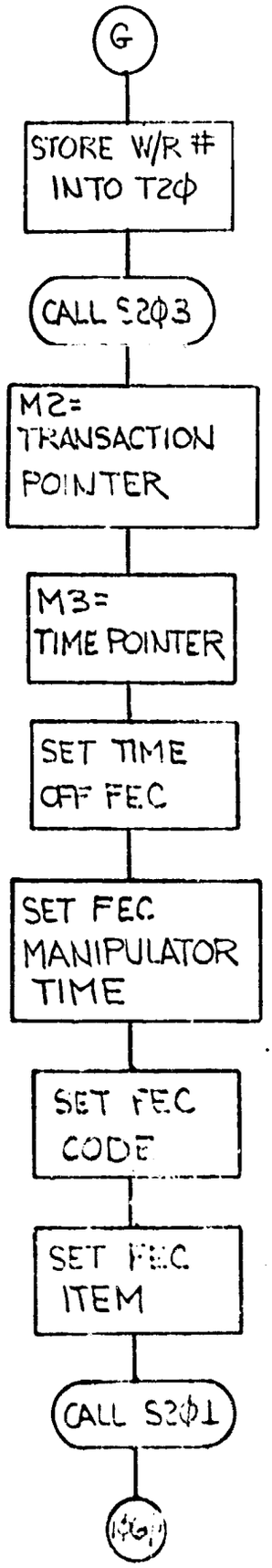

56φ
AVE FLOATING PT. FIELD LENGTH
FLOATING POINT TABLE ENTRY 3
>
27φ
N12= LDU/FLOATING POINT FIELD
64φ
N11= TOTAL FLOATING POINT BITS
ADD FLOATING POINT BITS TO TOTAL DATA STORAGE IN W/R BASE
65φ
A12 = φ
N13=LAST FILE VECTOR TABLE ENTRY
655
LOOP EXIT
YES
66φ
F
F
N14 = REAL FILE #
N15= # OF BUFFERS
N16= BUFFER LENGTH
A12= TOTAL CHARACTERS OF BUFFER STORAGE
DECREMENT N13
655
66φ
A13= TOTAL BITS OF BUFFER STORAGE
SET # OF PAGES OF I/O STORAGE INTO W/R BASE
CURRENT CARD TYPE = GENERATE
YES
67φ
READ GENERATE CARD

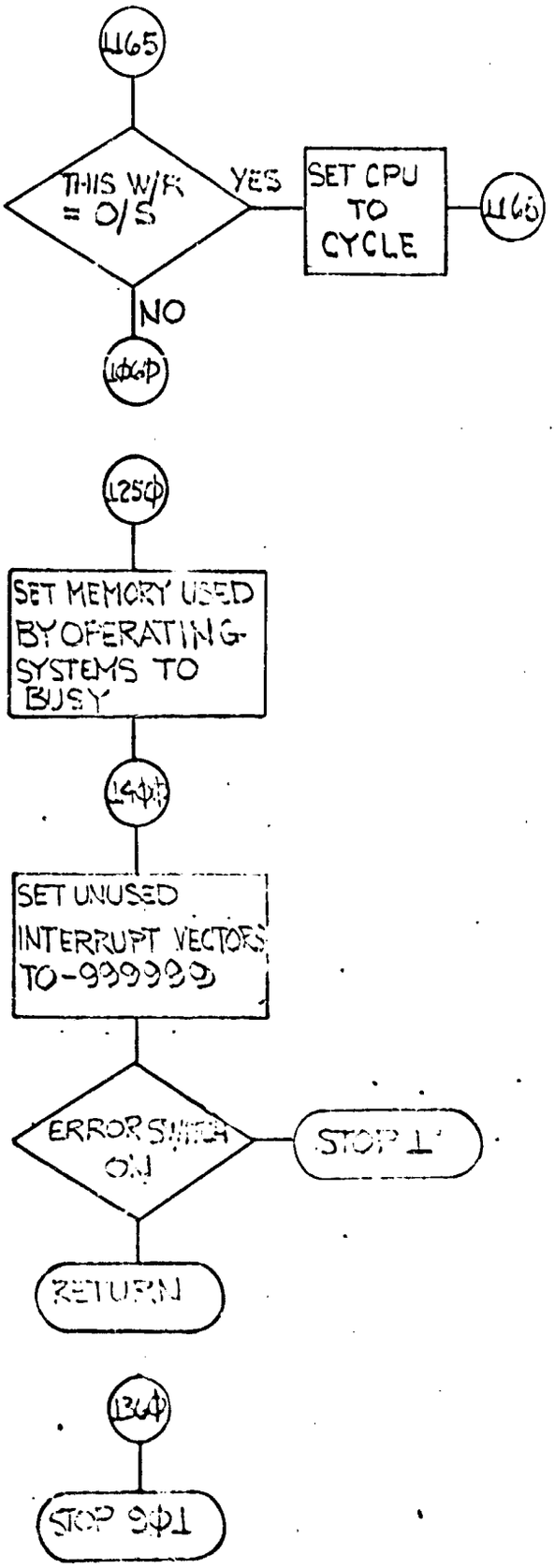

67Φ
GENERATE CARD
NO
221
YES
VALID W/R BASE
NO
222
N12= SPREAD
A14= SPREAD COUNTER TO DDDE
FIXED INTERVAL
YES
68Φ
NO
POISSON DISTRIBUTION
NO
YES
SET SPREAD NEGATIVE IN W/R BASE
69Φ
68Φ
SET SPREAD POSITIVE IN W/R BASE
69Φ
SET LIMIT IN W/R BASE
695
PRIORITY = Φ
NO
7ΦΦ
YES
SET STANDARD PRIORITY INTO W/R BASE
3ΦΦ
7ΦΦ
SET PRIORITY AS GIVEN INTO W/R BASE
3ΦΦ

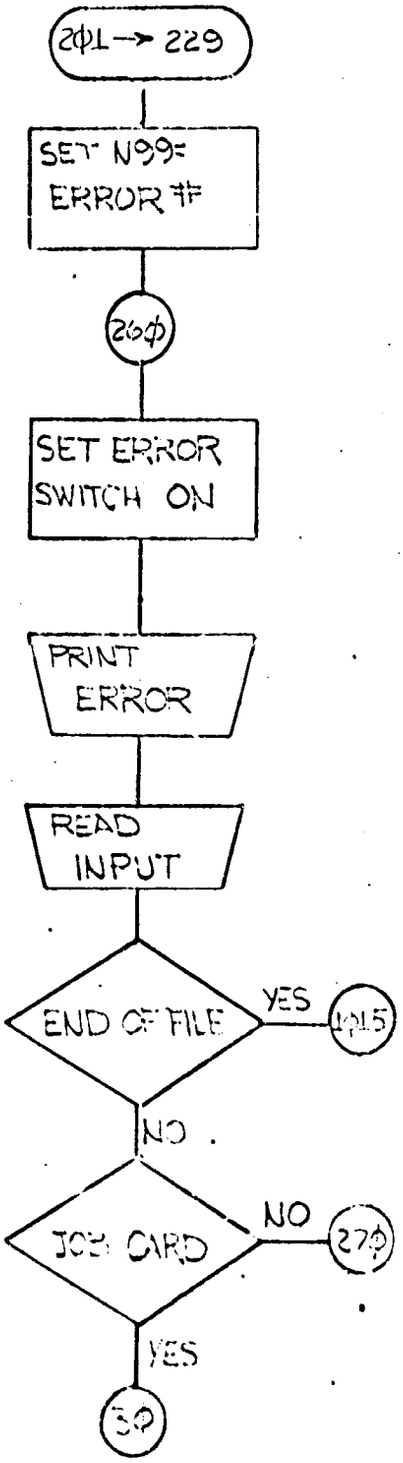

STORE FIRST INSTRUCTION LOCATION INTO W/R BASE
SET STATEMENT COUNTER EQUAL TO 1
READ STATEMENT
NEW JOB CARD
YES
NO
END OF FILE
YES
NO
VALID W/R #
NO
YES
VALID SEQUENCE #
NO
YES
STORE ADDRESS INTO SYMBOL TABLE
INCREMENT STATEMENT COUNTER
M2= OP-CODE
4 LOCATION INSTRUCTION
YES
NO
1 LOCATION INSTRUCTION
YES
3 LOCATION INSTRUCTION
YES
SPECIAL 2 LOCATION INSTRUCTION
INVALID OP-CD
YES

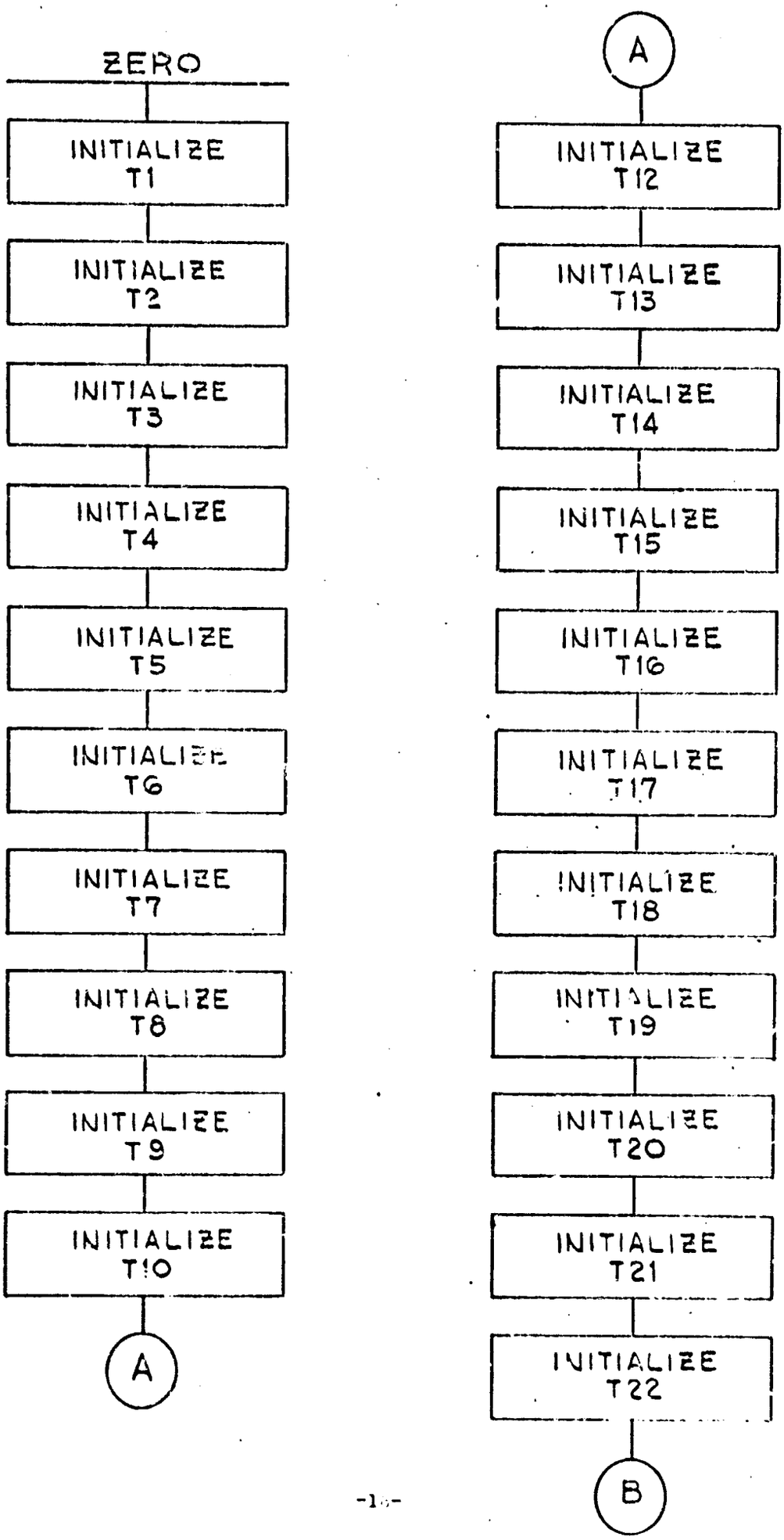

85Φ
86Φ
87Φ
82Φ
82Φ
9ΦΦ
9Φ5
MOVE INSTRUCTION TO STATEMENT TABLE AND FLAG TRANSFER ADDRESS WITH A MINUS SIGN
81Φ
91Φ
MOVE TERMINATE TO STATEMENT TABLE
BACKSPACE INPUT
RESOLVE ALL NEGATIVE OPERANDS FROM STATEMENT TABLE
W/R ADDRESS OUTSIDE W/R
YES
227
1ΦΦΦ
1ΦΦΦ
W/R # OPERATING SYSTEM
YES
192Φ
194Φ
192Φ
INSERT ADDRESSES INTO INTERRUPT VECTOR
194Φ

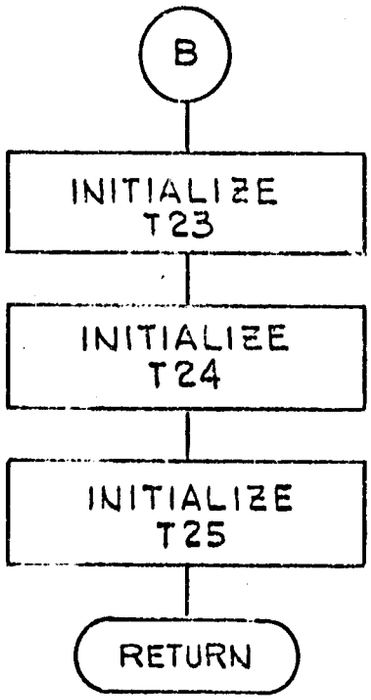

194Φ
READ INPUT
END OF FILE
YES
1Φ15
NO
JOB CARD
NO
228
YES
ZERO STATEMENT TABLE
3Φ
1Φ15
RESET MEMORY FIELDS USED
SET W/R COUNTER TO 1
1Φ51
1Φ5Φ
W/R BASE USED
YES
11ΦΦ
INCREMENT W/R COUNTER
END OF TABLE
125Φ
NO
1Φ5Φ
11ΦΦ
PROGRAM TYPE = D/S
YES
1165
NO
PROGRAM TYPE = SUMMARY WORKER
YES
1265
NO
CREATION LIMIT = Ø
YES
G

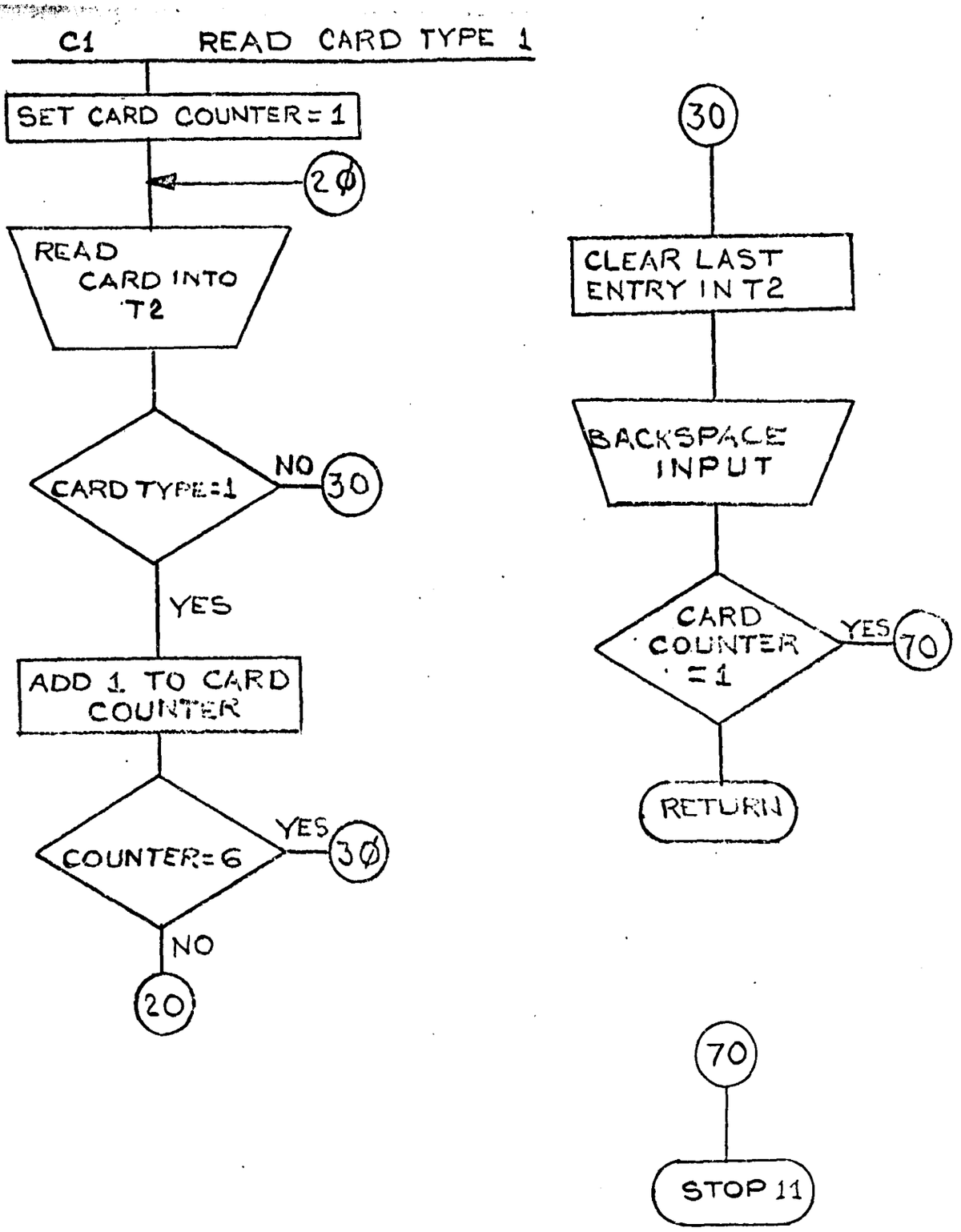

G
STORE W/R #
INTO T2Φ
CALL S2Φ3
M2=
TRANSACTION
POINTER
M3=
TIME POINTER
SET TIME
OFF FEC
SET FEC
MANIPULATOR
TIME
SET FEC
CODE
SET FEC
ITEM
CALL S2Φ1
1464

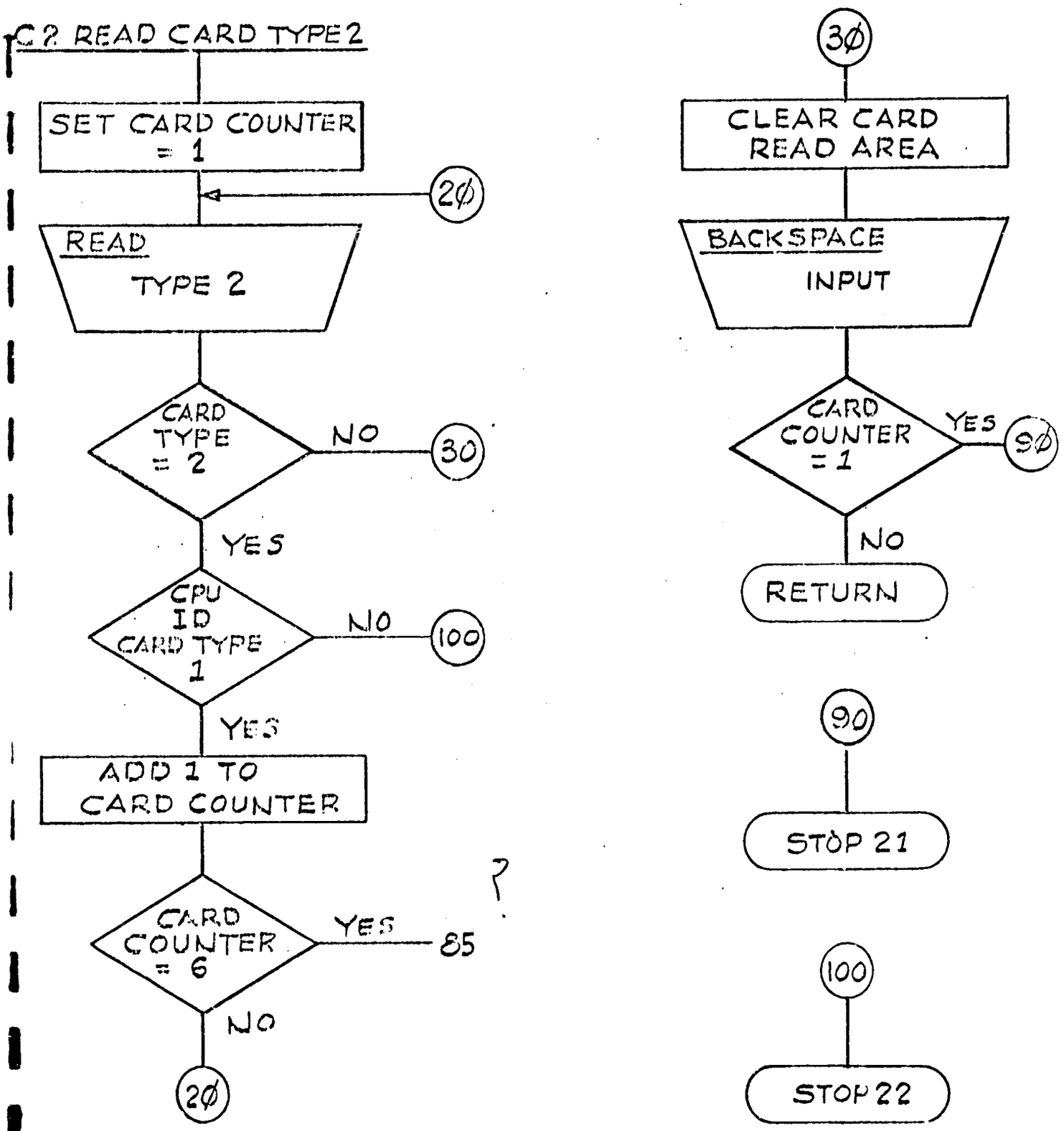

1165
THIS W/R
= O/S
YES
SET CPU
TO
CYCLE
1165
NO
1Φ6Φ
125Φ
SET MEMORY USED
BY OPERATING
SYSTEMS TO
BUSY
14Φ8
SET UNUSED
INTERRUPT VECTORS
TO-999999
ERROR SWITCH
ON
STOP 1
RETURN
136Φ
STOP 9Φ1

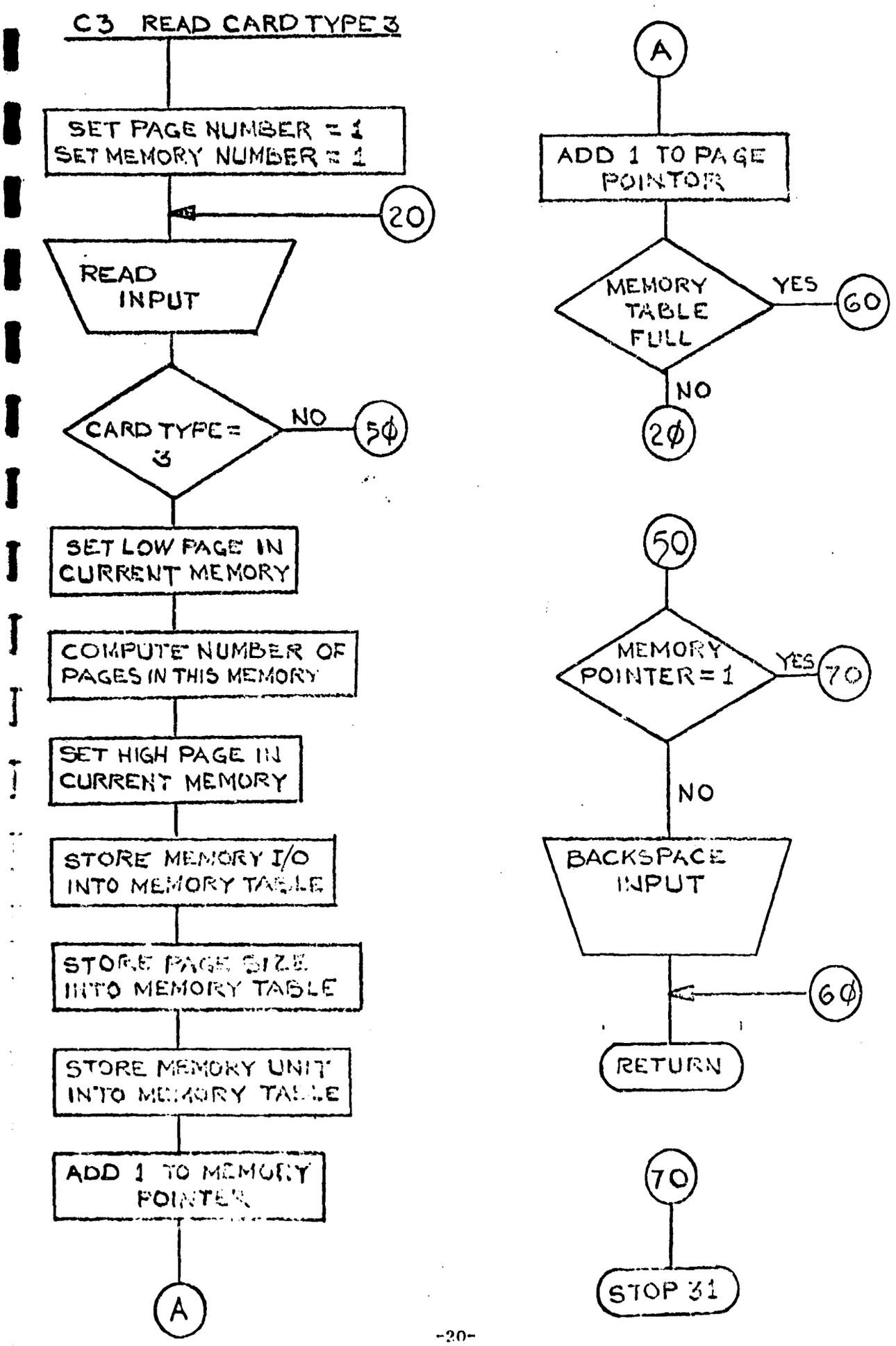

20L→ 229
SET N99= ERROR #
260
SET ERROR SWITCH ON
PRINT ERROR
READ INPUT
END OF FILE
YES
415
NO
JOB CARD
NO
270
YES
30


ZERO
INITIALIZE T1
INITIALIZE T2
INITIALIZE T3
INITIALIZE T4
INITIALIZE T5
INITIALIZE T6
INITIALIZE T7
INITIALIZE T8
INITIALIZE T9
INITIALIZE T10
A
A
INITIALIZE T12
INITIALIZE T13
INITIALIZE T14
INITIALIZE T15
INITIALIZE T16
INITIALIZE T17
INITIALIZE T18
INITIALIZE T19
INITIALIZE T20
INITIALIZE T21
INITIALIZE T22
B

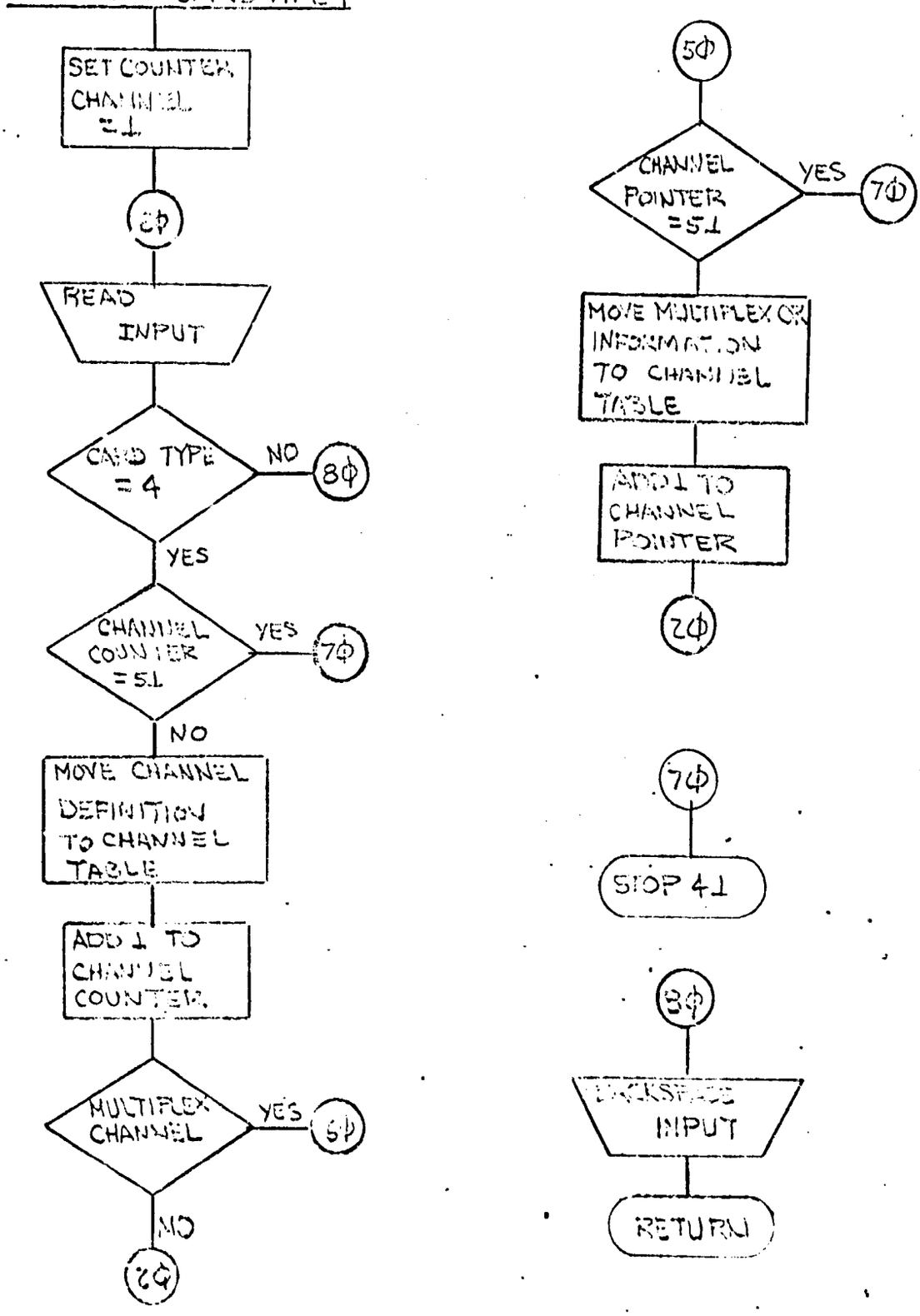

B
INITIALIZE T23
INITIALIZE T24
INITIALIZE T25
RETURN


C1 READ CARD TYPE 1
SET CARD COUNTER = 1
20
READ CARD INTO T2
CARD TYPE = 1
NO
30
YES
ADD 1 TO CARD COUNTER
COUNTER = 6
YES
30
NO
20
30
CLEAR LAST ENTRY IN T2
BACKSPACE INPUT
CARD COUNTER = 1
YES
70
RETURN
70
STOP 11

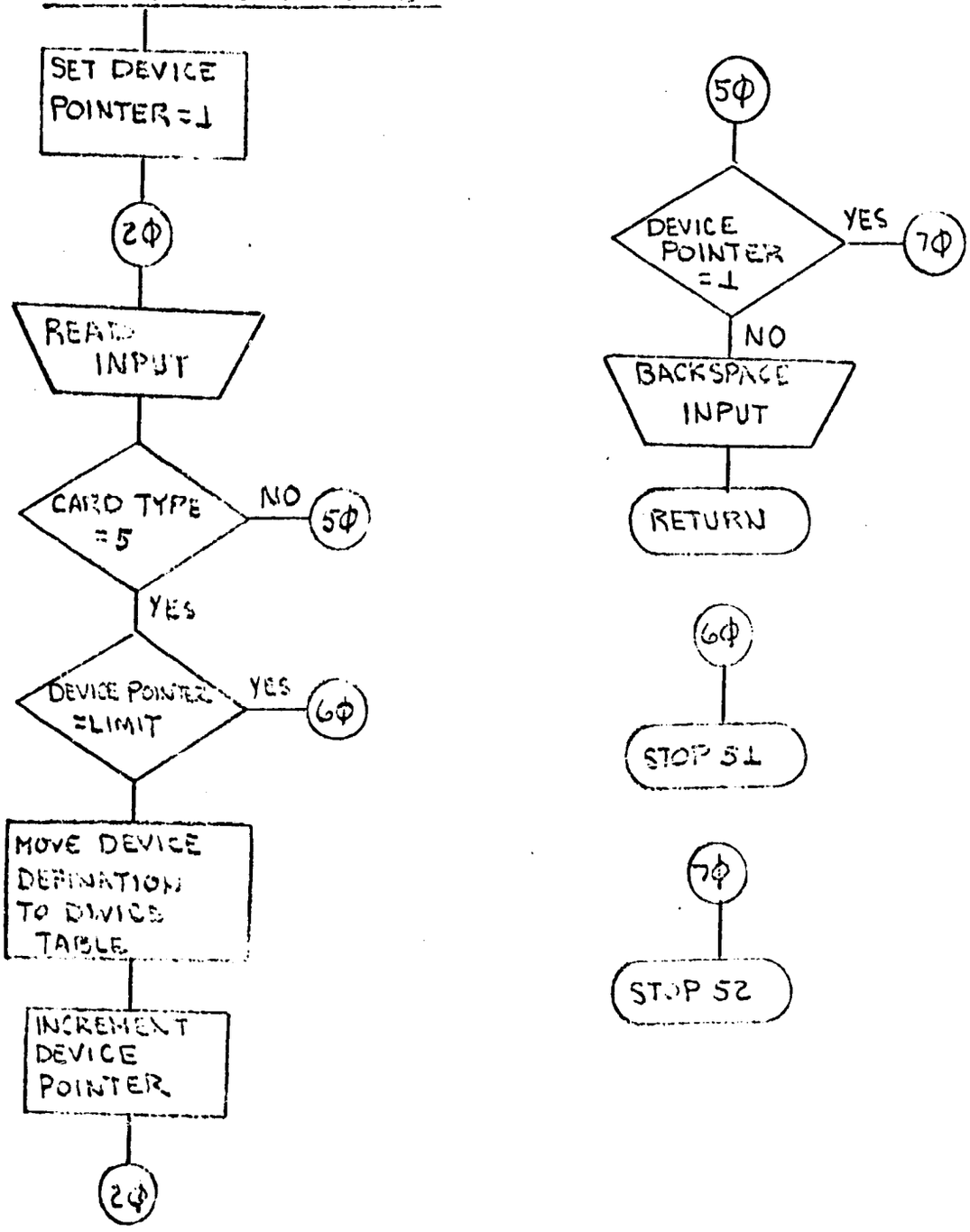

C2 READ CARD TYPE 2
SET CARD COUNTER = 1
20
READ TYPE 2
CARD TYPE = 2
NO
30
YES
CPU ID CARD TYPE 1
NO
100
YES
ADD 1 TO CARD COUNTER
CARD COUNTER = 6
YES
85
NO
20
30
CLEAR CARD READ AREA
BACKSPACE INPUT
CARD COUNTER = 1
YES
90
NO
RETURN
90
STOP 21
100
STOP 22

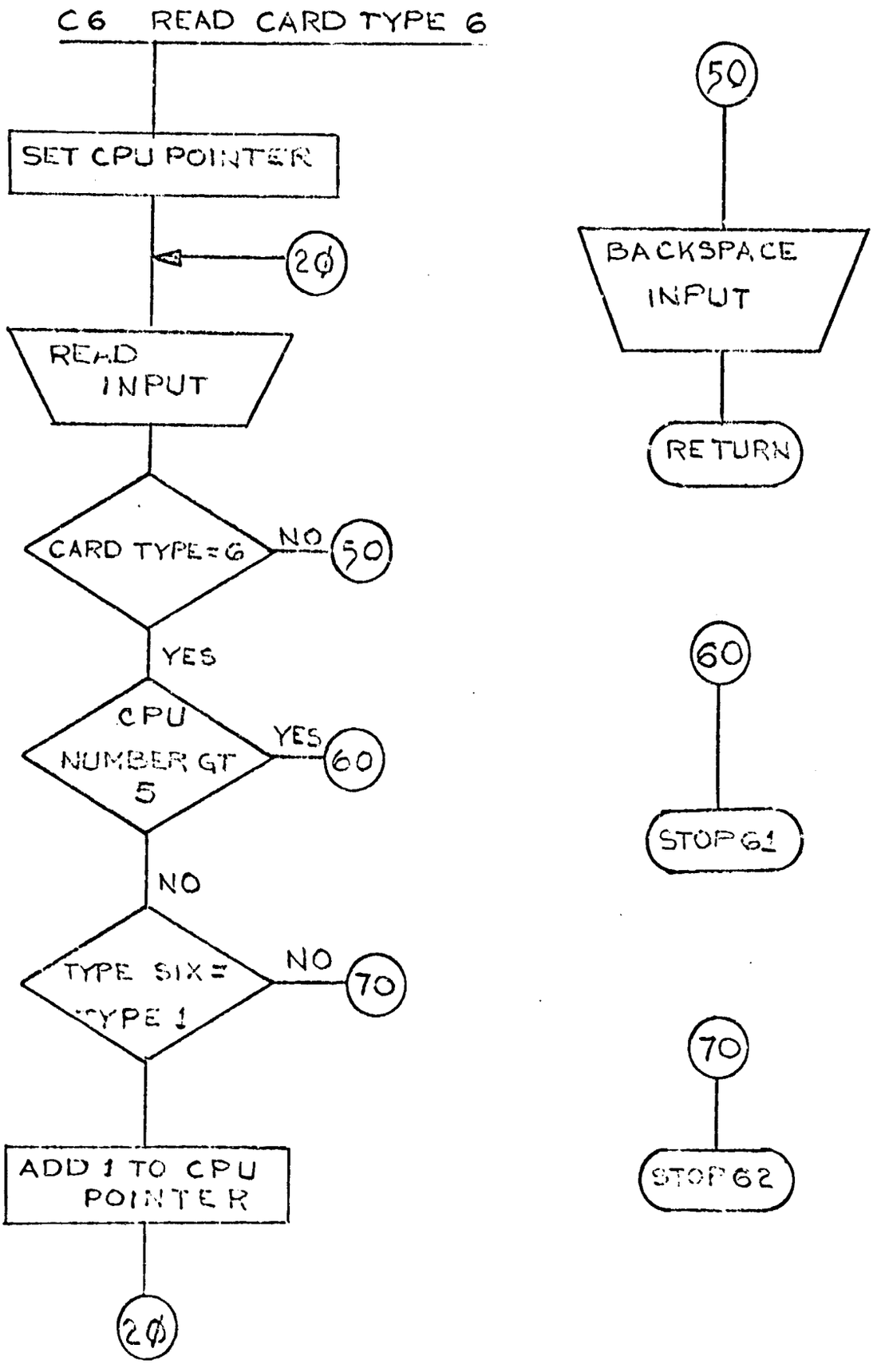

C3 READ CARD TYPE 3
SET PAGE NUMBER = 1
SET MEMORY NUMBER = 1
20
READ INPUT
CARD TYPE = 3
NO
50
SET LOW PAGE IN CURRENT MEMORY
COMPUTE NUMBER OF PAGES IN THIS MEMORY
SET HIGH PAGE IN CURRENT MEMORY
STORE MEMORY I/O INTO MEMORY TABLE
STORE PAGE SIZE INTO MEMORY TABLE
STORE MEMORY UNIT INTO MEMORY TABLE
ADD 1 TO MEMORY POINTER
A
A
ADD 1 TO PAGE POINTER
MEMORY TABLE FULL
YES
60
NO
20
50
MEMORY POINTER = 1
YES
70
NO
BACKSPACE INPUT
60
RETURN
70
STOP 31

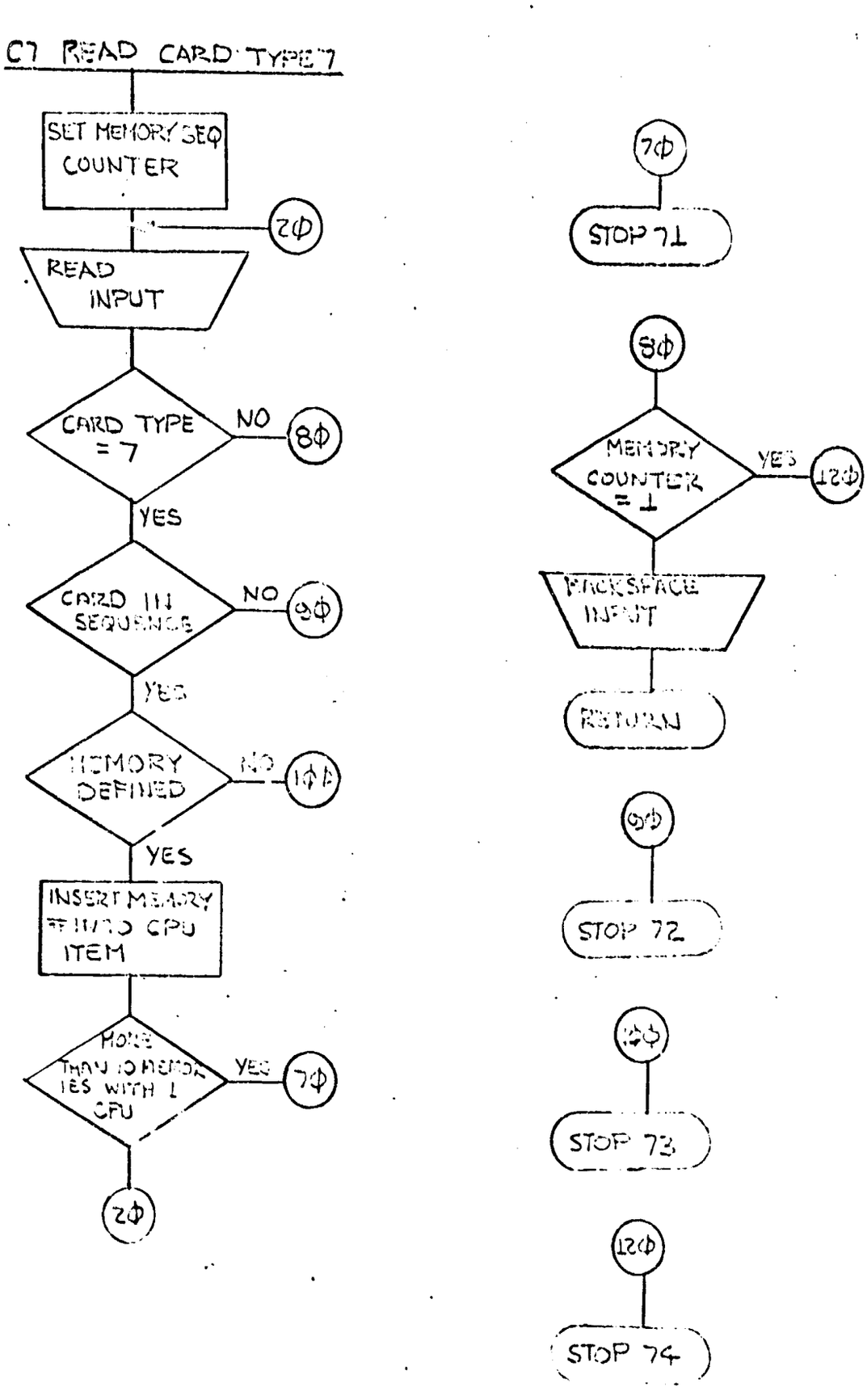
C4 READ CARD TYPE 4

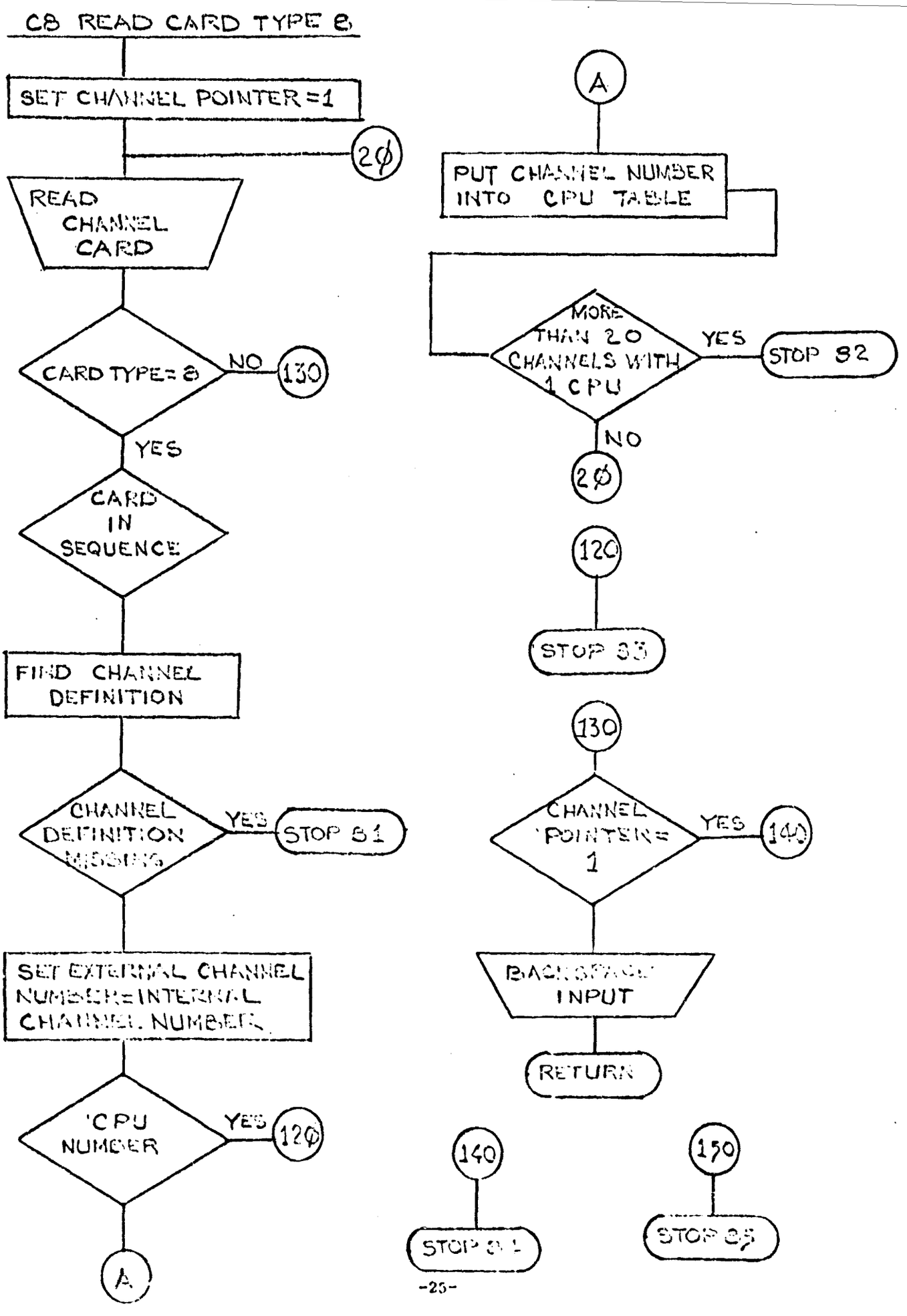


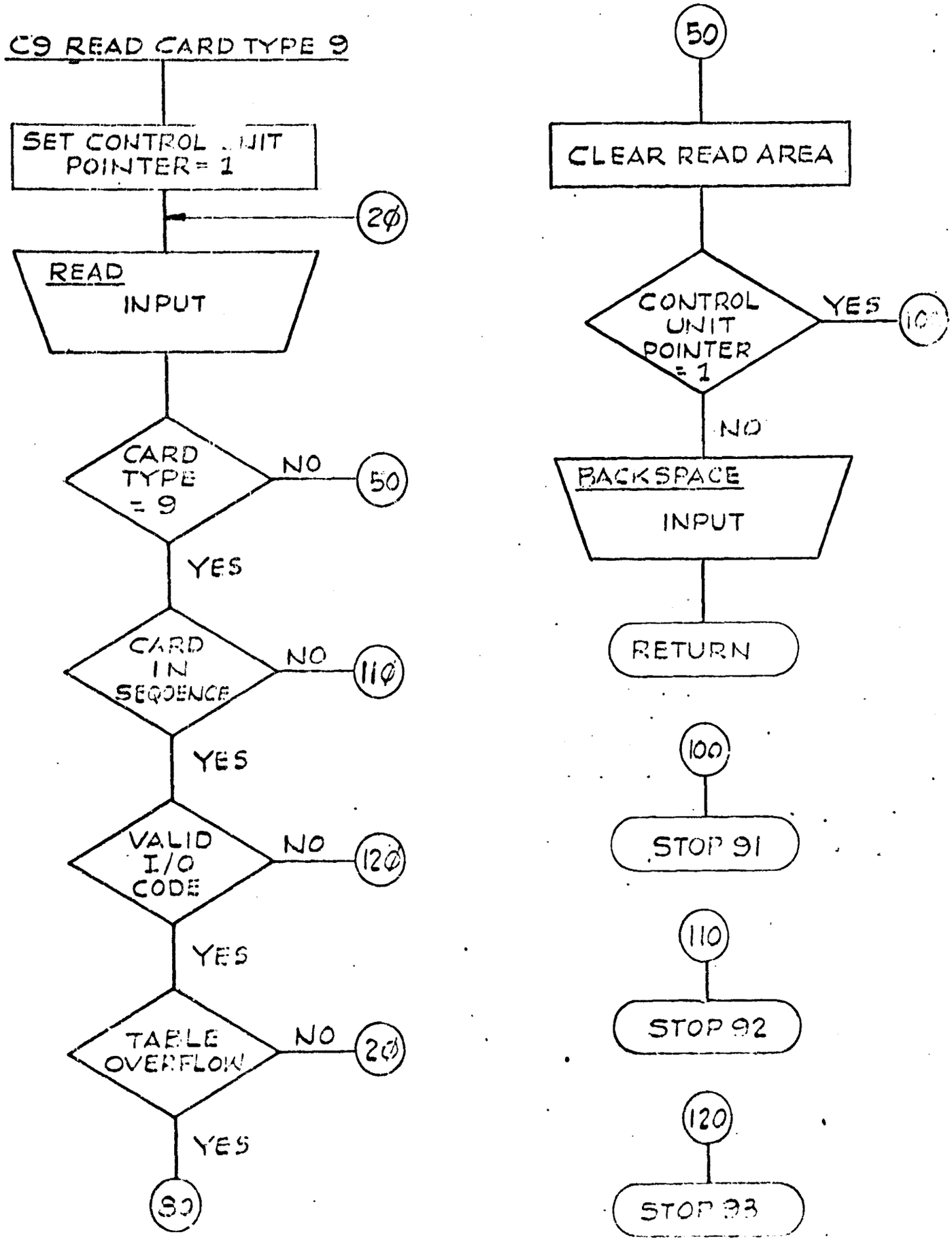
C5 READ CARD TYPE 5



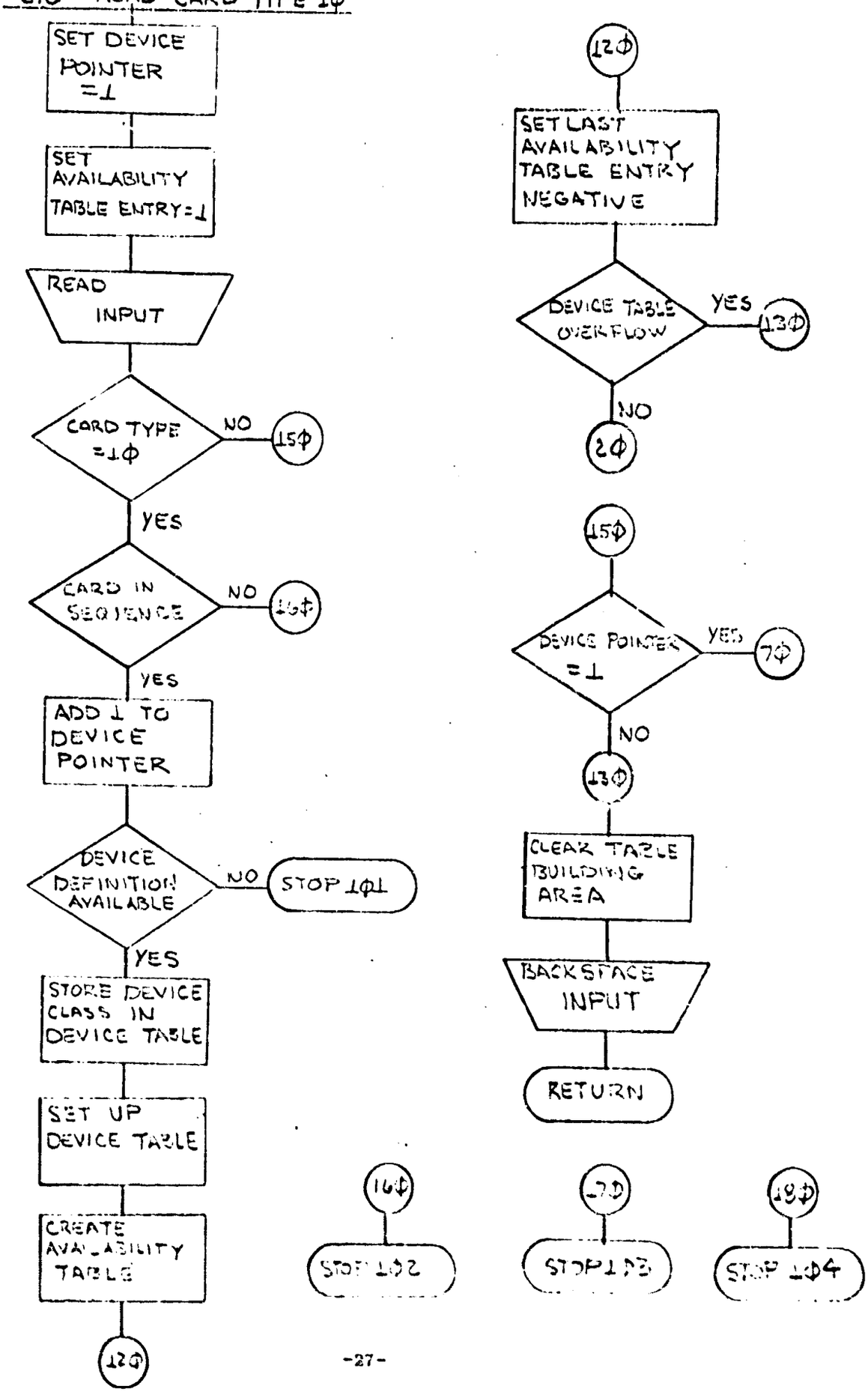

C6 READ CARD TYPE 6
SET CPU POINTER
20
READ INPUT
CARD TYPE=6
NO
50
YES
CPU NUMBER GT 5
YES
60
NO
TYPE SIX = TYPE 1
NO
70
ADD 1 TO CPU POINTER
20
50
BACKSPACE INPUT
RETURN
60
STOP 61
70
STOP 62

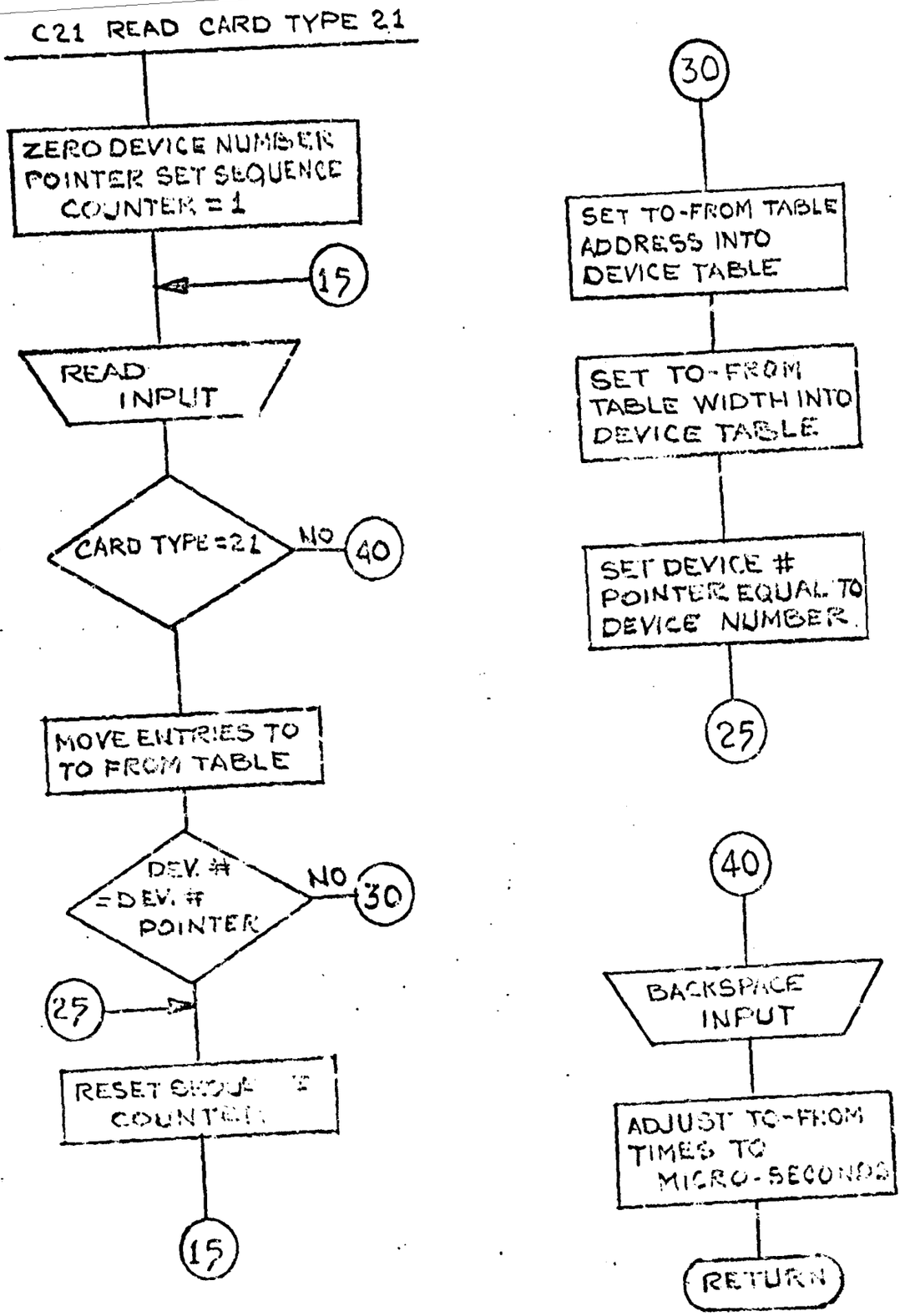

C7 READ CARD TYPE 7
SET MEMORY SEQ COUNTER
2Φ
READ INPUT
CARD TYPE = 7
NO
8Φ
YES
CARD IN SEQUENCE
NO
9Φ
YES
MEMORY DEFINED
NO
10Φ
YES
INSERT MEMORY FIELD CPU ITEM
MORE THAN 10 MEMORIES WITH 1 CPU
YES
7Φ
2Φ
7Φ
STOP 71
8Φ
MEMORY COUNTER = 1
YES
12Φ
BACKSPACE INPUT
RETURN
9Φ
STOP 72
12Φ
STOP 73
12Φ
STOP 74


C8 READ CARD TYPE 8
SET CHANNEL POINTER=1
20
READ CHANNEL CARD
CARD TYPE=8
NO
130
YES
CARD IN SEQUENCE
FIND CHANNEL DEFINITION
CHANNEL DEFINITION MISSING
YES
STOP 81
SET EXTERNAL CHANNEL NUMBER=INTERNAL CHANNEL NUMBER
CPU NUMBER
YES
120
A
A
PUT CHANNEL NUMBER INTO CPU TABLE
MORE THAN 20 CHANNELS WITH 1 CPU
YES
STOP 82
NO
20
120
STOP 83
130
CHANNEL POINTER=1
YES
140
BACKSPACE INPUT
RETURN
140
STOP 84
150
STOP 85

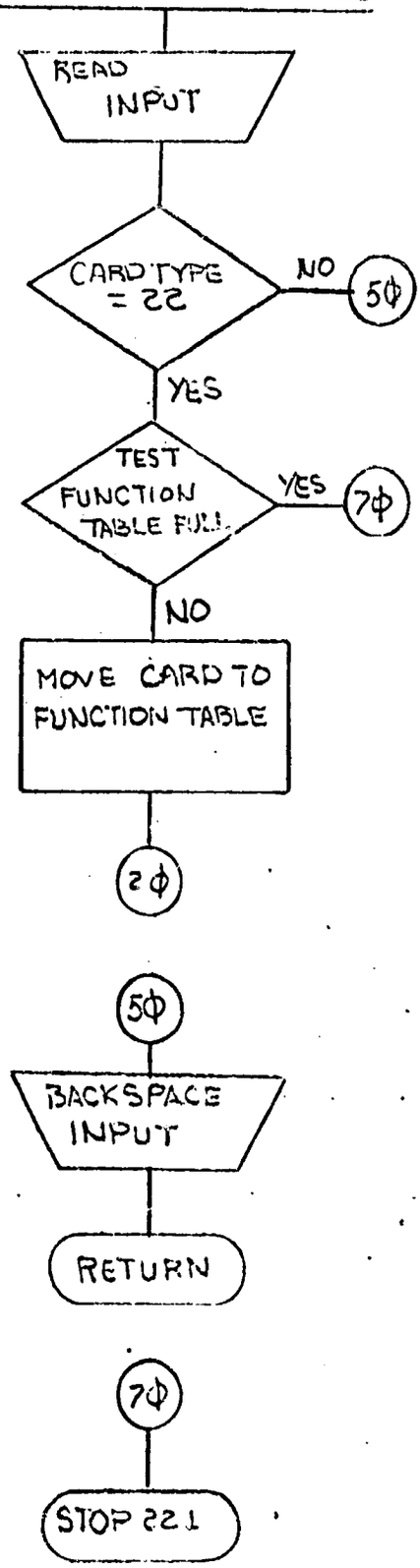

C9 READ CARD TYPE 9
SET CONTROL UNIT POINTER = 1
20
READ INPUT
CARD TYPE = 9
NO
50
YES
CARD IN SEQUENCE
NO
110
YES
VALID I/O CODE
NO
120
YES
TABLE OVERFLOW
NO
20
YES
90
50
CLEAR READ AREA
CONTROL UNIT POINTER = 1
YES
10
NO
BACKSPACE INPUT
RETURN
100
STOP 91
110
STOP 92
120
STOP 93

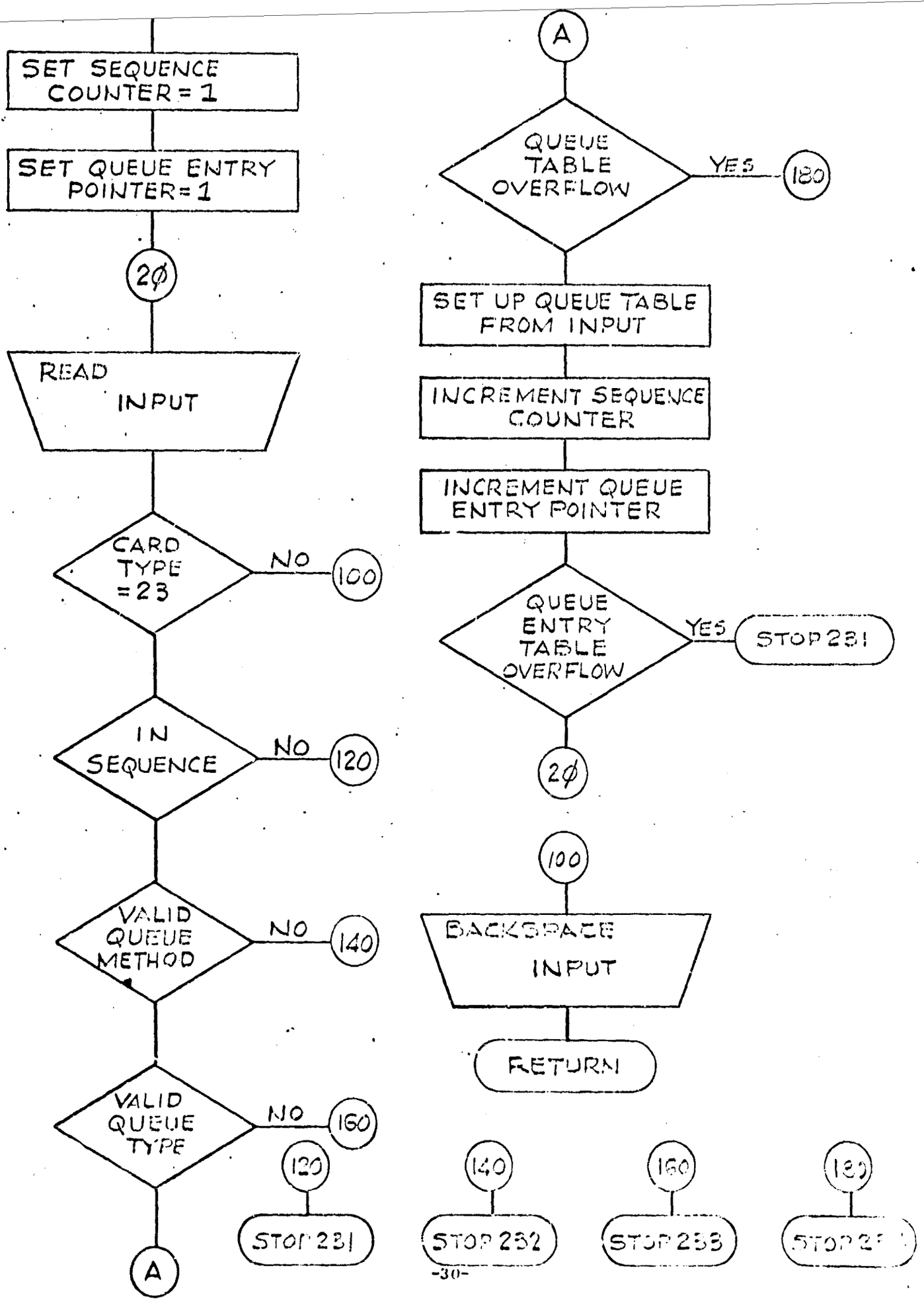
C10 READ CARD TYPE 1φ



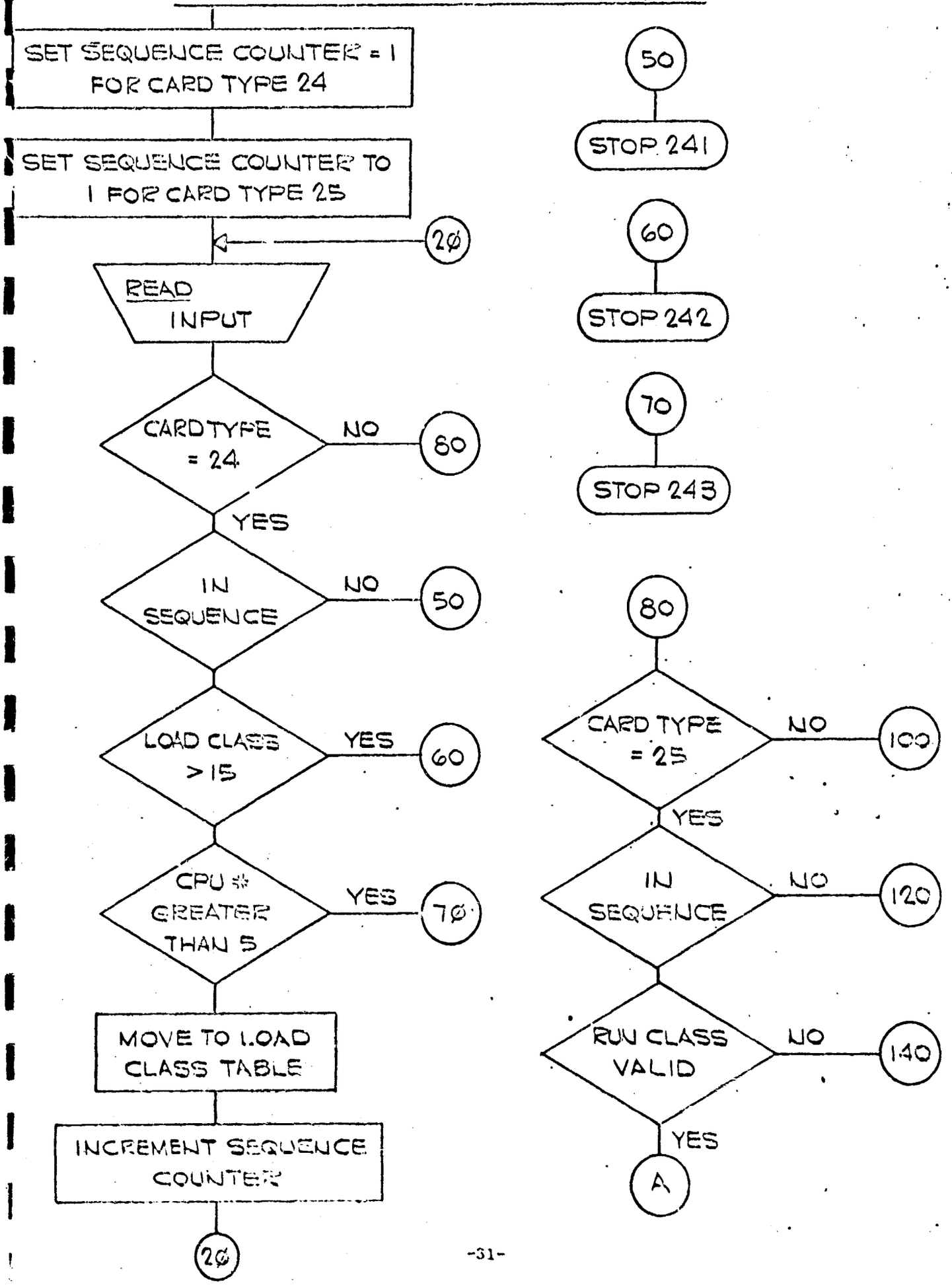

C21 READ CARD TYPE 21
ZERO DEVICE NUMBER POINTER SET SEQUENCE COUNTER = 1
15
READ INPUT
CARD TYPE = 21
NO
40
MOVE ENTRIES TO TO FROM TABLE
DEV. # = DEV. # POINTER
NO
30
25
RESET SEQUENCE COUNTER
15
30
SET TO-FROM TABLE ADDRESS INTO DEVICE TABLE
SET TO-FROM TABLE WIDTH INTO DEVICE TABLE
SET DEVICE # POINTER EQUAL TO DEVICE NUMBER
25
40
BACKSPACE INPUT
ADJUST TO-FROM TIMES TO MICRO-SECONDS
RETURN

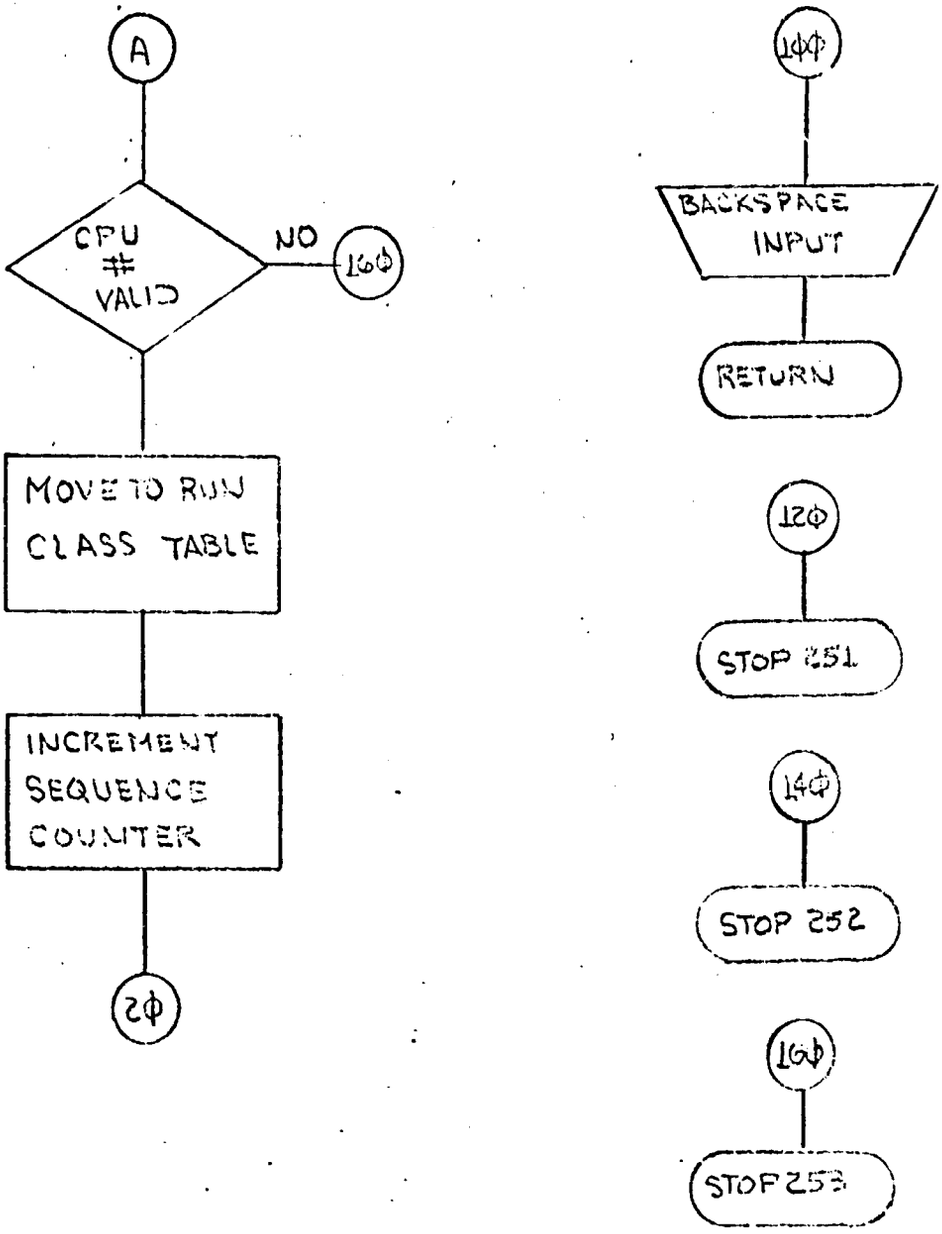
C22 READ CARD TYPE 22



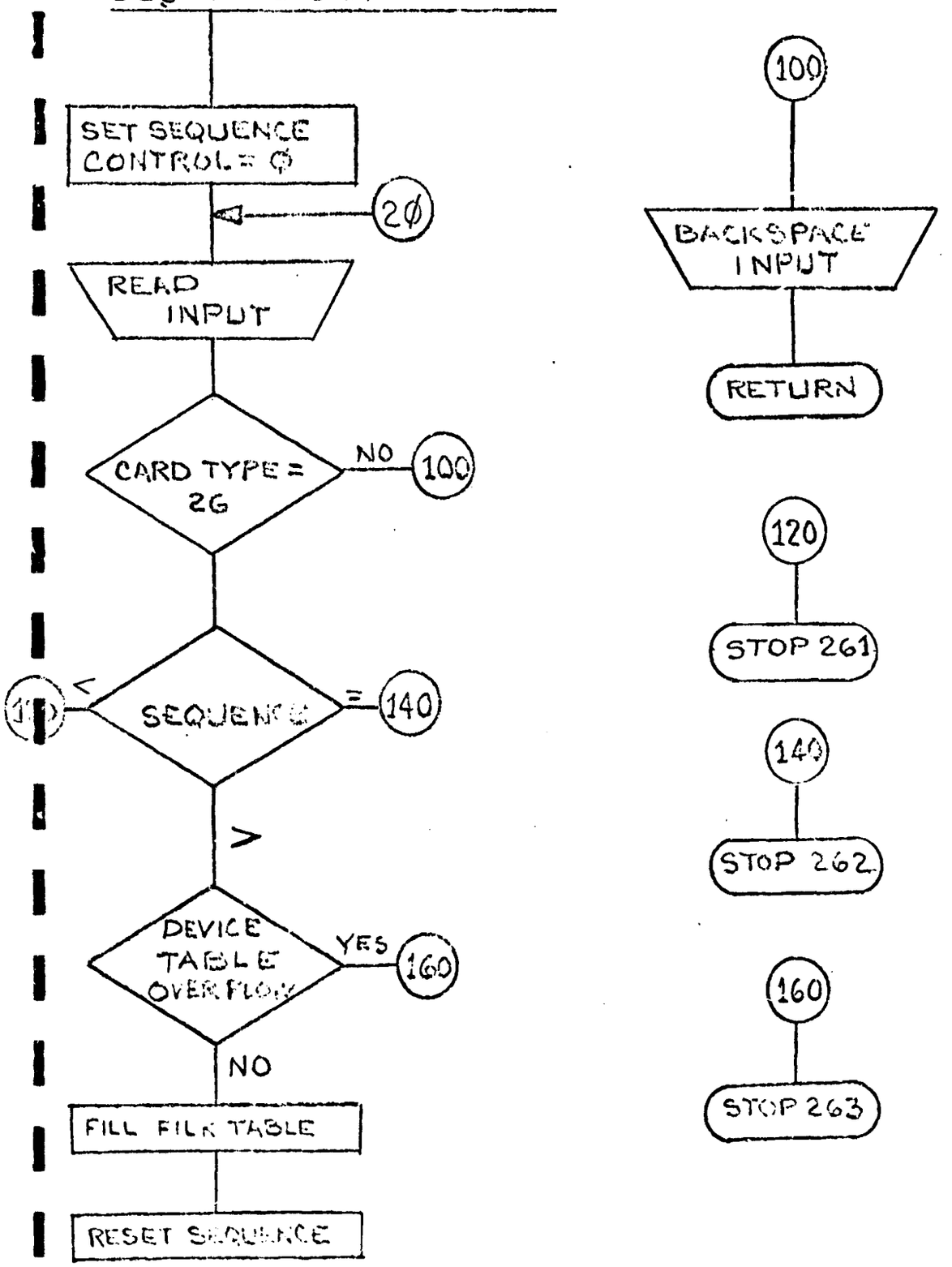

SET SEQUENCE COUNTER = 1
SET QUEUE ENTRY POINTER = 1
20
READ INPUT
CARD TYPE = 23
NO
100
IN SEQUENCE
NO
120
VALID QUEUE METHOD
NO
140
VALID QUEUE TYPE
NO
160
120
STOP 231
A
A
QUEUE TABLE OVERFLOW
YES
180
SET UP QUEUE TABLE FROM INPUT
INCREMENT SEQUENCE COUNTER
INCREMENT QUEUE ENTRY POINTER
QUEUE ENTRY TABLE OVERFLOW
YES
STOP 231
20
100
BACKSPACE INPUT
RETURN
140
STOP 232
160
STOP 233
180

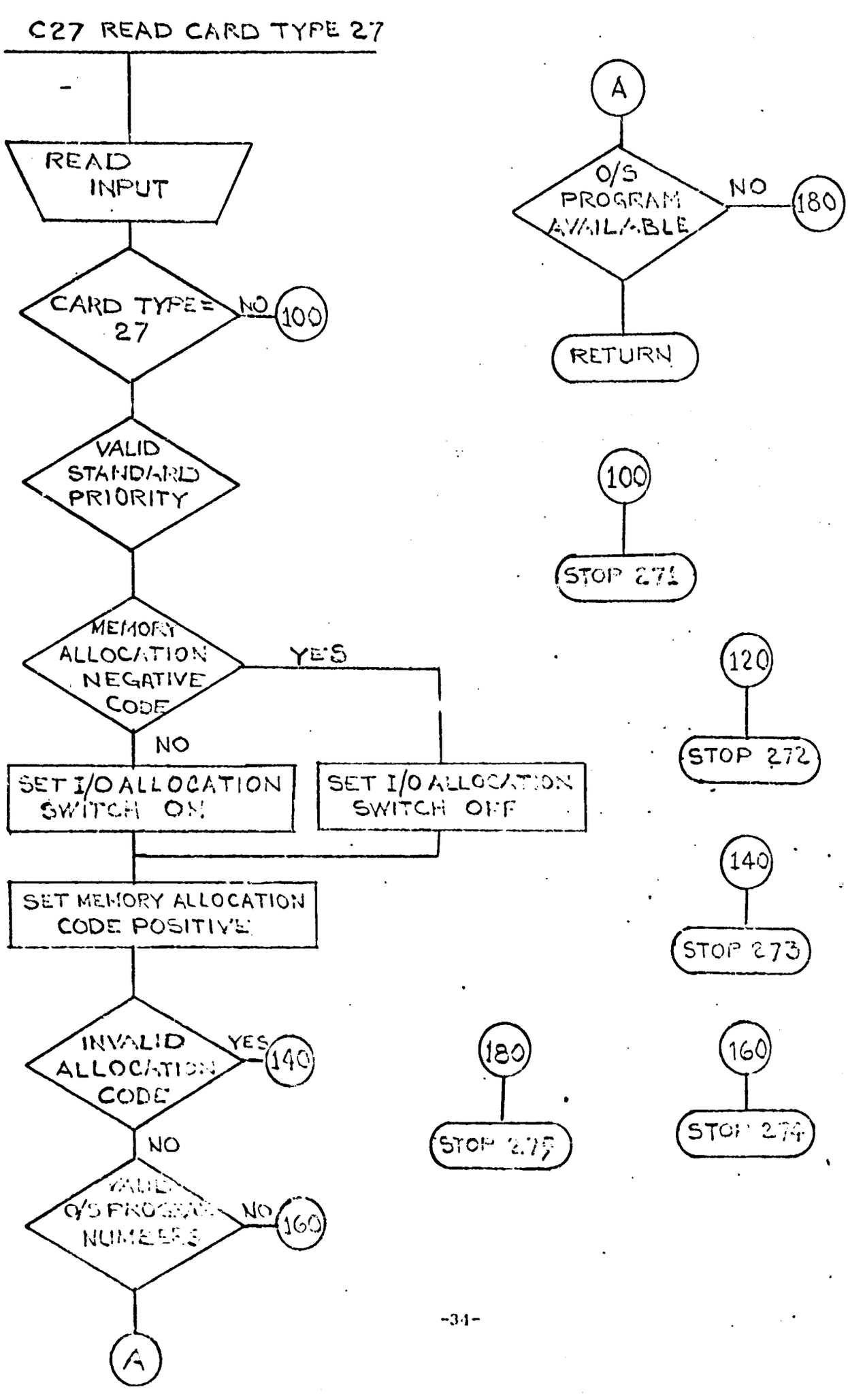
C24 READ CARD TYPES 24 & 25



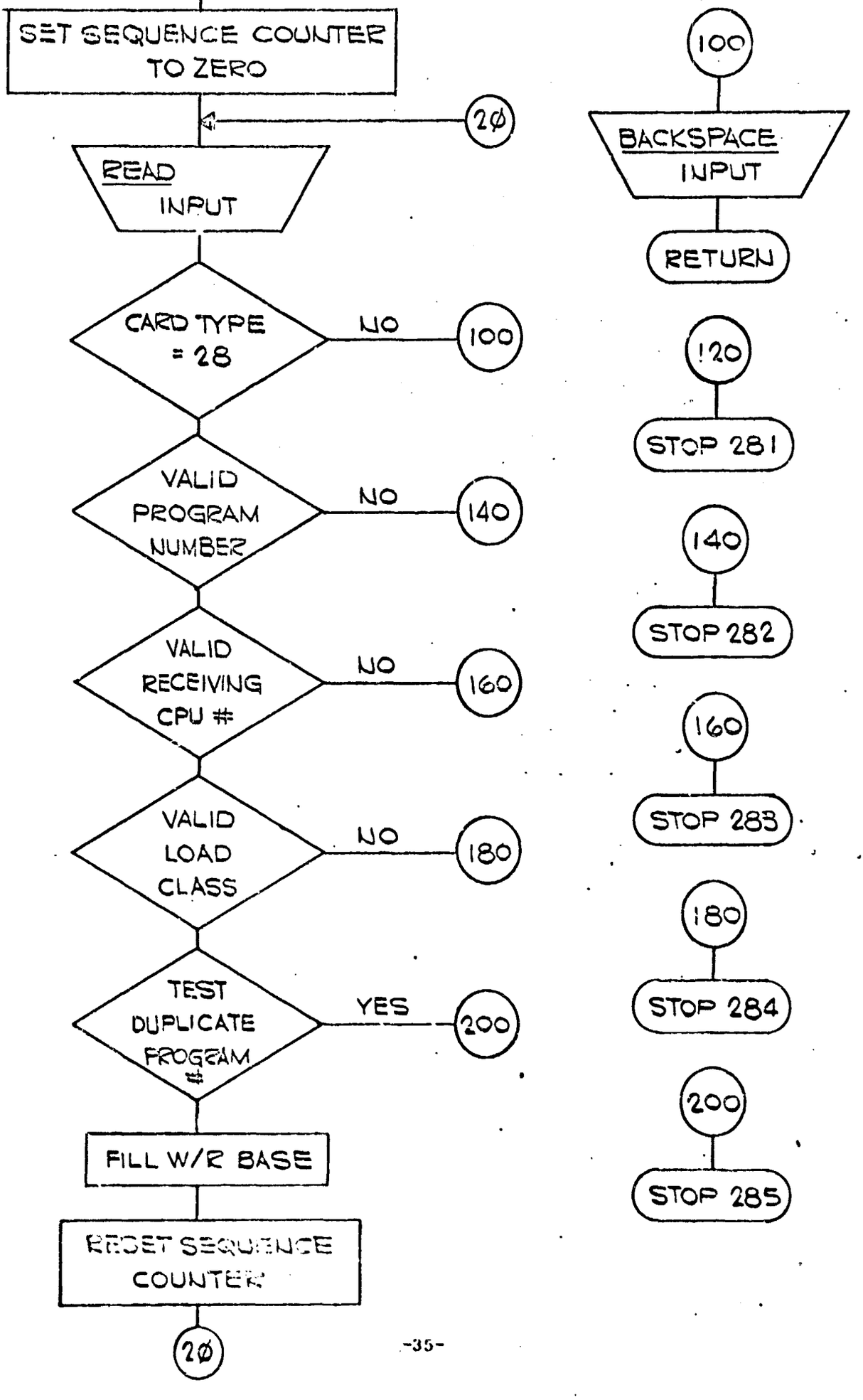

A
CPU # VALID
NO
160
MOVE TO RUN CLASS TABLE
INCREMENT SEQUENCE COUNTER
20
100
BACKSPACE INPUT
RETURN
120
STOP 251
140
STOP 252
160
STOP 253

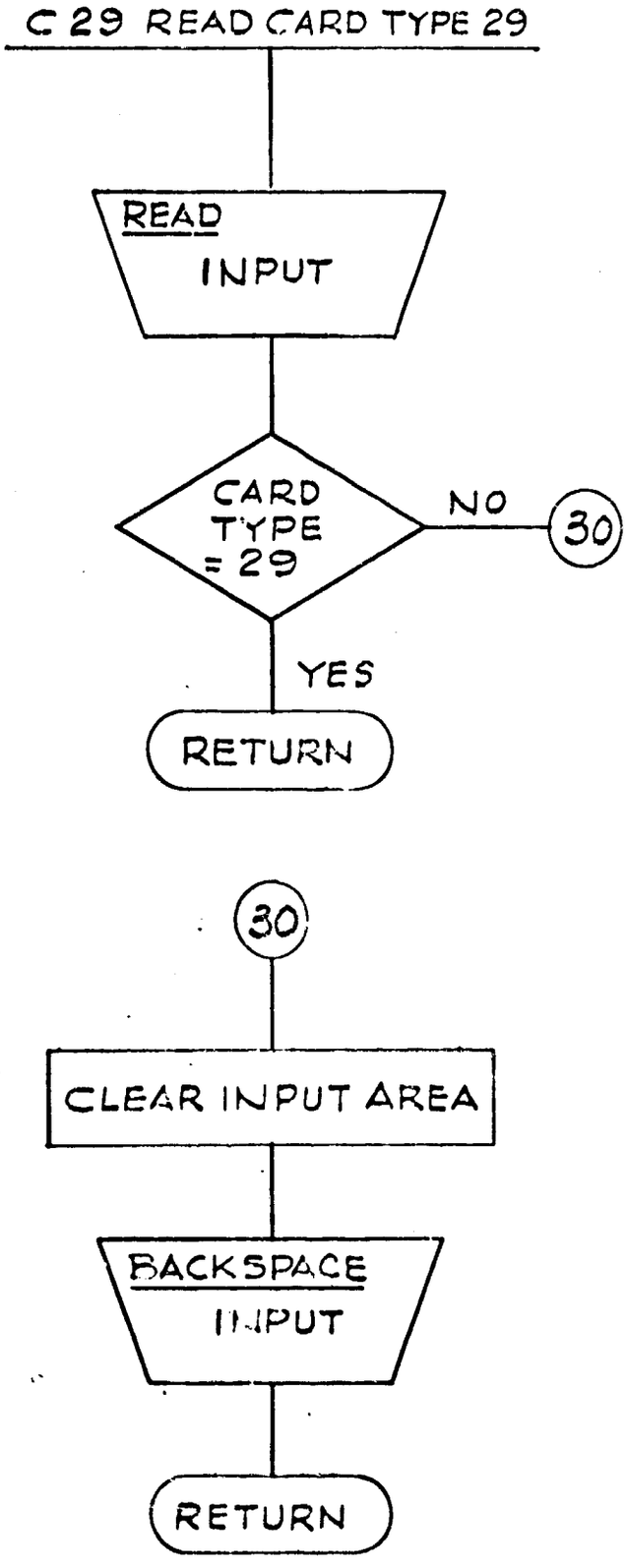
C26 READ CARD TYPE 26

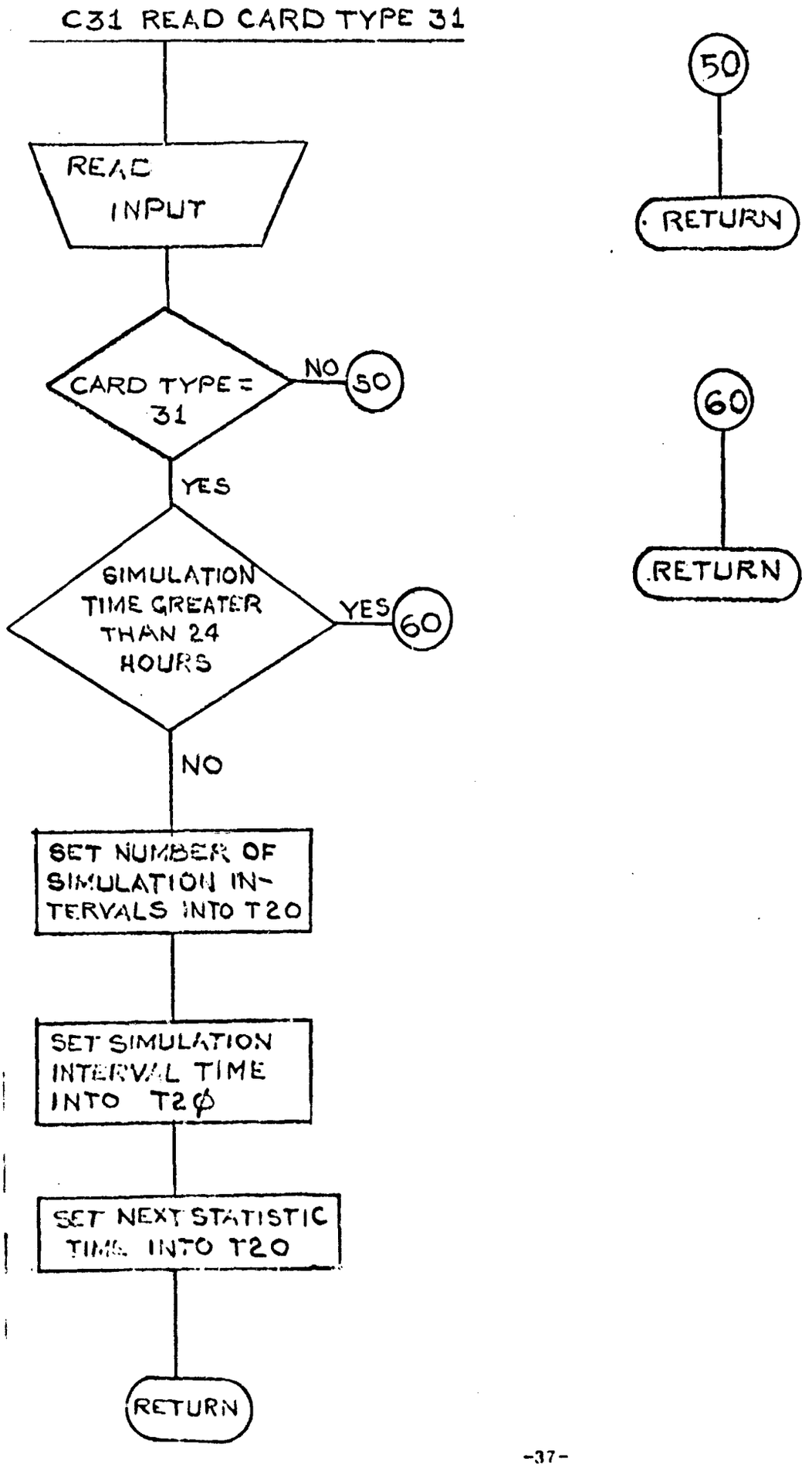



C27 READ CARD TYPE 27
READ INPUT
CARD TYPE = 27
NO
100
VALID STANDARD PRIORITY
MEMORY ALLOCATION NEGATIVE CODE
YES
NO
SET I/O ALLOCATION SWITCH ON
SET I/O ALLOCATION SWITCH OFF
SET MEMORY ALLOCATION CODE POSITIVE
INVALID ALLOCATION CODE
YES
140
NO
VALID O/S PROGRAM NUMBERS
NO
160
A
A
O/S PROGRAM AVAILABLE
NO
180
RETURN
100
STOP 271
120
STOP 272
140
STOP 273
180
STOP 275
160
STOP 274

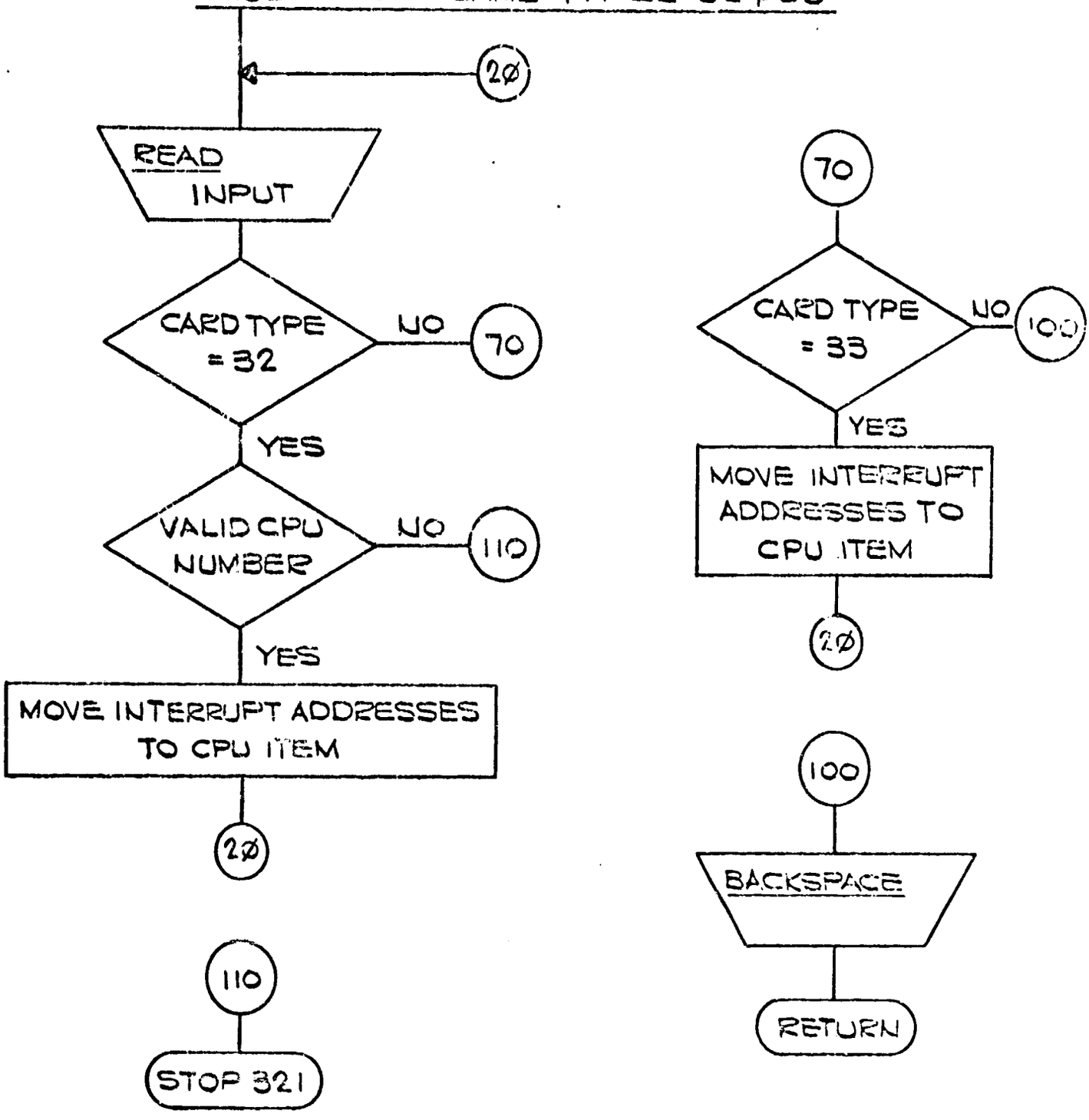
C28 READ CARD TYPE 28



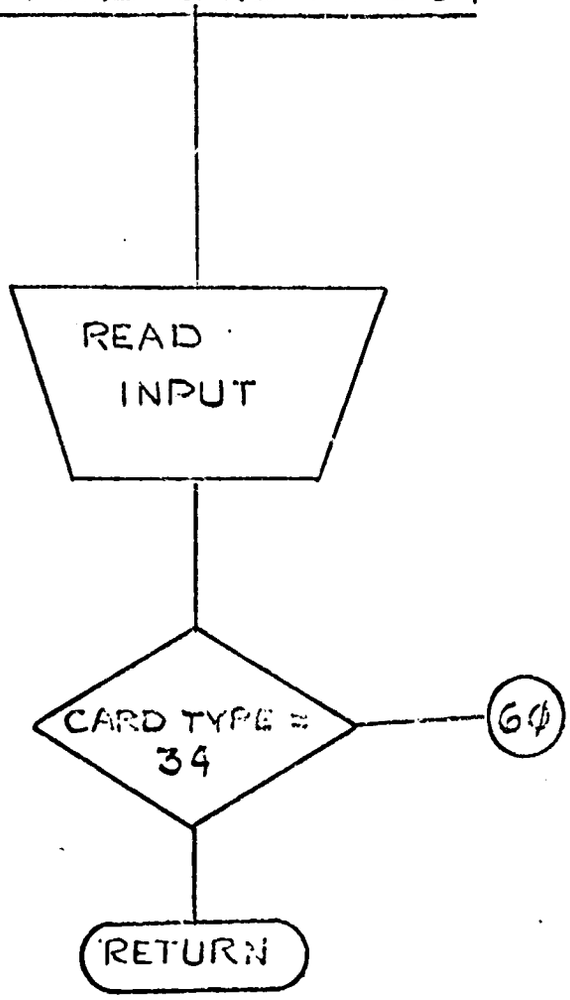

C 29 READ CARD TYPE 29
READ
INPUT
CARD
TYPE
= 29
NO
30
YES
RETURN
30
CLEAR INPUT AREA
BACKSPACE
INPUT
RETURN

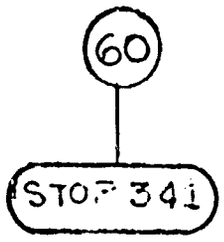

C31 READ CARD TYPE 31
READ INPUT
CARD TYPE = 31
NO
50
YES
SIMULATION TIME GREATER THAN 24 HOURS
YES
60
NO
SET NUMBER OF SIMULATION INTERVALS INTO T20
SET SIMULATION INTERVAL TIME INTO T2ϕ
SET NEXT STATISTIC TIME INTO T20
RETURN
50
RETURN
60
RETURN

C32 READ CARD TYPES 32 & 33

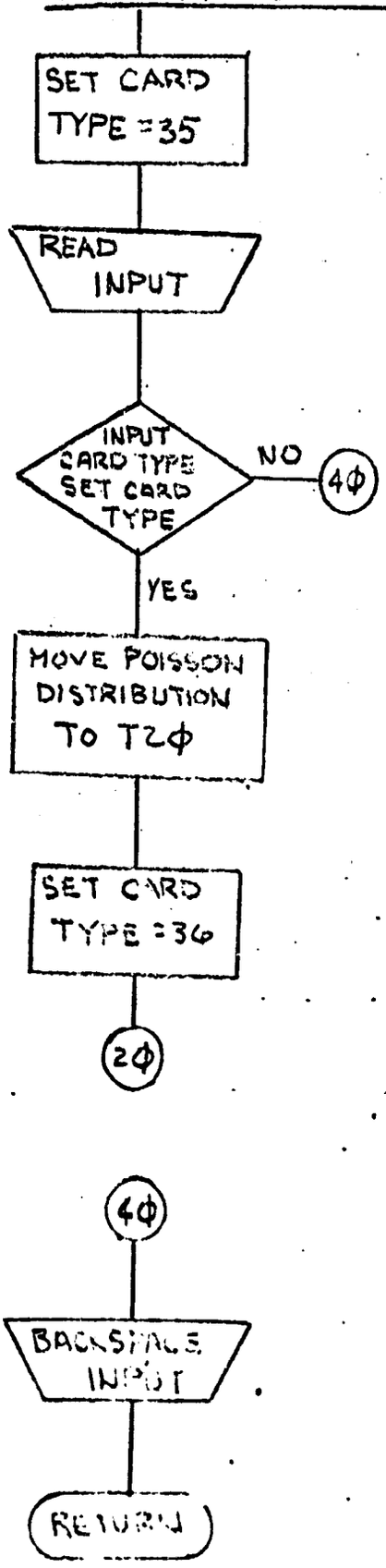


C34 READ CARD TYPE 34

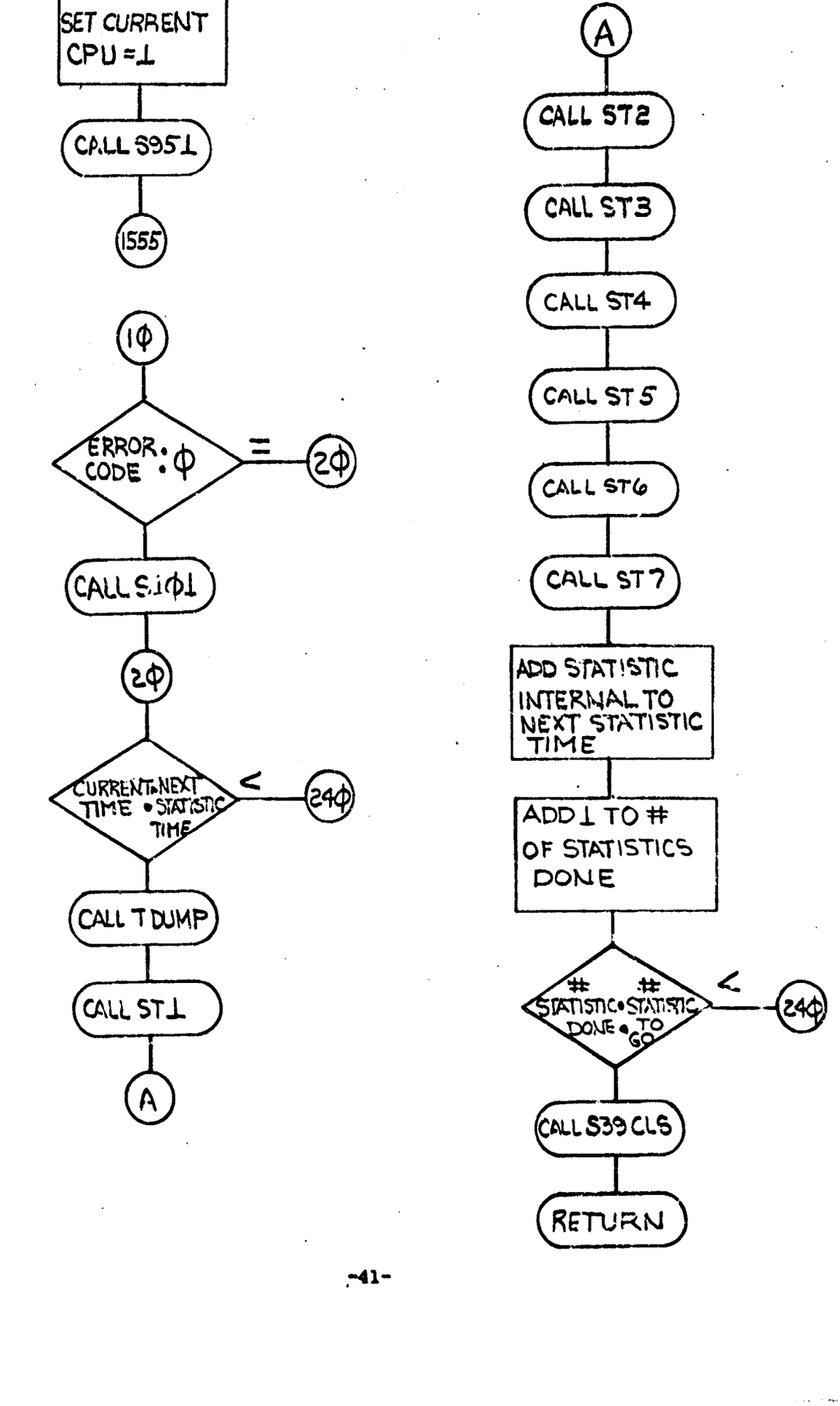


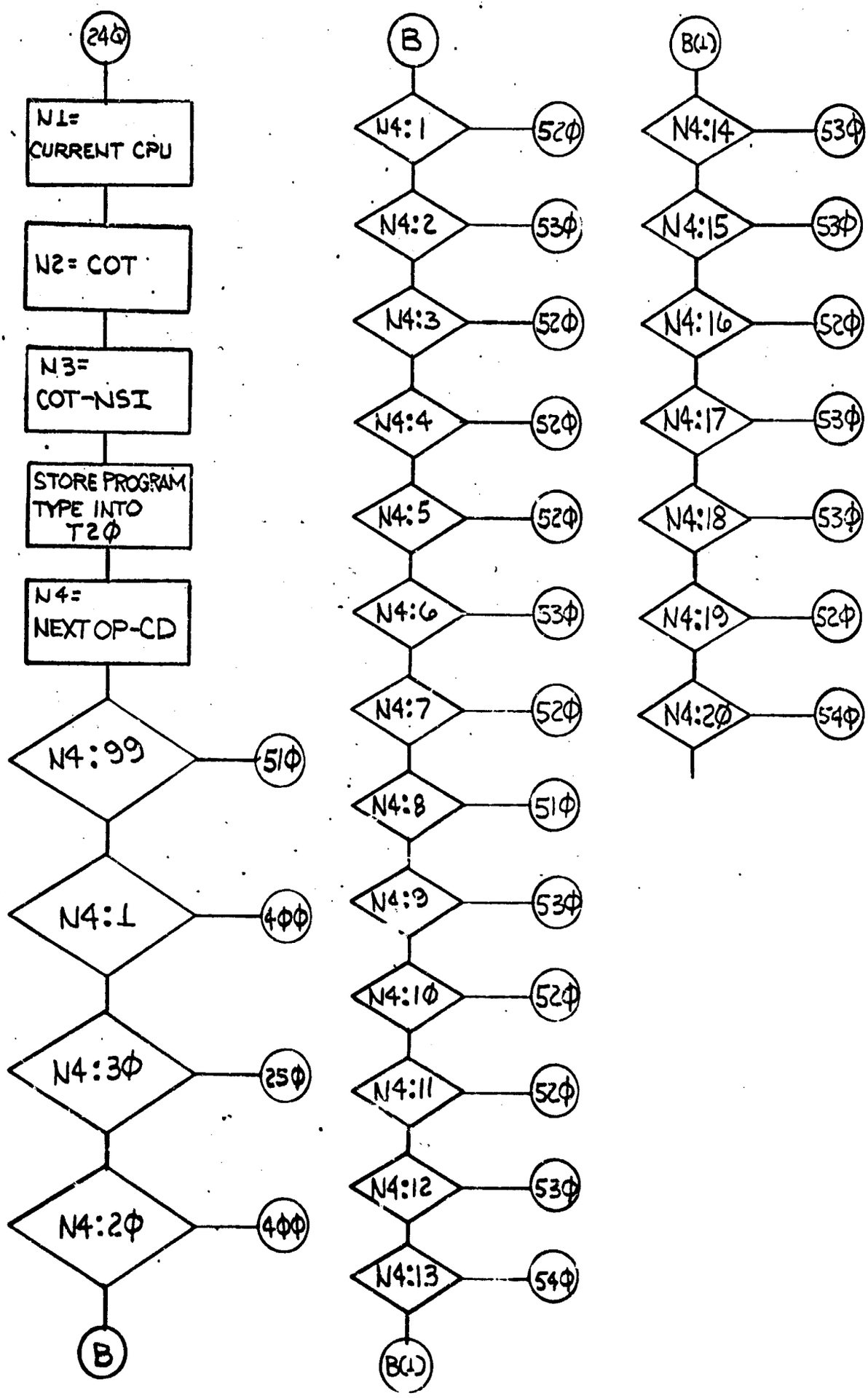


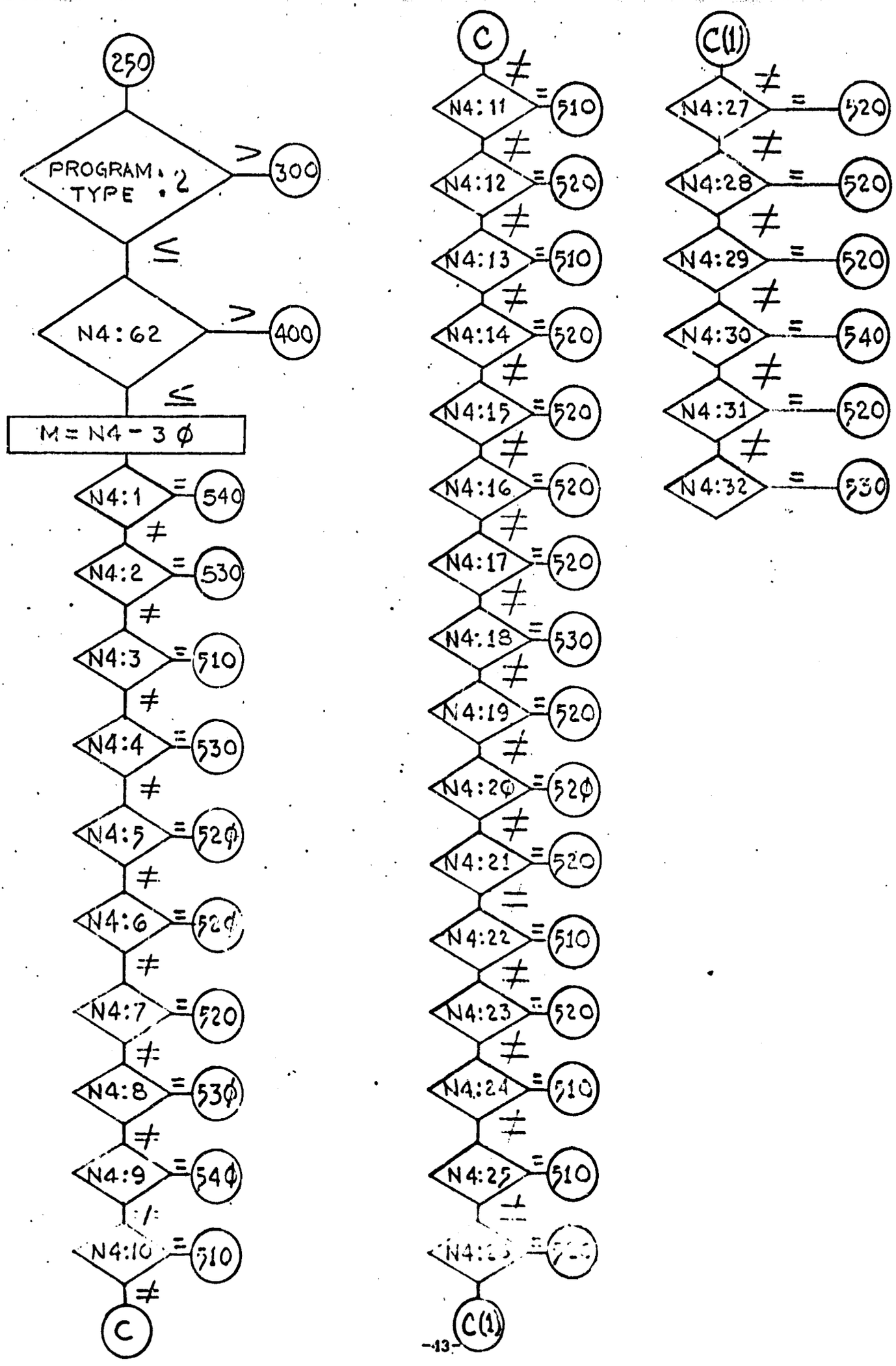
C35 READ CARD TYPE 35 & 36

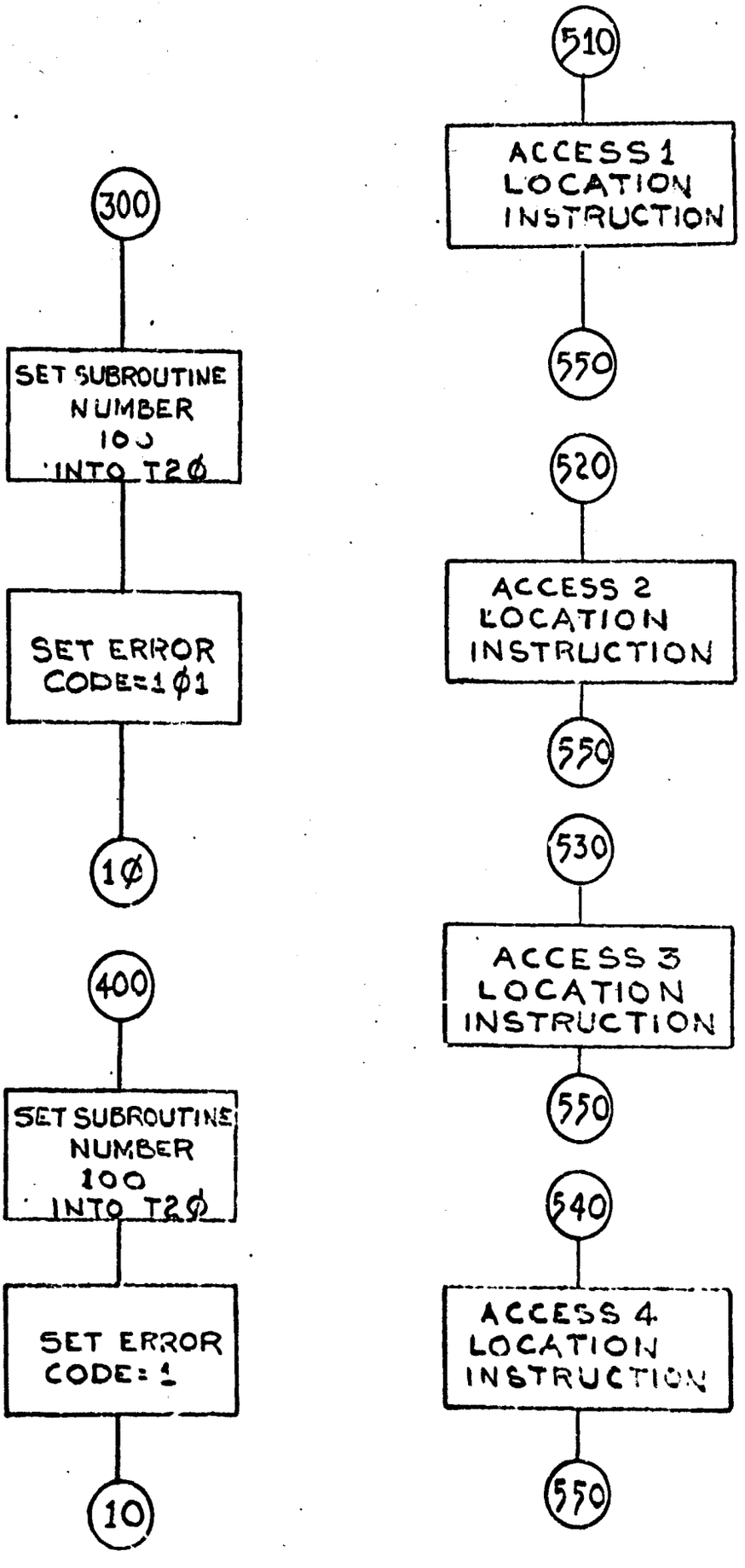


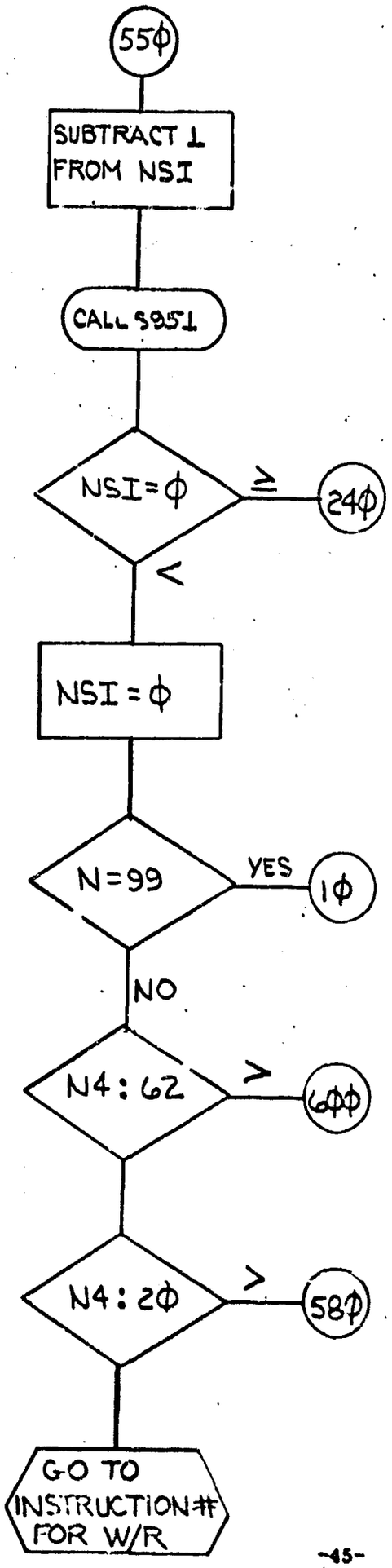
SM2 SIMULATION EXEC

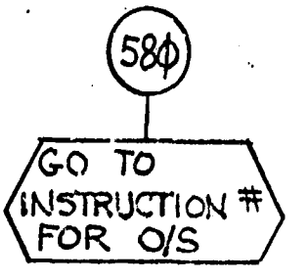


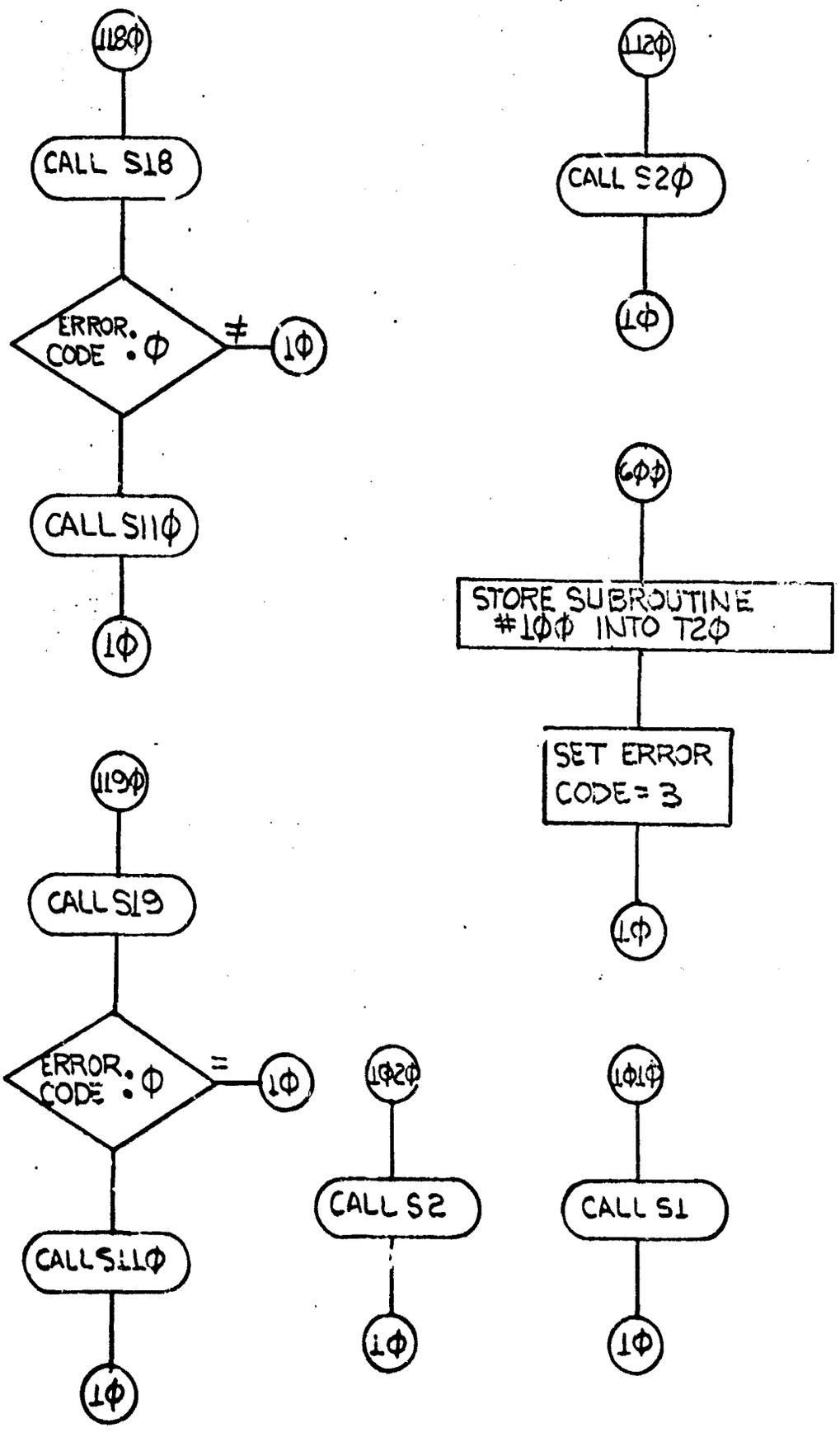

24Ø
N1= CURRENT CPU
N2= COT
N3= COT-NSI
STORE PROGRAM TYPE INTO T2Ø
N4= NEXT OP-CD
N4:99
51Ø
N4:1
4ØØ
N4:3Ø
25Ø
N4:2Ø
4ØØ
B
B
N4:1
52Ø
N4:2
53Ø
N4:3
52Ø
N4:4
52Ø
N4:5
52Ø
N4:6
53Ø
N4:7
52Ø
N4:8
51Ø
N4:9
53Ø
N4:1Ø
52Ø
N4:11
52Ø
N4:12
53Ø
N4:13
54Ø
B(1)
B(1)
N4:14
53Ø
N4:15
53Ø
N4:16
52Ø
N4:17
53Ø
N4:18
53Ø
N4:19
52Ø
N4:2Ø
54Ø

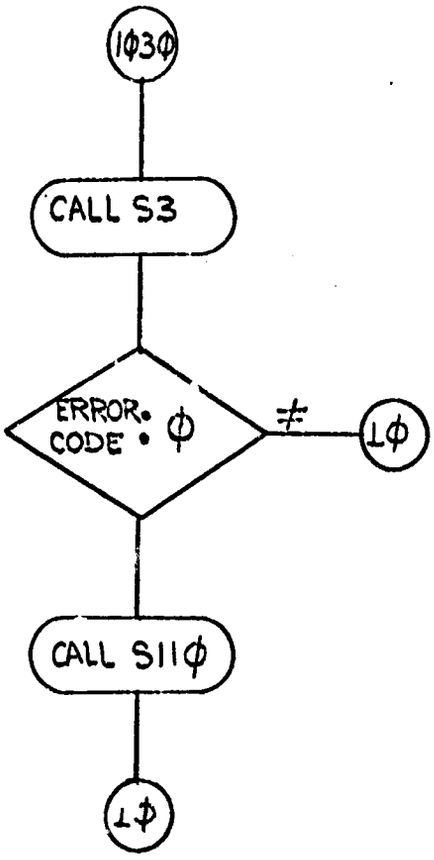

250
PROGRAM TYPE : 2
>
300
≤
N4:62
>
400
≤
M = N4 - 3 Ø
N4:1
=
540
≠
N4:2
=
530
≠
N4:3
=
510
≠
N4:4
=
530
≠
N4:5
=
52Ø
≠
N4:6
=
52Ø
≠
N4:7
=
520
≠
N4:8
=
53Ø
≠
N4:9
=
54Ø
≠
N4:10
=
510
≠
C
C
≠
N4:11
=
510
≠
N4:12
=
520
≠
N4:13
=
510
≠
N4:14
=
520
≠
N4:15
=
520
≠
N4:16
=
520
≠
N4:17
=
520
≠
N4:18
=
530
≠
N4:19
=
520
≠
N4:2Ø
=
52Ø
≠
N4:21
=
520
≠
N4:22
=
510
≠
N4:23
=
520
≠
N4:24
=
510
≠
N4:25
=
510
≠
C(1)
C(1)
≠
N4:27
=
520
≠
N4:28
=
520
≠
N4:29
=
520
≠
N4:30
=
540
≠
N4:31
=
520
≠
N4:32
=
530


300
SET SUBROUTINE NUMBER 105 INTO T2Ø
SET ERROR CODE=1Ø1
1Ø
400
SET SUBROUTINE NUMBER 100 INTO T2Ø
SET ERROR CODE=1
10
510
ACCESS 1 LOCATION INSTRUCTION
550
520
ACCESS 2 LOCATION INSTRUCTION
550
530
ACCESS 3 LOCATION INSTRUCTION
550
540
ACCESS 4 LOCATION INSTRUCTION
550

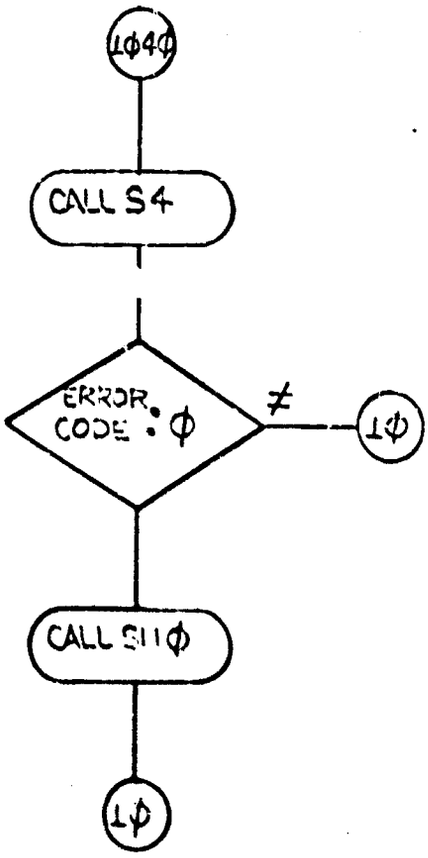

55Φ
SUBTRACT 1 FROM NSI
CALL S951
NSI=Φ
≥
24Φ
<
NSI=Φ
N=99
YES
1Φ
NO
N4:62
>
6ΦΦ
N4:2Φ
>
58Φ
GO TO INSTRUCTION# FOR W/R

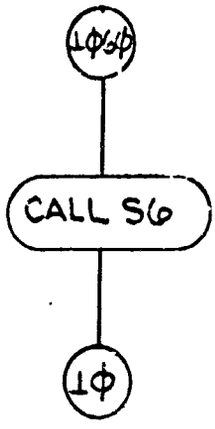

58Φ
GO TO INSTRUCTION # FOR O/S

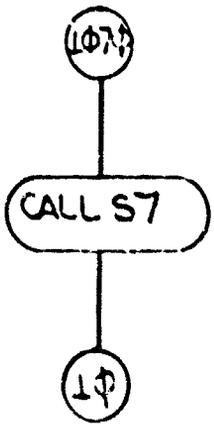

118Φ
CALL S18
ERROR CODE = Φ
≠
1Φ
CALL S11Φ
1Φ
112Φ
CALL S2Φ
1Φ
6ΦΦ
STORE SUBROUTINE #1ΦΦ INTO T2Φ
SET ERROR CODE = 3
1Φ
119Φ
CALL S19
ERROR CODE = Φ
=
1Φ
CALL S11Φ
1Φ
1Φ2Φ
CALL S2
1Φ
1Φ1Φ
CALL S1
1Φ

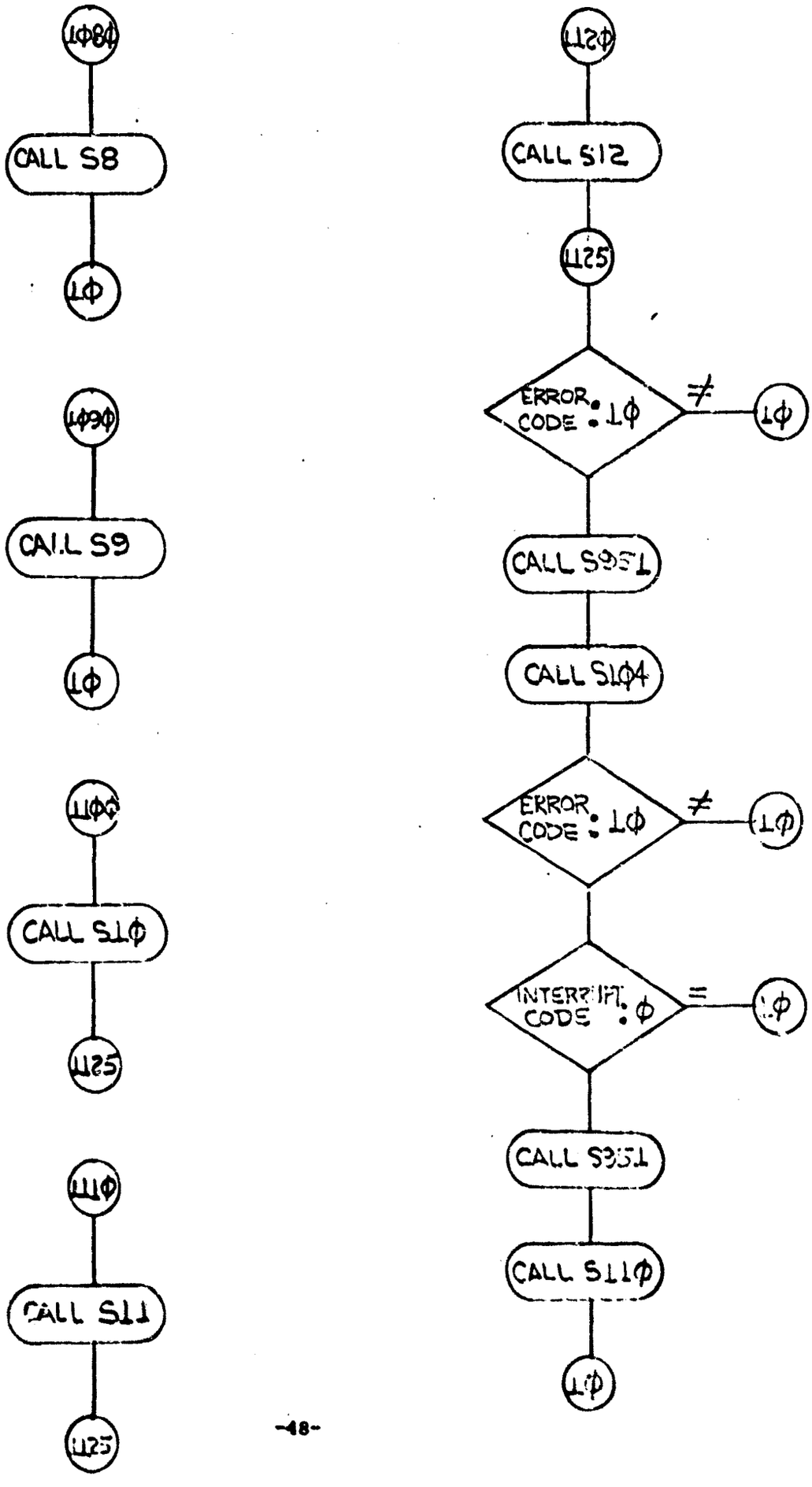

1Φ3Φ
CALL S3
ERROR CODE = Φ
≠
1Φ
CALL S11Φ
1Φ

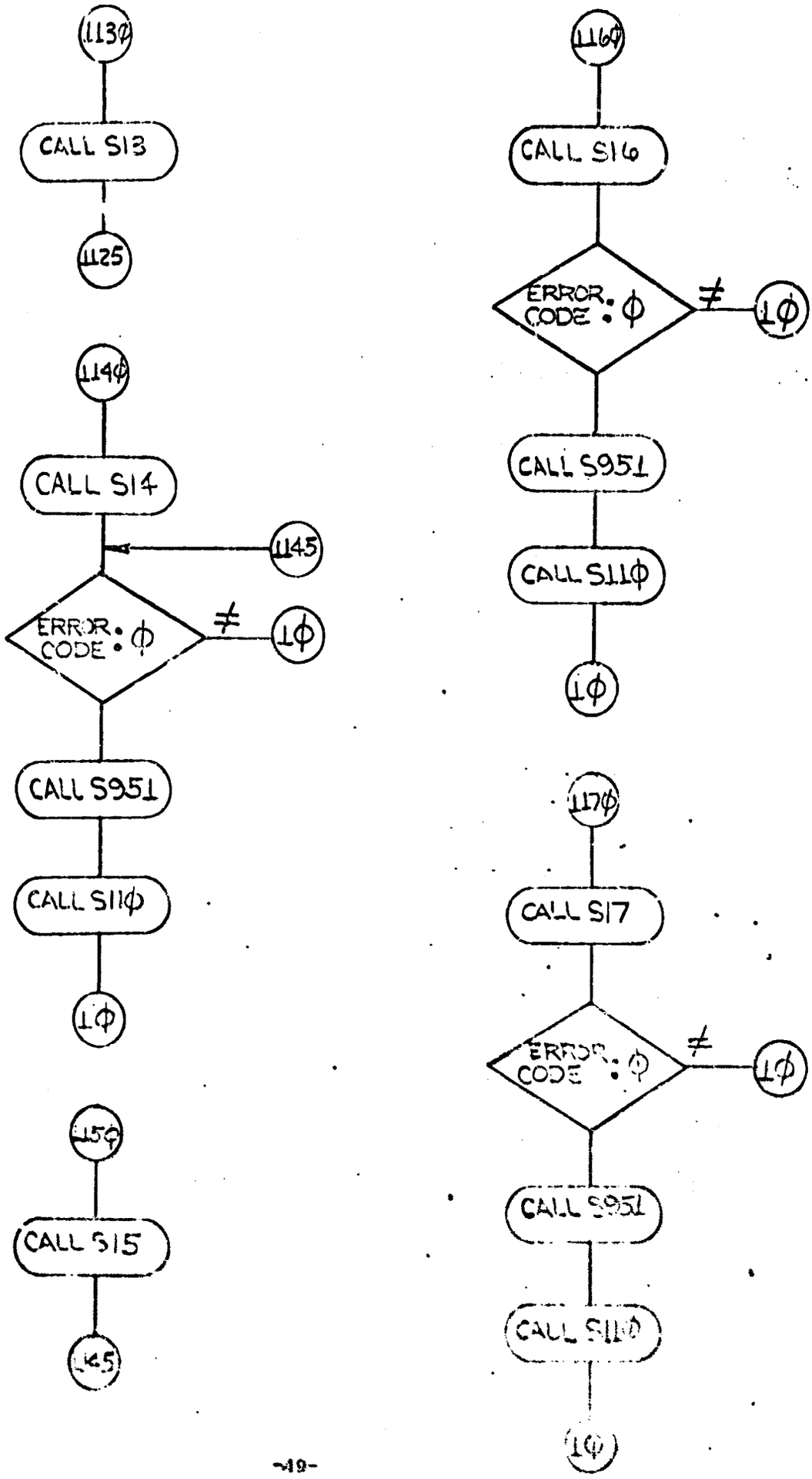

1Φ5Φ
CALL S5
CALL S951
CALL S11Φ
1Φ

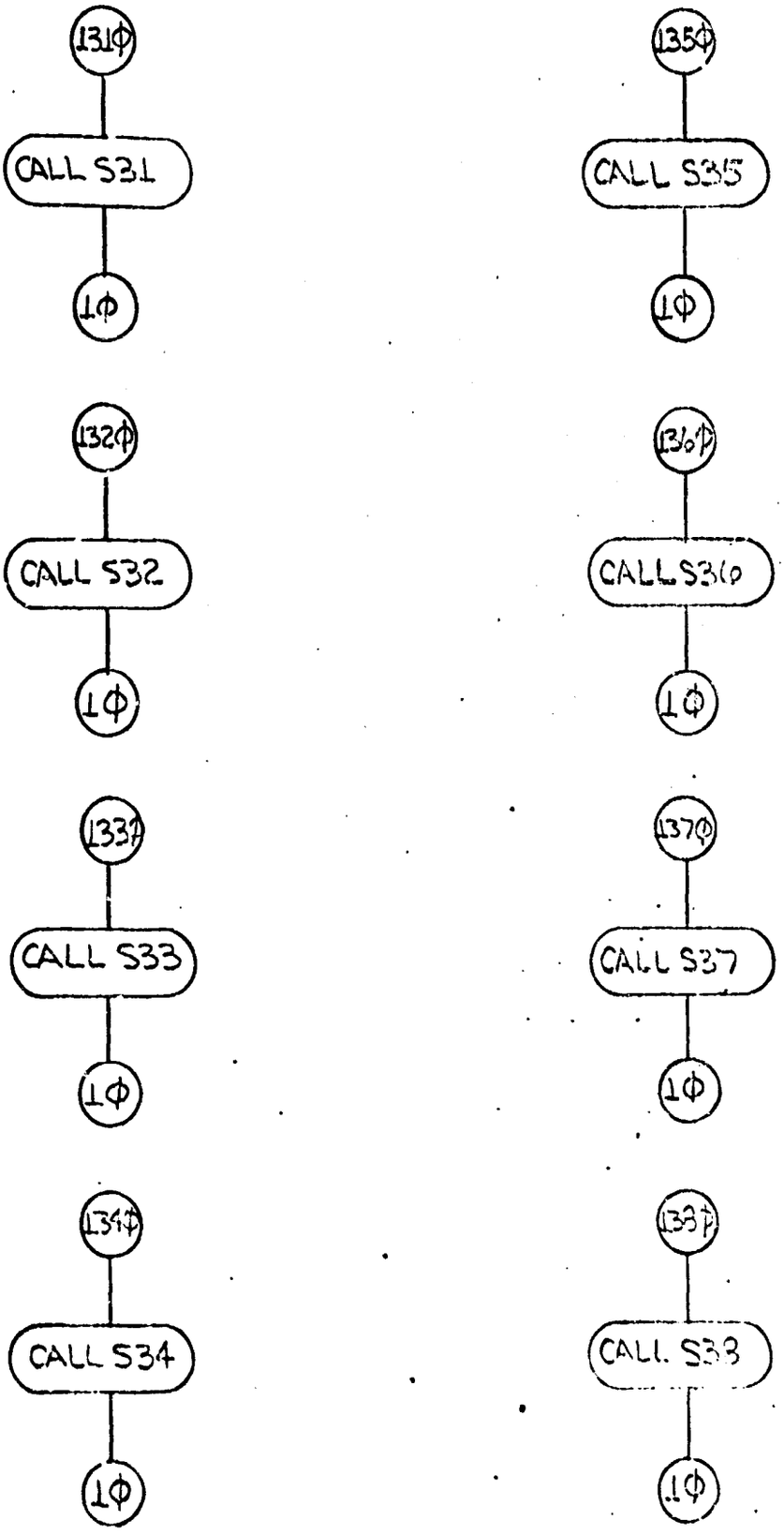

1Φ4Φ
CALL S4
ERROR CODE = Φ
≠
1Φ
CALL S11Φ
1Φ

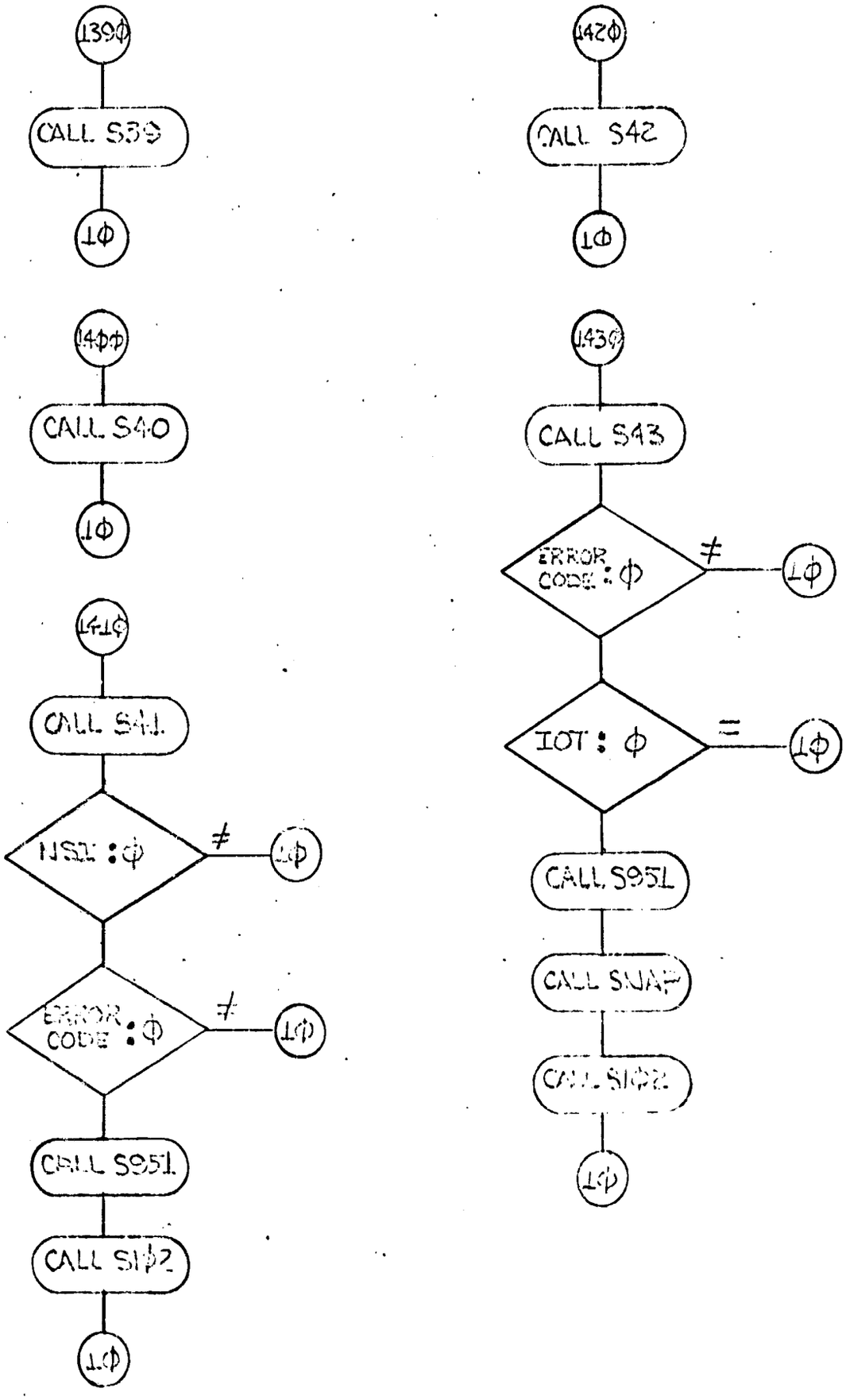

1Φ6Φ
CALL S6
1Φ

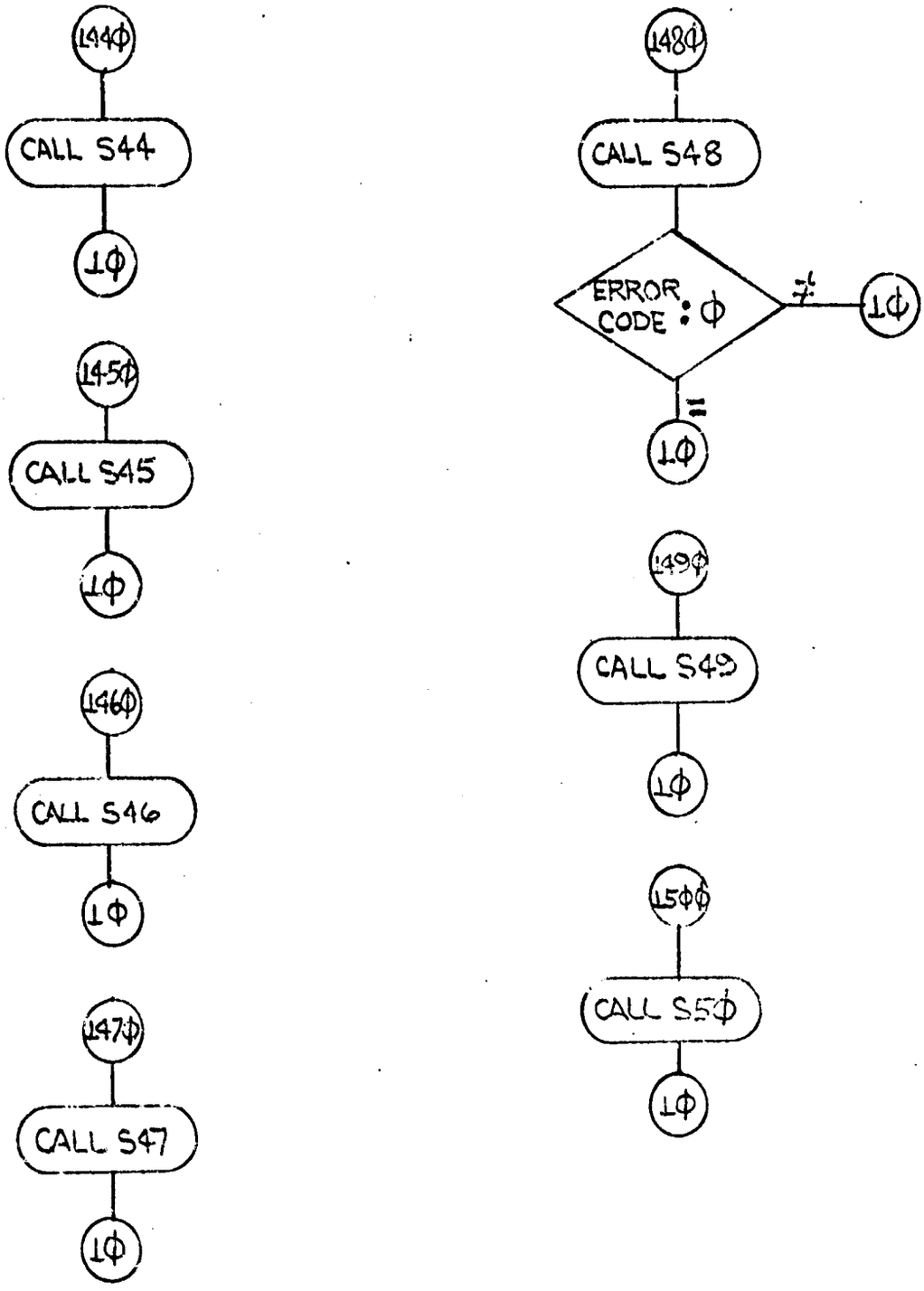

1Φ7Φ
CALL S7
1Φ

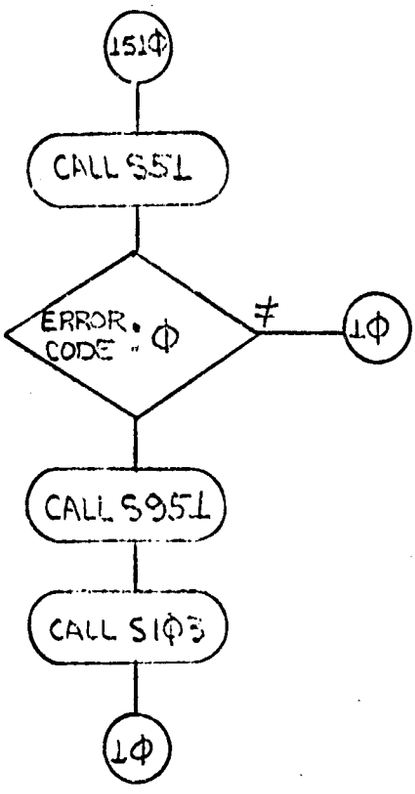

1Φ8Φ
CALL S8
1Φ
1Φ9Φ
CALL S9
1Φ
11ΦΦ
CALL S1Φ
1125
111Φ
CALL S11
1125
1125
CALL S12
1125
ERROR CODE : 1Φ
≠
1Φ
CALL S9S1
CALL S1Φ4
ERROR CODE : 1Φ
≠
1Φ
INTERRUPT CODE : Φ
=
1Φ
CALL S3S1
CALL S11Φ
1Φ

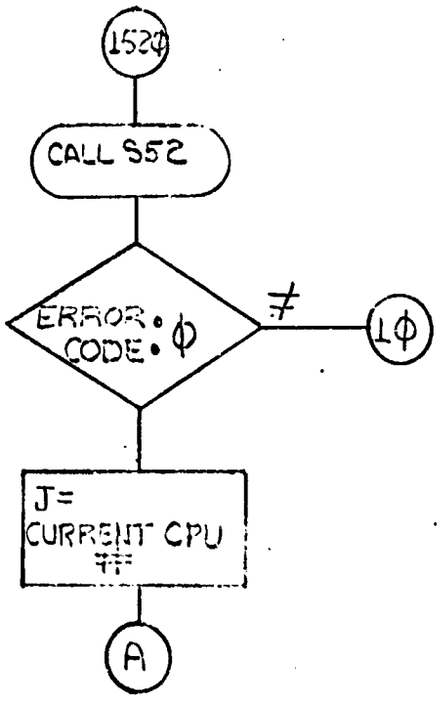

1130
CALL S13
1125
1140
CALL S14
1145
ERROR CODE = Φ
≠
1Φ
CALL S951
CALL S11Φ
1Φ
150
CALL S15
145
1160
CALL S16
ERROR CODE = Φ
≠
1Φ
CALL S951
CALL S11Φ
1Φ
1170
CALL S17
ERROR CODE = Φ
≠
1Φ
CALL S951
CALL S11Φ
1Φ

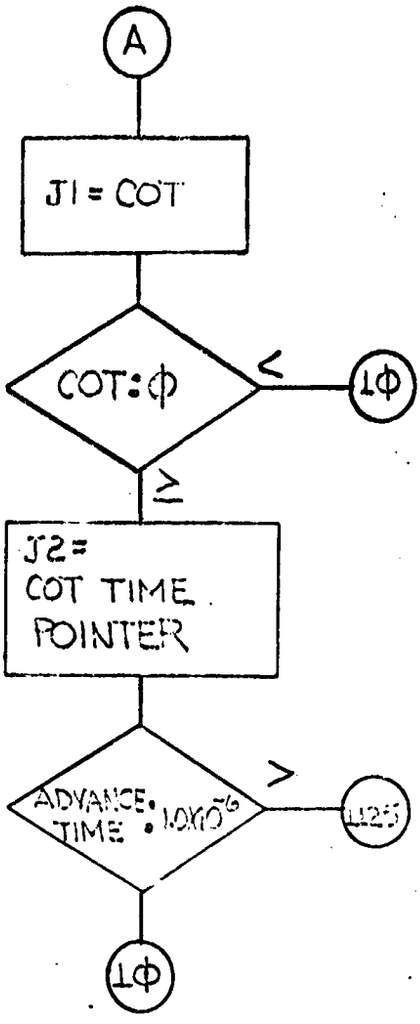

131Φ
CALL S31
1Φ
135Φ
CALL S35
1Φ
132Φ
CALL S32
1Φ
136Φ
CALL S36
1Φ
133Φ
CALL S33
1Φ
137Φ
CALL S37
1Φ
134Φ
CALL S34
1Φ
138Φ
CALL S38
1Φ

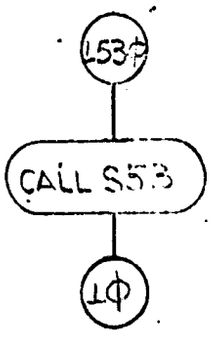

1390
CALL S39
1Φ
1420
CALL S42
1Φ
140Φ
CALL S40
1Φ
1430
CALL S43
ERROR CODE : Φ
≠
1Φ
IOT : Φ
=
1Φ
141Φ
CALL S41
NSE : Φ
≠
1Φ
ERROR CODE : Φ
≠
1Φ
CALL S95L
CALL S102
1Φ
CALL S95L
CALL SNAP
CALL S102
1Φ

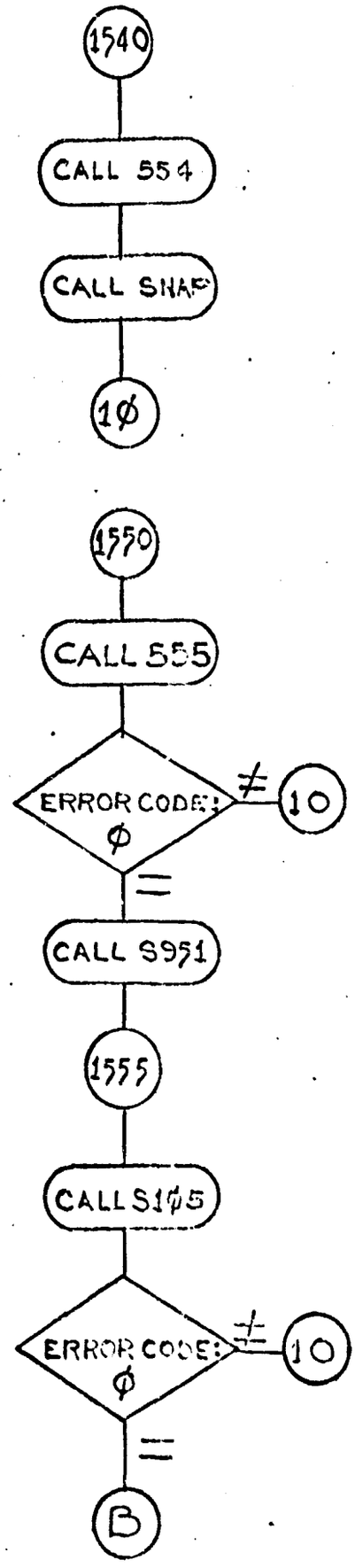

1440
CALL S44
10
1450
CALL S45
10
1460
CALL S46
10
1470
CALL S47
10
1480
CALL S48
ERROR CODE : 0
≠
10
=
10
1490
CALL S49
10
15000
CALL S50
10

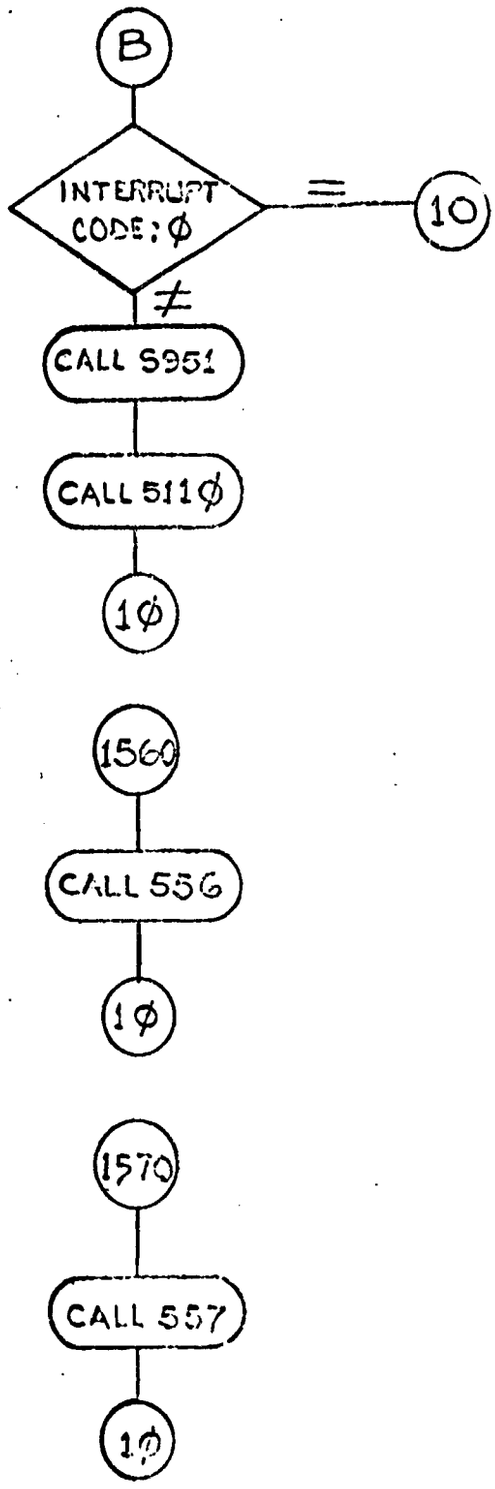

151Φ
CALL S51
ERROR CODE : Φ
≠
1Φ
CALL S95L
CALL S1Φ3
1Φ

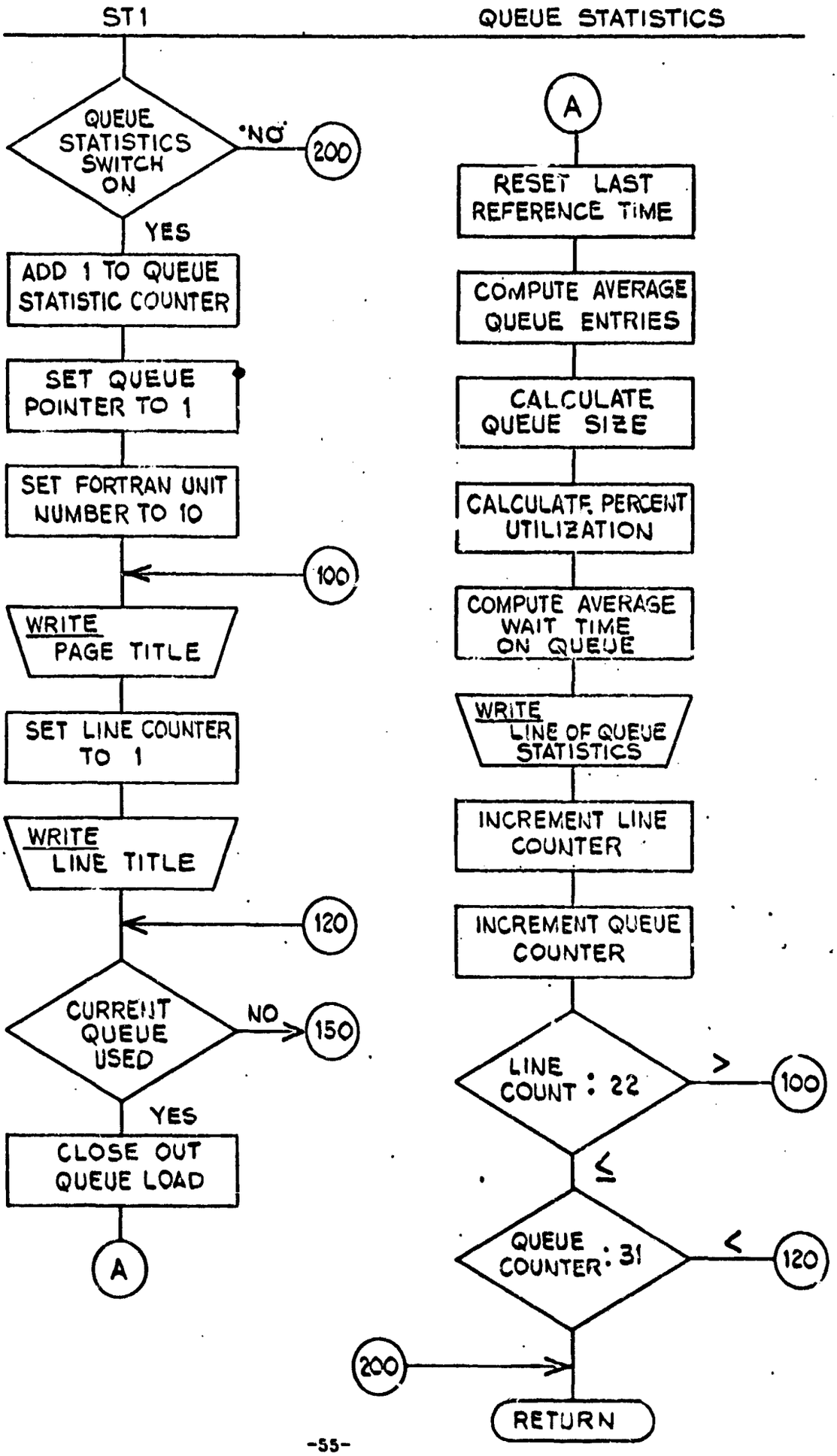

152Φ
CALL S52
ERROR CODE : Φ
≠
1Φ
J=
CURRENT CPU
##
A

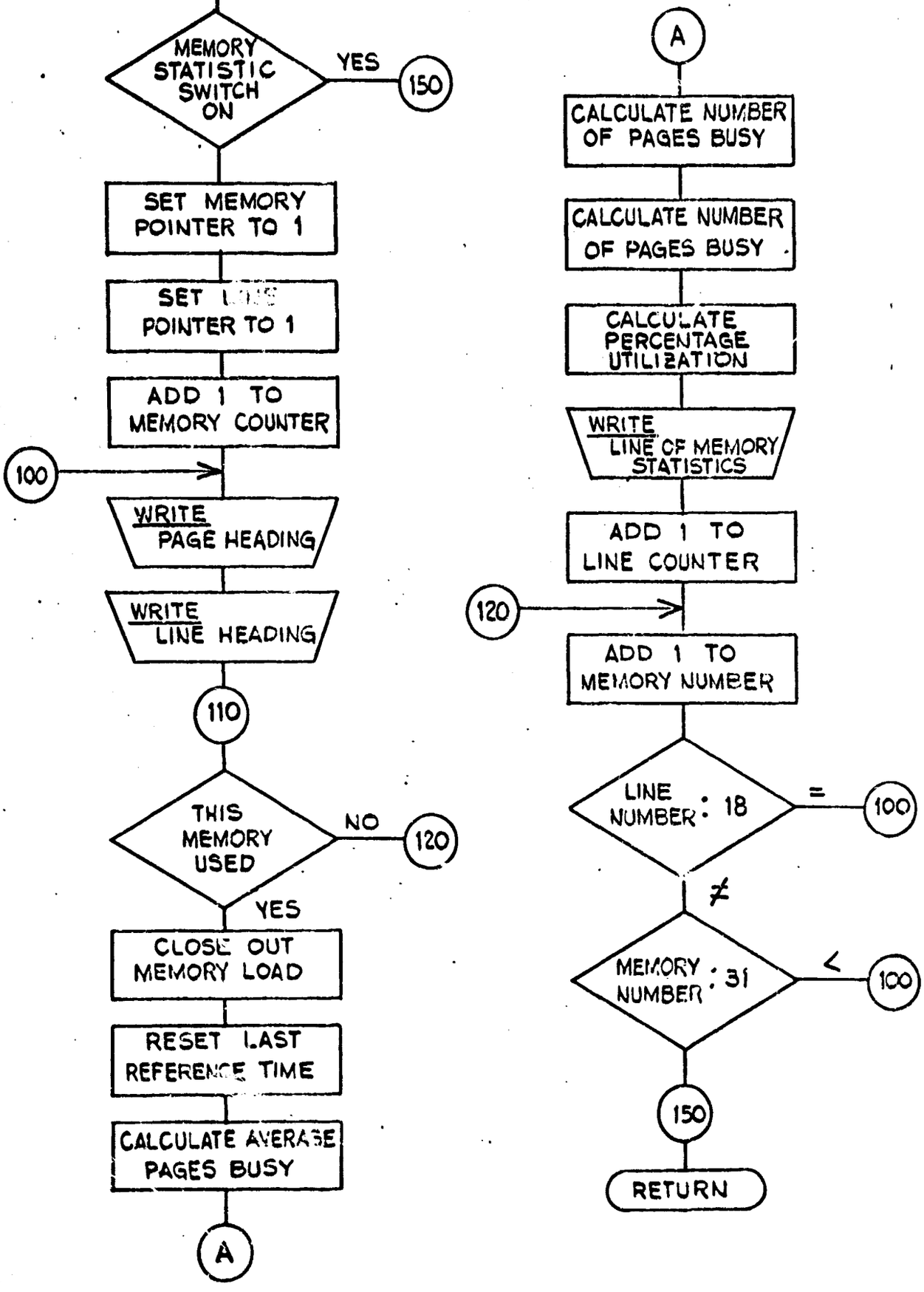

A
J1 = COT
COT : Φ
<
1Φ
≥
J2=
COT TIME
POINTER
ADVANCE TIME : 1.0X10^6
>
1125
1Φ

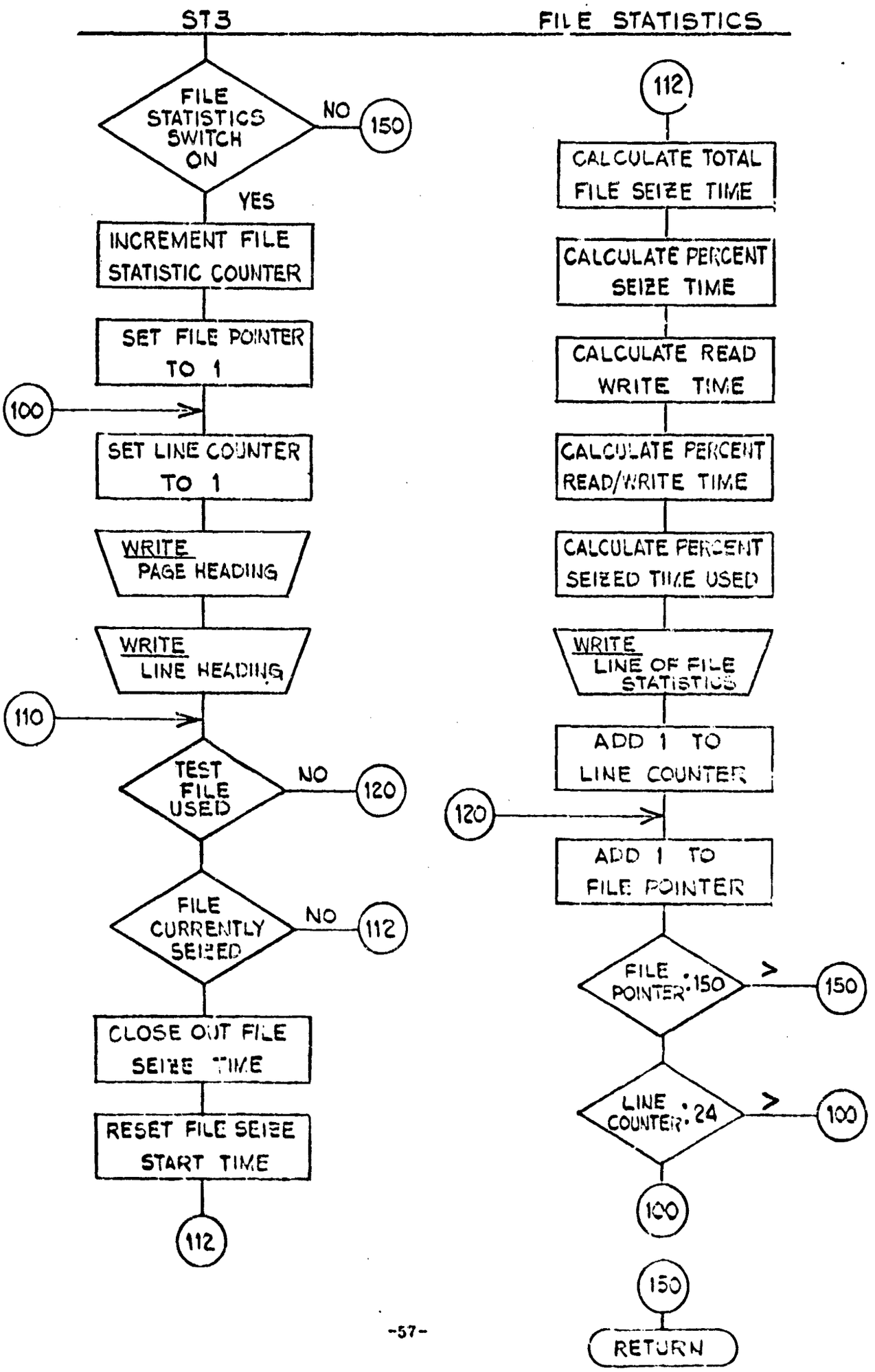

153#
CALL S53
1Φ

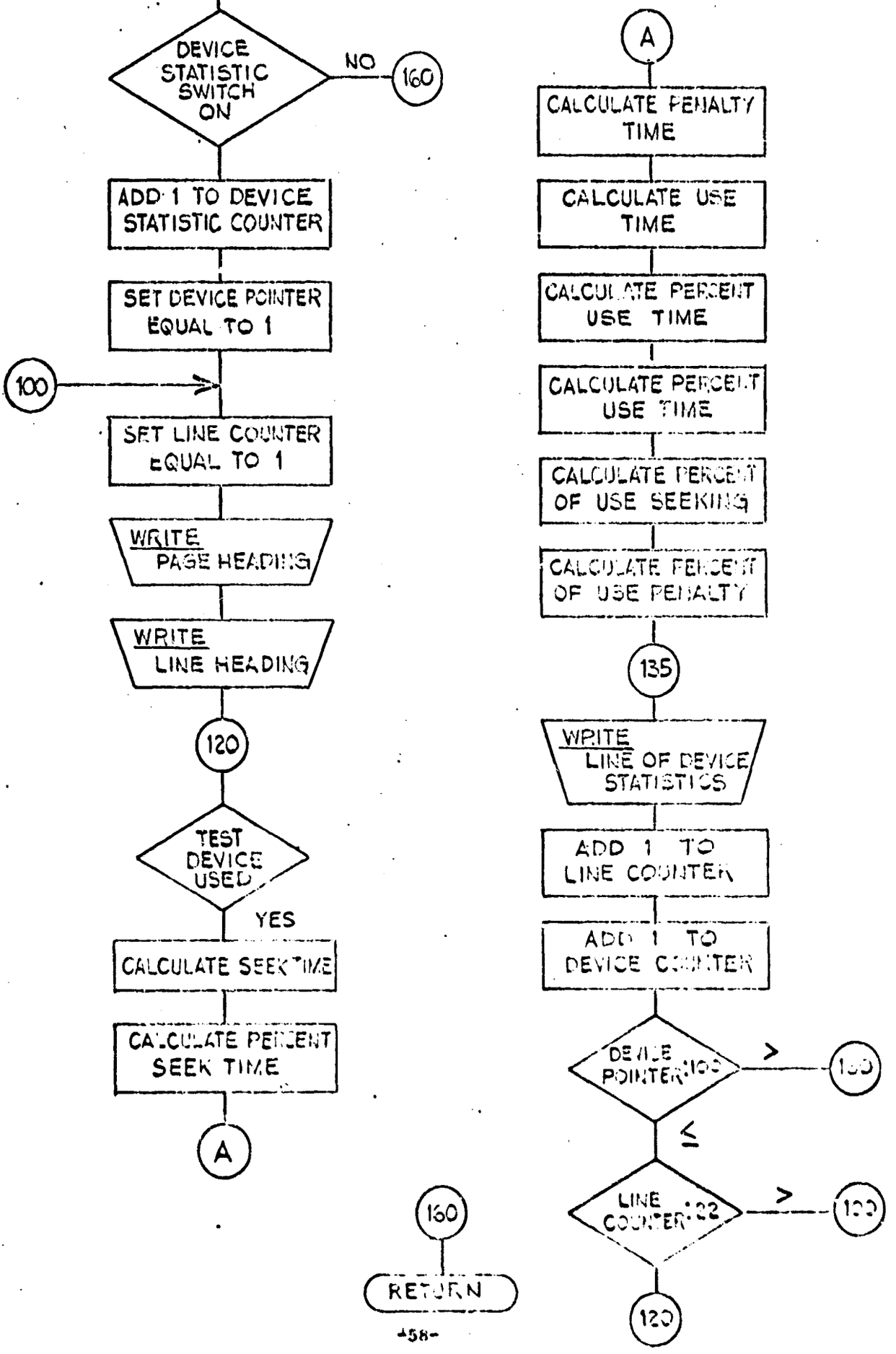

1540
CALL 554
CALL SNAP
1Ø
1550
CALL 555
ERROR CODE: Ø
≠
10
=
CALL S951
1555
CALL S1Ø5
ERROR CODE: Ø
≠
10
=
B

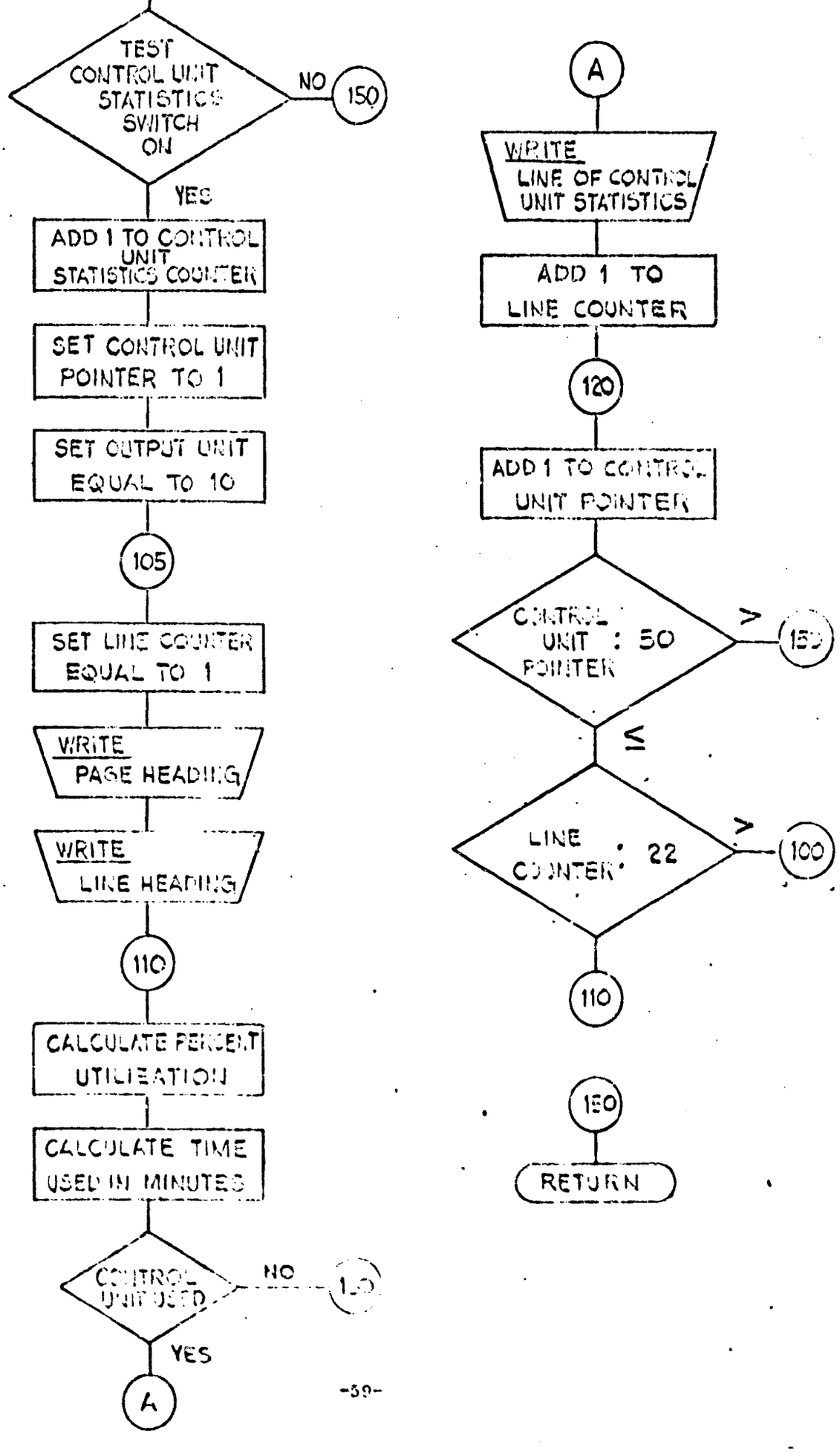

B
INTERRUPT CODE: Ø
=
10
≠
CALL S951
CALL 511Ø
1Ø
1560
CALL 556
1Ø
1570
CALL 557
1Ø

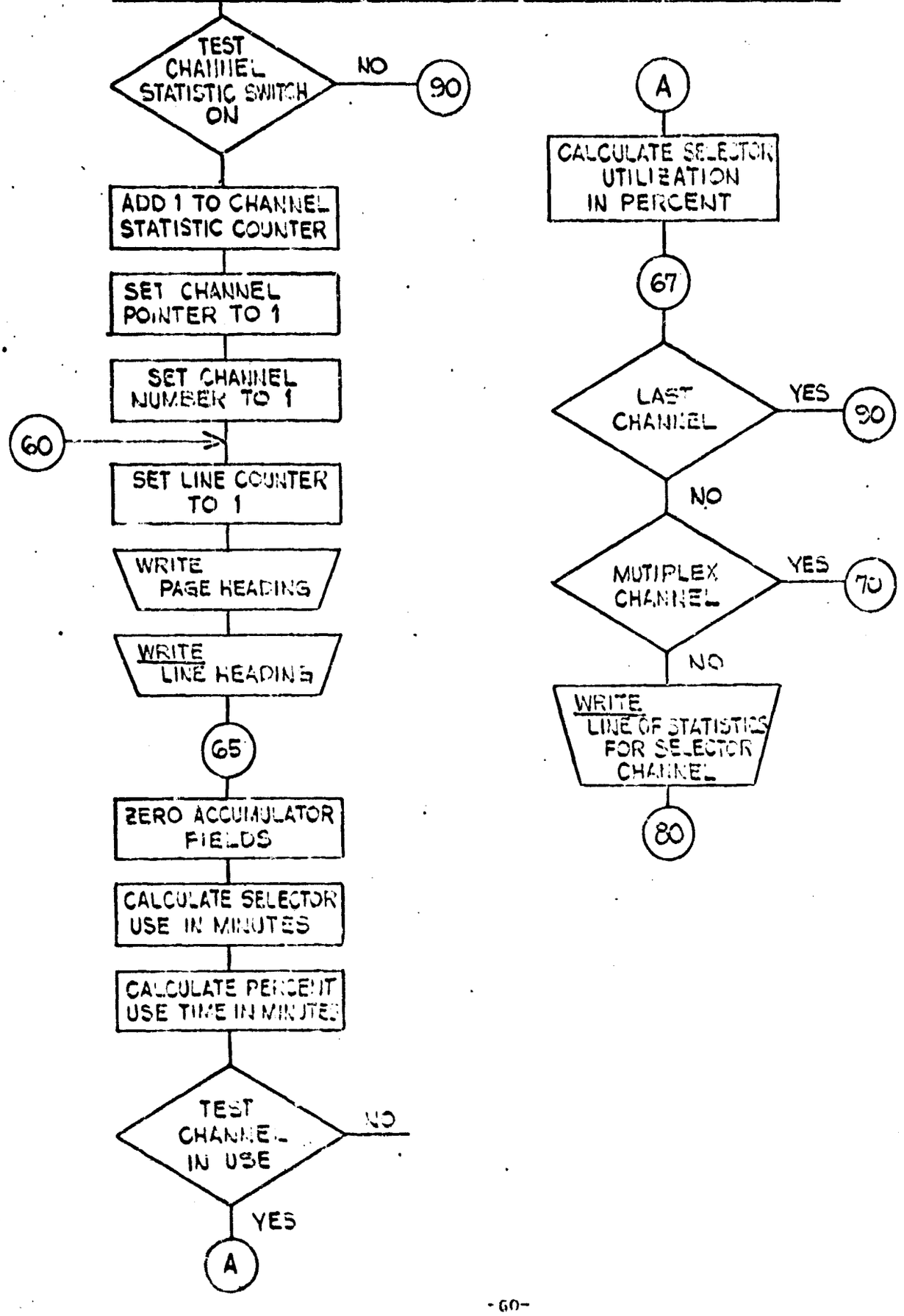

ST1
QUEUE STATISTICS
QUEUE STATISTICS SWITCH ON
"NO"
200
YES
ADD 1 TO QUEUE STATISTIC COUNTER
SET QUEUE POINTER TO 1
SET FORTRAN UNIT NUMBER TO 10
100
WRITE PAGE TITLE
SET LINE COUNTER TO 1
WRITE LINE TITLE
120
CURRENT QUEUE USED
NO
150
YES
CLOSE OUT QUEUE LOAD
A
A
RESET LAST REFERENCE TIME
COMPUTE AVERAGE QUEUE ENTRIES
CALCULATE QUEUE SIZE
CALCULATE PERCENT UTILIZATION
COMPUTE AVERAGE WAIT TIME ON QUEUE
WRITE LINE OF QUEUE STATISTICS
INCREMENT LINE COUNTER
INCREMENT QUEUE COUNTER
LINE COUNT : 22
>
100
≤
QUEUE COUNTER : 31
<
120
200
RETURN


MEMORY STATISTIC SWITCH ON
YES
150
SET MEMORY POINTER TO 1
SET LINE POINTER TO 1
ADD 1 TO MEMORY COUNTER
100
WRITE PAGE HEADING
WRITE LINE HEADING
110
THIS MEMORY USED
NO
120
YES
CLOSE OUT MEMORY LOAD
RESET LAST REFERENCE TIME
CALCULATE AVERAGE PAGES BUSY
A
A
CALCULATE NUMBER OF PAGES BUSY
CALCULATE NUMBER OF PAGES BUSY
CALCULATE PERCENTAGE UTILIZATION
WRITE LINE OF MEMORY STATISTICS
ADD 1 TO LINE COUNTER
120
ADD 1 TO MEMORY NUMBER
LINE NUMBER : 18
=
100
≠
MEMORY NUMBER : 31
<
100
150
RETURN


ST3
FILE STATISTICS
FILE STATISTICS SWITCH ON
NO
150
YES
INCREMENT FILE STATISTIC COUNTER
SET FILE POINTER TO 1
100
SET LINE COUNTER TO 1
WRITE PAGE HEADING
WRITE LINE HEADING
110
TEST FILE USED
NO
120
FILE CURRENTLY SEIZED
NO
112
CLOSE OUT FILE SEIZE TIME
RESET FILE SEIZE START TIME
112
112
CALCULATE TOTAL FILE SEIZE TIME
CALCULATE PERCENT SEIZE TIME
CALCULATE READ WRITE TIME
CALCULATE PERCENT READ/WRITE TIME
CALCULATE PERCENT SEIZED TIME USED
WRITE LINE OF FILE STATISTICS
ADD 1 TO LINE COUNTER
120
ADD 1 TO FILE POINTER
FILE POINTER .150
>
150
LINE COUNTER .24
>
100
100
150
RETURN

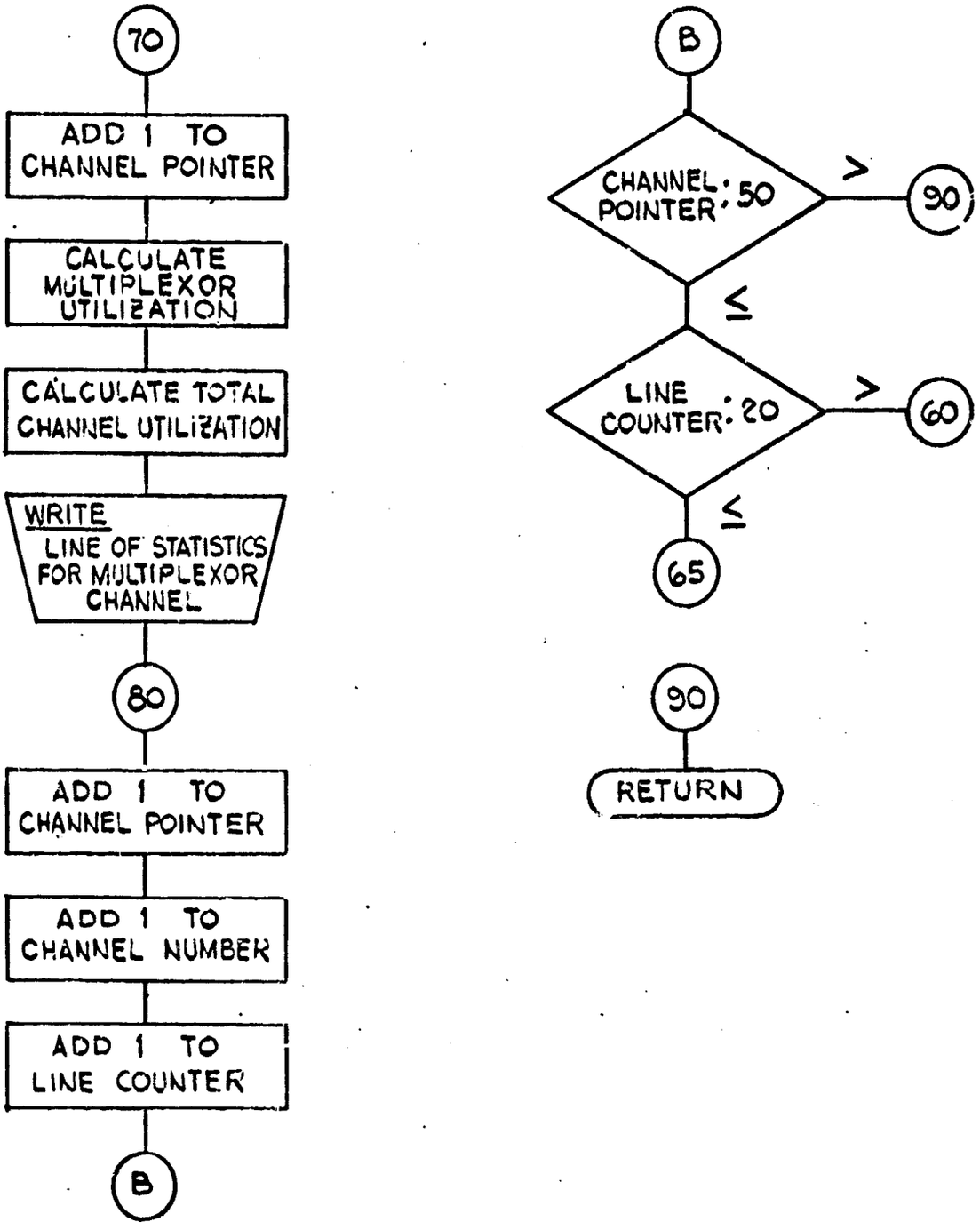

DEVICE STATISTIC SWITCH ON
NO
160
ADD 1 TO DEVICE STATISTIC COUNTER
SET DEVICE POINTER EQUAL TO 1
100
SET LINE COUNTER EQUAL TO 1
WRITE PAGE HEADING
WRITE LINE HEADING
120
TEST DEVICE USED
YES
CALCULATE SEEK TIME
CALCULATE PERCENT SEEK TIME
A
A
CALCULATE PENALTY TIME
CALCULATE USE TIME
CALCULATE PERCENT USE TIME
CALCULATE PERCENT USE TIME
CALCULATE PERCENT OF USE SEEKING
CALCULATE PERCENT OF USE PENALTY
135
WRITE LINE OF DEVICE STATISTICS
ADD 1 TO LINE COUNTER
ADD 1 TO DEVICE COUNTER
DEVICE POINTER:100
>
100
≤
LINE COUNTER:22
>
100
120
160
RETURN

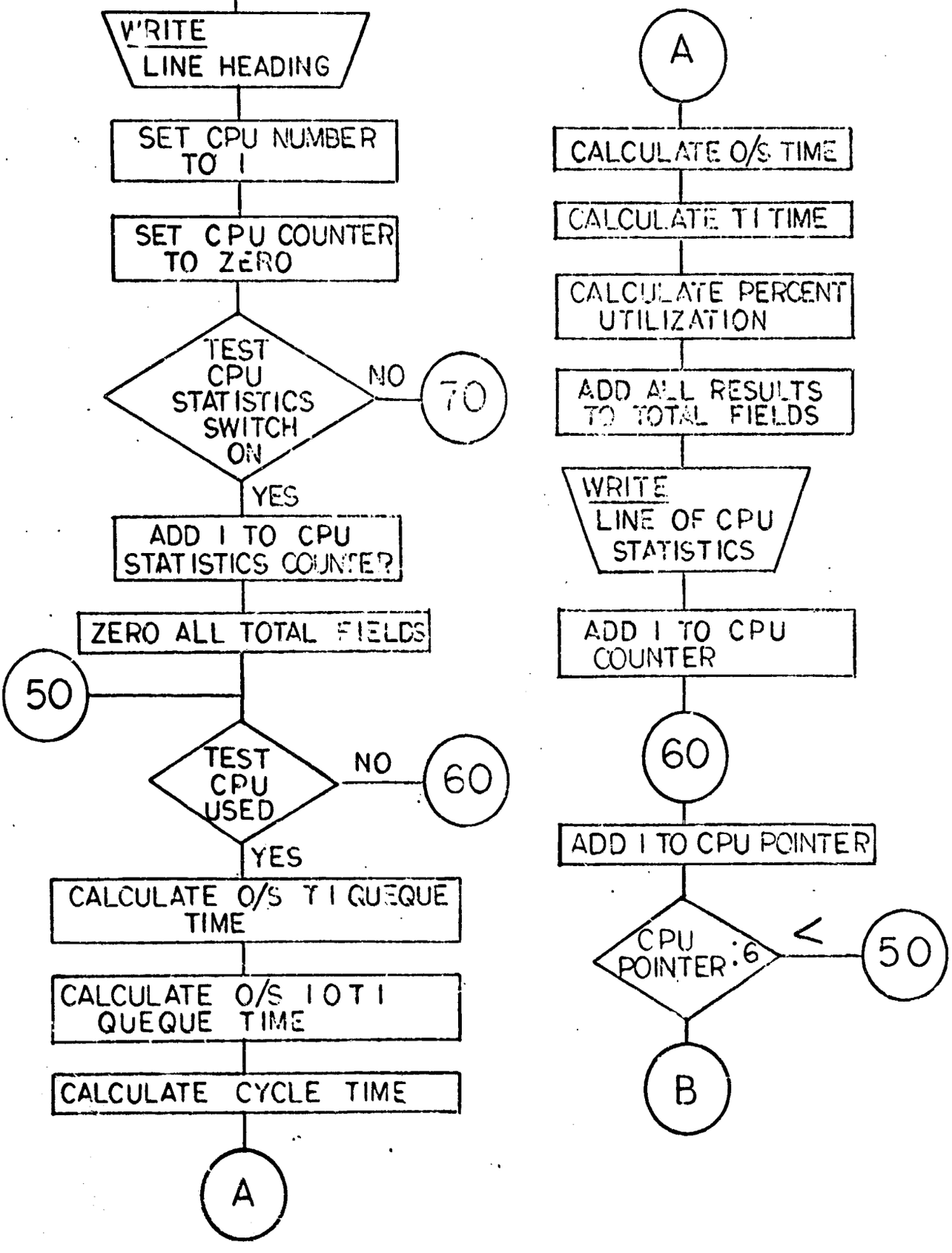

TEST CONTROL UNIT STATISTICS SWITCH ON
NO
150
YES
ADD 1 TO CONTROL UNIT STATISTICS COUNTER
SET CONTROL UNIT POINTER TO 1
SET OUTPUT UNIT EQUAL TO 10
105
SET LINE COUNTER EQUAL TO 1
WRITE PAGE HEADING
WRITE LINE HEADING
110
CALCULATE PERCENT UTILIZATION
CALCULATE TIME USED IN MINUTES
CONTROL UNIT USED
NO
YES
A
A
WRITE LINE OF CONTROL UNIT STATISTICS
ADD 1 TO LINE COUNTER
120
ADD 1 TO CONTROL UNIT POINTER
CONTROL UNIT POINTER : 50
>
≤
LINE COUNTER : 22
>
100
110
150
RETURN


TEST CHANNEL STATISTIC SWITCH ON
NO
90
ADD 1 TO CHANNEL STATISTIC COUNTER
SET CHANNEL POINTER TO 1
SET CHANNEL NUMBER TO 1
60
SET LINE COUNTER TO 1
WRITE PAGE HEADING
WRITE LINE HEADING
65
ZERO ACCUMULATOR FIELDS
CALCULATE SELECTOR USE IN MINUTES
CALCULATE PERCENT USE TIME IN MINUTES
TEST CHANNEL IN USE
NO
YES
A
A
CALCULATE SELECTOR UTILIZATION IN PERCENT
67
LAST CHANNEL
YES
90
NO
MUTIPLEX CHANNEL
YES
70
NO
WRITE LINE OF STATISTICS FOR SELECTOR CHANNEL
80

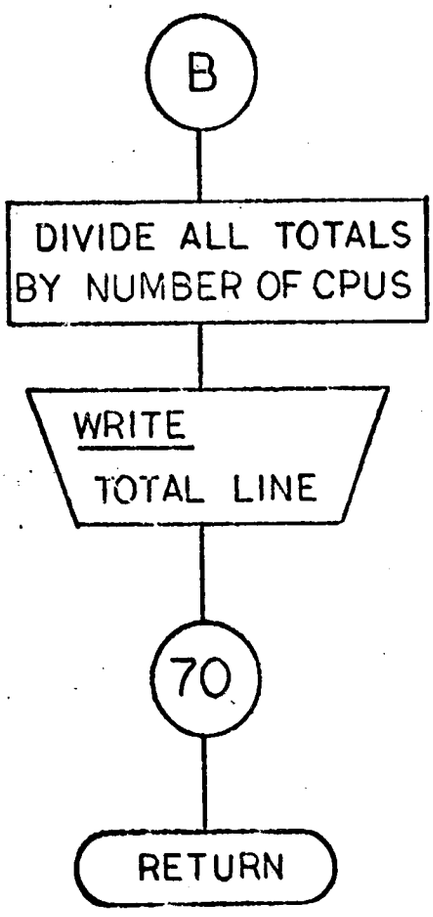
PG 1

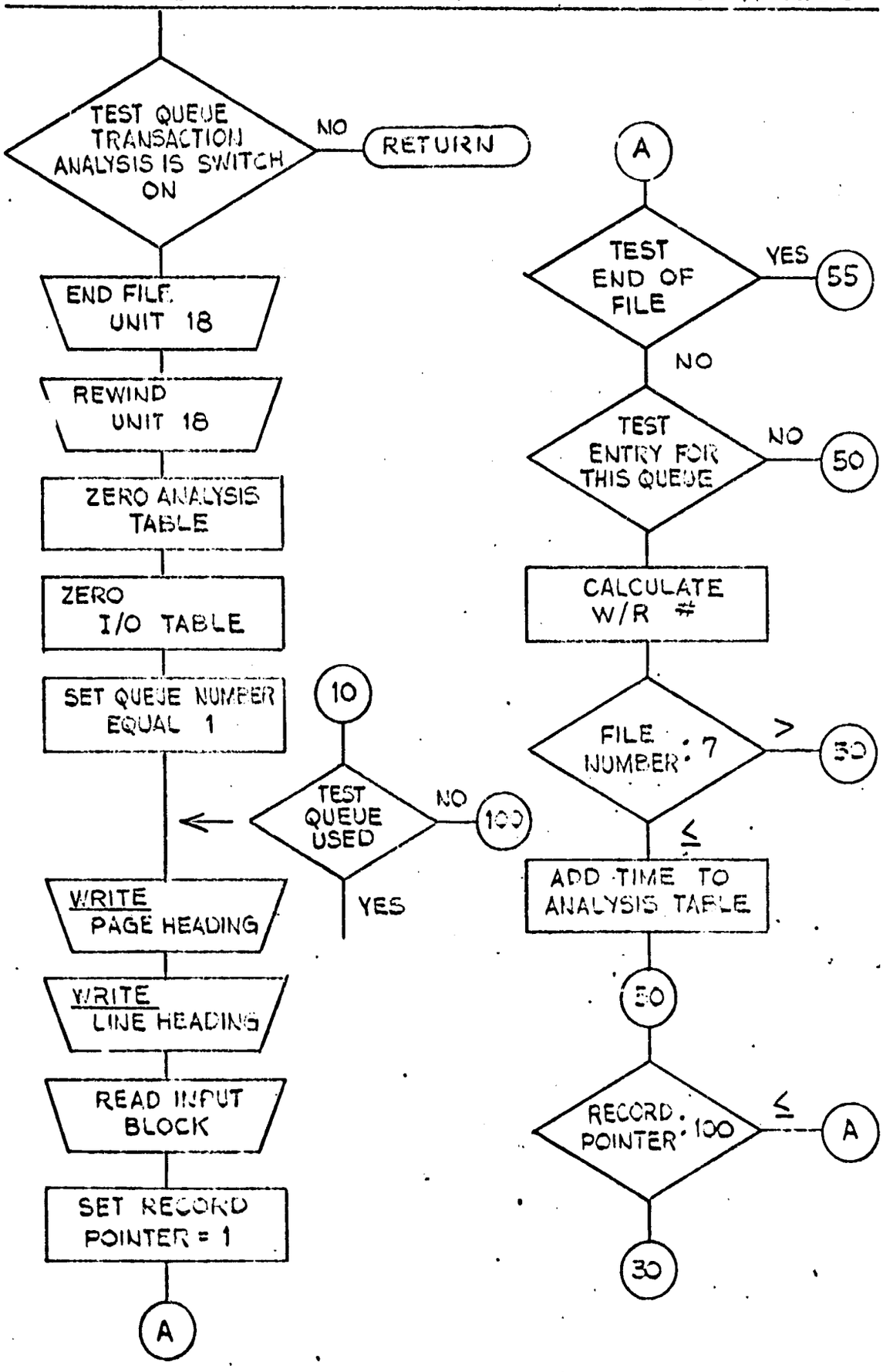
CHANNEL STATISTICS (CONTINUED)




WRITE LINE HEADING
SET CPU NUMBER TO 1
SET CPU COUNTER TO ZERO
TEST CPU STATISTICS SWITCH ON
NO
70
YES
ADD 1 TO CPU STATISTICS COUNTER?
ZERO ALL TOTAL FIELDS
50
TEST CPU USED
NO
60
YES
CALCULATE O/S TI QUEQUE TIME
CALCULATE O/S IOTI QUEQUE TIME
CALCULATE CYCLE TIME
A
A
CALCULATE O/S TIME
CALCULATE TI TIME
CALCULATE PERCENT UTILIZATION
ADD ALL RESULTS TO TOTAL FIELDS
WRITE LINE OF CPU STATISTICS
ADD 1 TO CPU COUNTER
60
ADD 1 TO CPU POINTER
CPU POINTER :6
<
50
B

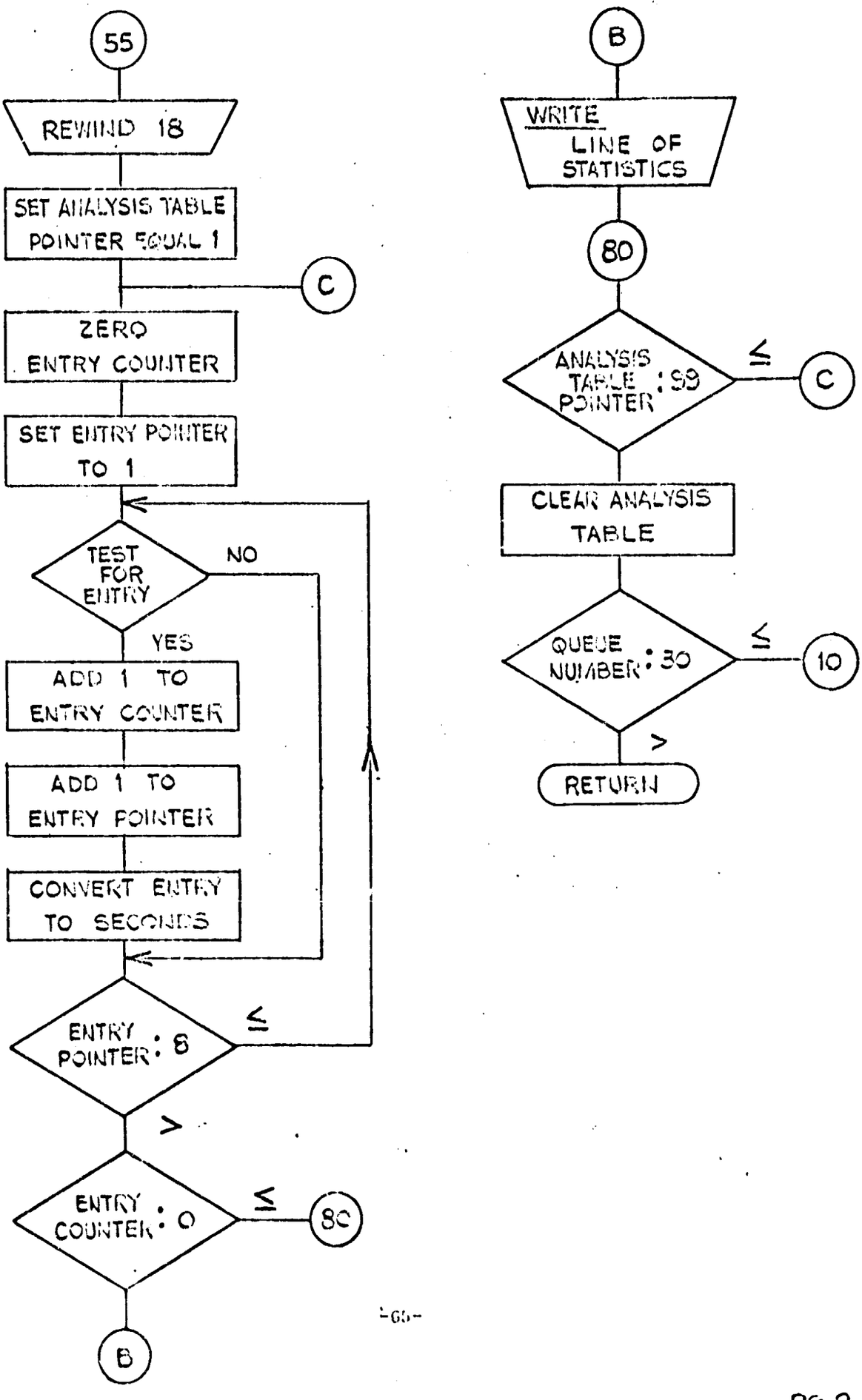
CPU STATISTICS (CONTINUED)



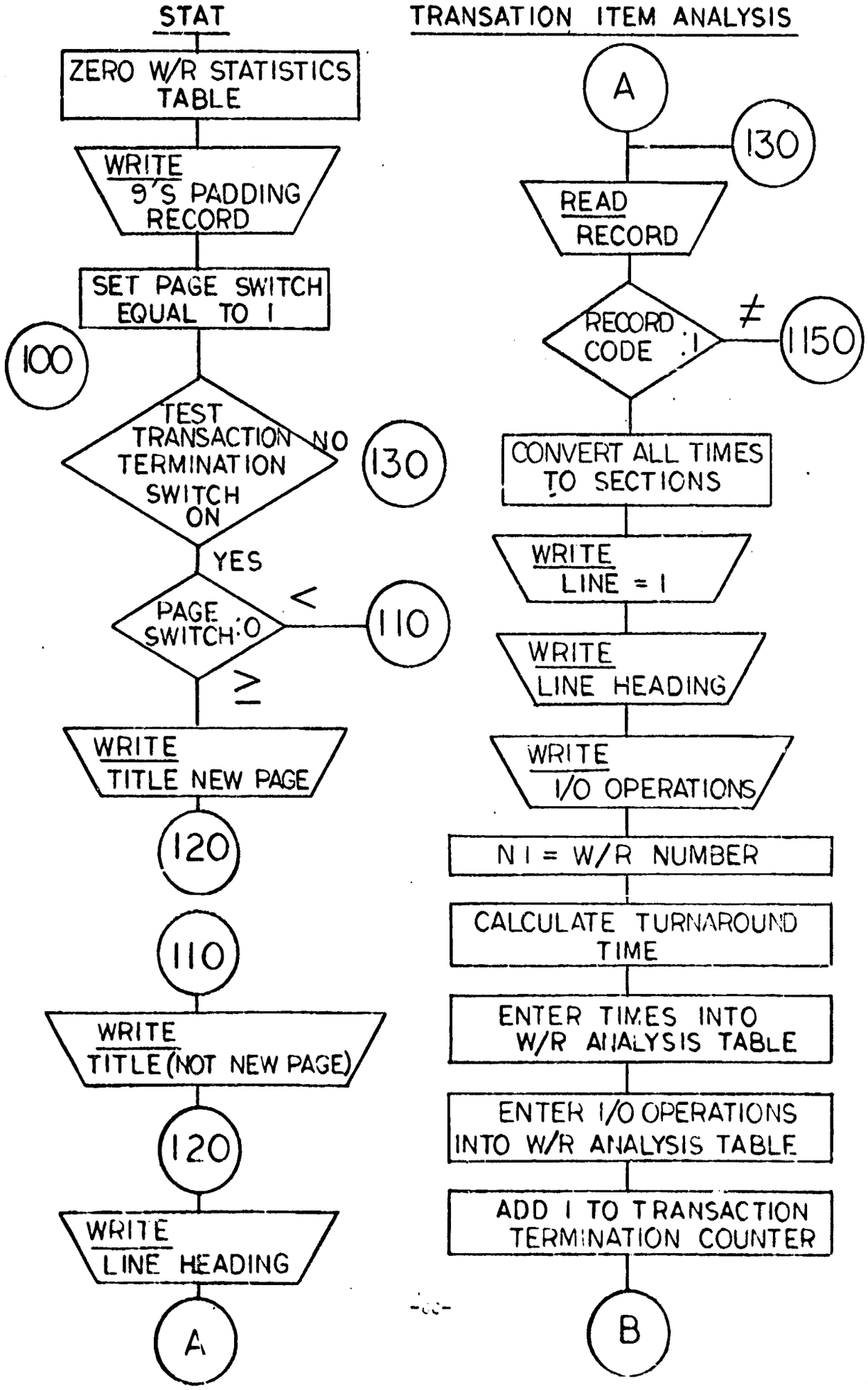

TEST QUEUE TRANSACTION ANALYSIS IS SWITCH ON
NO
RETURN
END FILE UNIT 18
REWIND UNIT 18
ZERO ANALYSIS TABLE
ZERO I/O TABLE
SET QUEUE NUMBER EQUAL 1
WRITE PAGE HEADING
WRITE LINE HEADING
READ INPUT BLOCK
SET RECORD POINTER = 1
A
10
TEST QUEUE USED
NO
100
YES
A
TEST END OF FILE
YES
55
NO
TEST ENTRY FOR THIS QUEUE
NO
50
CALCULATE W/R #
FILE NUMBER : 7
>
30
≤
ADD TIME TO ANALYSIS TABLE
30
RECORD POINTER : 100
≤
A
30

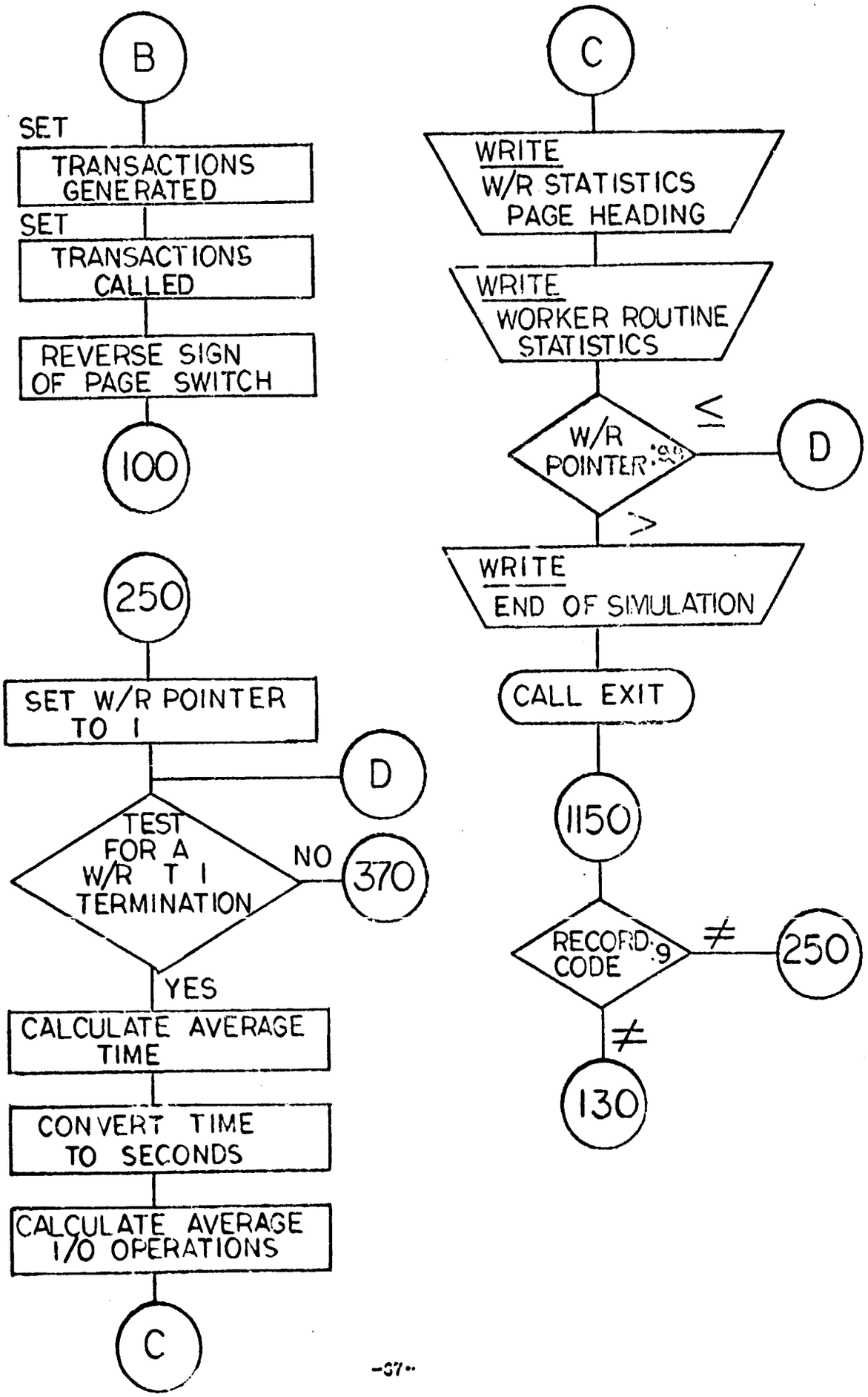
Pg 1

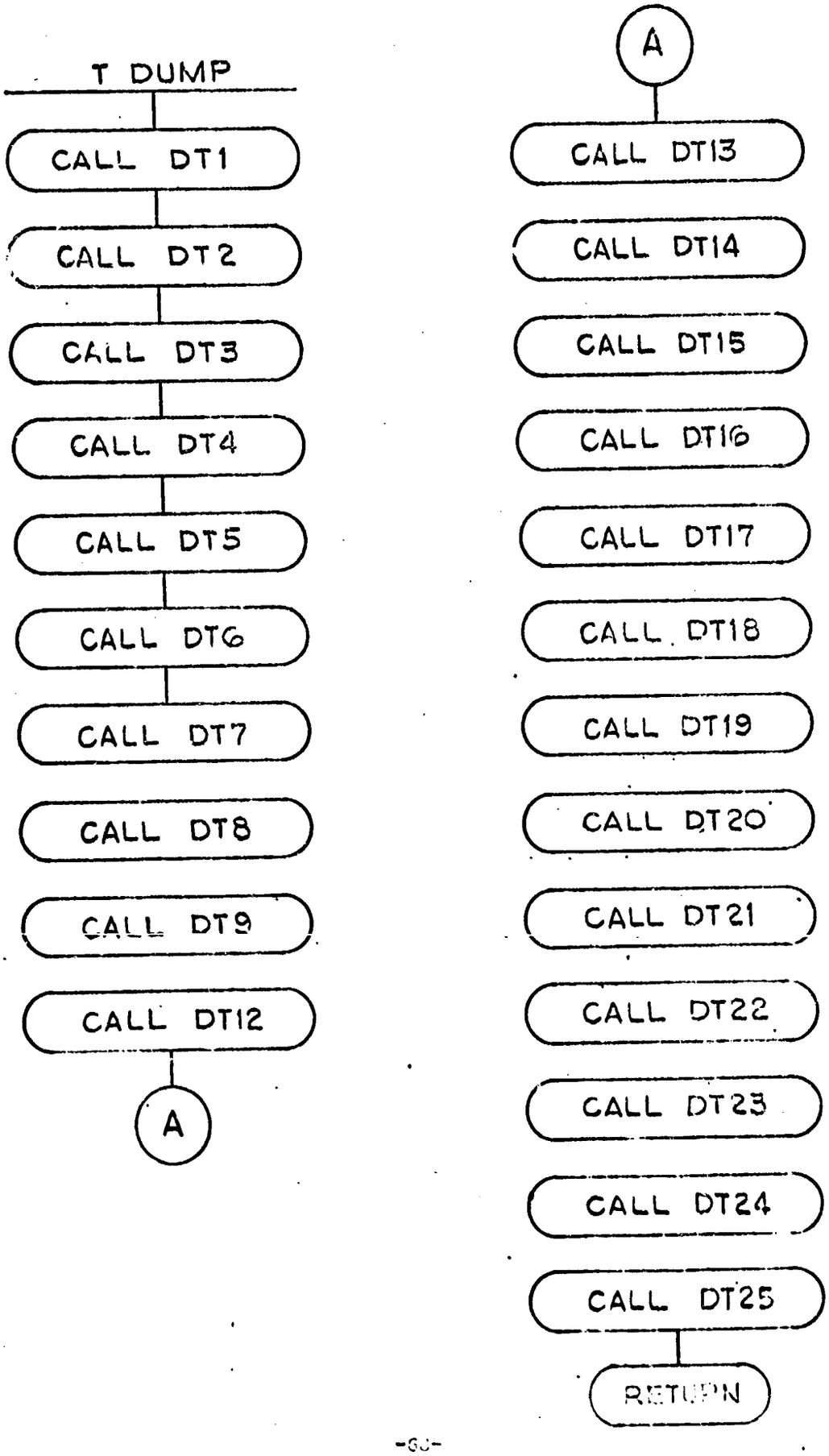
QUEUE TRANSACTION ANALYSIS (CONTINUED)

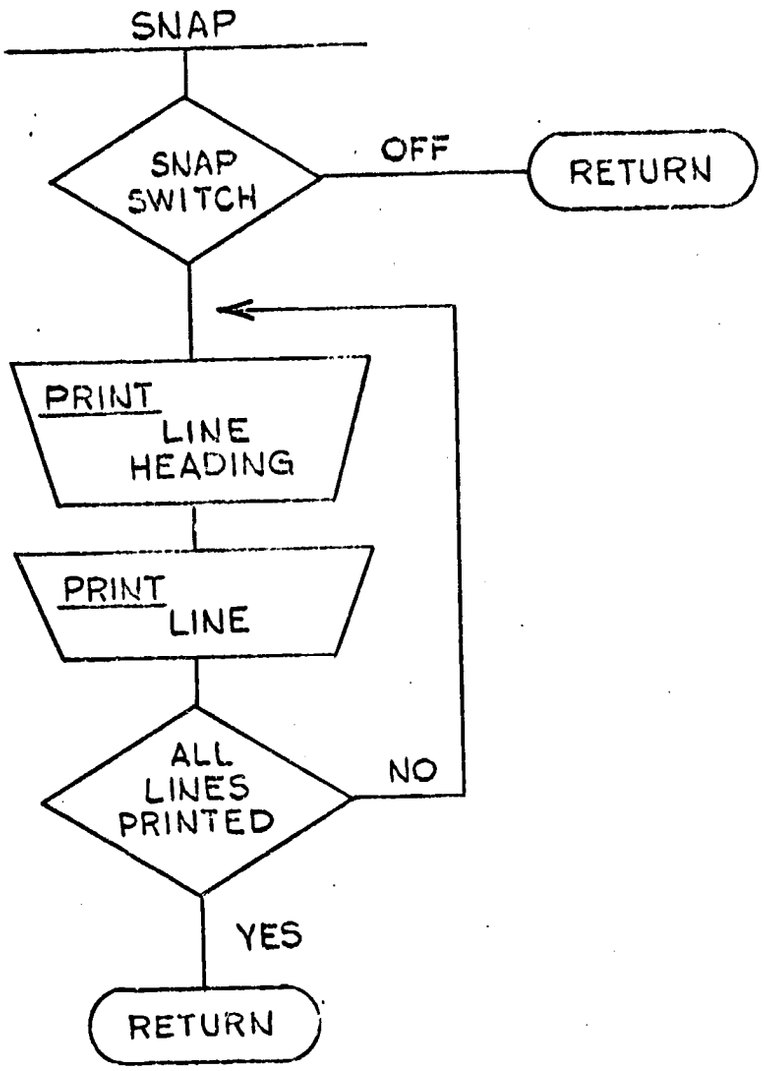


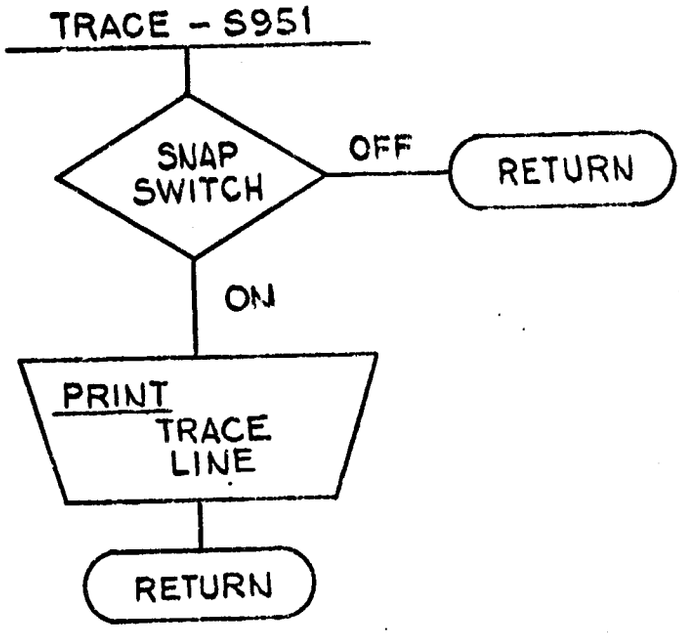
PG 2

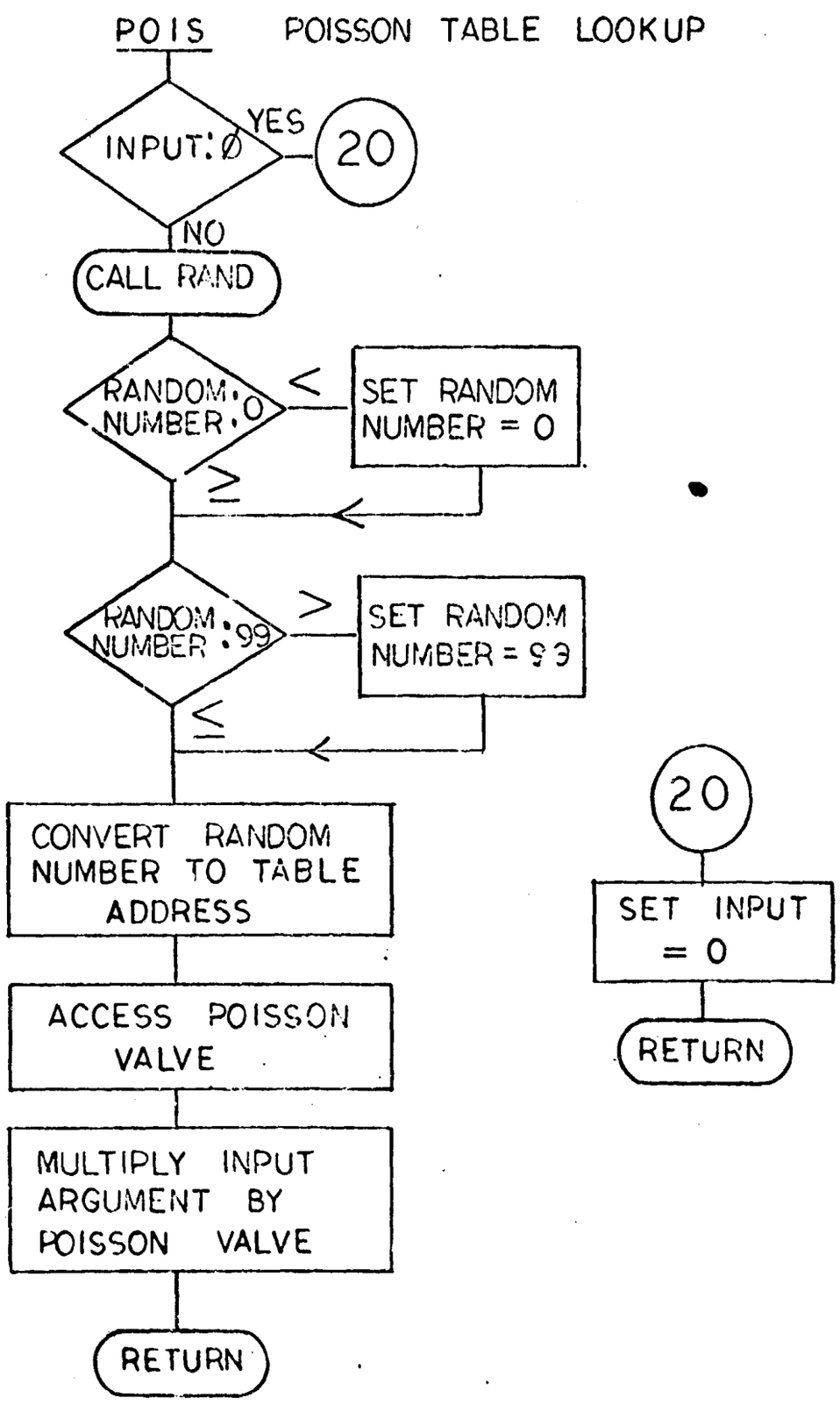

STAT
ZERO W/R STATISTICS TABLE
WRITE 9'S PADDING RECORD
SET PAGE SWITCH EQUAL TO 1
100
TEST TRANSACTION TERMINATION SWITCH ON
NO
130
YES
PAGE SWITCH : 0
<
110
≥
WRITE TITLE NEW PAGE
120
110
WRITE TITLE (NOT NEW PAGE)
120
WRITE LINE HEADING
A
TRANSATION ITEM ANALYSIS
A
130
READ RECORD
RECORD CODE : 1
≠
1150
CONVERT ALL TIMES TO SECTIONS
WRITE LINE = 1
WRITE LINE HEADING
WRITE I/O OPERATIONS
N1 = W/R NUMBER
CALCULATE TURNAROUND TIME
ENTER TIMES INTO W/R ANALYSIS TABLE
ENTER I/O OPERATIONS INTO W/R ANALYSIS TABLE
ADD 1 TO TRANSACTION TERMINATION COUNTER
B

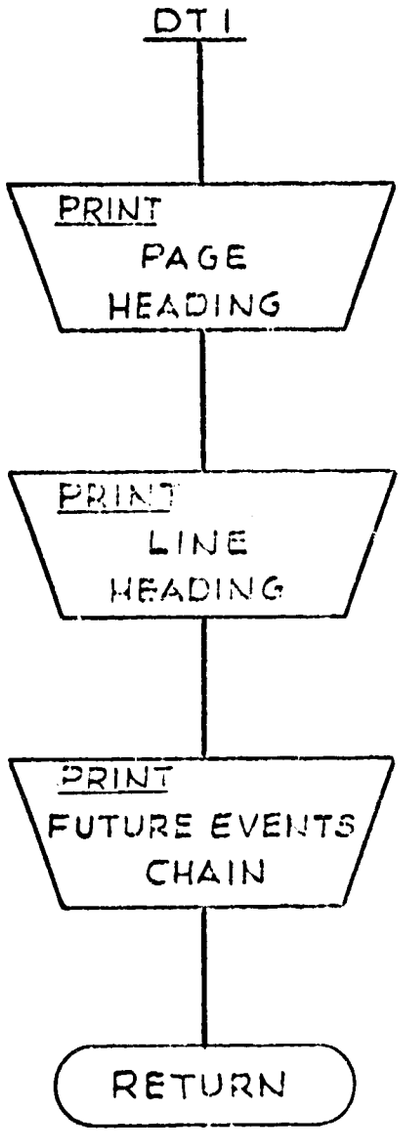

B
SET
TRANSACTIONS GENERATED
SET
TRANSACTIONS CALLED
REVERSE SIGN OF PAGE SWITCH
100
250
SET W/R POINTER TO 1
D
TEST FOR A W/R T I TERMINATION
NO
370
YES
CALCULATE AVERAGE TIME
CONVERT TIME TO SECONDS
CALCULATE AVERAGE I/O OPERATIONS
C
C
WRITE W/R STATISTICS PAGE HEADING
WRITE WORKER ROUTINE STATISTICS
W/R POINTER
≤
D
>
WRITE END OF SIMULATION
CALL EXIT
1150
RECORD CODE
≠
250
≠
130

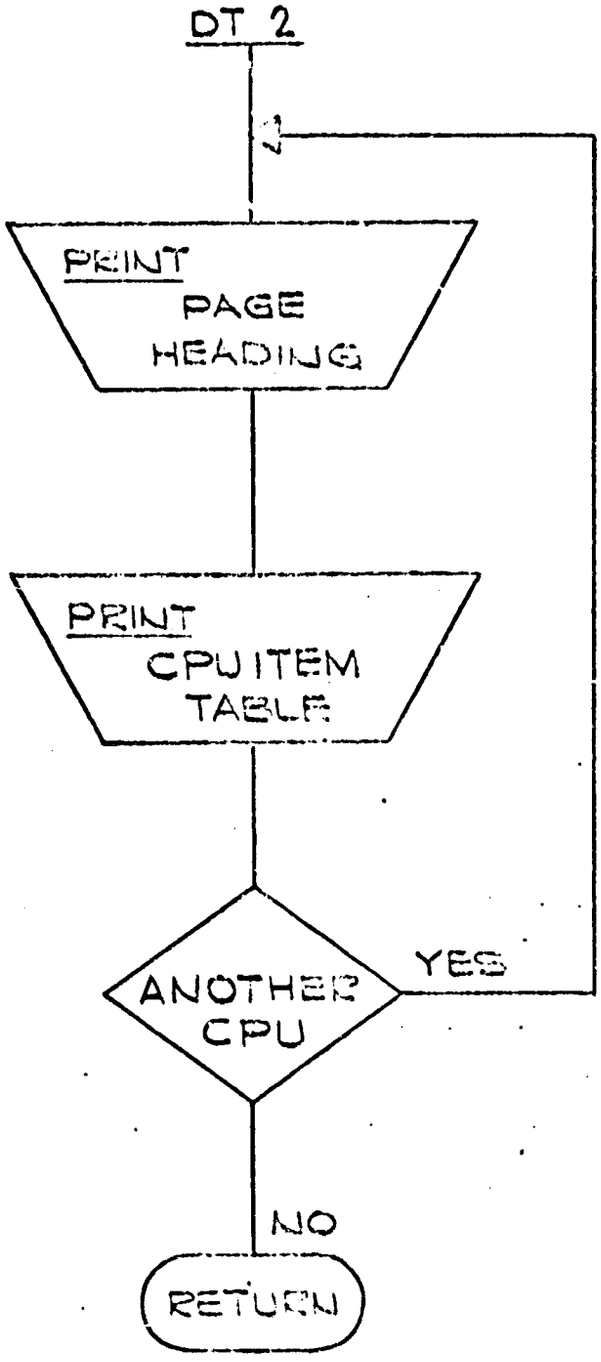

T DUMP
CALL DT1
CALL DT2
CALL DT3
CALL DT4
CALL DT5
CALL DT6
CALL DT7
CALL DT8
CALL DT9
CALL DT12
A
A
CALL DT13
CALL DT14
CALL DT15
CALL DT16
CALL DT17
CALL DT18
CALL DT19
CALL DT20
CALL DT21
CALL DT22
CALL DT23
CALL DT24
CALL DT25
RETURN

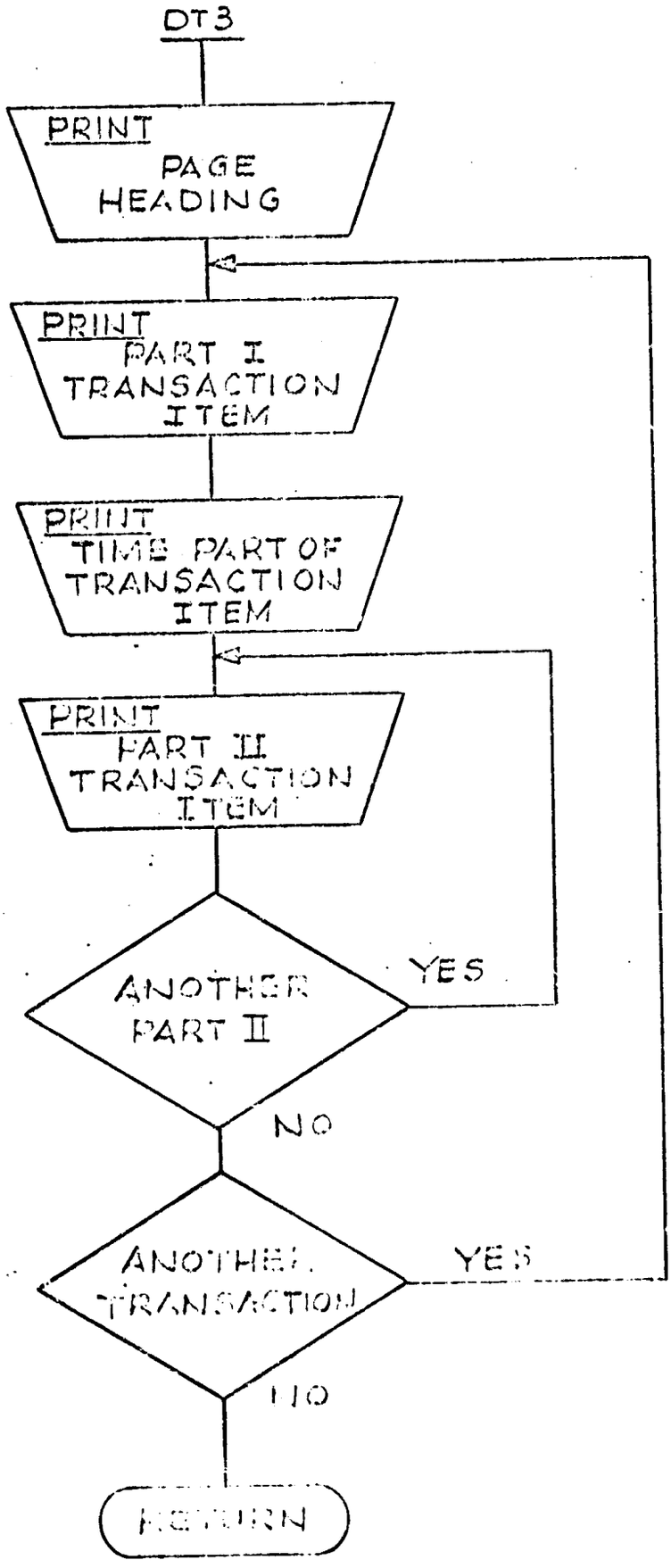

SNAP
SNAP SWITCH
OFF
RETURN
PRINT LINE HEADING
PRINT LINE
ALL LINES PRINTED
NO
YES
RETURN

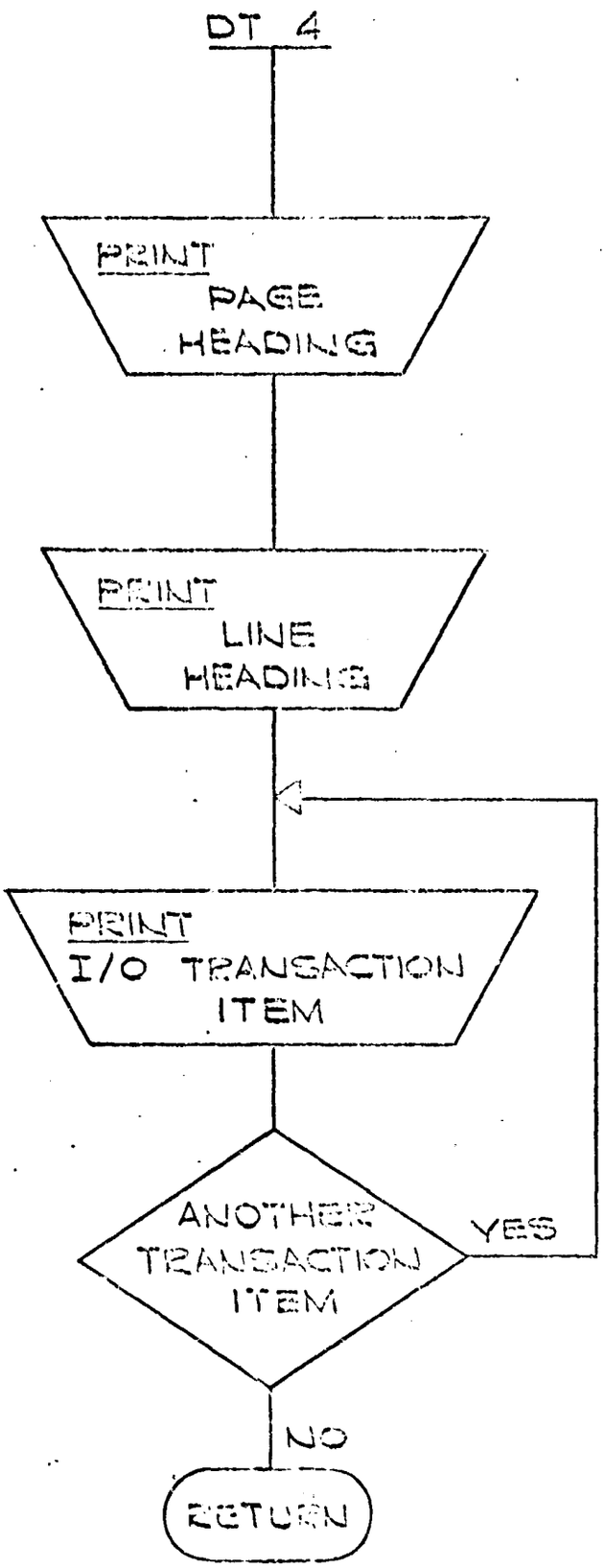

TRACE - S951
SNAP SWITCH
OFF
RETURN
ON
PRINT TRACE LINE
RETURN

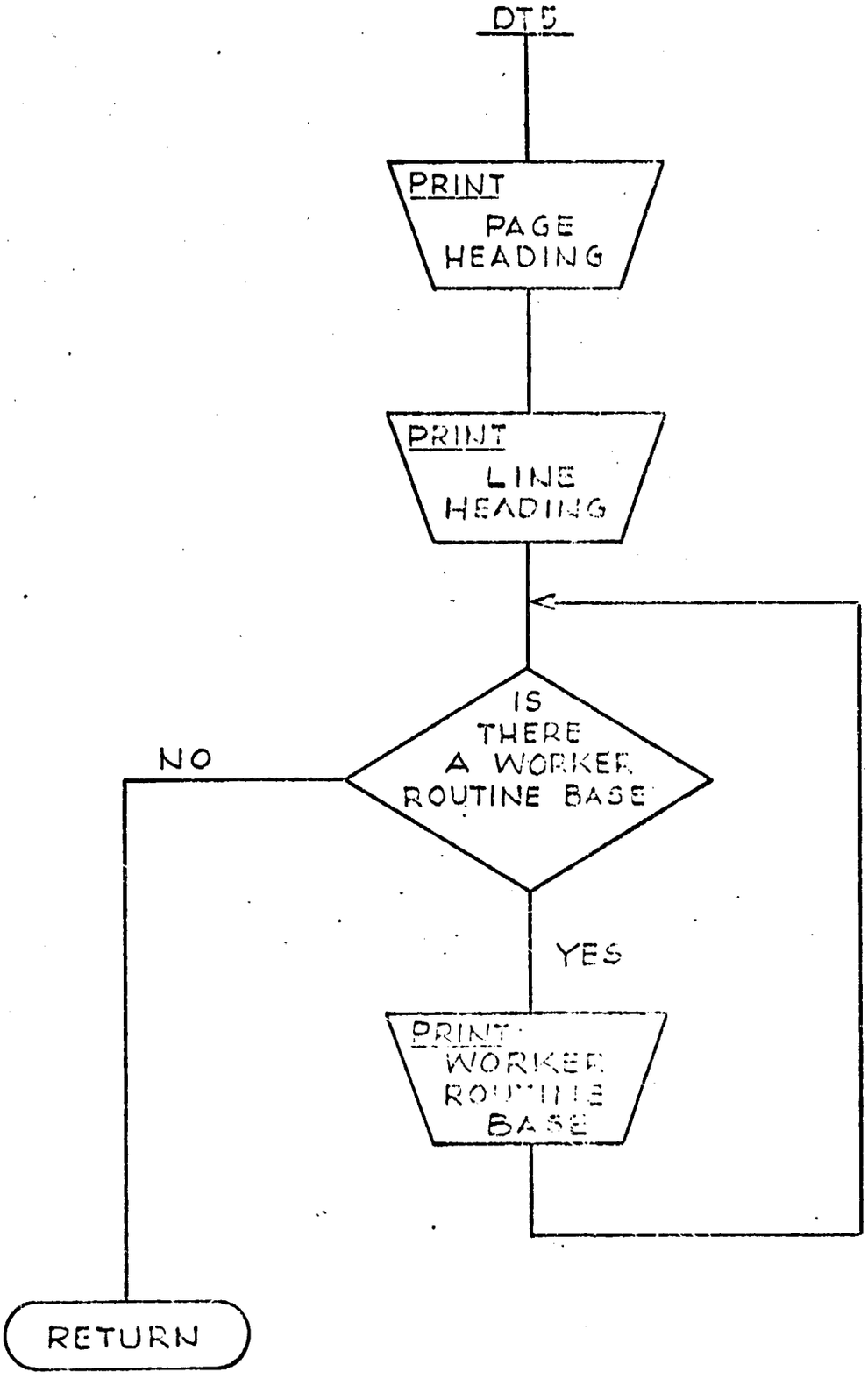

POIS
POISSON TABLE LOOKUP
INPUT: Ø
YES
20
NO
CALL RAND
RANDOM NUMBER: 0
<
SET RANDOM NUMBER = 0
≥
RANDOM NUMBER: 99
>
SET RANDOM NUMBER = 99
≤
CONVERT RANDOM NUMBER TO TABLE ADDRESS
ACCESS POISSON VALVE
MULTIPLY INPUT ARGUMENT BY POISSON VALVE
RETURN
20
SET INPUT = 0
RETURN

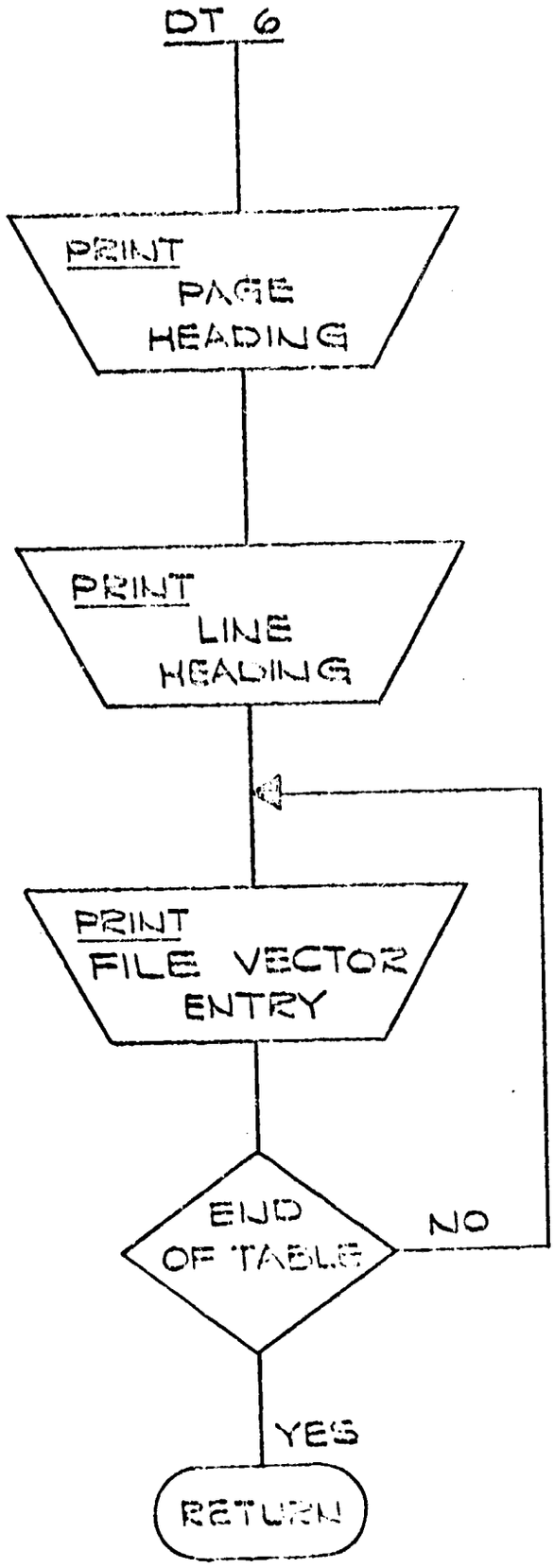

DTI
PRINT
PAGE
HEADING
PRINT
LINE
HEADING
PRINT
FUTURE EVENTS
CHAIN
RETURN

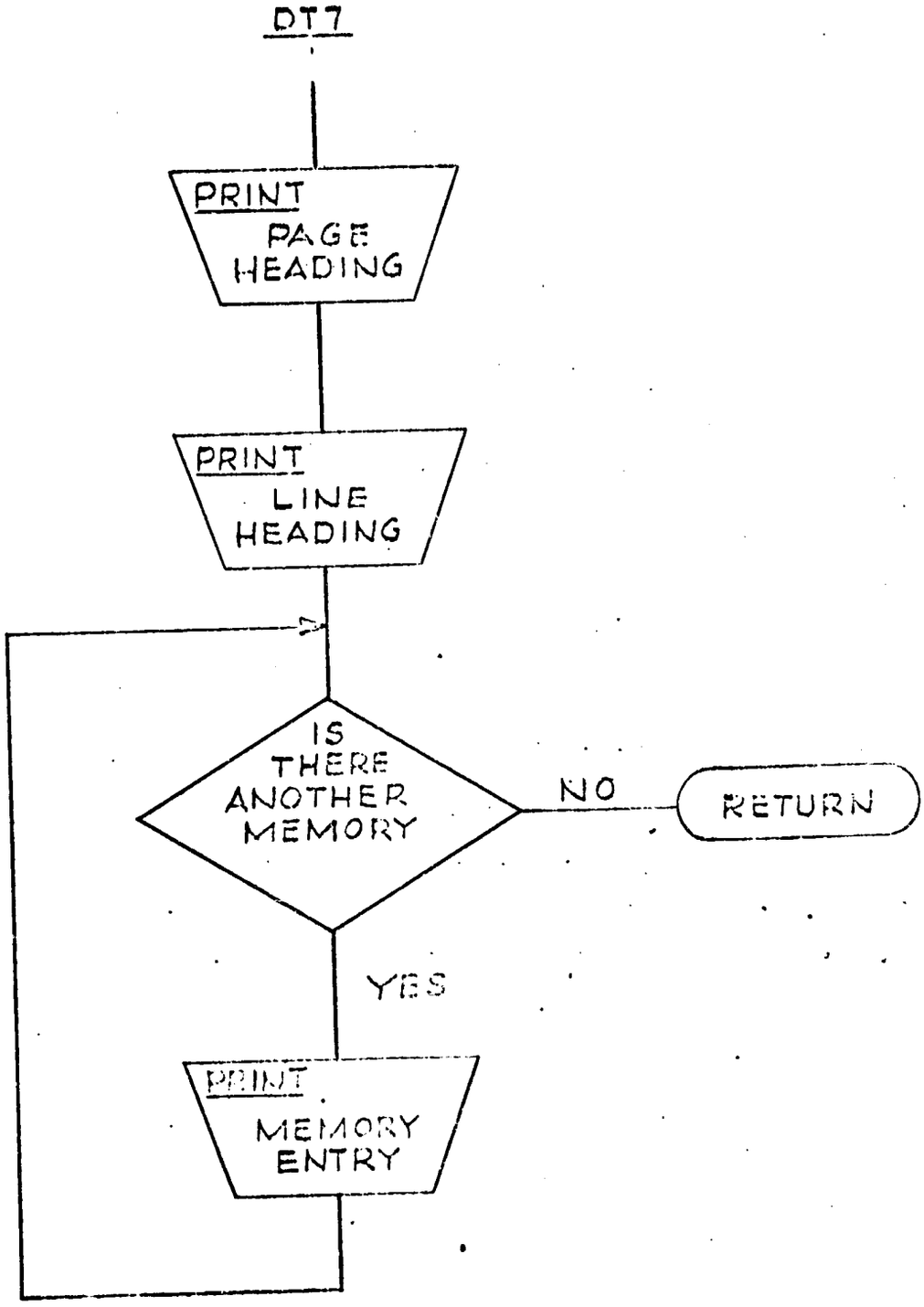

DT 2
PRINT
PAGE
HEADING
PRINT
CPU ITEM
TABLE
ANOTHER
CPU
YES
NO
RETURN

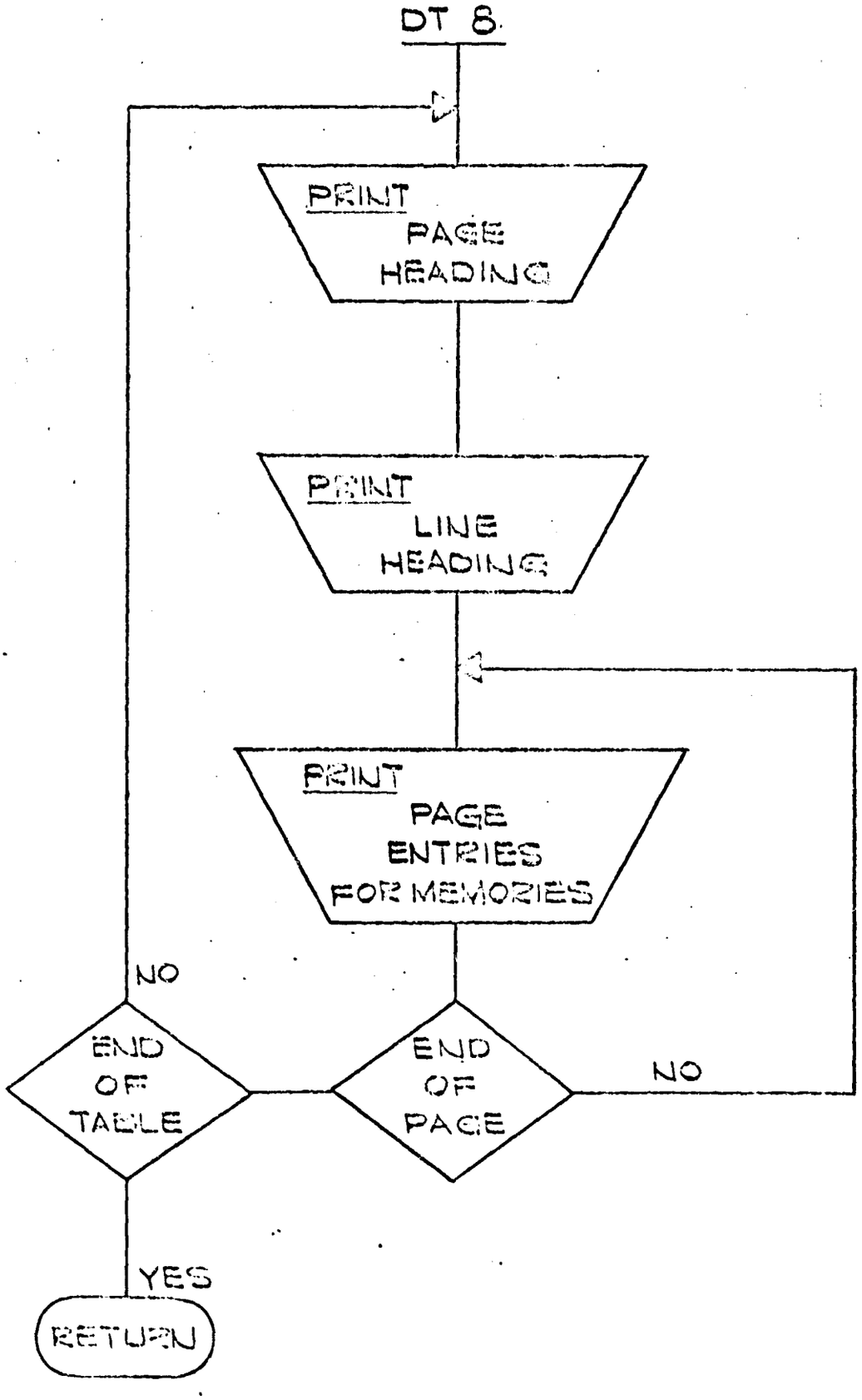

DT3
PRINT
PAGE
HEADING
PRINT
PART I
TRANSACTION
ITEM
PRINT
TIME PART OF
TRANSACTION
ITEM
PRINT
PART II
TRANSACTION
ITEM
ANOTHER
PART II
YES
NO
ANOTHER
TRANSACTION
YES
NO
RETURN

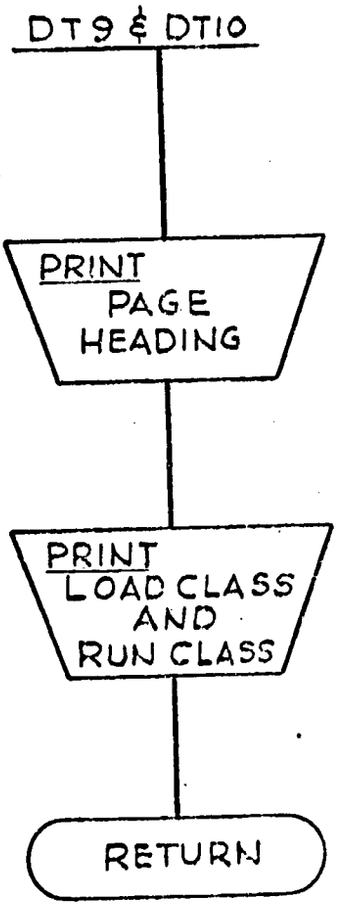

DT 4
PRINT
PAGE
HEADING
PRINT
LINE
HEADING
PRINT
I/O TRANSACTION
ITEM
ANOTHER
TRANSACTION
ITEM
YES
NO
RETURN

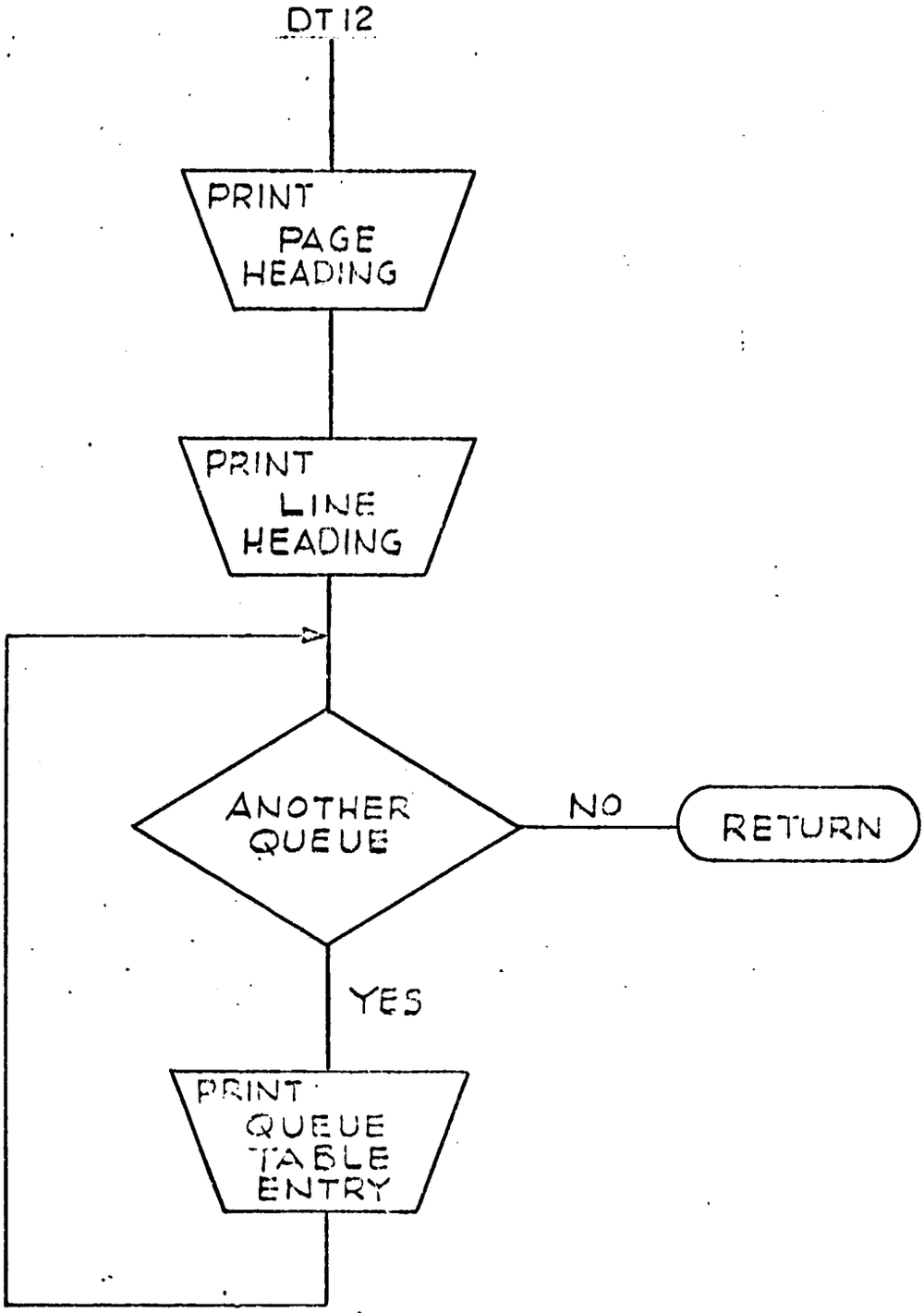

DT5
PRINT
PAGE
HEADING
PRINT
LINE
HEADING
IS
THERE
A WORKER
ROUTINE BASE
NO
YES
PRINT
WORKER
ROUTINE
BASE
RETURN

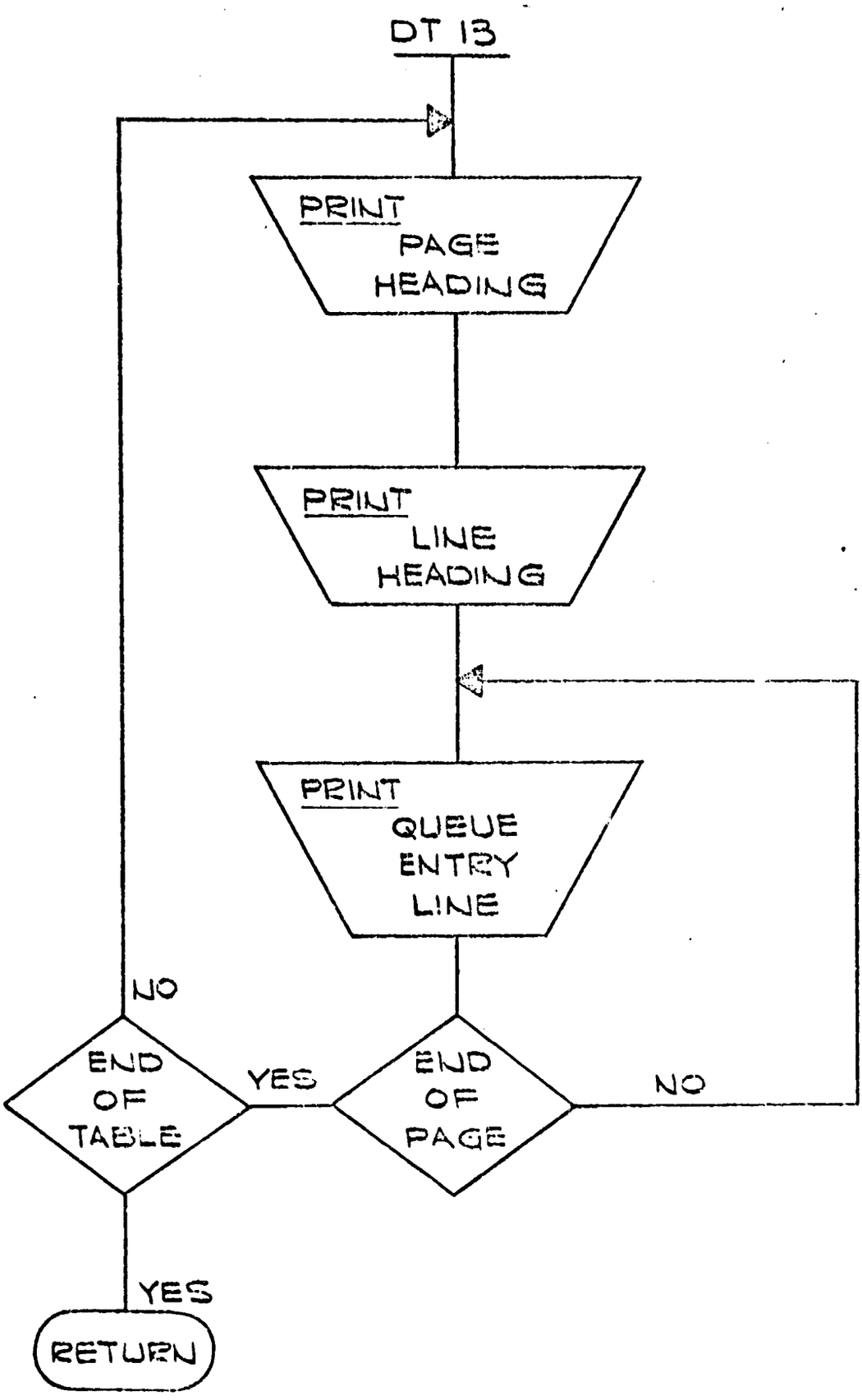

DT 6
PRINT
PAGE
HEADING
PRINT
LINE
HEADING
PRINT
FILE VECTOR
ENTRY
END
OF TABLE
NO
YES
RETURN

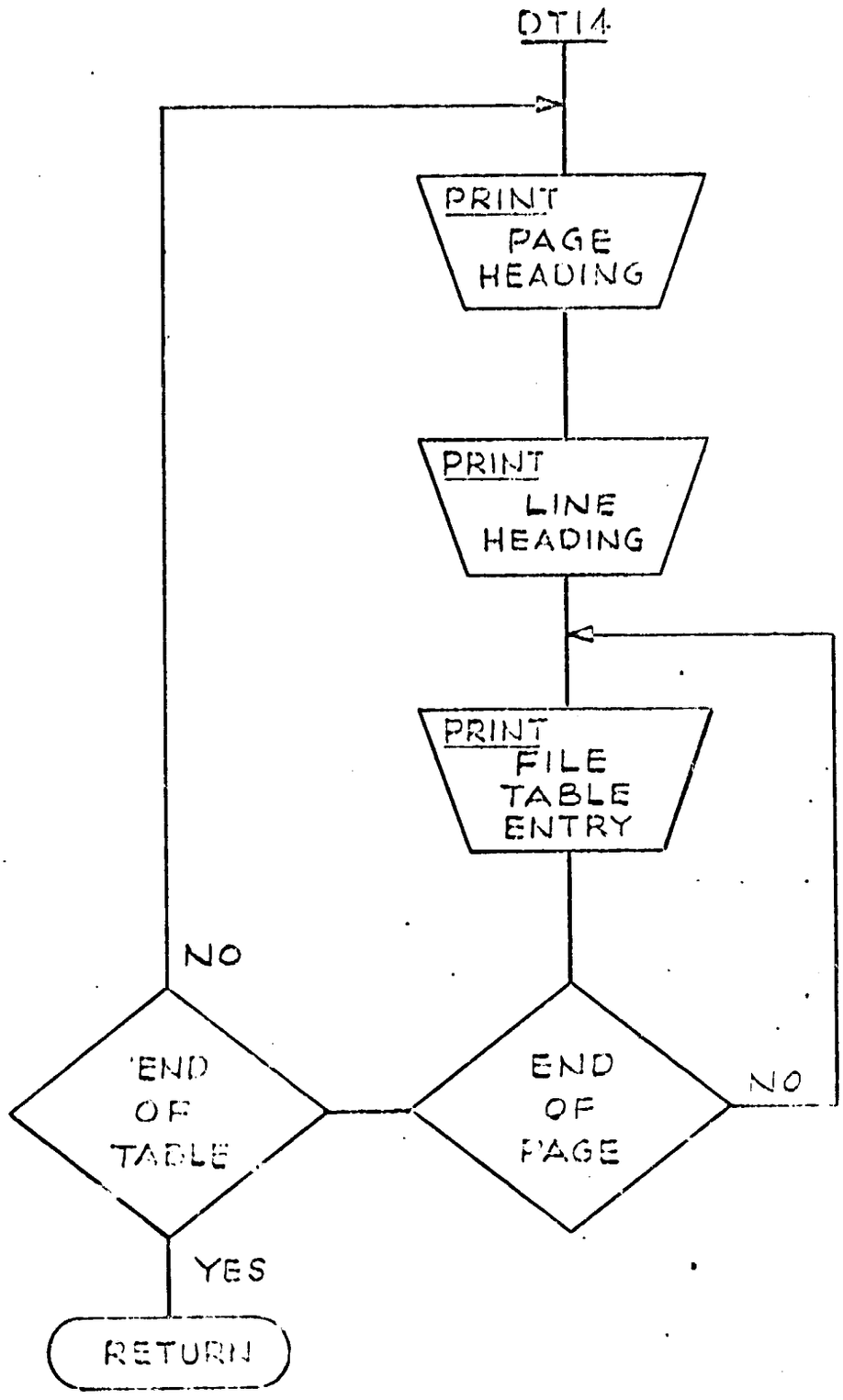

DT7
PRINT
PAGE
HEADING
PRINT
LINE
HEADING
IS
THERE
ANOTHER
MEMORY
NO
RETURN
YES
PRINT
MEMORY
ENTRY

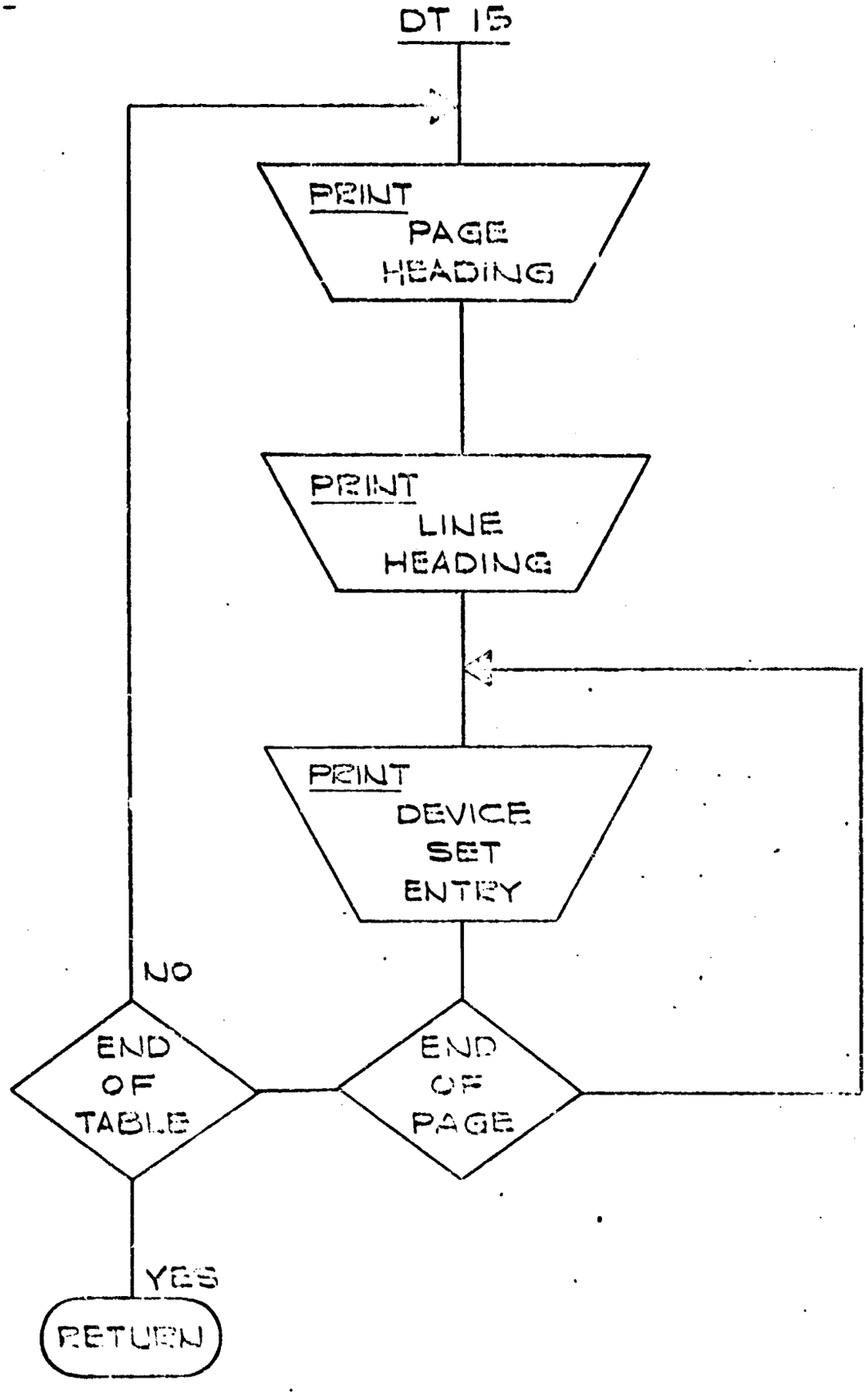

DT 8
PRINT
PAGE
HEADING
PRINT
LINE
HEADING
PRINT
PAGE
ENTRIES
FOR MEMORIES
END
OF
PAGE
NO
END
OF
TABLE
NO
YES
RETURN

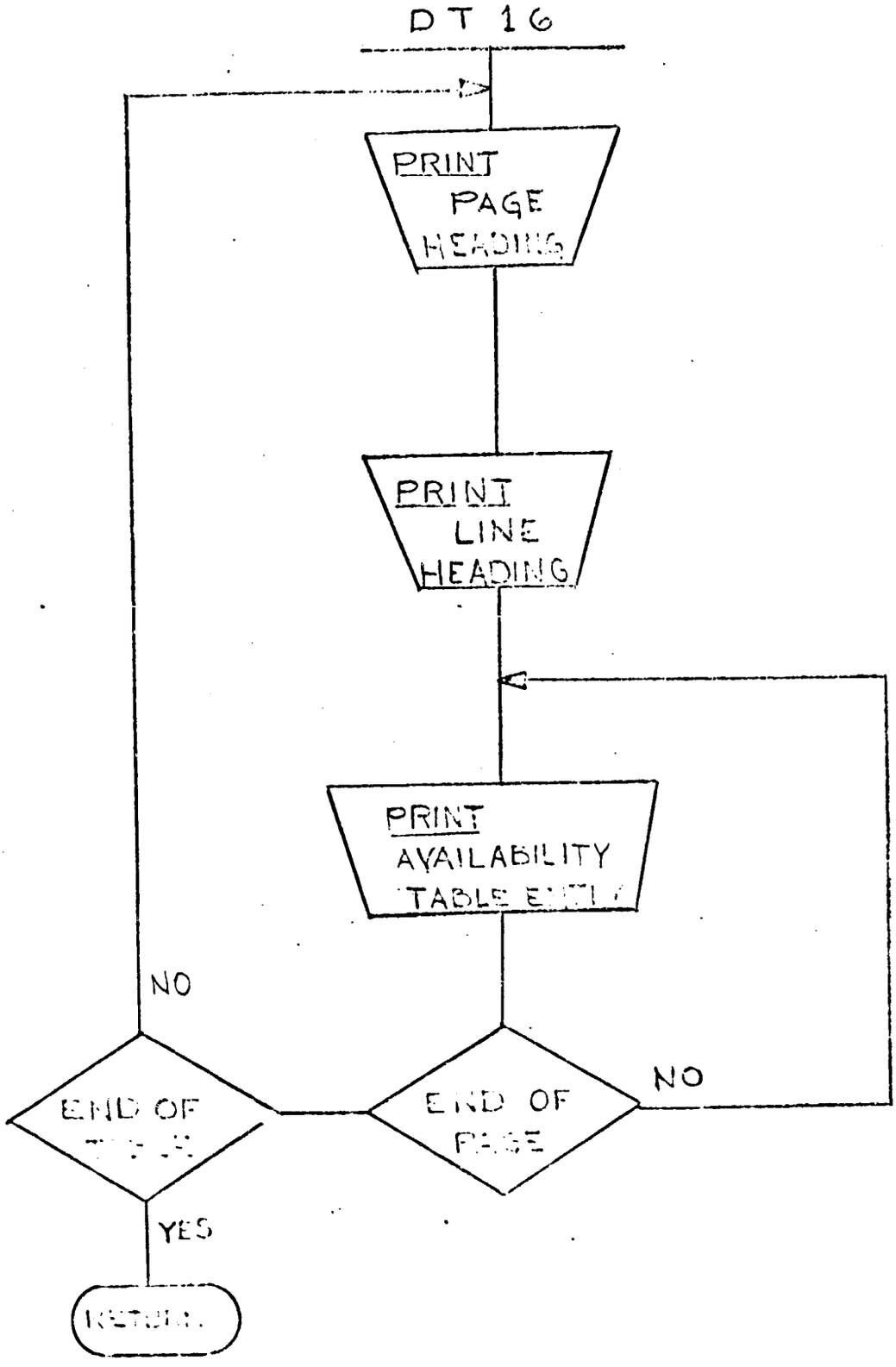

DT9 & DT10
PRINT PAGE HEADING
PRINT LOAD CLASS AND RUN CLASS
RETURN

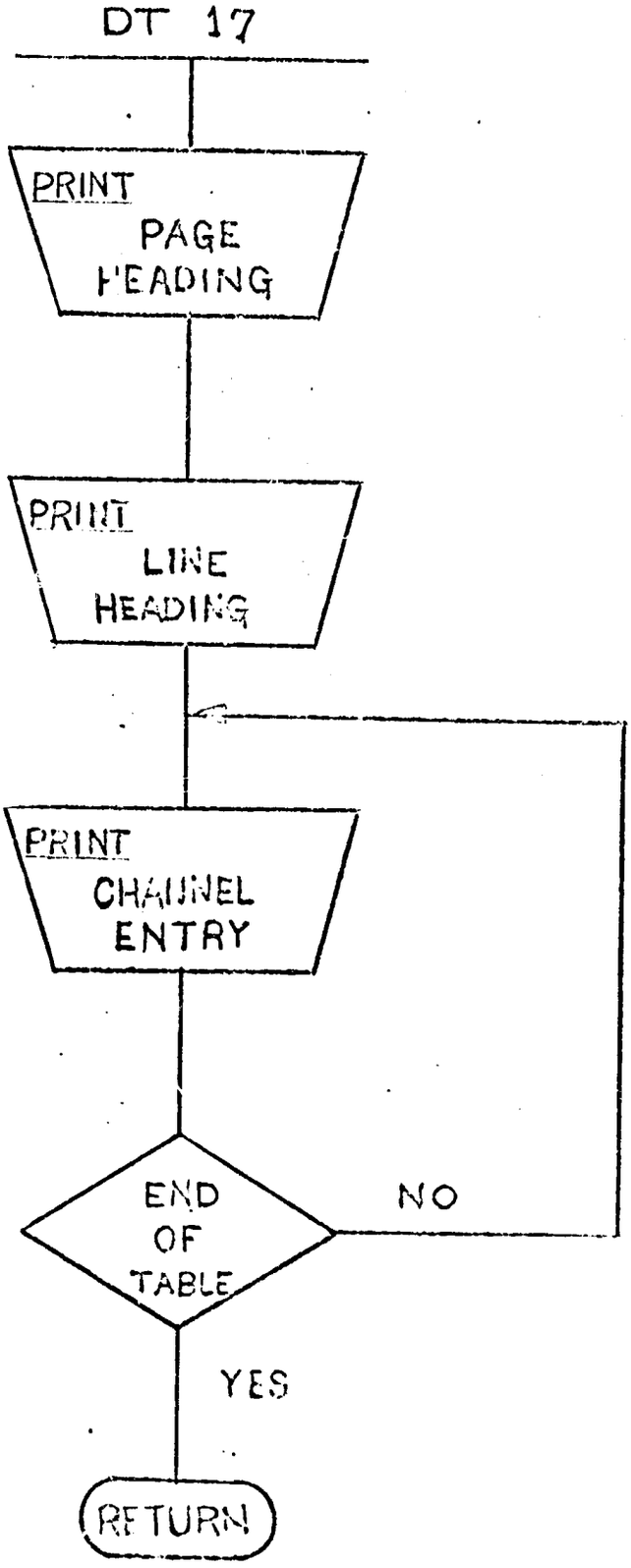

DT12
PRINT PAGE HEADING
PRINT LINE HEADING
ANOTHER QUEUE
NO
RETURN
YES
PRINT QUEUE TABLE ENTRY

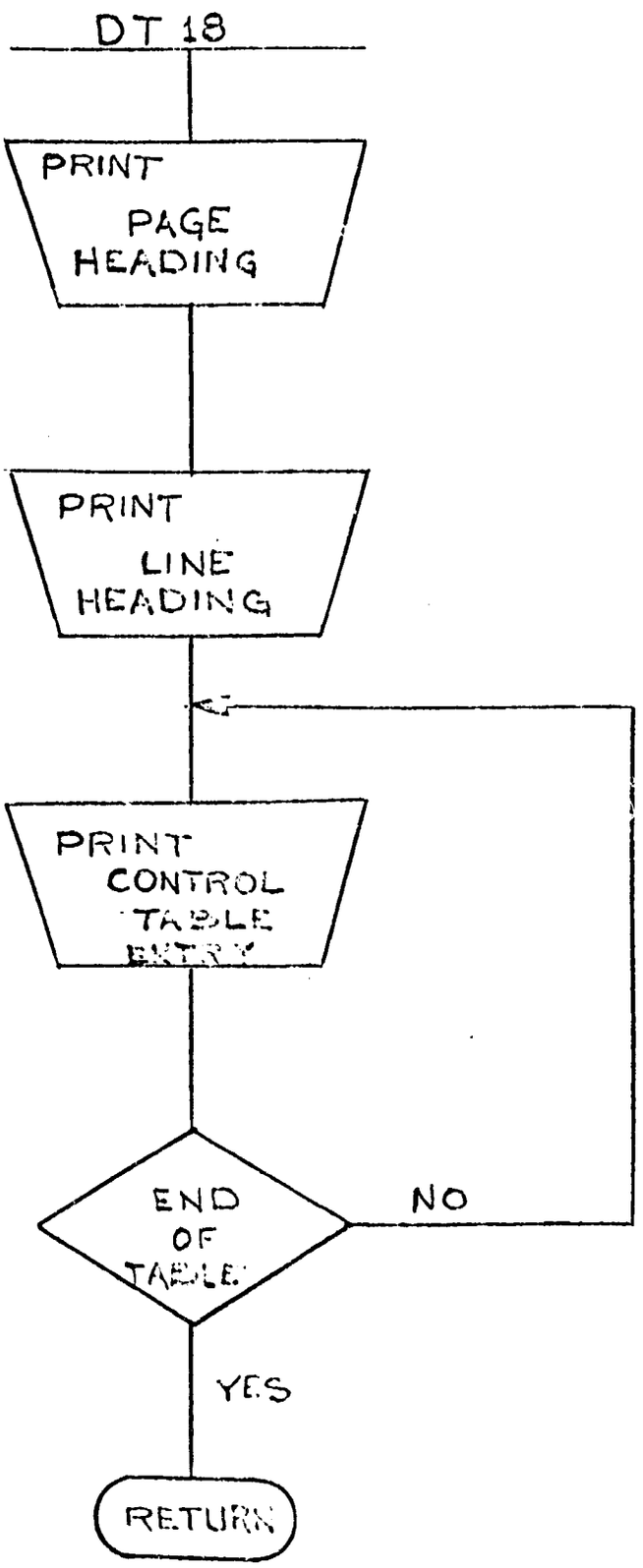

DT 13
PRINT PAGE HEADING
PRINT LINE HEADING
PRINT QUEUE ENTRY LINE
END OF PAGE
NO
YES
END OF TABLE
NO
YES
RETURN

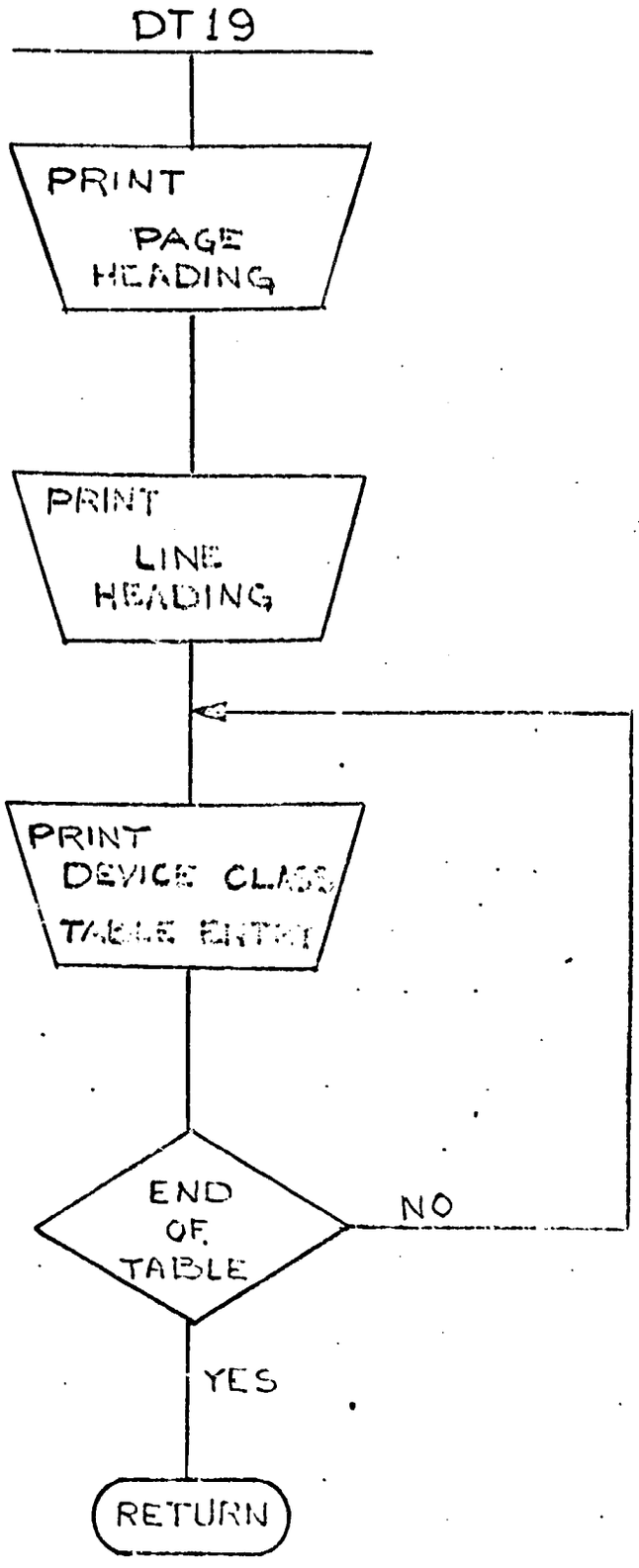

DT14
PRINT
PAGE
HEADING
PRINT
LINE
HEADING
PRINT
FILE
TABLE
ENTRY
END
OF
PAGE
NO
END
OF
TABLE
NO
YES
RETURN

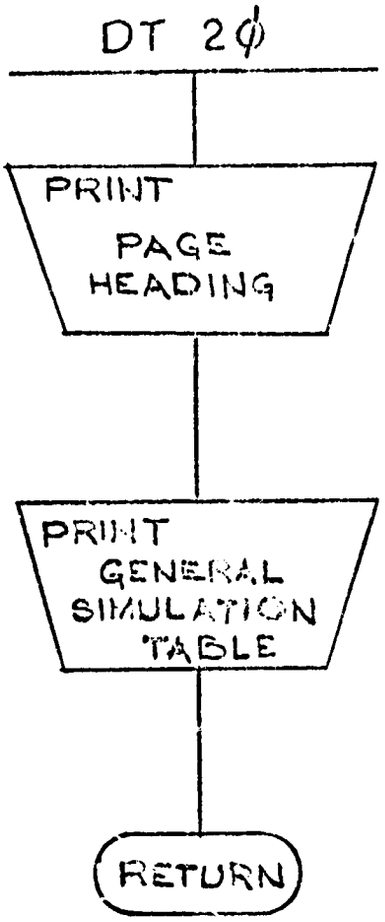

DT 15
PRINT
PAGE
HEADING
PRINT
LINE
HEADING
PRINT
DEVICE
SET
ENTRY
END
OF
PAGE
NO
END
OF
TABLE
YES
RETURN

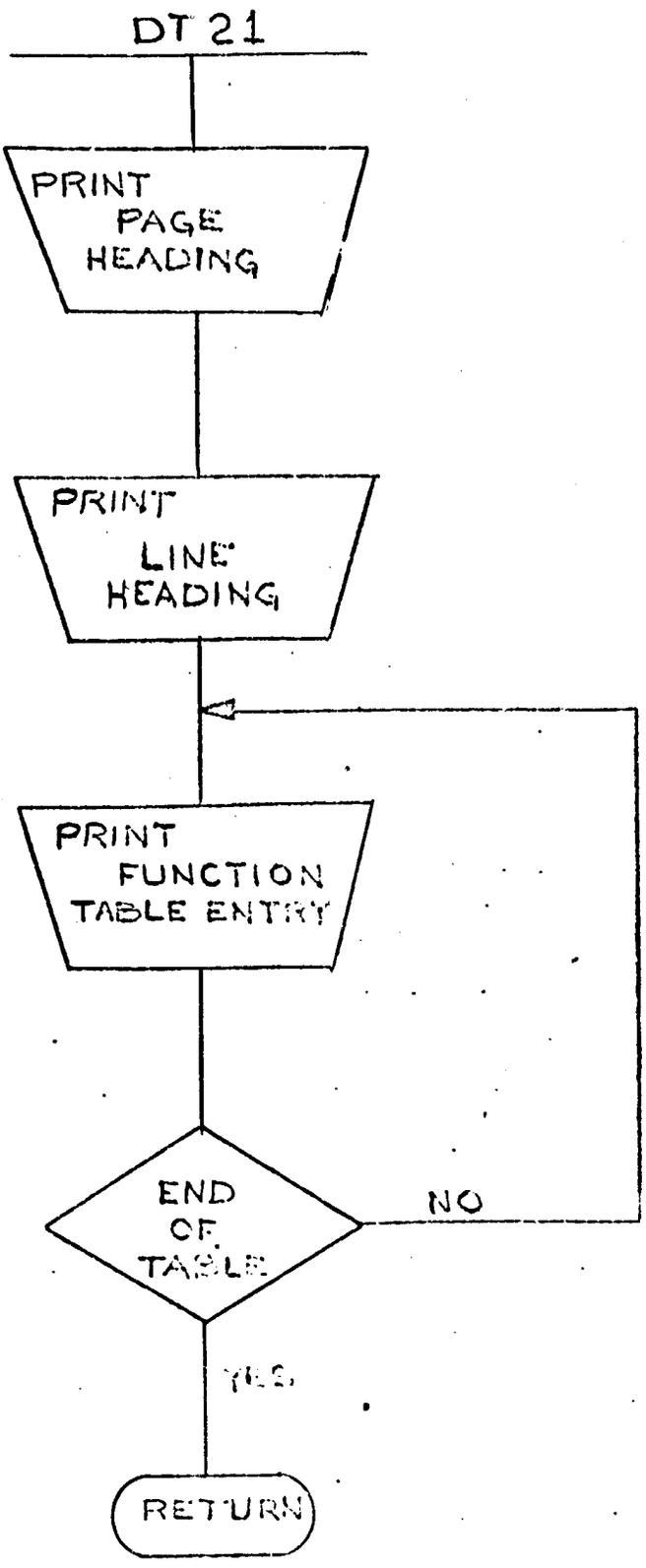

DT 16
PRINT PAGE HEADING
PRINT LINE HEADING
PRINT AVAILABILITY
NO
END OF
END OF
NO
YES

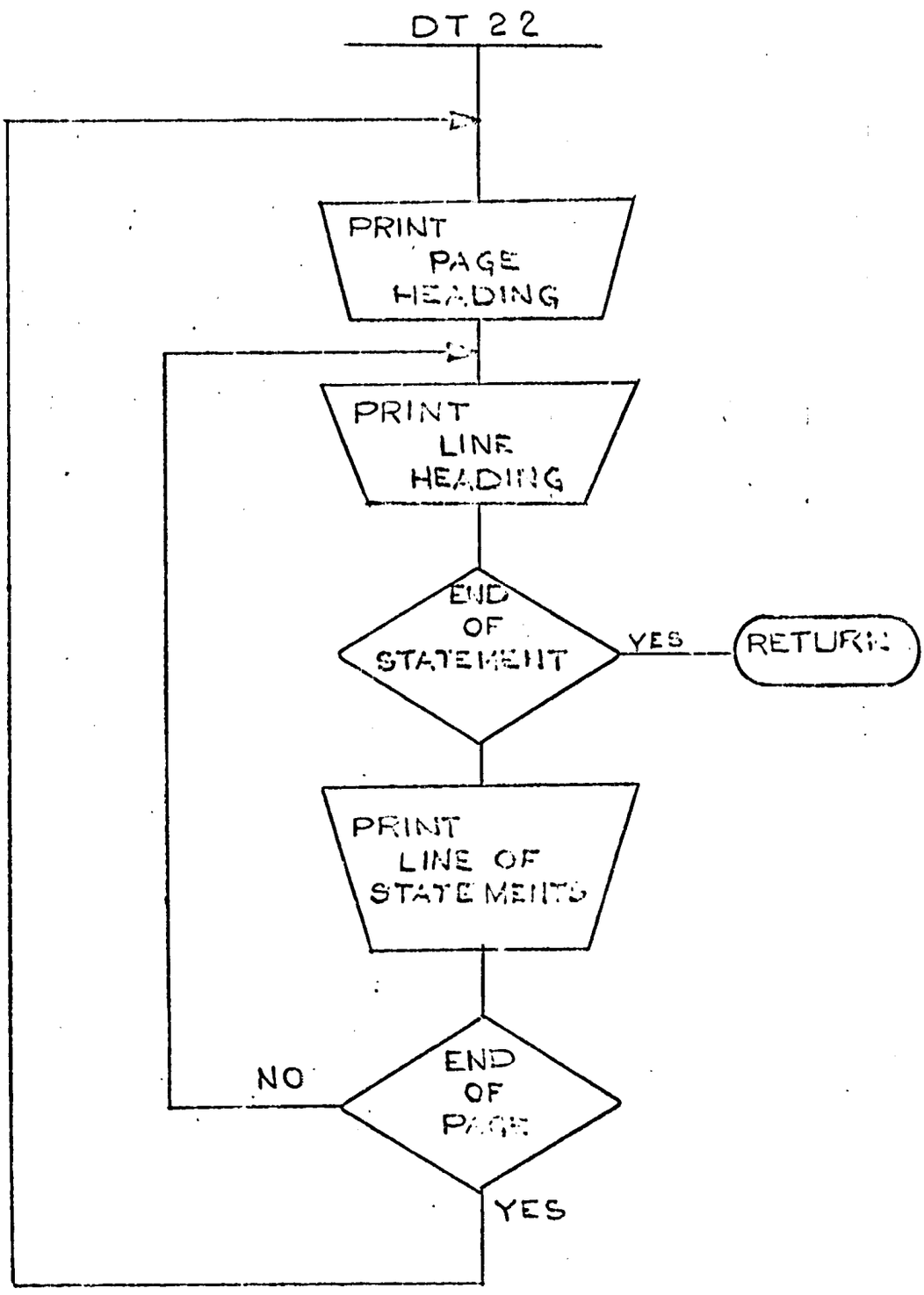

DT 17
PRINT
PAGE
HEADING
PRINT
LINE
HEADING
PRINT
CHANNEL
ENTRY
END
OF
TABLE
NO
YES
RETURN

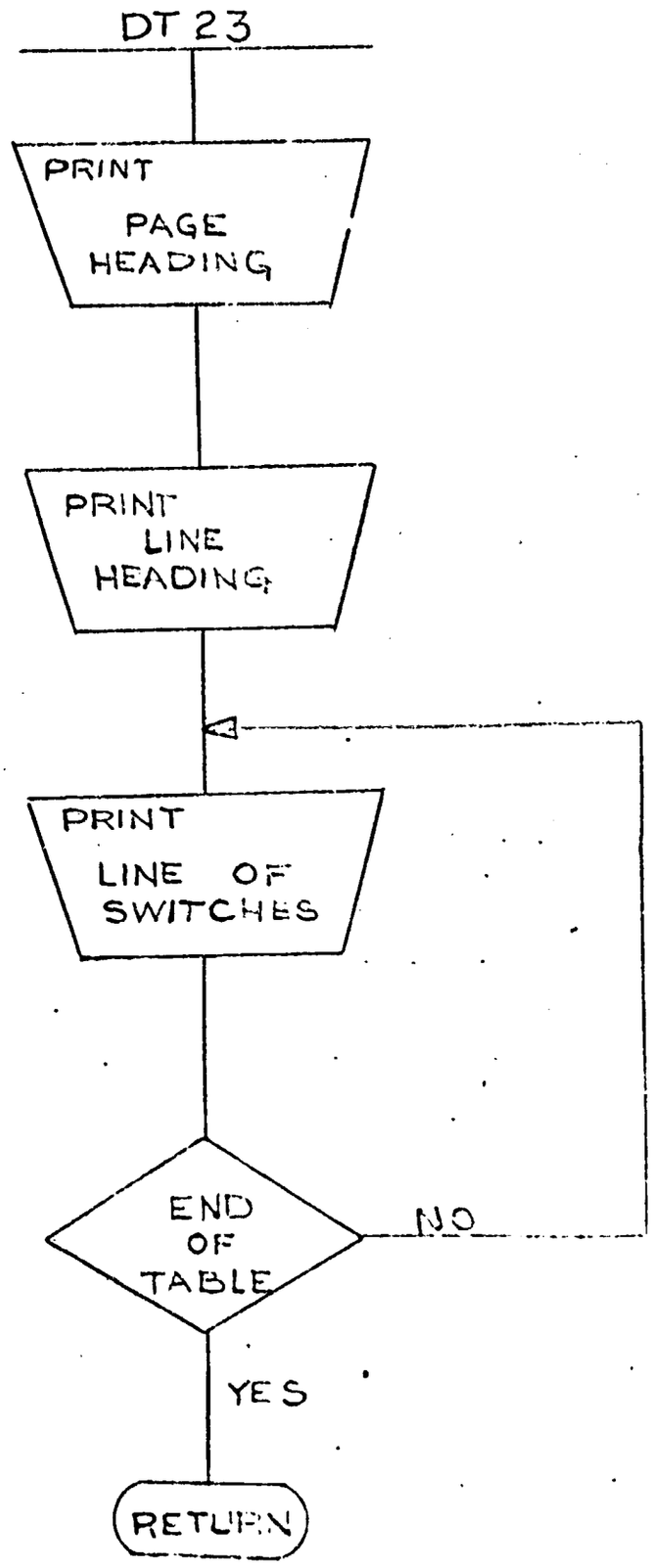

DT 18
PRINT
PAGE
HEADING
PRINT
LINE
HEADING
PRINT
CONTROL
TABLE
ENTRY
END
OF
TABLE
NO
YES
RETURN

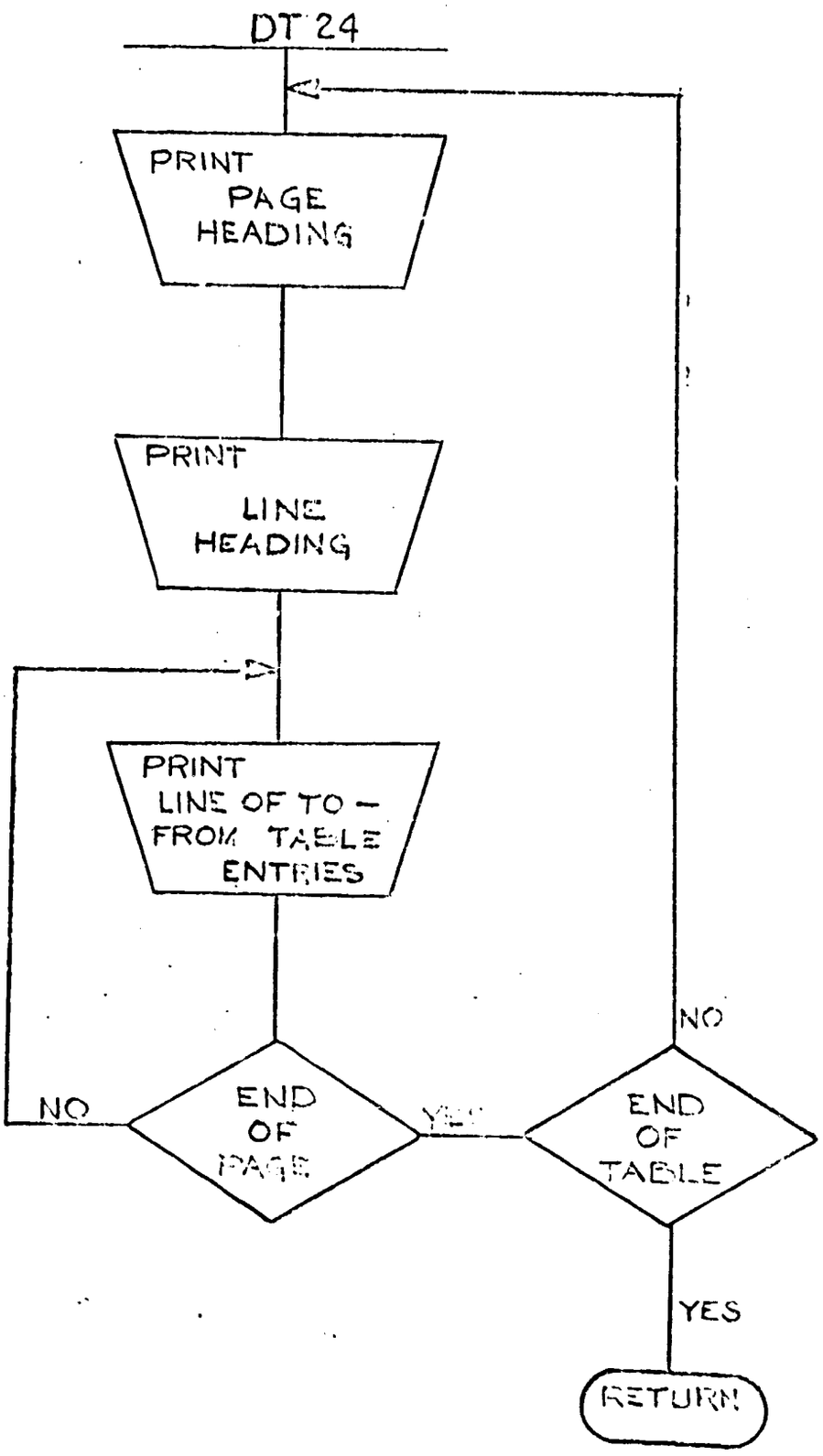

DT19
PRINT
PAGE
HEADING
PRINT
LINE
HEADING
PRINT
DEVICE CLASS
TABLE ENTRY
END
OF
TABLE
NO
YES
RETURN

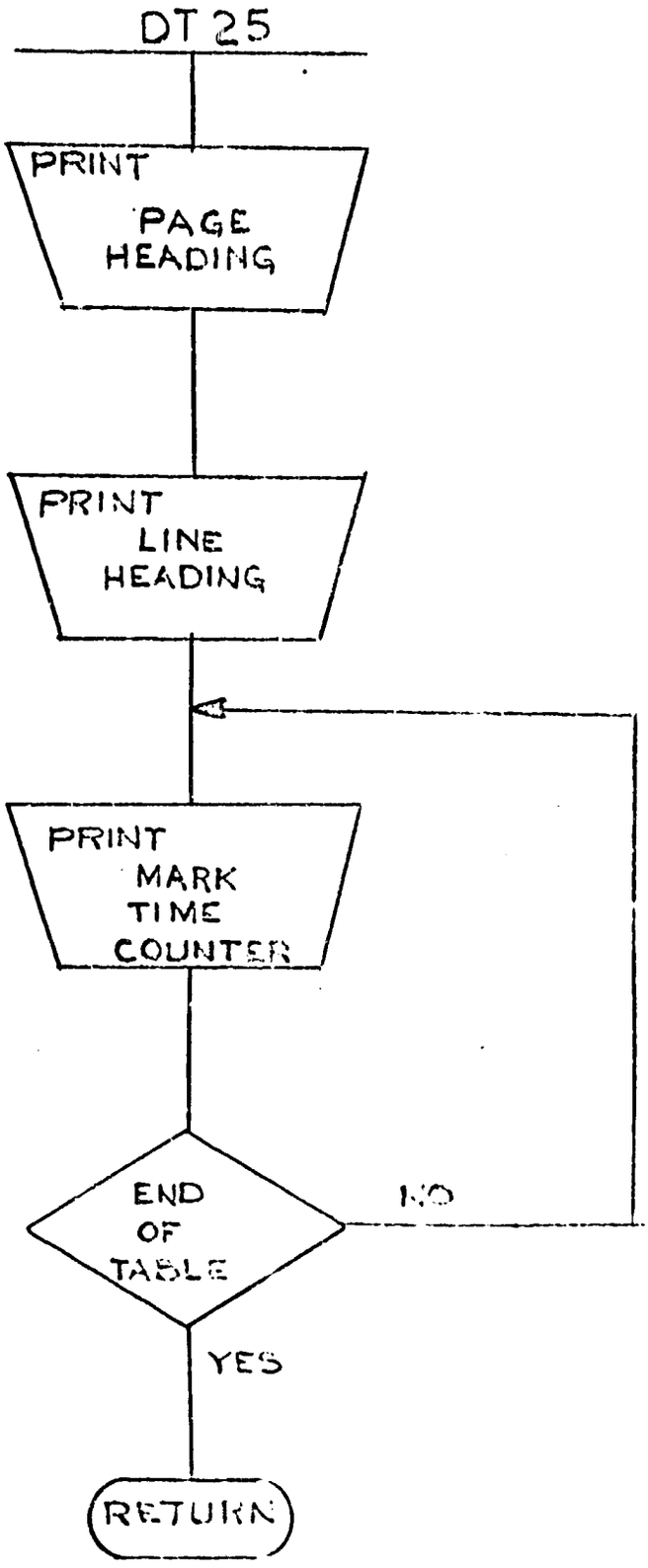

DT 2ϕ
PRINT
PAGE
HEADING
PRINT
GENERAL
SIMULATION
TABLE
RETURN

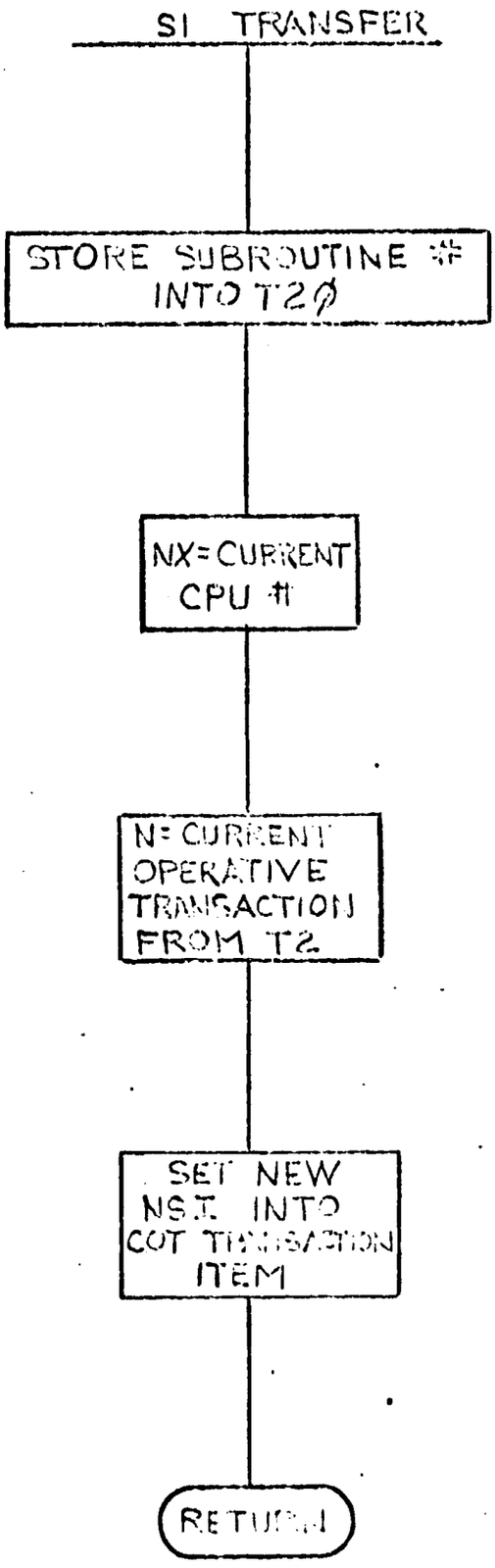

DT 21
PRINT PAGE HEADING
PRINT LINE HEADING
PRINT FUNCTION TABLE ENTRY
END OF TABLE
NO
YES
RETURN


DT 22
PRINT PAGE HEADING
PRINT LINE HEADING
END OF STATEMENT
YES
RETURN
PRINT LINE OF STATEMENTS
END OF PAGE
NO
YES

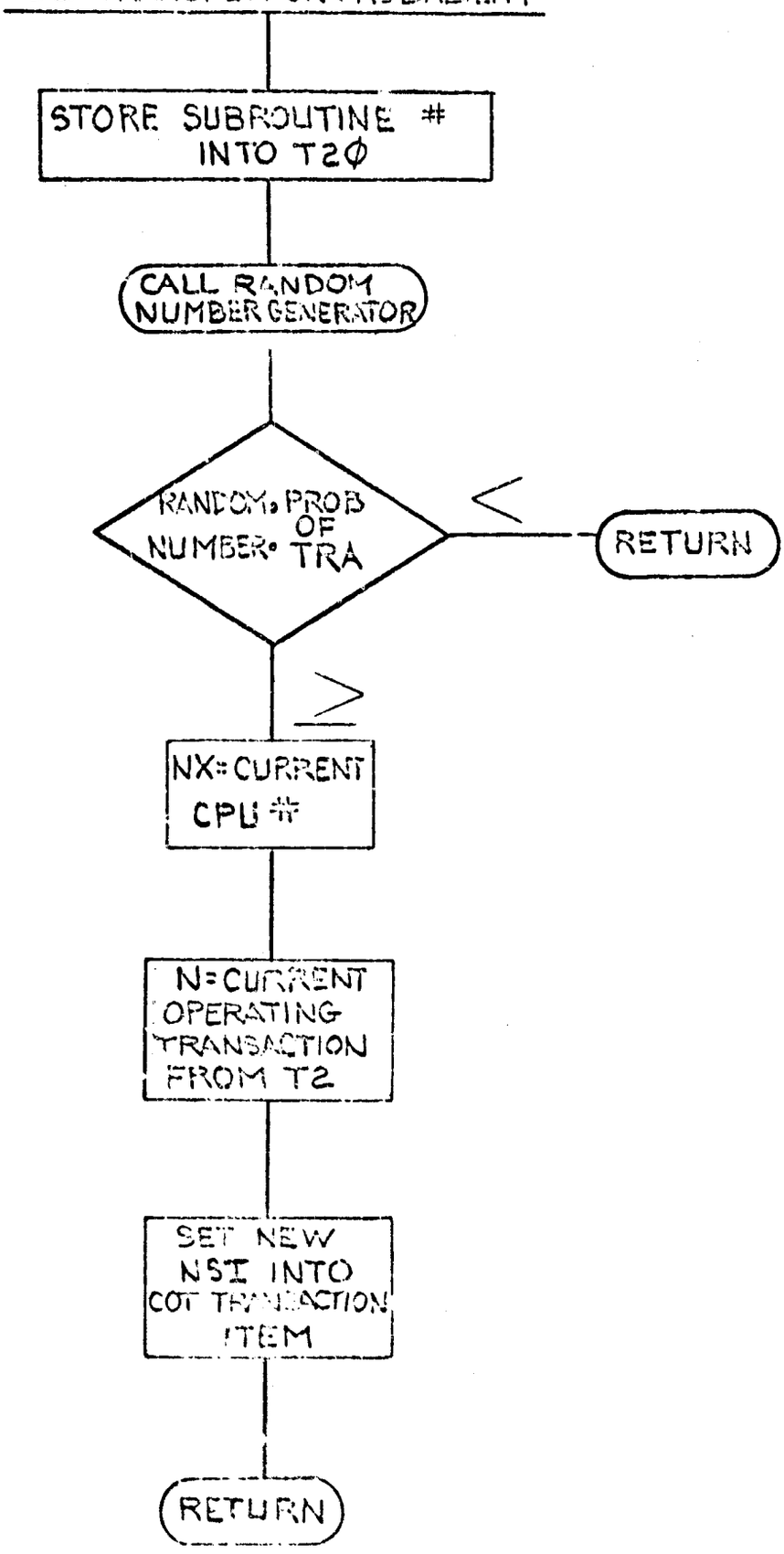

DT 23
PRINT
PAGE
HEADING
PRINT
LINE
HEADING
PRINT
LINE OF
SWITCHES
END
OF
TABLE
NO
YES
RETURN

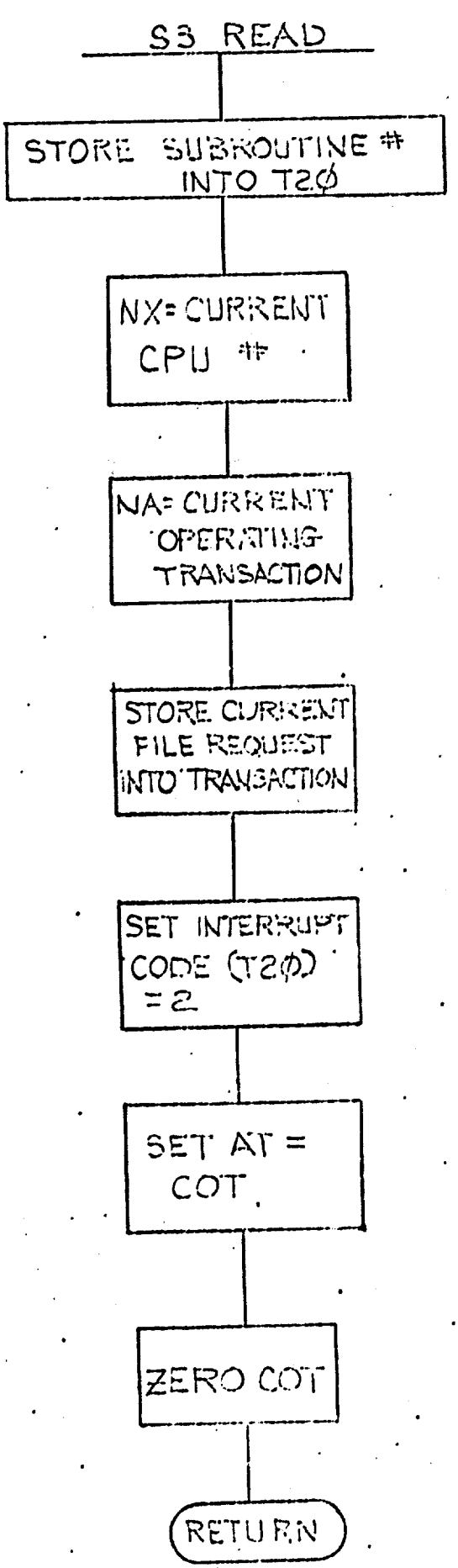

DT 24
PRINT PAGE HEADING
PRINT LINE HEADING
PRINT LINE OF TO – FROM TABLE ENTRIES
END OF PAGE
NO
YES
END OF TABLE
NO
YES
RETURN

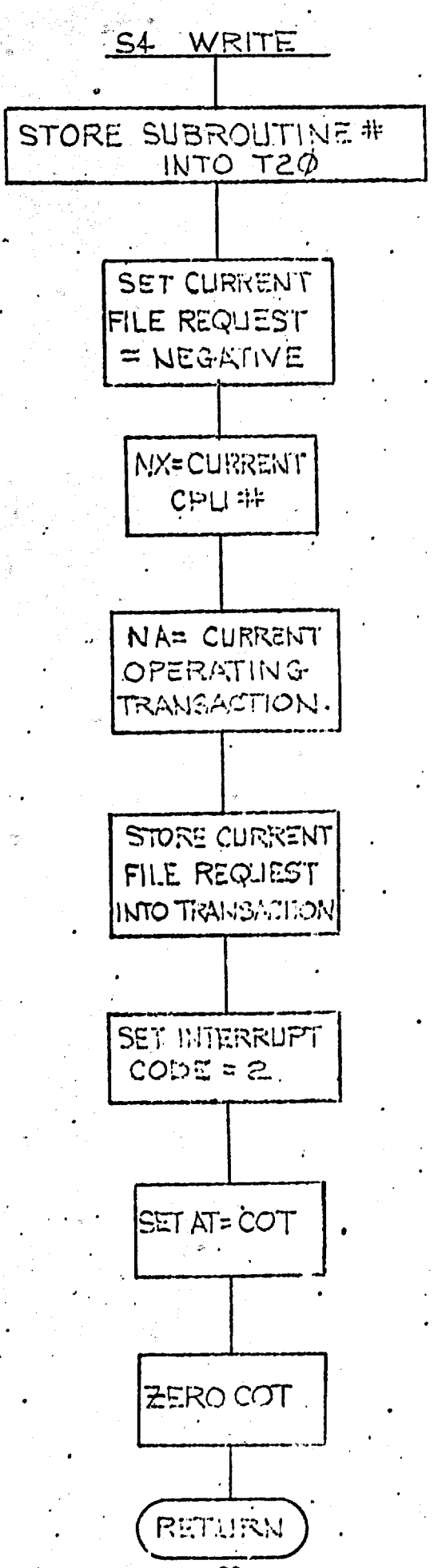

DT 25
PRINT
PAGE
HEADING
PRINT
LINE
HEADING
PRINT
MARK
TIME
COUNTER
END
OF
TABLE
NO
YES
RETURN

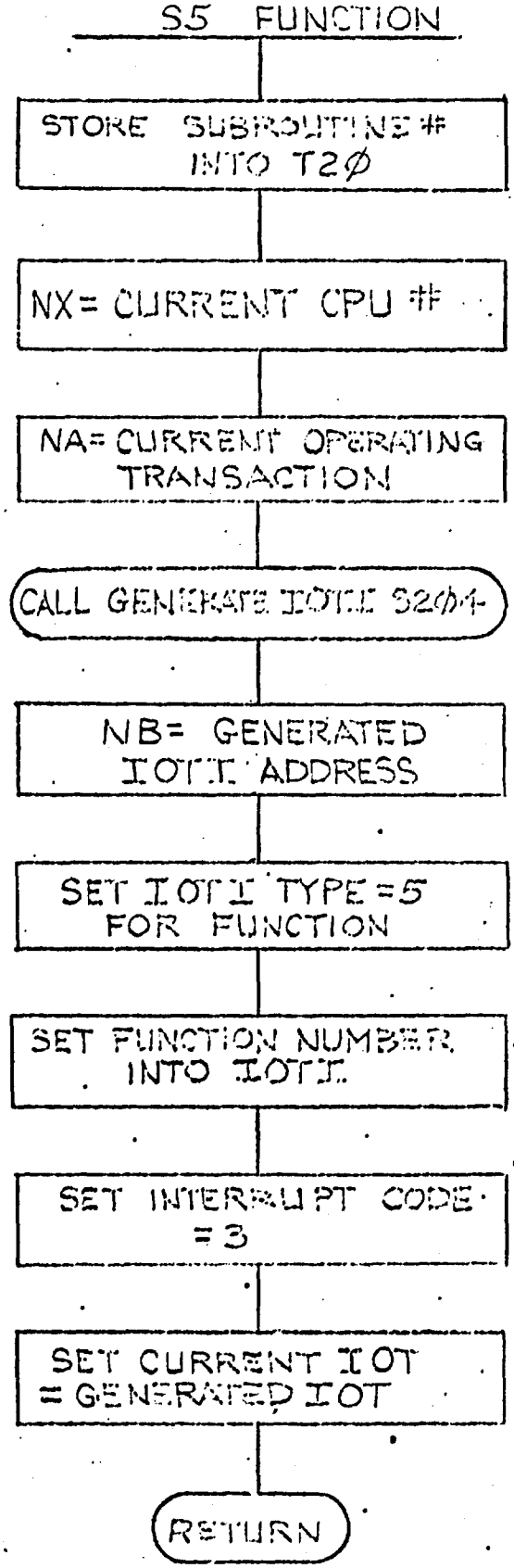

SI TRANSFER
STORE SUBROUTINE # INTO T2Ø
NX = CURRENT CPU #
N = CURRENT OPERATIVE TRANSACTION FROM T2
SET NEW NSI INTO COT TRANSACTION ITEM
RETURN

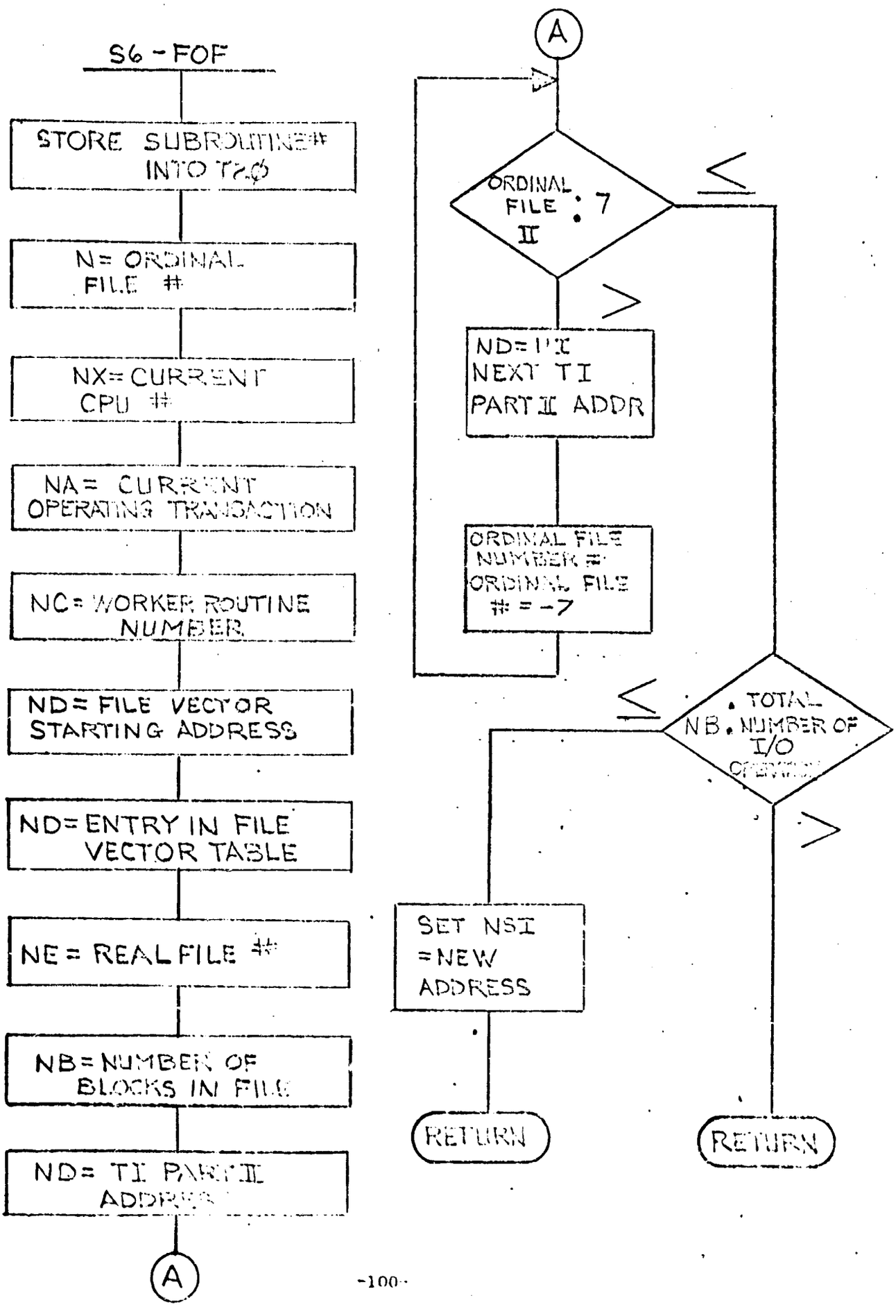
S2 TRANSFER ON PROBABILITY

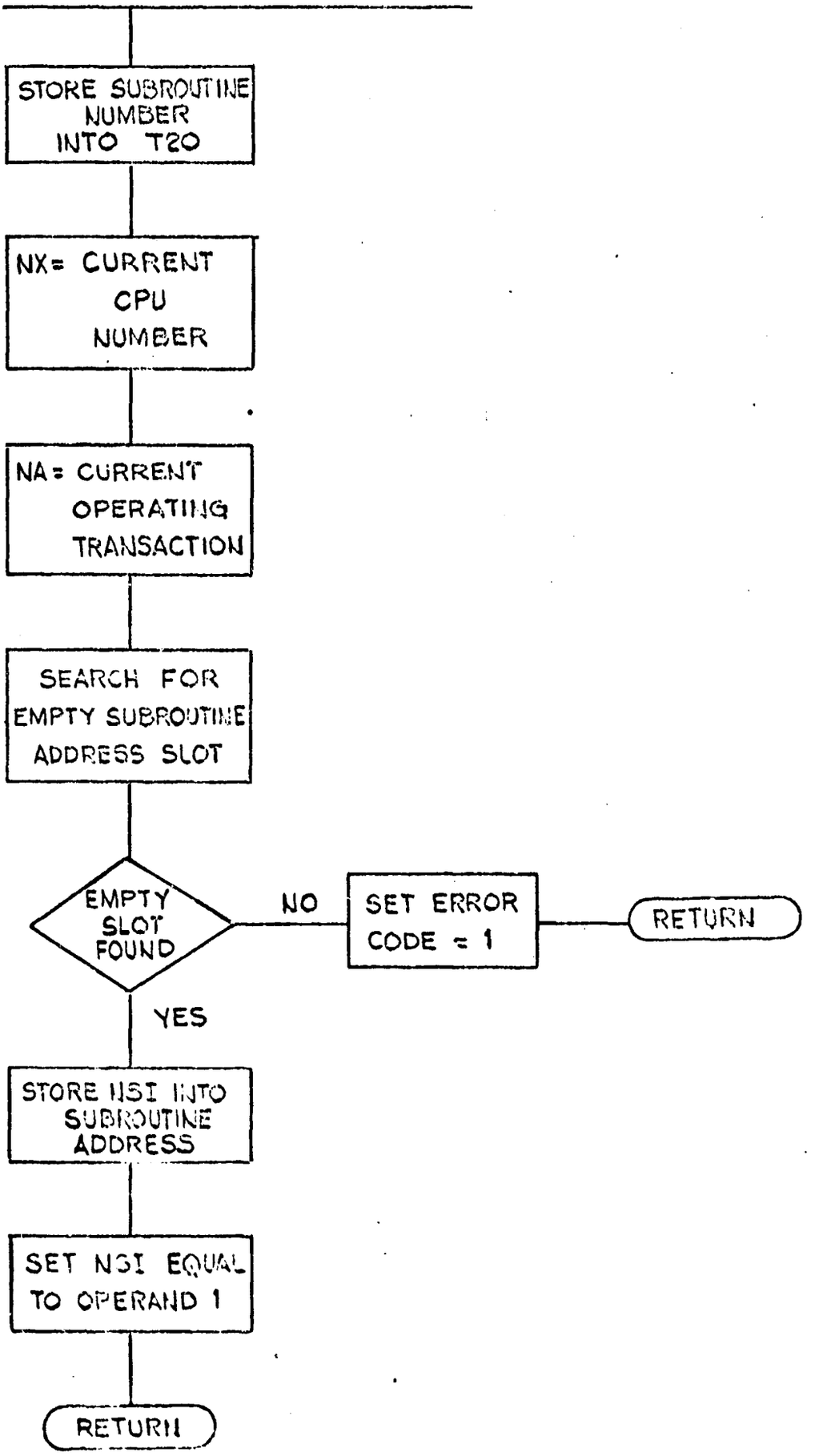


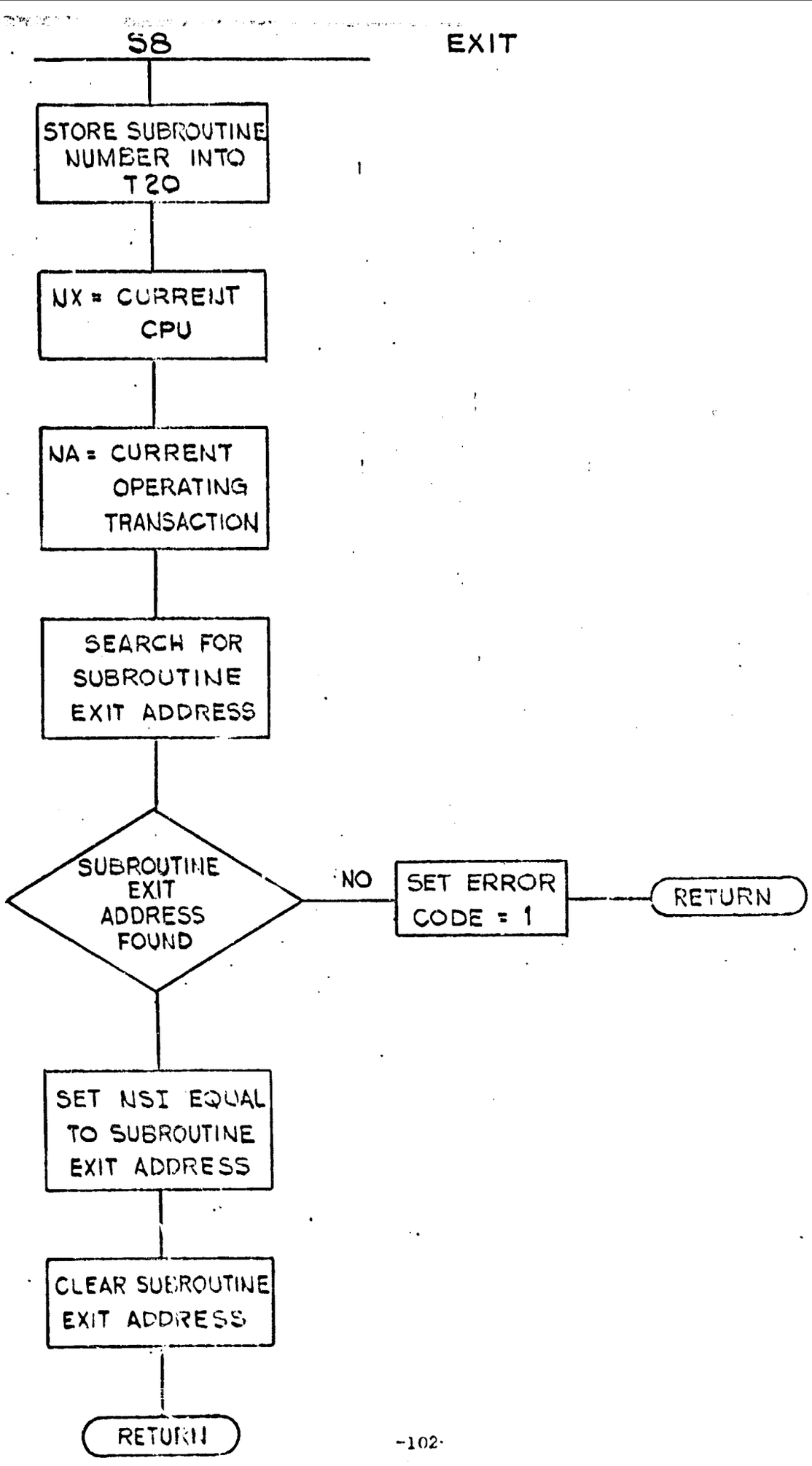

S3 READ
STORE SUBROUTINE # INTO T2Ø
NX= CURRENT CPU #
NA= CURRENT OPERATING TRANSACTION
STORE CURRENT FILE REQUEST INTO TRANSACTION
SET INTERRUPT CODE (T2Ø) =2
SET AT = COT
ZERO COT
RETURN

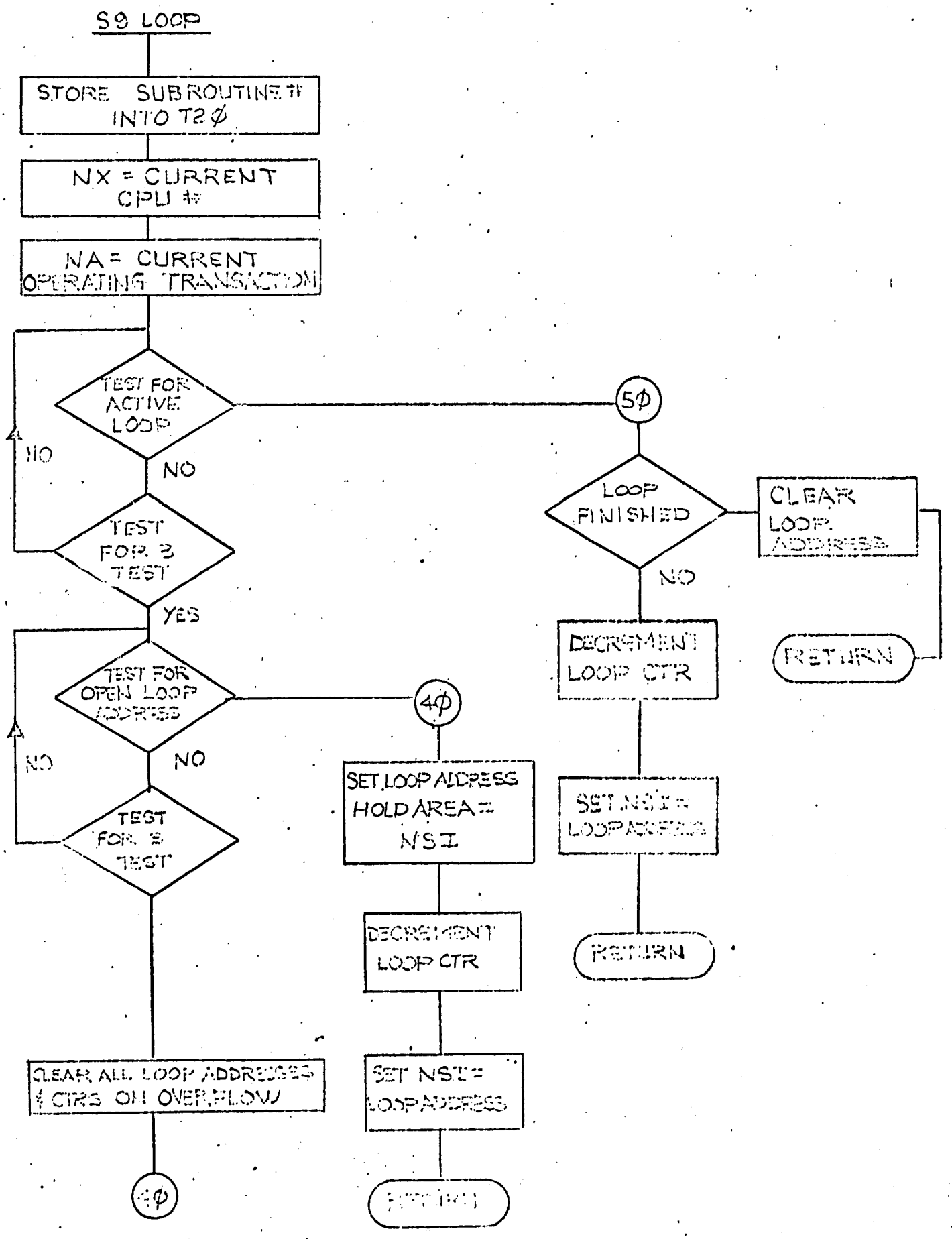

S4 WRITE
STORE SUBROUTINE # INTO T2Ø
SET CURRENT FILE REQUEST = NEGATIVE
NX=CURRENT CPU #
NA= CURRENT OPERATING TRANSACTION.
STORE CURRENT FILE REQUEST INTO TRANSACTION
SET INTERRUPT CODE = 2
SET AT= COT
ZERO COT
RETURN

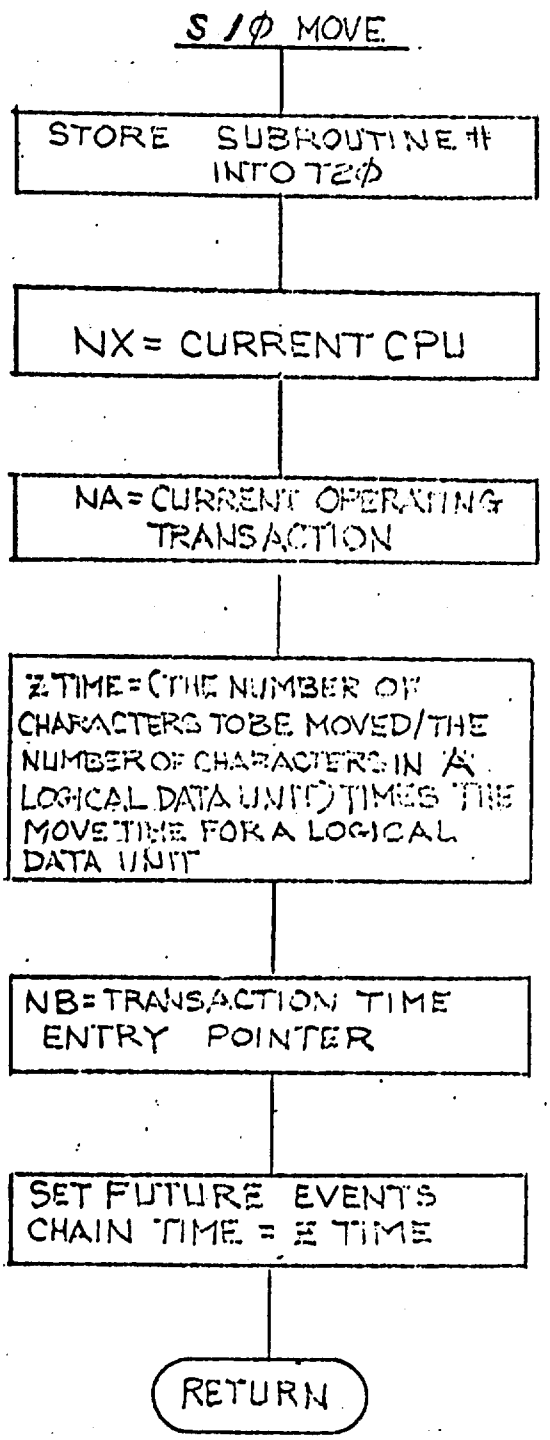

S5 FUNCTION
STORE SUBROUTINE # INTO T2Ø
NX= CURRENT CPU #
NA= CURRENT OPERATING TRANSACTION
CALL GENERATE IOTI S2Ø4
NB= GENERATED IOTI ADDRESS
SET IOTI TYPE=5 FOR FUNCTION
SET FUNCTION NUMBER INTO IOTI
SET INTERRUPT CODE =3
SET CURRENT IOT = GENERATED IOT
RETURN

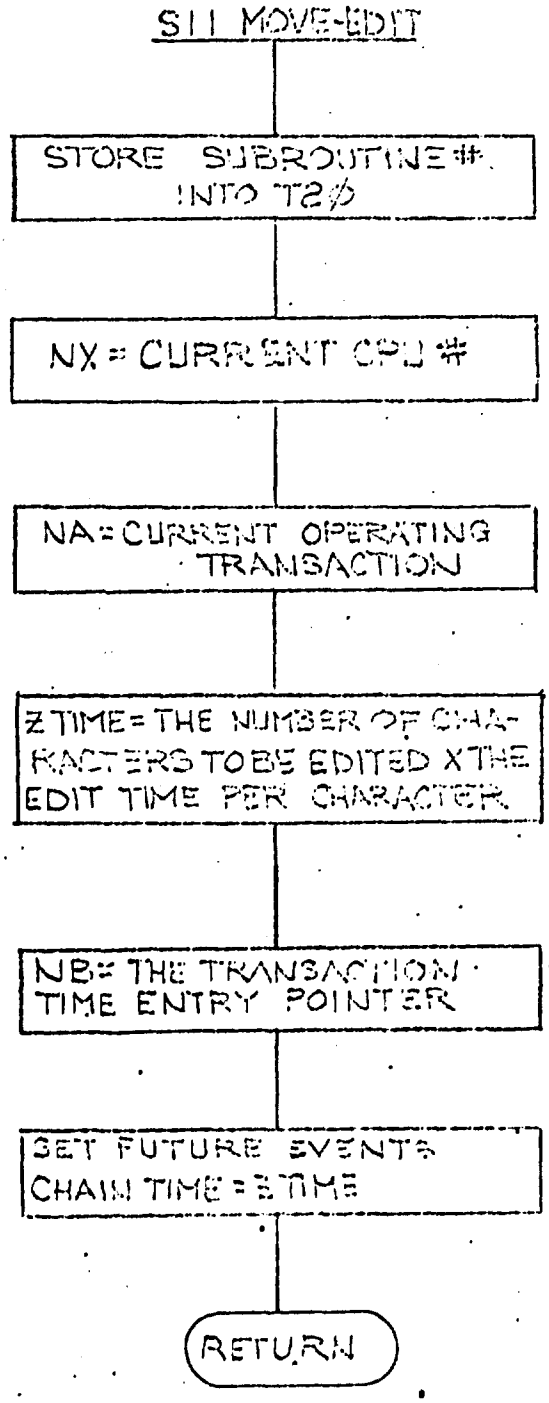

S6 - FOF
STORE SUBROUTINE# INTO T2Φ
N= ORDINAL FILE #
NX= CURRENT CPU #
NA= CURRENT OPERATING TRANSACTION
NC= WORKER ROUTINE NUMBER
ND= FILE VECTOR STARTING ADDRESS
ND= ENTRY IN FILE VECTOR TABLE
NE= REAL FILE #
NB= NUMBER OF BLOCKS IN FILE
ND= TI PART II ADDRESS
A
A
ORDINAL FILE # : 7
II
ND= I'I NEXT TI PART II ADDR
ORDINAL FILE NUMBER = ORDINAL FILE # = -7
NB . NUMBER OF I/O
SET NSI =NEW ADDRESS
RETURN
RETURN

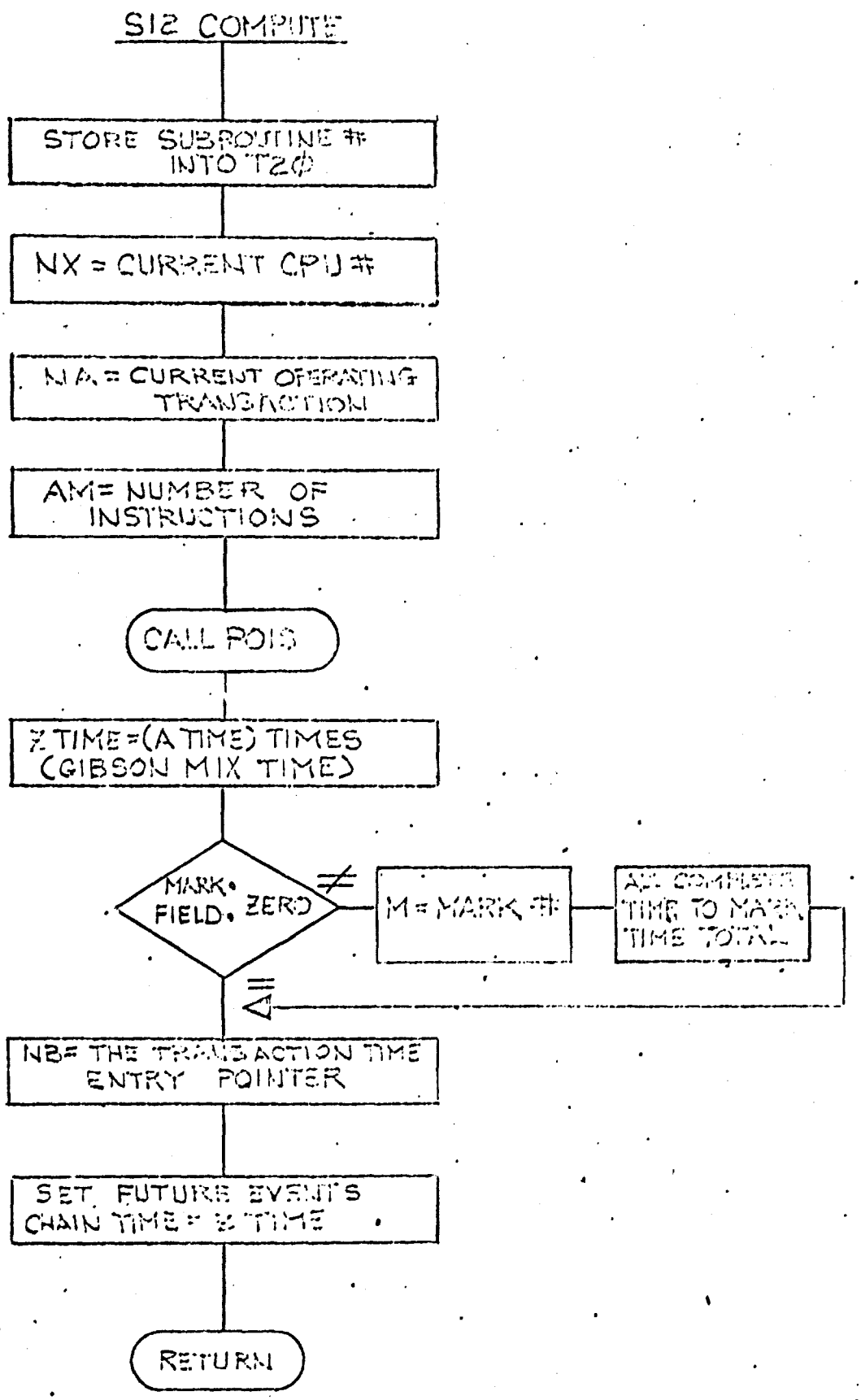

STORE SUBROUTINE NUMBER INTO T20
NX= CURRENT CPU NUMBER
NA= CURRENT OPERATING TRANSACTION
SEARCH FOR EMPTY SUBROUTINE ADDRESS SLOT
EMPTY SLOT FOUND
NO
SET ERROR CODE = 1
RETURN
YES
STORE NSI INTO SUBROUTINE ADDRESS
SET NSI EQUAL TO OPERAND 1
RETURN

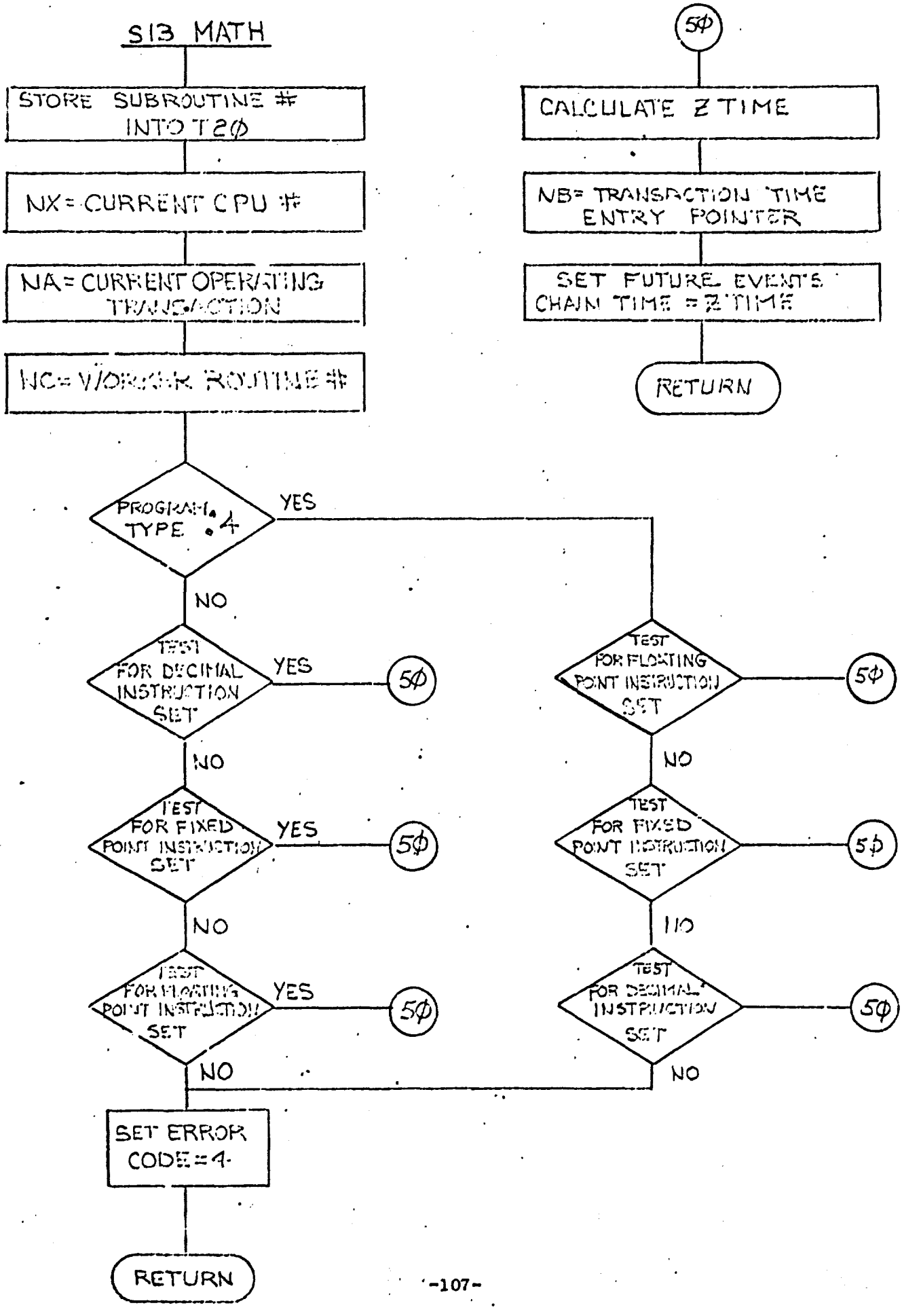

S8
EXIT
STORE SUBROUTINE NUMBER INTO T20
NX = CURREIJT CPU
NA = CURRENT OPERATING TRANSACTION
SEARCH FOR SUBROUTINE EXIT ADDRESS
SUBROUTINE EXIT ADDRESS FOUND
NO
SET ERROR CODE = 1
RETURN
SET NSI EQUAL TO SUBROUTINE EXIT ADDRESS
CLEAR SUBROUTINE EXIT ADDRESS
RETURN

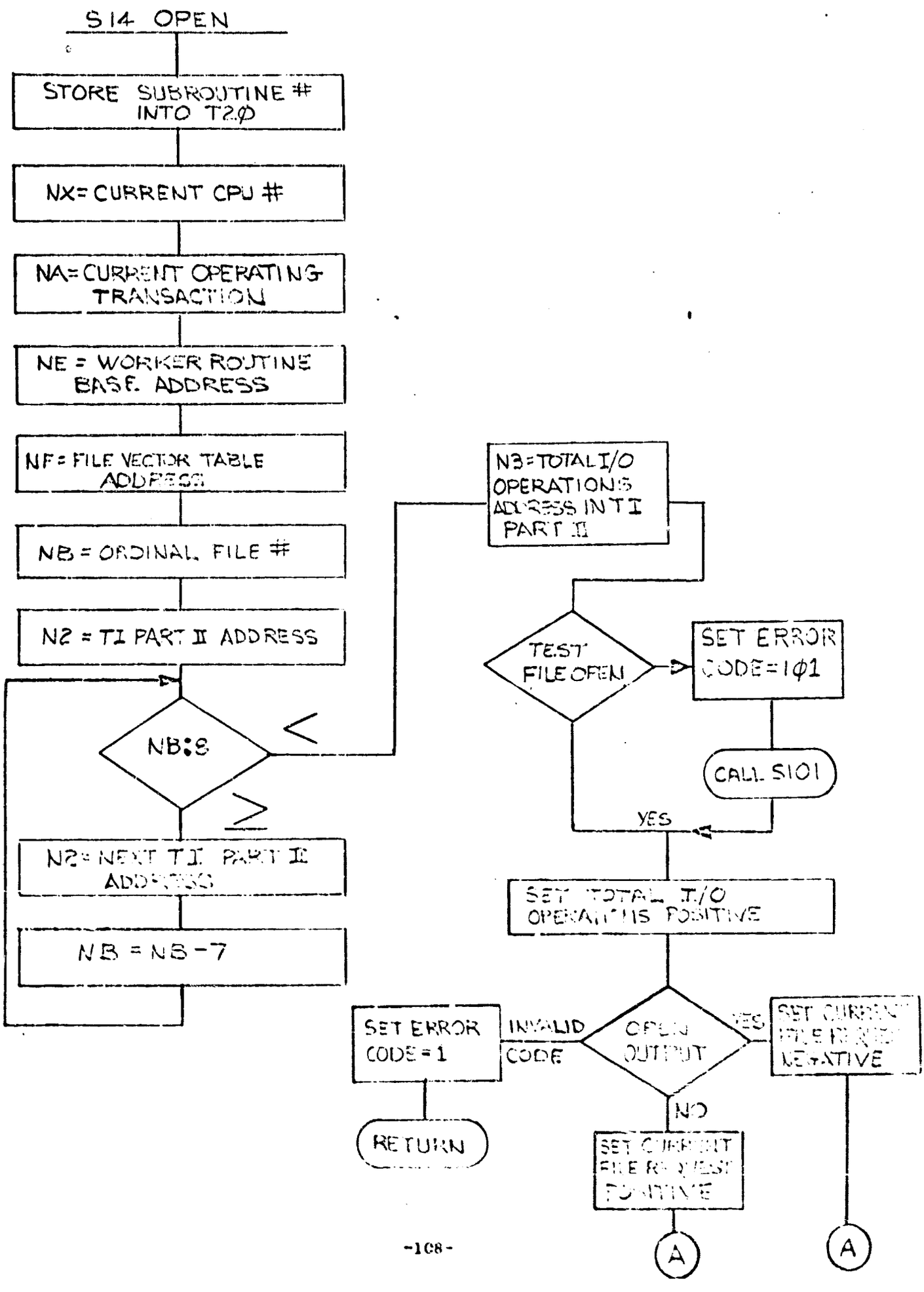

S9 LOOP
STORE SUBROUTINE TF INTO T2Ø
NX = CURRENT CPU #
NA = CURRENT OPERATING TRANSACTION
TEST FOR ACTIVE LOOP
5Ø
YES
NO
LOOP FINISHED
CLEAR LOOP ADDRESS
NO
DECREMENT LOOP CTR
RETURN
TEST FOR 3 TEST
YES
TEST FOR OPEN LOOP ADDRESS
4Ø
NO
NO
SET LOOP ADDRESS HOLD AREA = NSI
SET NSI = LOOP ADDRESS
TEST FOR 3 TEST
DECREMENT LOOP CTR
RETURN
CLEAR ALL LOOP ADDRESSES & CTRS ON OVERFLOW
SET NSI = LOOP ADDRESS
4Ø

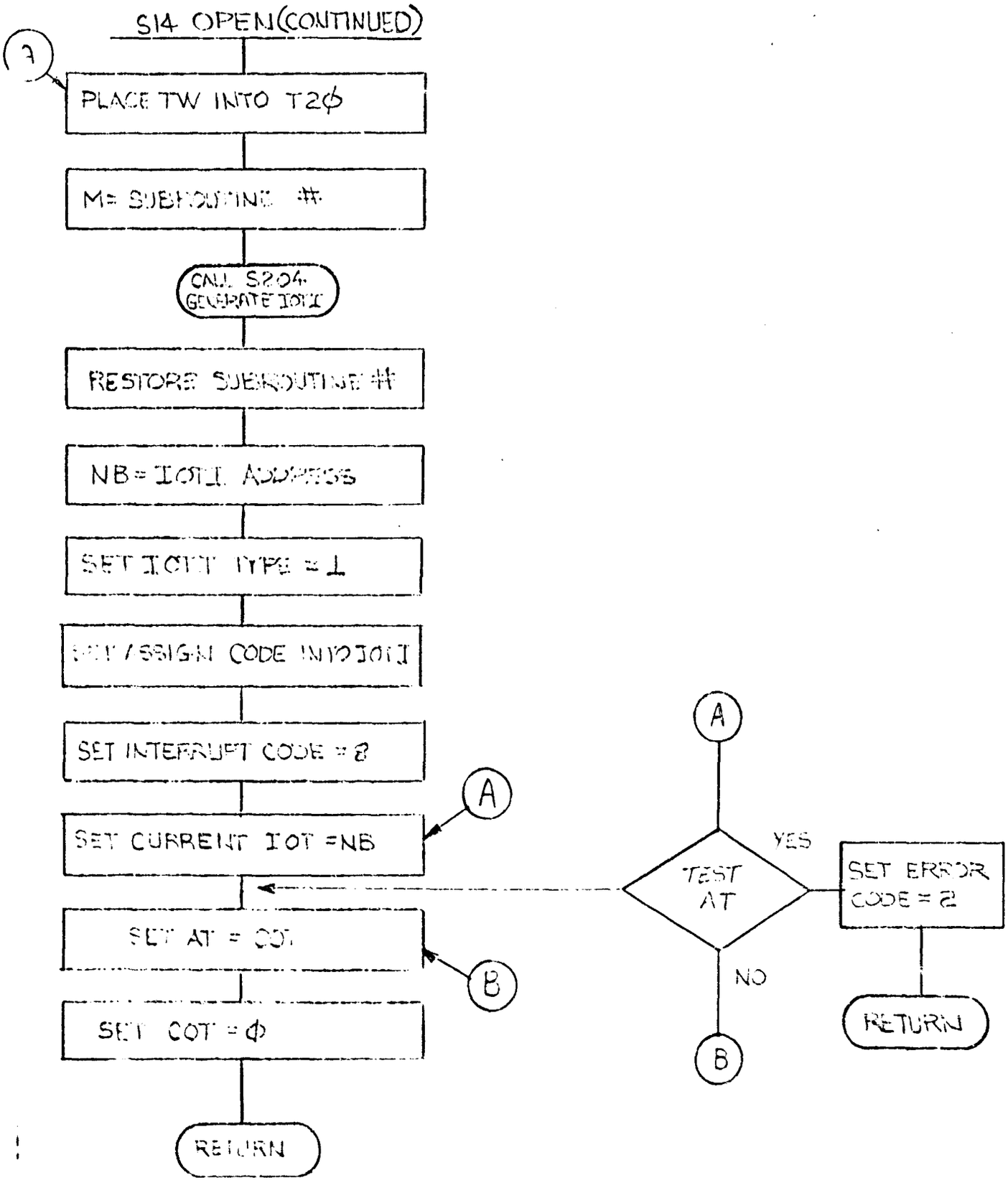

S/Ø MOVE
STORE SUBROUTINE # INTO 72Ø
NX = CURRENT CPU
NA = CURRENT OPERATING TRANSACTION
Z TIME = (THE NUMBER OF CHARACTERS TO BE MOVED/THE NUMBER OF CHARACTERS IN A LOGICAL DATA UNIT) TIMES THE MOVE TIME FOR A LOGICAL DATA UNIT
NB = TRANSACTION TIME ENTRY POINTER
SET FUTURE EVENTS CHAIN TIME = E TIME
RETURN

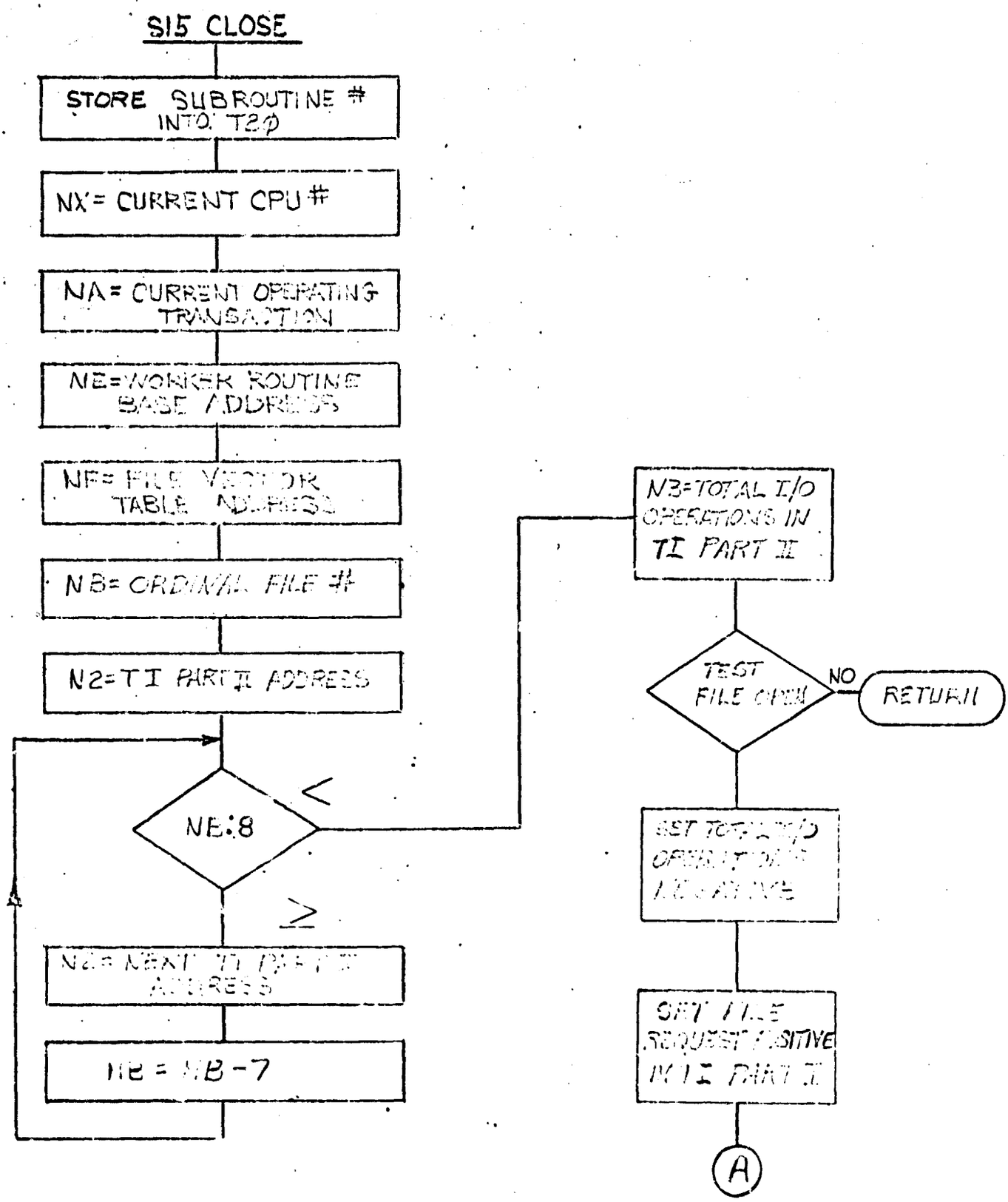

S11 MOVE-EDIT
STORE SUBROUTINE #
INTO T2Ø
NX = CURRENT CPU #
NA = CURRENT OPERATING
TRANSACTION
ZTIME = THE NUMBER OF CHA-
RACTERS TO BE EDITED X THE
EDIT TIME PER CHARACTER
NB = THE TRANSACTION
TIME ENTRY POINTER
SET FUTURE EVENTS
CHAIN TIME = ETIME
RETURN

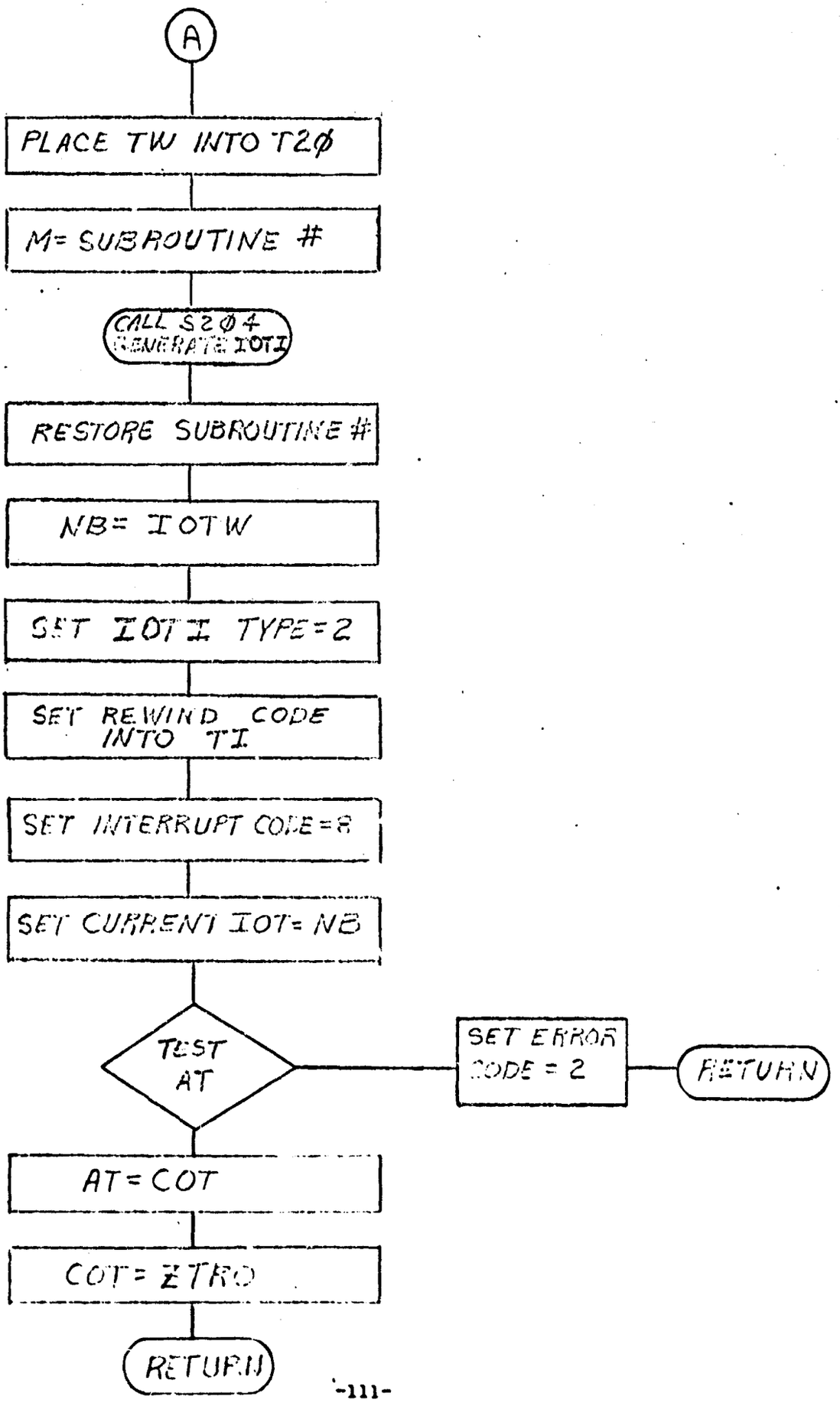

S12 COMPUTE
STORE SUBROUTINE # INTO T2Ø
NX = CURRENT CPU #
NA = CURRENT OPERATING TRANSACTION
AM = NUMBER OF INSTRUCTIONS
CALL POIS
% TIME = (A TIME) TIMES (GIBSON MIX TIME)
MARK FIELD ZERO
≠
M = MARK #
ALL COMPLETE TIME TO MARK TIME TOTAL
NB = THE TRANSACTION TIME ENTRY POINTER
SET FUTURE EVENTS CHAIN TIME = % TIME
RETURN

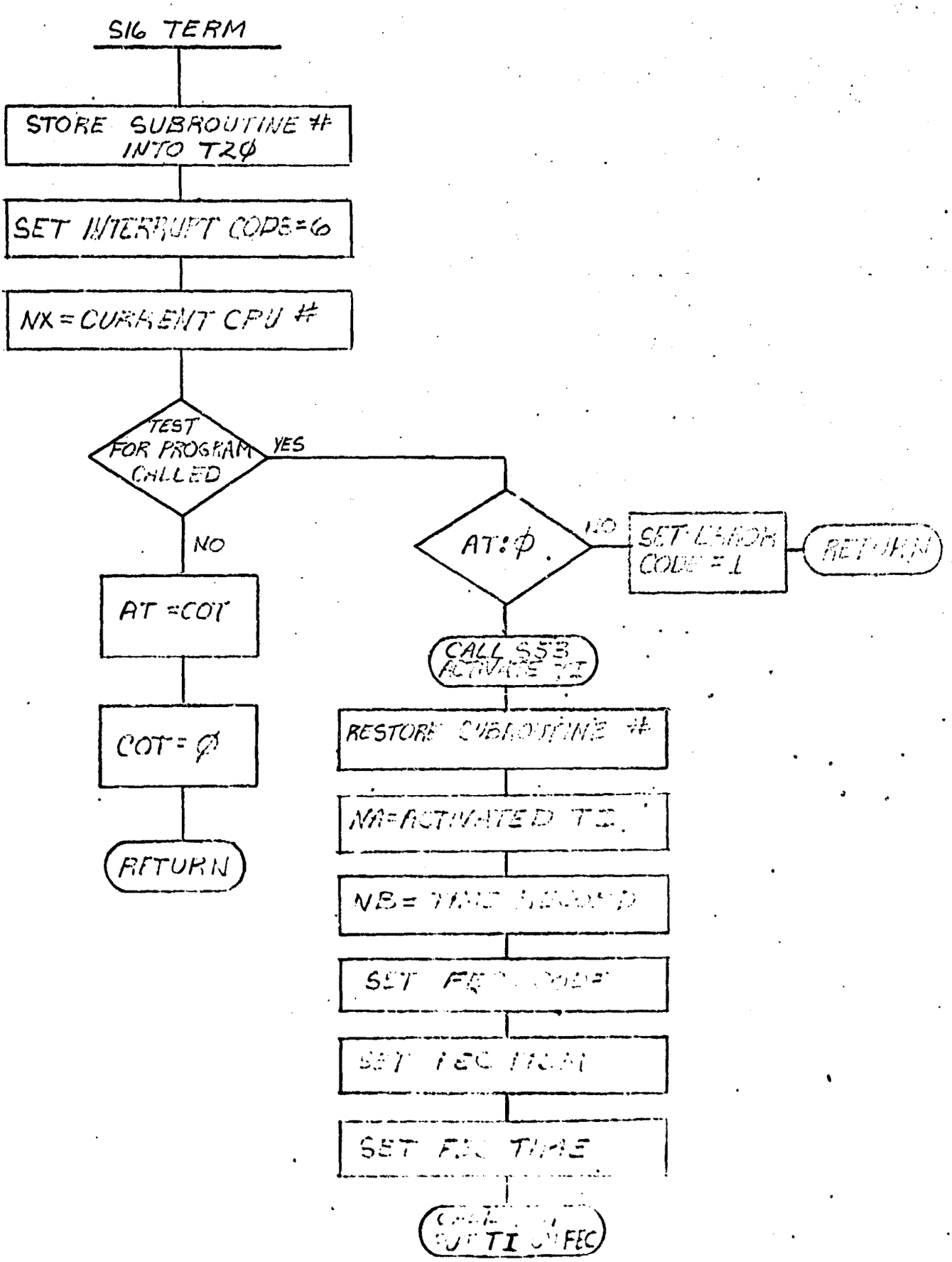

S13 MATH
STORE SUBROUTINE # INTO T2Ø
NX = CURRENT CPU #
NA = CURRENT OPERATING TRANSACTION
NC = WORKER ROUTINE #
PROGRAM TYPE = 4
YES
NO
TEST FOR DECIMAL INSTRUCTION SET
YES
5Ø
NO
TEST FOR FIXED POINT INSTRUCTION SET
YES
5Ø
NO
TEST FOR FLOATING POINT INSTRUCTION SET
YES
5Ø
NO
SET ERROR CODE = 4
RETURN
5Ø
CALCULATE Z TIME
NB = TRANSACTION TIME ENTRY POINTER
SET FUTURE EVENTS CHAIN TIME = Z TIME
RETURN
TEST FOR FLOATING POINT INSTRUCTION SET
5Ø
NO
TEST FOR FIXED POINT INSTRUCTION SET
5Ø
NO
TEST FOR DECIMAL INSTRUCTION SET
5Ø
NO


S14 OPEN
STORE SUBROUTINE # INTO T2Ø
NX= CURRENT CPU #
NA= CURRENT OPERATING TRANSACTION
NE = WORKER ROUTINE BASE ADDRESS
NF = FILE VECTOR TABLE ADDRESS
NB = ORDINAL FILE #
N2 = TI PART II ADDRESS
NB:8
<
≥
N2 = NEXT TI PART II ADDRESS
NB = NB-7
N3 = TOTAL I/O OPERATIONS ADDRESS IN TI PART II
TEST FILE OPEN
SET ERROR CODE = 1Ø1
CALL S101
YES
SET TOTAL I/O OPERATIONS POSITIVE
OPEN OUTPUT
YES
SET CURRENT FILE REQUEST NEGATIVE
NO
SET CURRENT FILE REQUEST POSITIVE
INVALID CODE
SET ERROR CODE = 1
RETURN
A
A

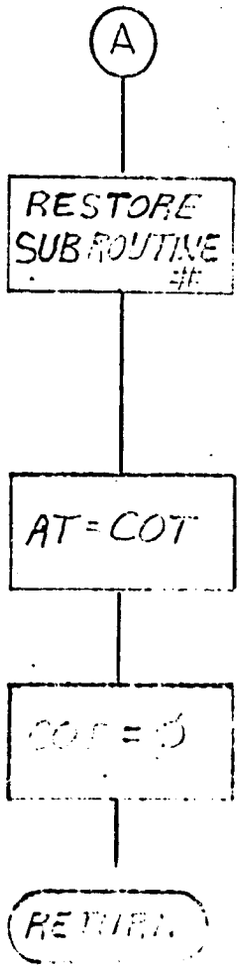

S14 OPEN(CONTINUED)
PLACE TW INTO T2ϕ
M= SUBROUTINE #
CALL S204 GENERATE IOTI
RESTORE SUBROUTINE #
NB = IOTI ADDRESS
SET IOTT TYPE = 1
SET/ASSIGN CODE INTO IOTI
SET INTERRUPT CODE = 8
SET CURRENT IOT = NB
SET AT = COT
SET COT = ϕ
RETURN
A
B
TEST AT
YES
NO
SET ERROR CODE = 2
RETURN

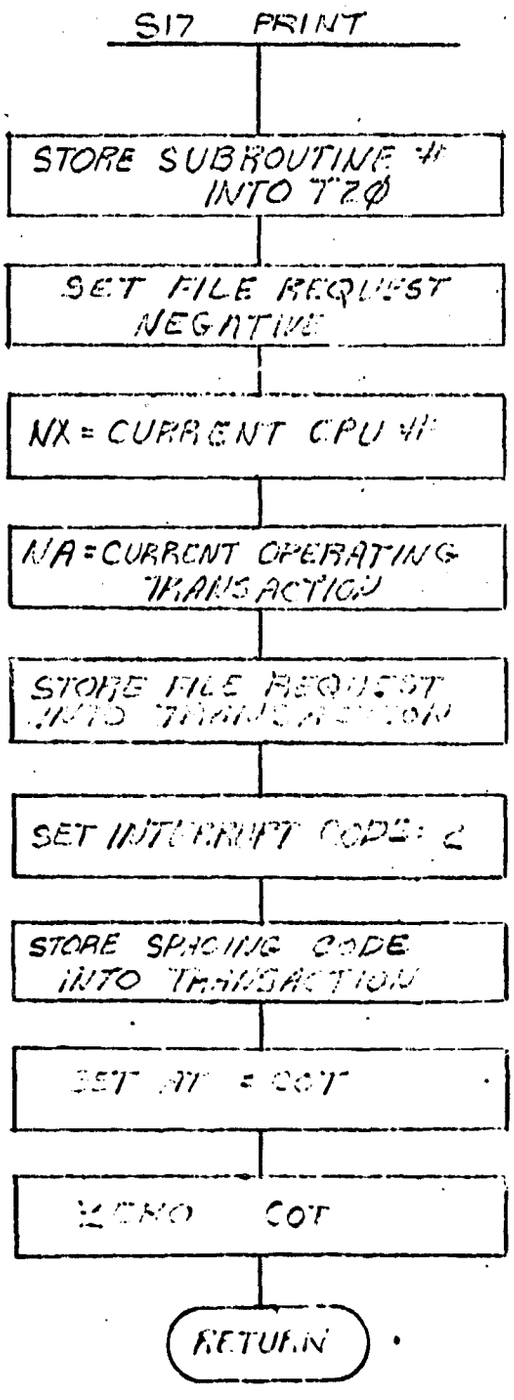

S15 CLOSE
STORE SUBROUTINE # INTO T2Ø
NX= CURRENT CPU #
NA= CURRENT OPERATING TRANSACTION
NE=WORKER ROUTINE BASE ADDRESS
NF= FILE VECTOR TABLE ADDRESS
NB= ORDINAL FILE #
N2=TI PART II ADDRESS
NB:8
<
≥
NB=TOTAL I/O OPERATIONS IN TI PART II
TEST FILE OPEN
NO
RETURN
A

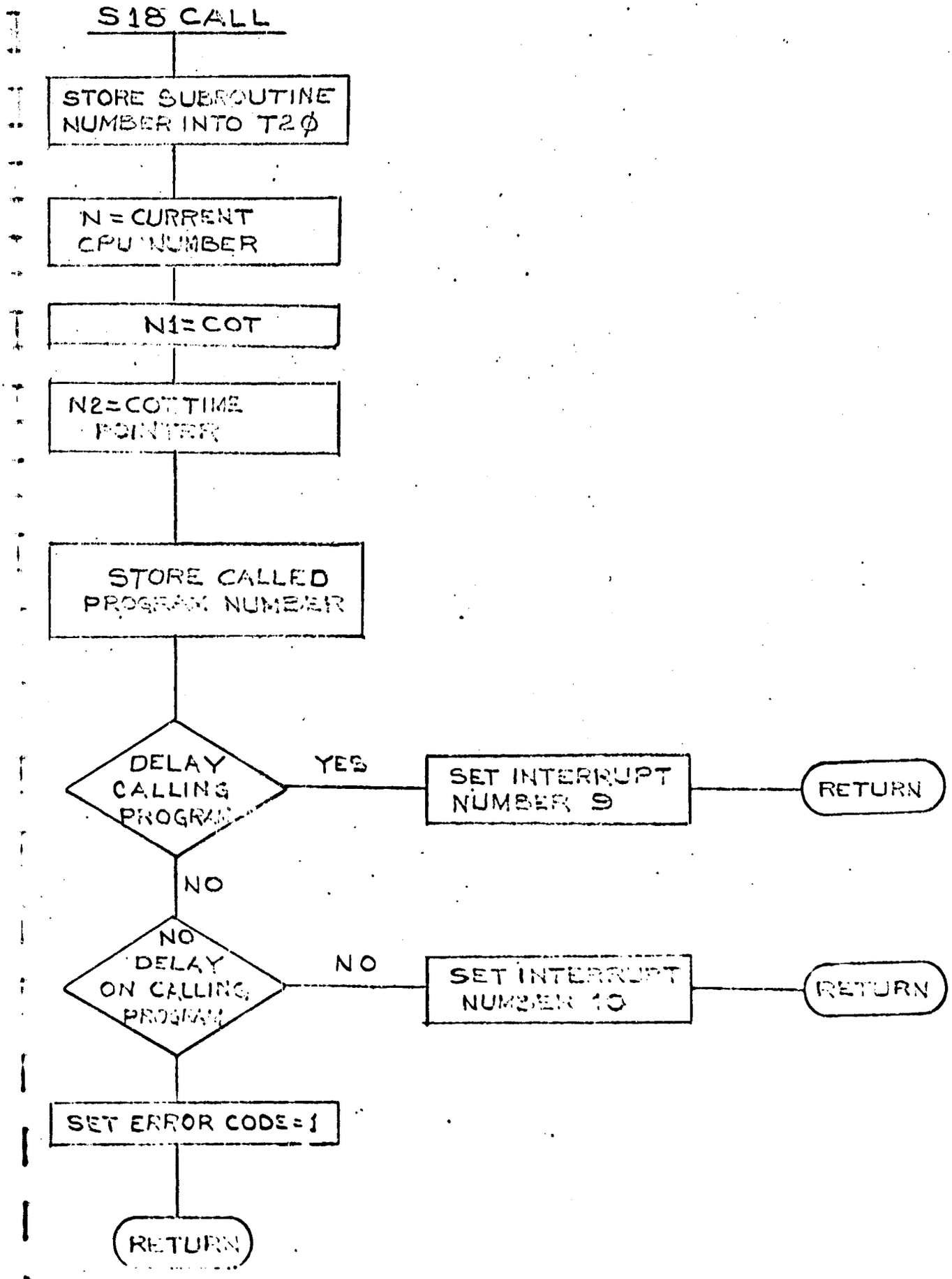

A
PLACE TW INTO T2Ø
M= SUBROUTINE #
CALL S2Ø4 GENERATE IOTI
RESTORE SUBROUTINE #
NB= IOTW
SET IOTI TYPE=2
SET REWIND CODE INTO TI
SET INTERRUPT CODE=8
SET CURRENT IOT= NB
TEST AT
SET ERROR CODE = 2
RETURN
AT= COT
COT= ZERO
RETURN

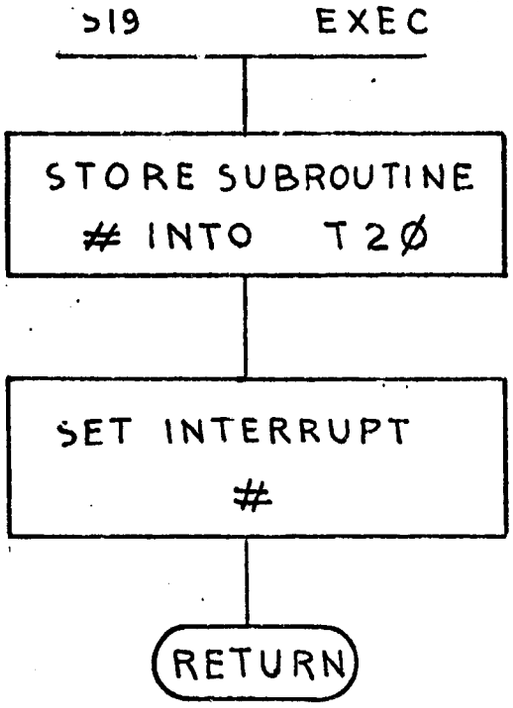

S16 TERM
STORE SUBROUTINE # INTO T2Ø
SET INTERRUPT CODE=6
NX=CURRENT CPU #
TEST FOR PROGRAM CALLED
YES
NO
AT=COT
COT=Ø
RETURN
AT:Ø.
NO
SET RANDOM CODE=1
RETURN
CALL SSB ACTIVATE TI
RESTORE SUBROUTINE #
NA=ACTIVATED TI.
NB=TIME REQUIRED
SET FE
SET FEC FROM
SET FEC TIME
FEC

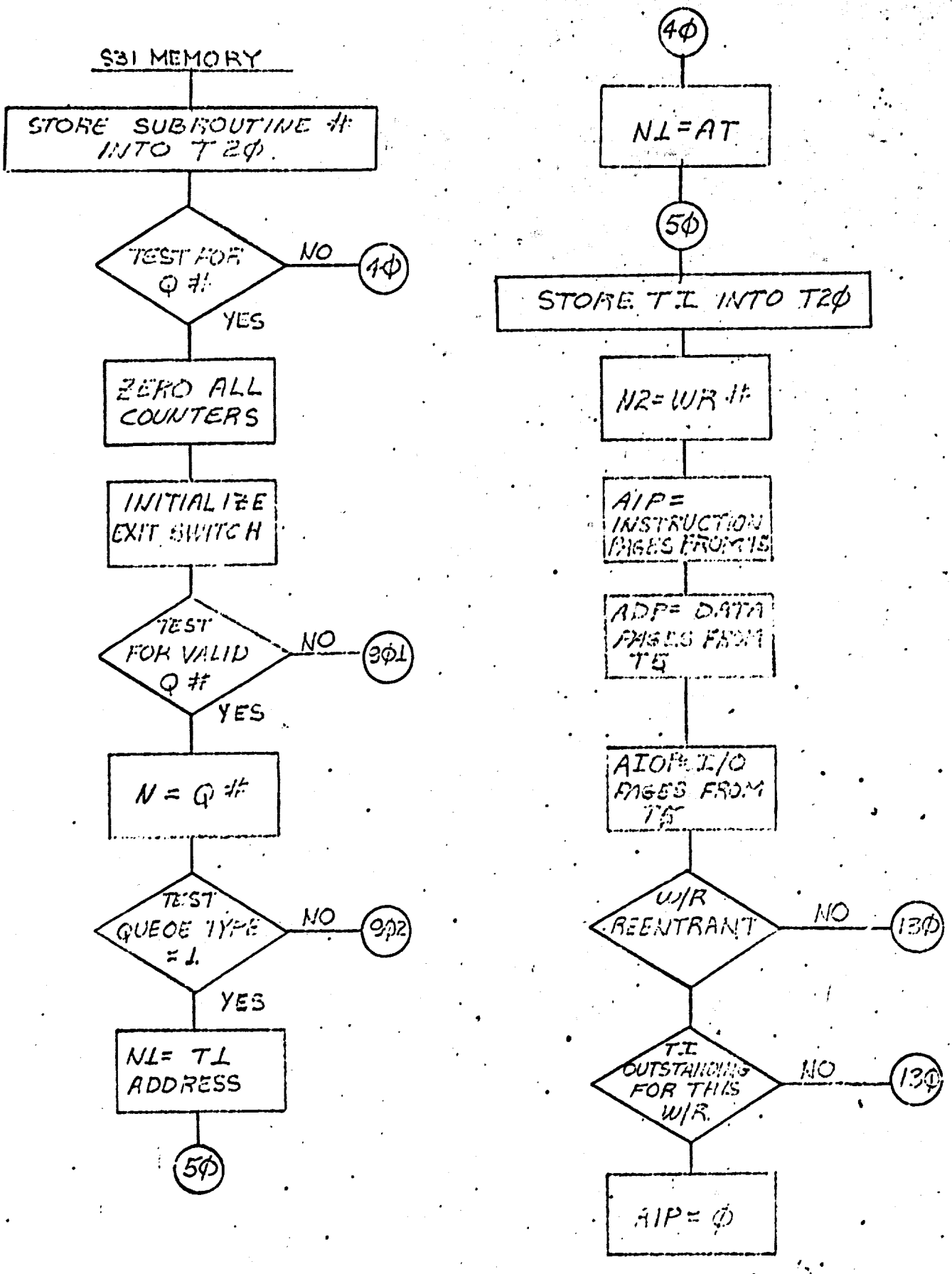
S16 TERM. (CONTINUED)

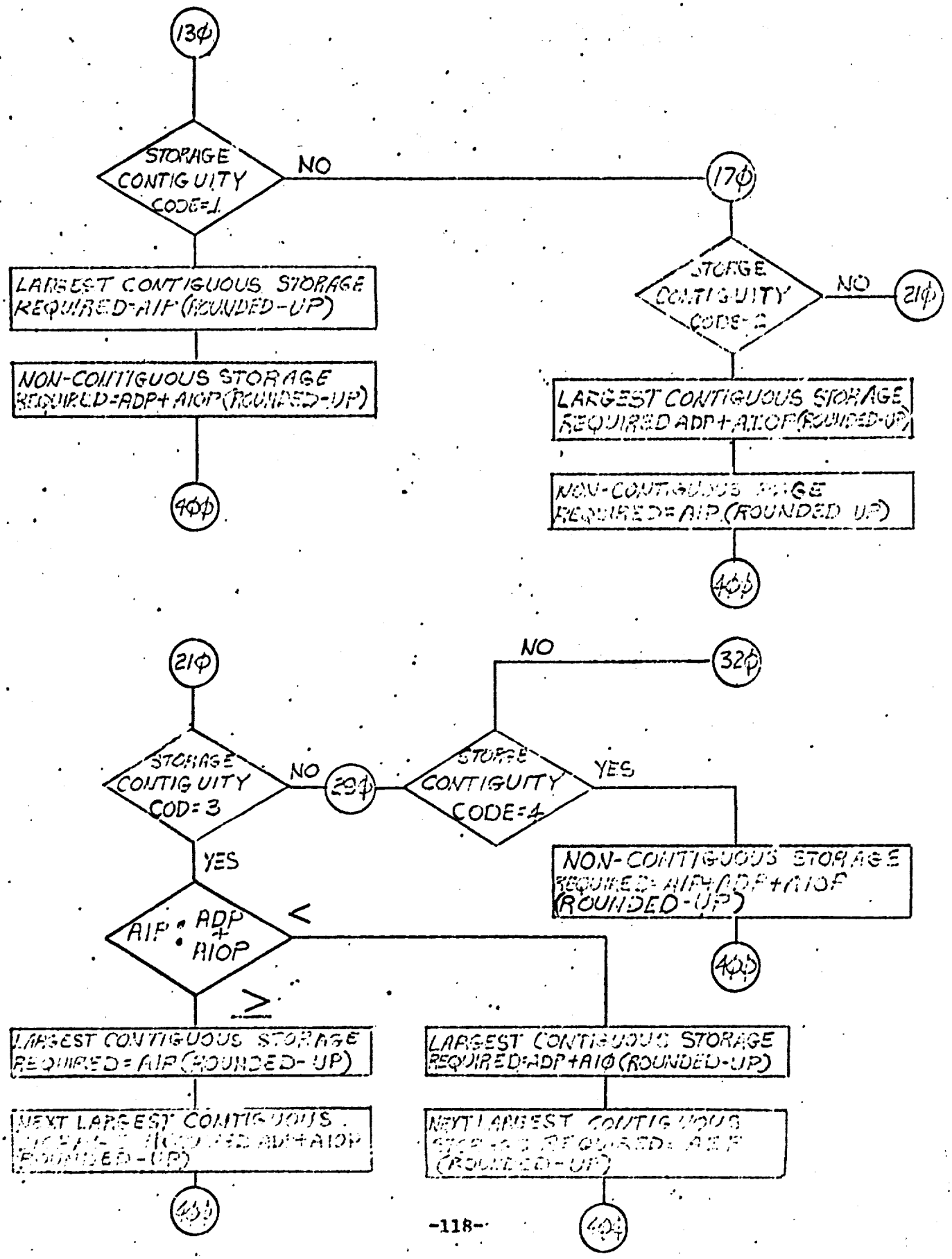


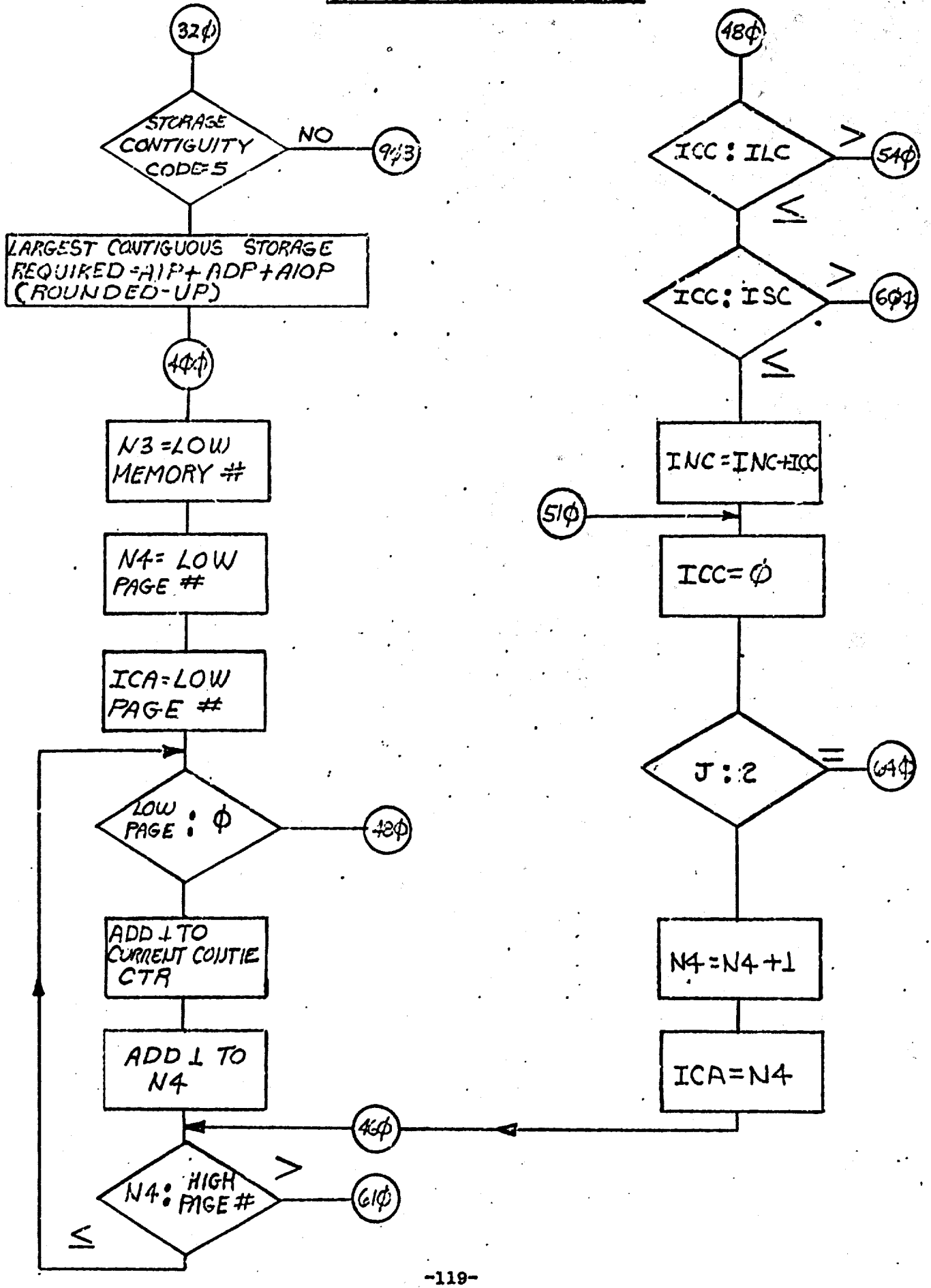

S17 PRINT
STORE SUBROUTINE # INTO T2Ø
SET FILE REQUEST NEGATIVE
NX = CURRENT CPU #
NA = CURRENT OPERATING TRANSACTION
STORE FILE REQUEST INTO TRANSACTION
SET INTERRUPT CODE = 2
STORE SPACING CODE INTO TRANSACTION
SET AT = COT
XCHG COT
RETURN


S18 CALL
STORE SUBROUTINE NUMBER INTO T2Ø
N = CURRENT CPU NUMBER
N1=COT
N2=COT TIME POINTER
STORE CALLED PROGRAM NUMBER
DELAY CALLING PROGRAM
YES
SET INTERRUPT NUMBER 9
RETURN
NO
NO DELAY ON CALLING PROGRAM
NO
SET INTERRUPT NUMBER 10
RETURN
SET ERROR CODE=1
RETURN

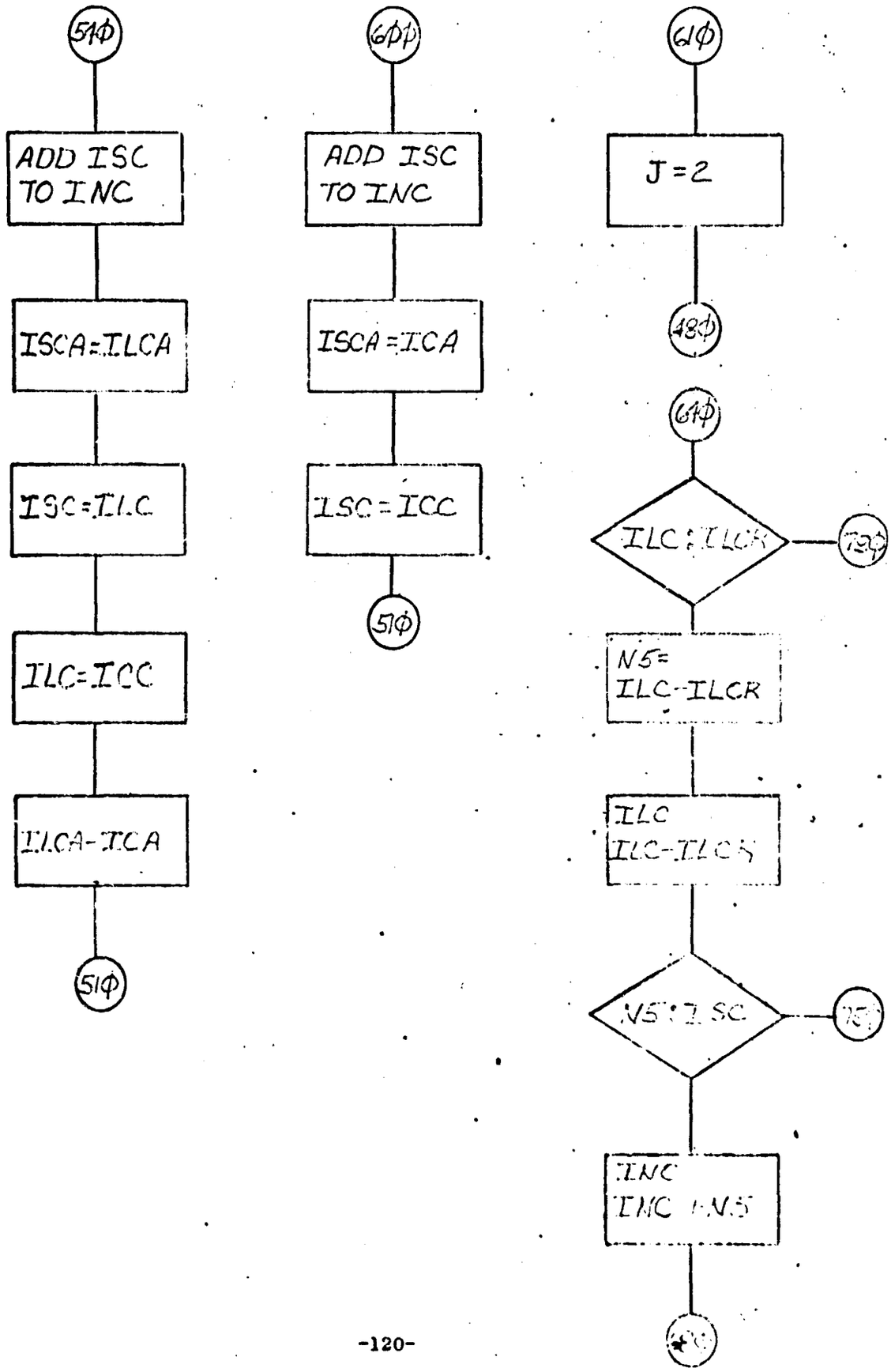

>19 EXEC
STORE SUBROUTINE # INTO T2Ø
SET INTERRUPT #
RETURN


S31 MEMORY
STORE SUBROUTINE # INTO T2Φ.
TEST FOR Q #
NO
1Φ
YES
ZERO ALL COUNTERS
INITIALIZE EXIT SWITCH
TEST FOR VALID Q #
NO
3Φ1
YES
N = Q #
TEST QUEOE TYPE = 1.
NO
9Φ2
YES
NL= TL ADDRESS
5Φ
4Φ
NL=AT
5Φ
STORE T.L INTO T2Φ
N2= WR #
AIP= INSTRUCTION PAGES FROM T5
ADP= DATA PAGES FROM T5
AIOP= I/O PAGES FROM T5
W/R REENTRANT
NO
13Φ
T.L OUTSTANDING FOR THIS W/R.
NO
13Φ
AIP= Φ

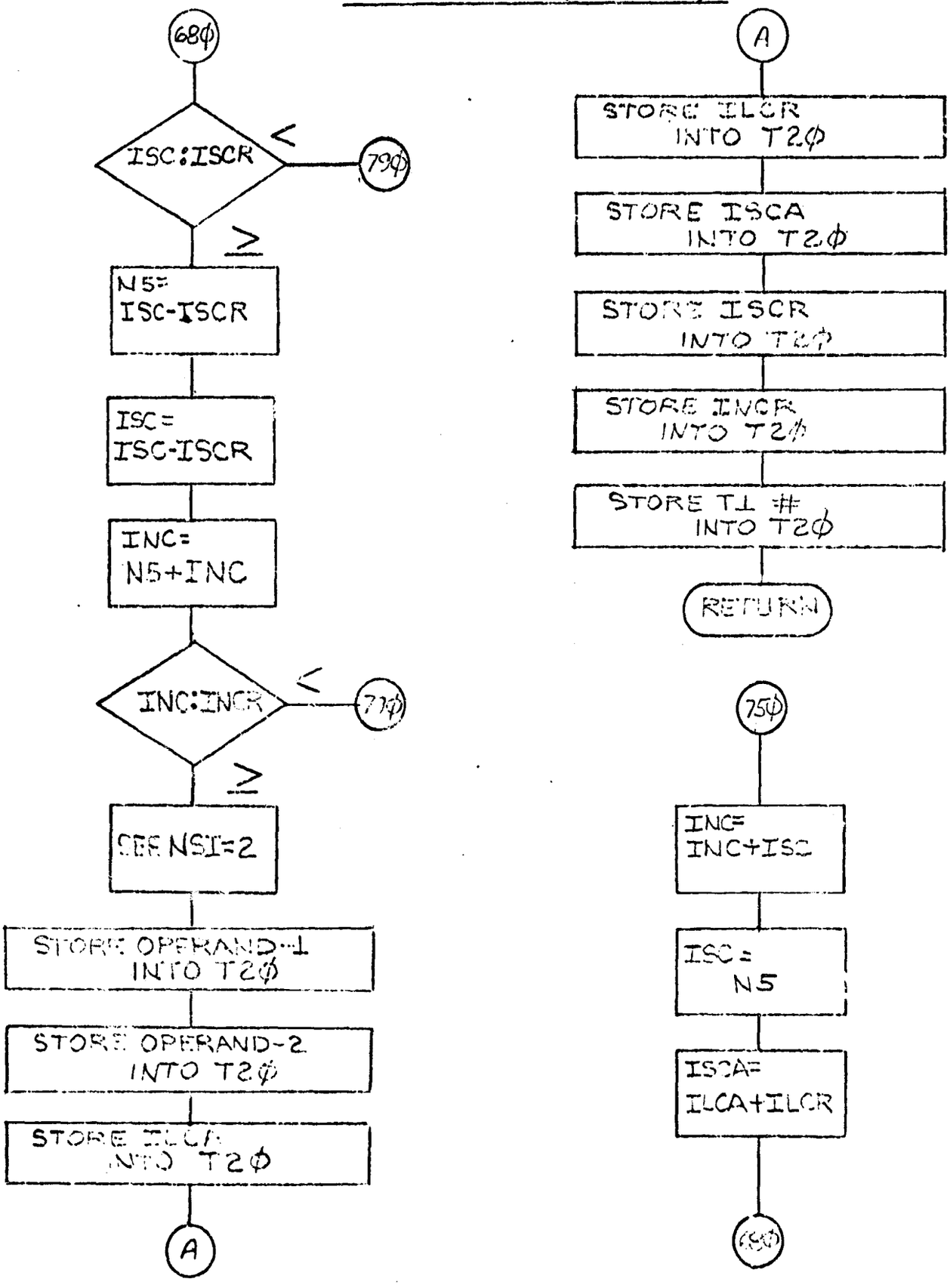

13Φ
STORAGE CONTIGUITY CODE=1
NO
17Φ
LARGEST CONTIGUOUS STORAGE REQUIRED=AIP (ROUNDED-UP)
NON-CONTIGUOUS STORAGE REQUIRED=ADP+AIOP (ROUNDED-UP)
40Φ
STORAGE CONTIGUITY CODE=2
NO
21Φ
LARGEST CONTIGUOUS STORAGE REQUIRED ADP+AIOP (ROUNDED-UP)
NON-CONTIGUOUS STORAGE REQUIRED=AIP (ROUNDED UP)
40Φ
21Φ
STORAGE CONTIGUITY CODE=3
NO
29Φ
STORAGE CONTIGUITY CODE=4
NO
32Φ
YES
NON-CONTIGUOUS STORAGE REQUIRED=AIP+ADP+AIOP (ROUNDED-UP)
40Φ
YES
AIP ? ADP + AIOP
<
≥
LARGEST CONTIGUOUS STORAGE REQUIRED=AIP (ROUNDED-UP)
NEXT LARGEST CONTIGUOUS
(ROUNDED-UP)
40Φ
LARGEST CONTIGUOUS STORAGE REQUIRED=ADP+AIO (ROUNDED-UP)
NEXT LARGEST CONTIGUOUS
(ROUNDED-UP)
40Φ

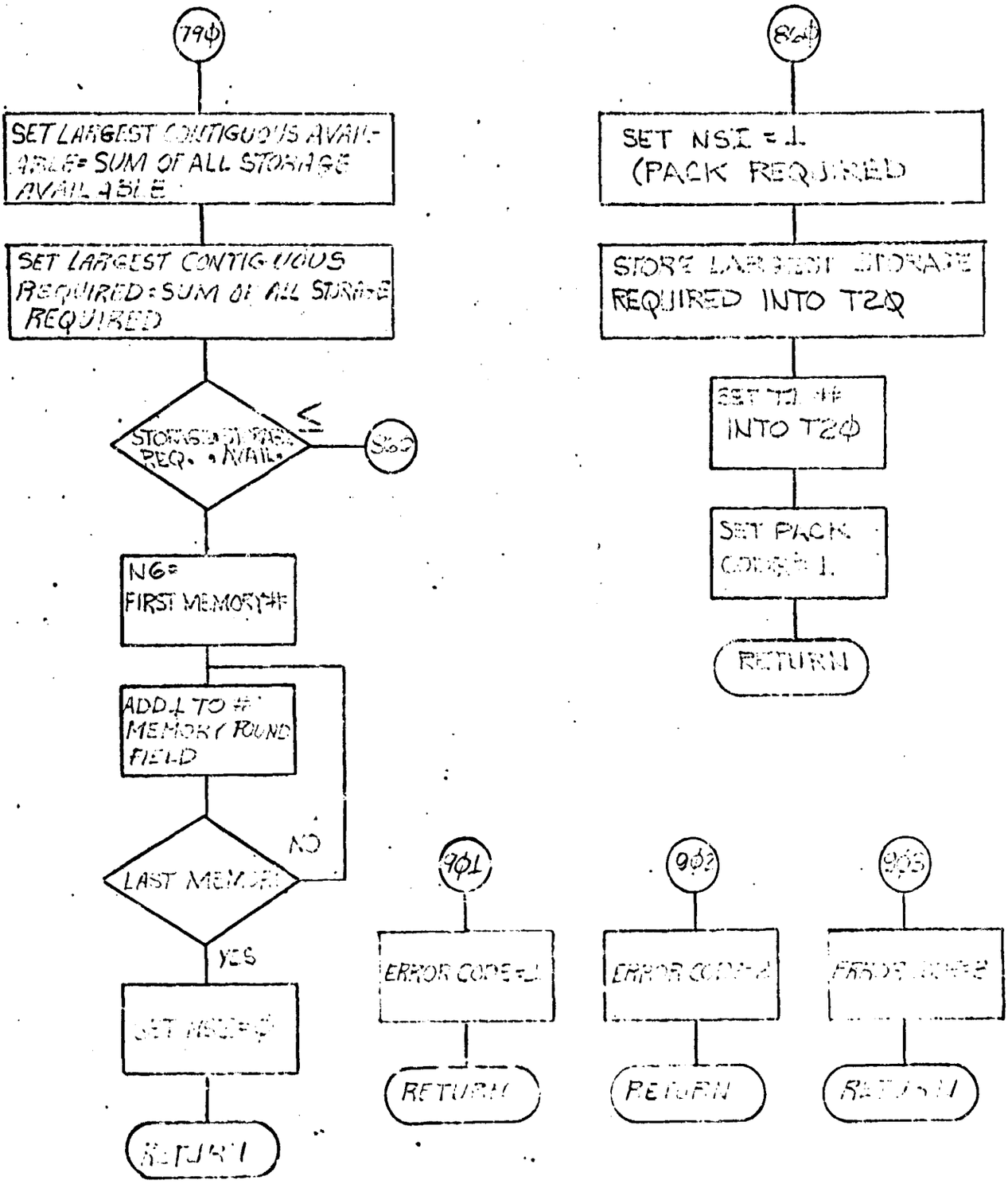

32φ
STORAGE CONTIGUITY CODE=5
NO
943
LARGEST CONTIGUOUS STORAGE REQUIRED=AIP+ADP+AIOP (ROUNDED-UP)
40φ
N3=LOW MEMORY #
N4=LOW PAGE #
ICA=LOW PAGE #
LOW PAGE : φ
42φ
ADD 1 TO CURRENT COUNTIE CTR
ADD 1 TO N4
46φ
N4 : HIGH PAGE #
>
61φ
≤
48φ
ICC : ILC
>
54φ
≤
ICC : ISC
>
60φ
≤
INC=INC+ICC
51φ
ICC=φ
J : 2
=
64φ
N4=N4+1
ICA=N4

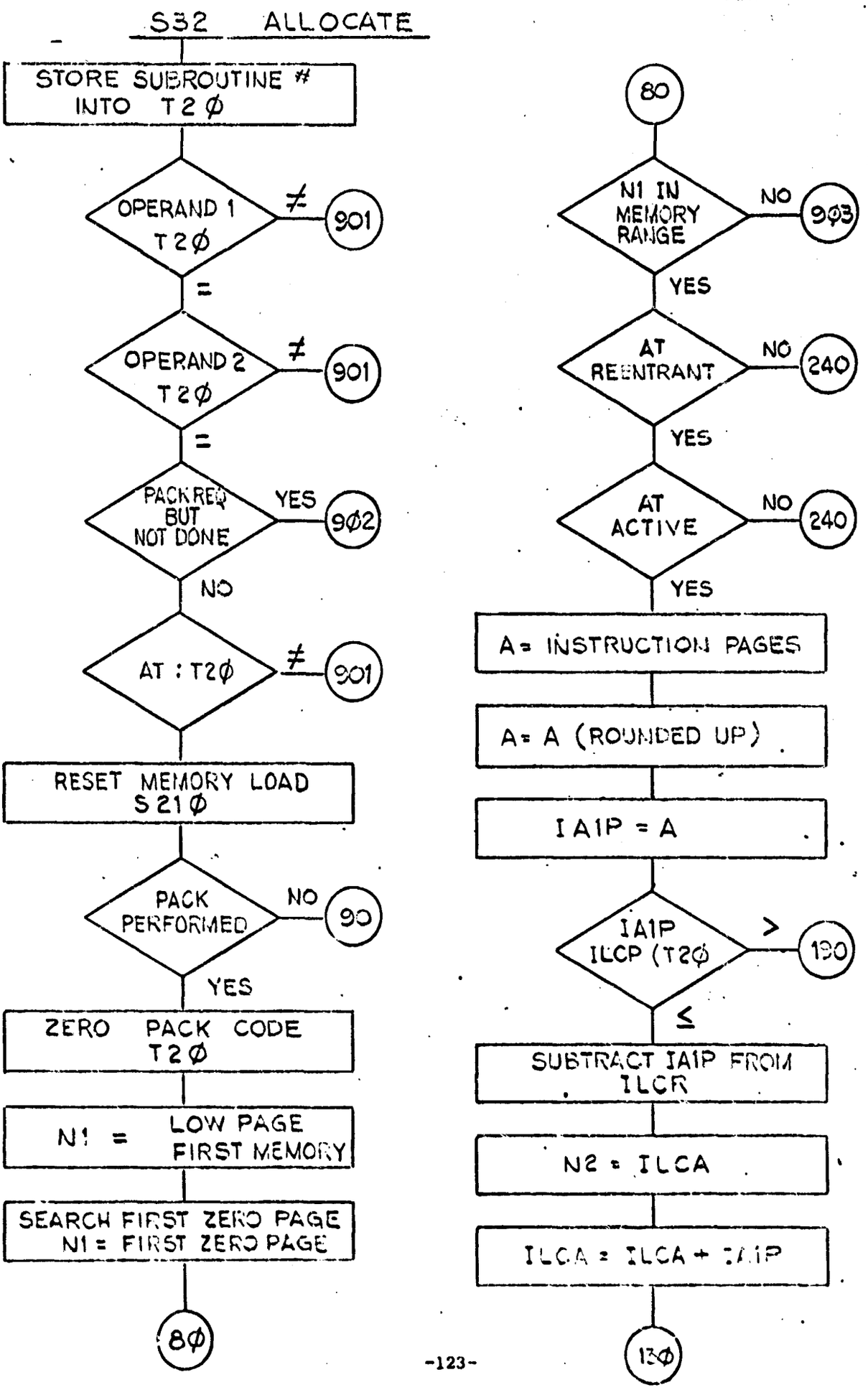
S31 MEMORY (CONTINUED)



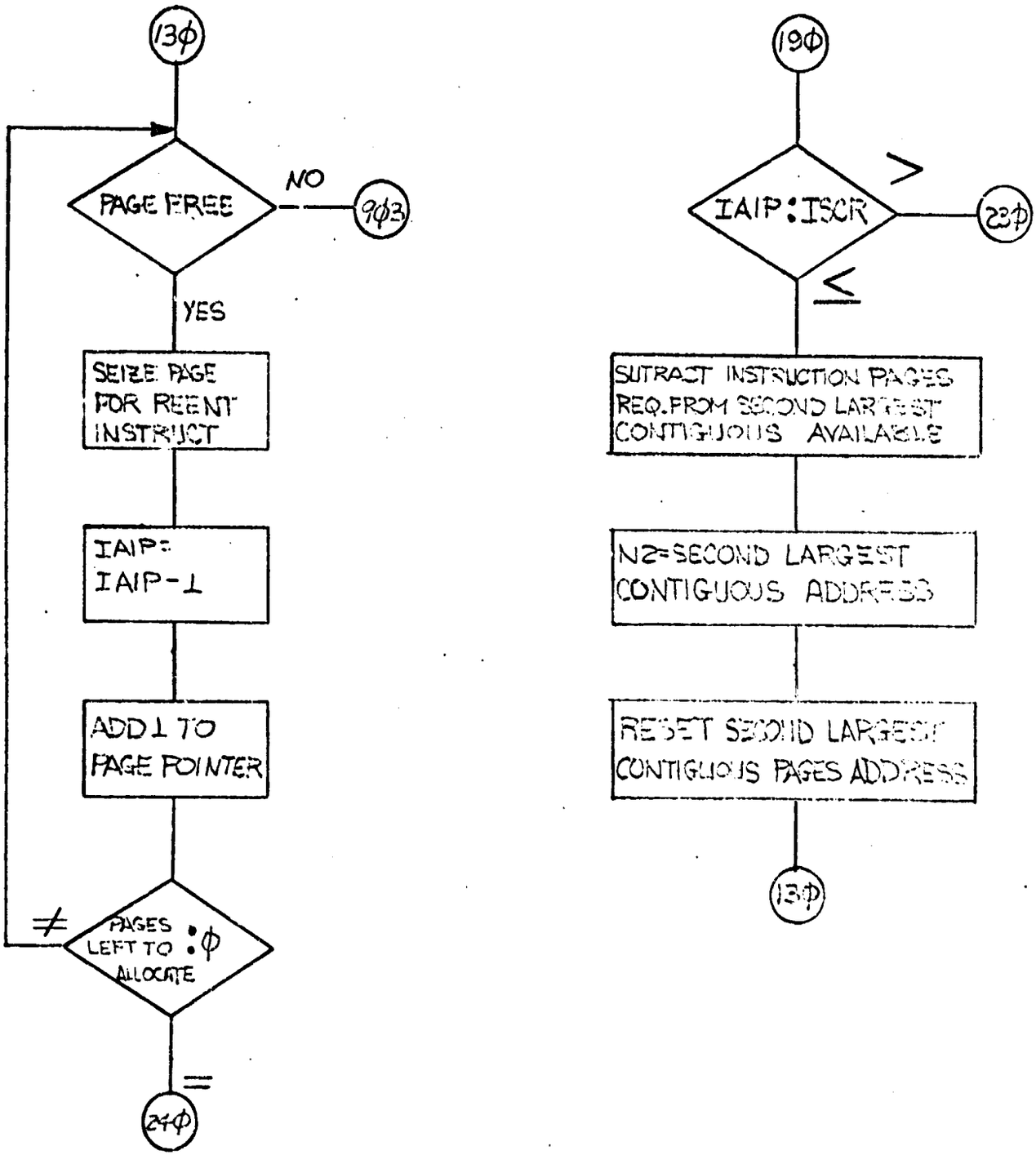
S3I. MEMORY (CONTINUED)



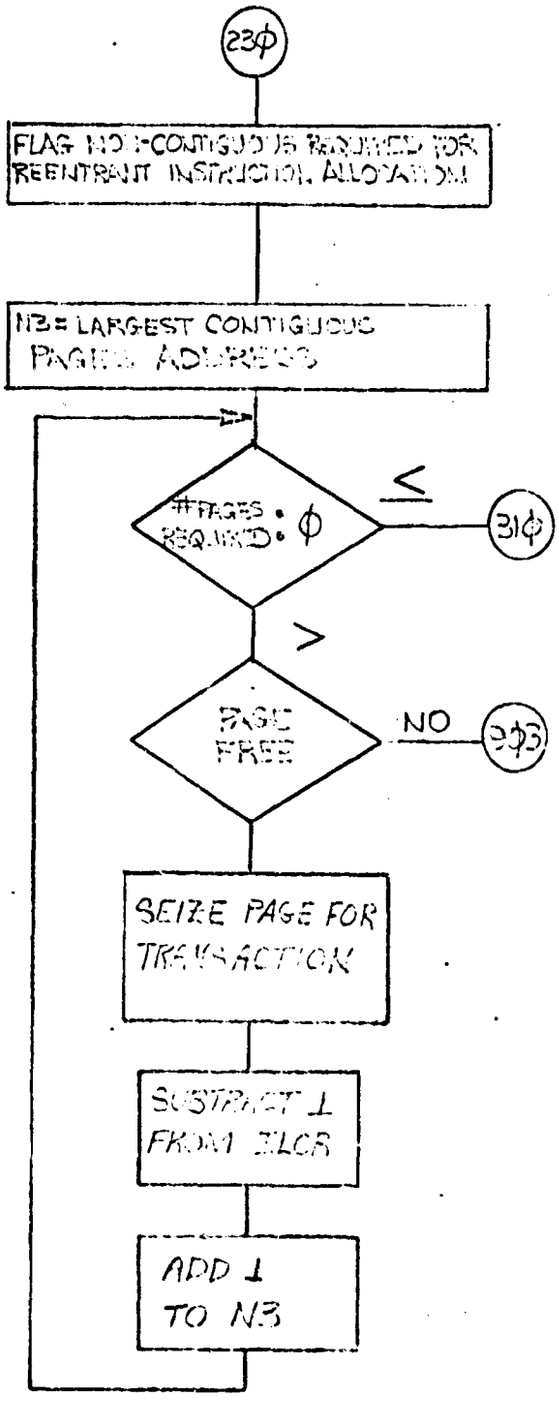

79Φ
SET LARGEST CONTIGUOUS AVAIL-
ABLE= SUM OF ALL STORAGE
AVAILABLE
SET LARGEST CONTIGUOUS
REQUIRED = SUM OF ALL STORAGE
REQUIRED
≤
NG=
FIRST MEMORY #
ADD 1 TO #
MEMORY ROUND
FIELD
NO
YES
RETURN
84
SET NSI = 1
(PACK REQUIRED
STORE LARGEST STORAGE
REQUIRED INTO T2Q
INTO T2Φ
SET PACK
RETURN
9Φ1
ERROR CODE = 1
RETURN
9Φ2
9Φ5

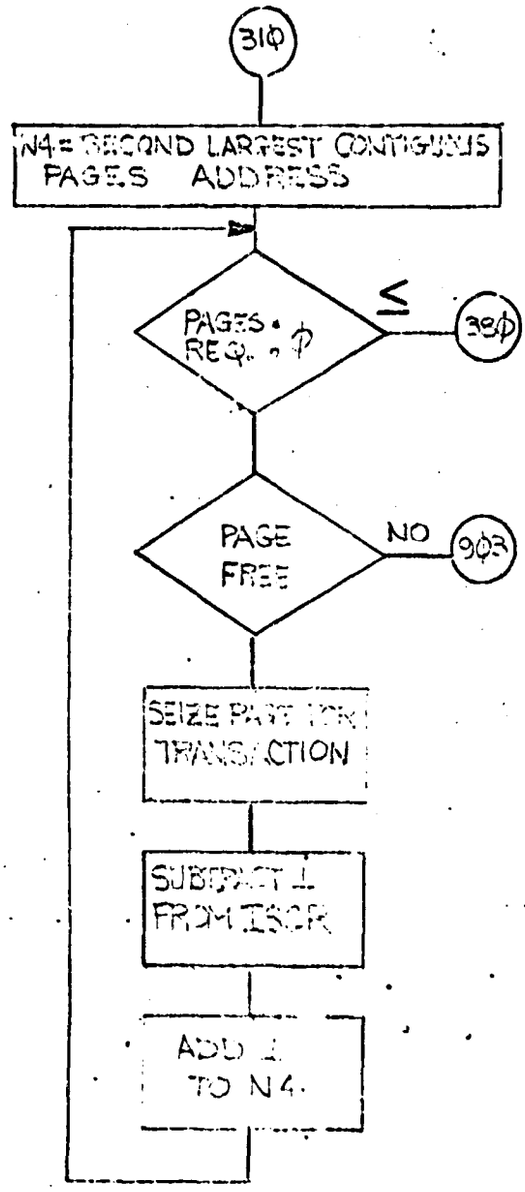

S32 ALLOCATE
STORE SUBROUTINE # INTO T2Ø
OPERAND 1 T2Ø
≠
901
=
OPERAND 2 T2Ø
≠
901
=
PACK REQ BUT NOT DONE
YES
9Ø2
NO
AT : T2Ø
≠
901
RESET MEMORY LOAD S21Ø
PACK PERFORMED
NO
9Ø
YES
ZERO PACK CODE T2Ø
N1 = LOW PAGE FIRST MEMORY
SEARCH FIRST ZERO PAGE N1 = FIRST ZERO PAGE
8Ø
80
N1 IN MEMORY RANGE
NO
9Ø3
YES
AT REENTRANT
NO
240
YES
AT ACTIVE
NO
240
YES
A = INSTRUCTION PAGES
A = A (ROUNDED UP)
IA1P = A
IA1P ILCP (T2Ø
>
190
≤
SUBTRACT IA1P FROM ILCR
N2 = ILCA
ILCA = ILCA + IA1P
13Ø

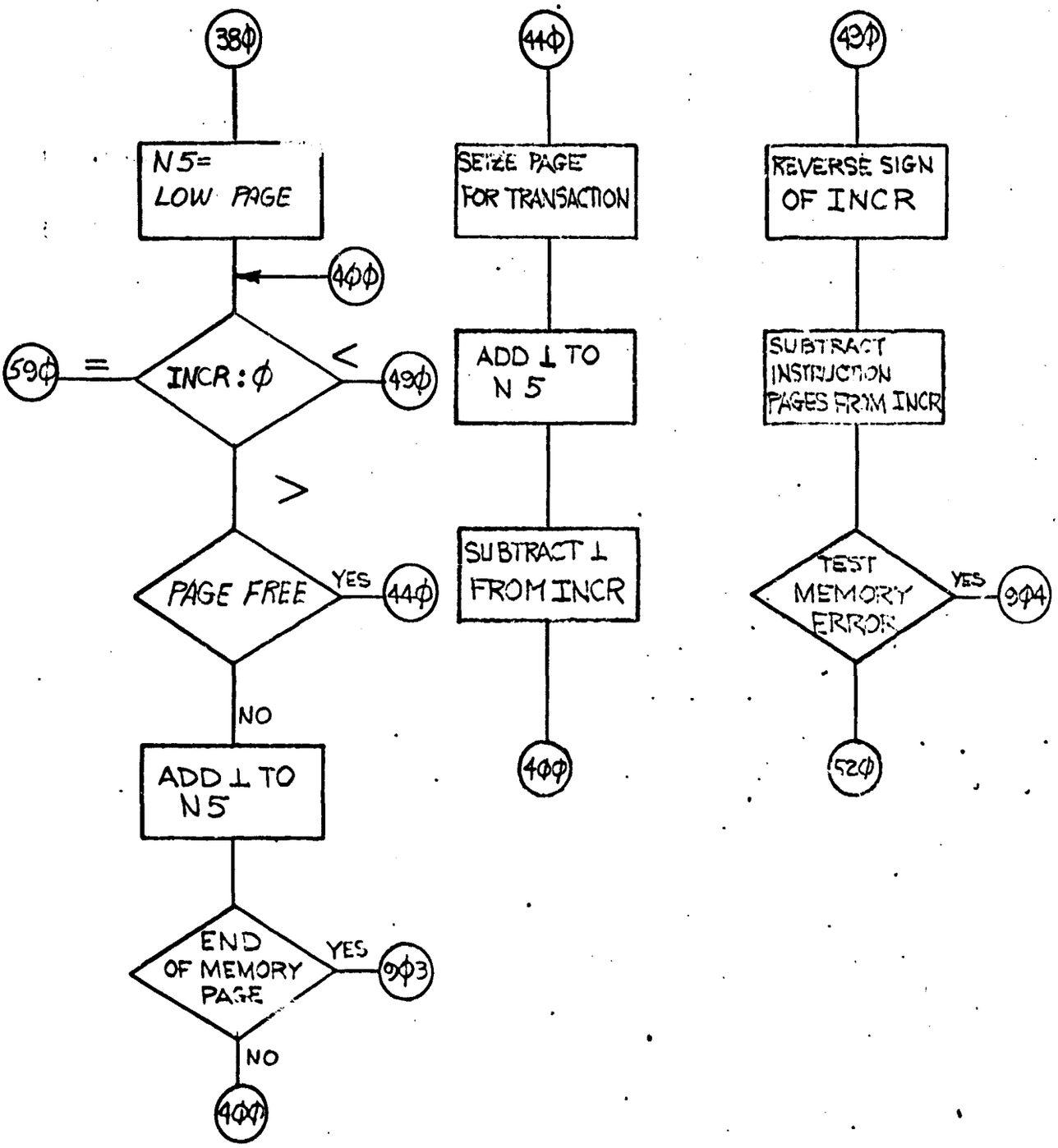
S32 ALLOCATE (CONTINUED)

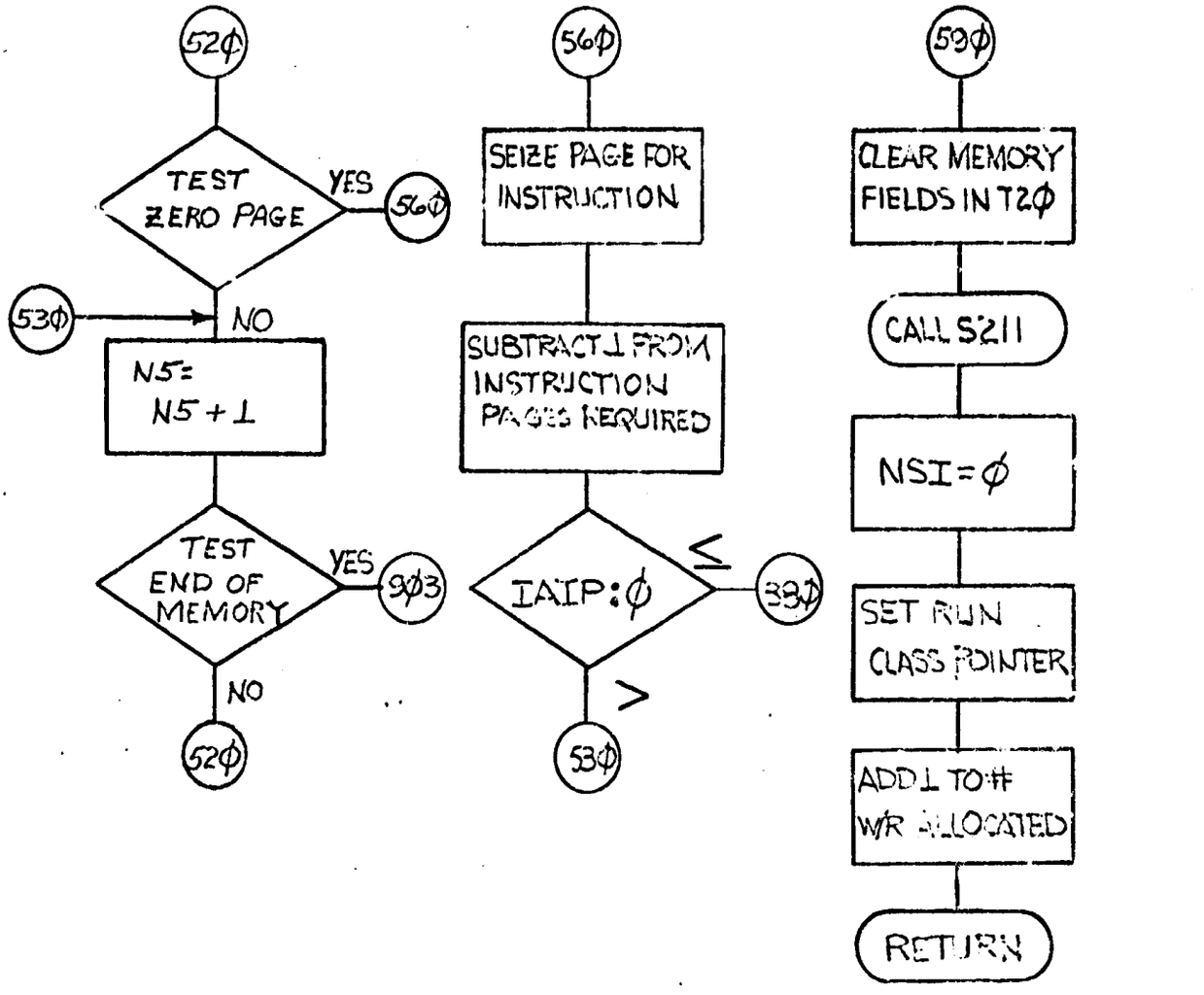


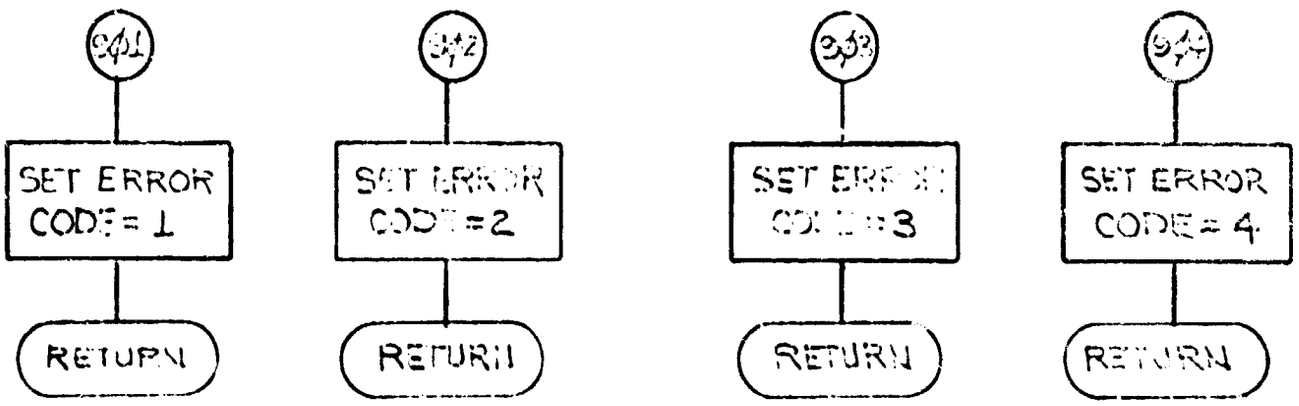
S32 ALLOCATE



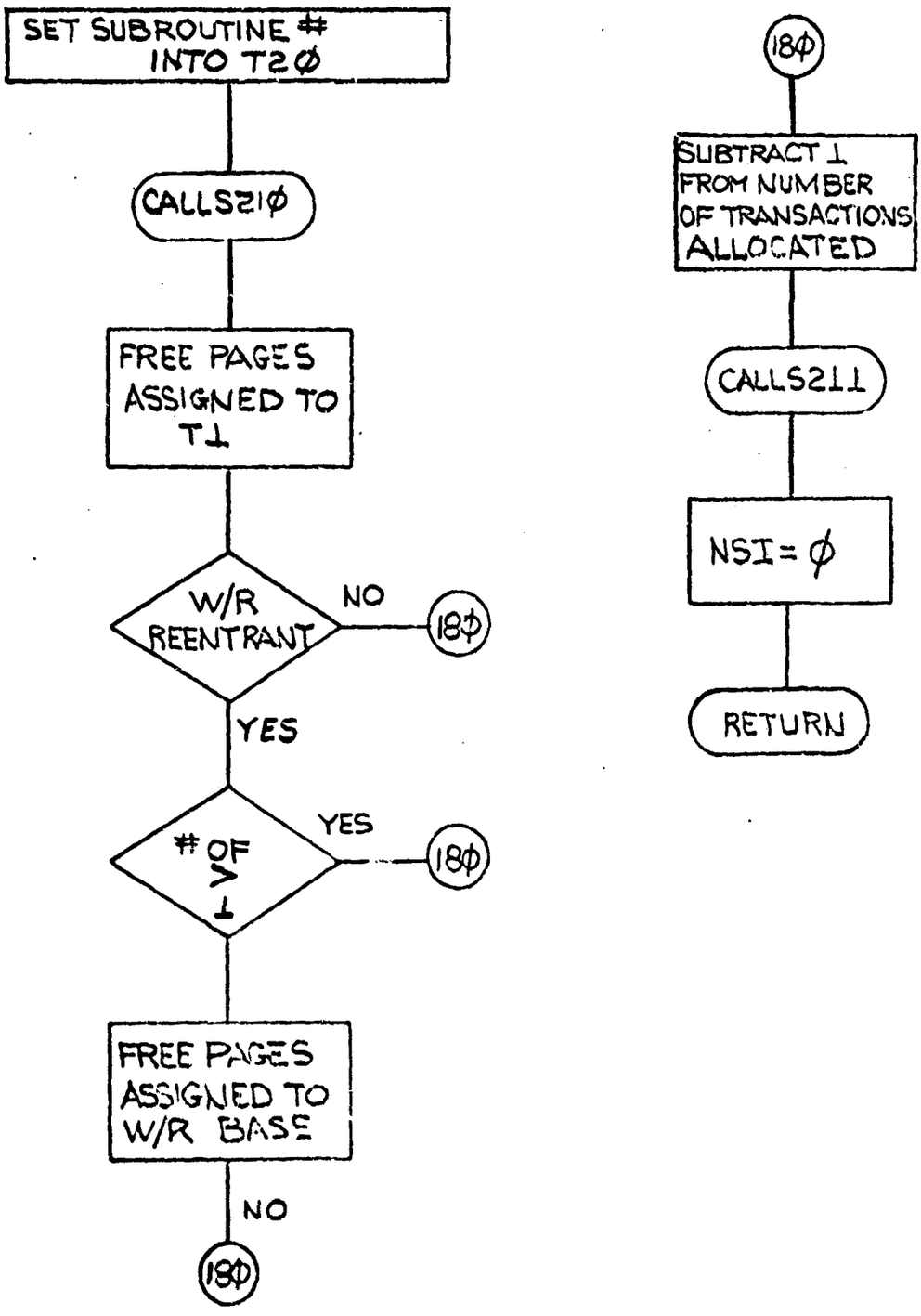



38φ
N5=
LOW PAGE
4φφ
59φ
=
INCR:φ
<
49φ
>
PAGE FREE
YES
44φ
NO
ADD ⊥ TO
N5
END
OF MEMORY
PAGE
YES
9φ3
NO
4φφ
44φ
SEIZE PAGE
FOR TRANSACTION
ADD ⊥ TO
N 5
SUBTRACT ⊥
FROM INCR
4φφ
43φ
REVERSE SIGN
OF INCR
SUBTRACT
INSTRUCTION
PAGES FROM INCR
TEST
MEMORY
ERROR
YES
9φ4
52φ

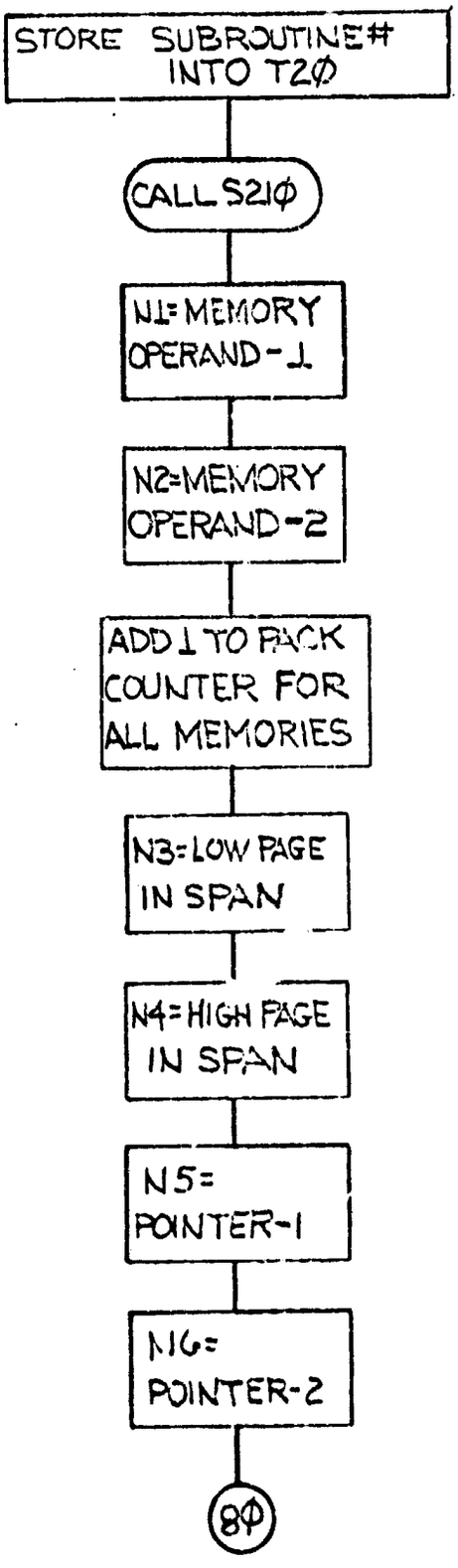

52Φ
TEST ZERO PAGE
YES
56Φ
53Φ
NO
N5= N5+1
TEST END OF MEMORY
YES
90Φ3
NO
52Φ
56Φ
SEIZE PAGE FOR INSTRUCTION
SUBTRACT 1 FROM INSTRUCTION PAGES REQUIRED
IAIP:Φ
≤
33Φ
>
53Φ
59Φ
CLEAR MEMORY FIELDS IN T2Φ
CALL S211
NSI=Φ
SET RUN CLASS POINTER
ADD 1 TO # W/R ALLOCATED
RETURN

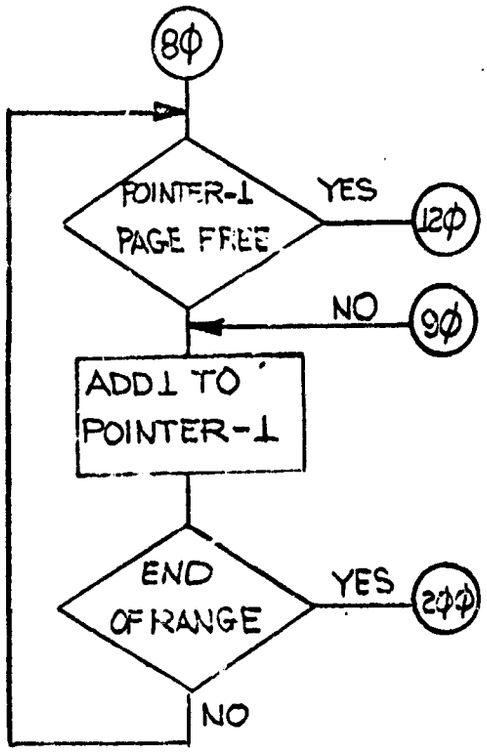

9Φ1
SET ERROR CODE=1
RETURN
9Φ2
SET ERROR CODE=2
RETURN
9Φ3
SET ERROR CODE=3
RETURN
9Φ4
SET ERROR CODE=4
RETURN

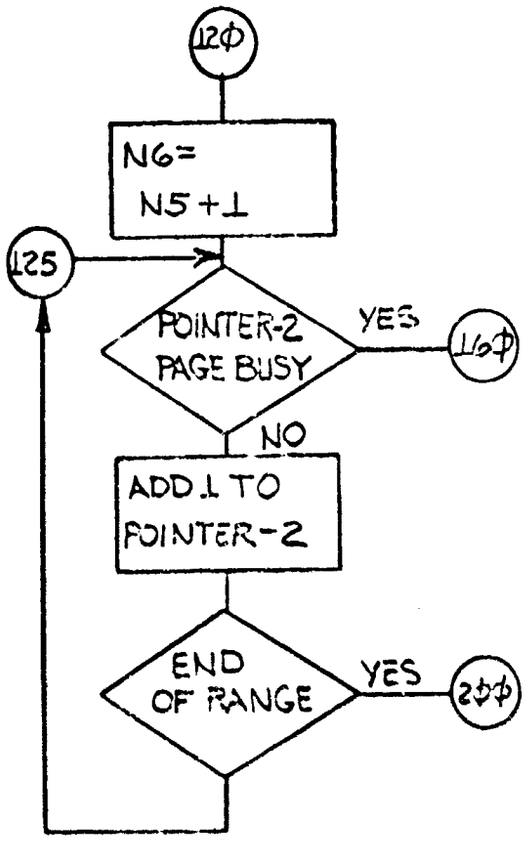
S33 DEALLOCATE



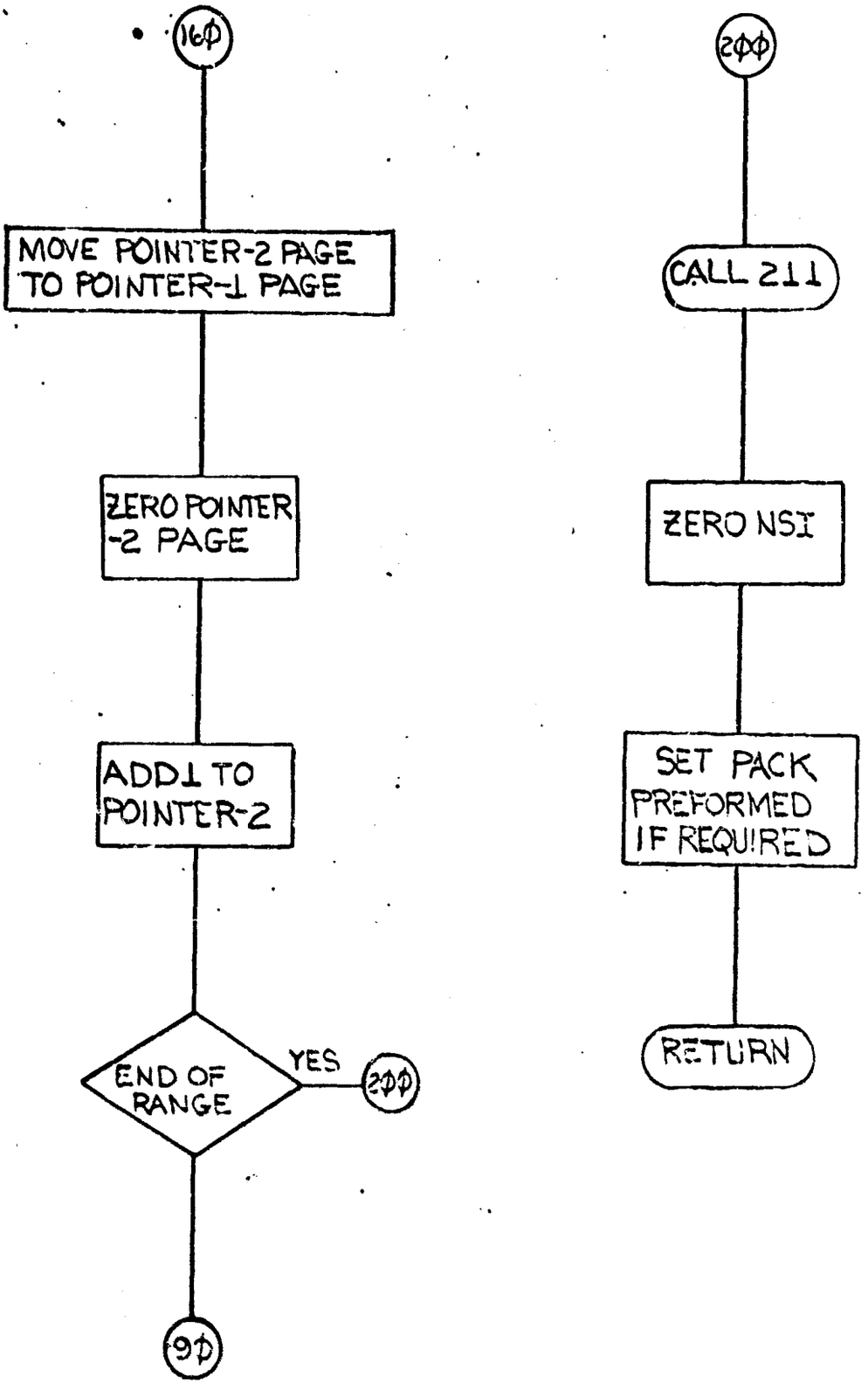
S34 PACK

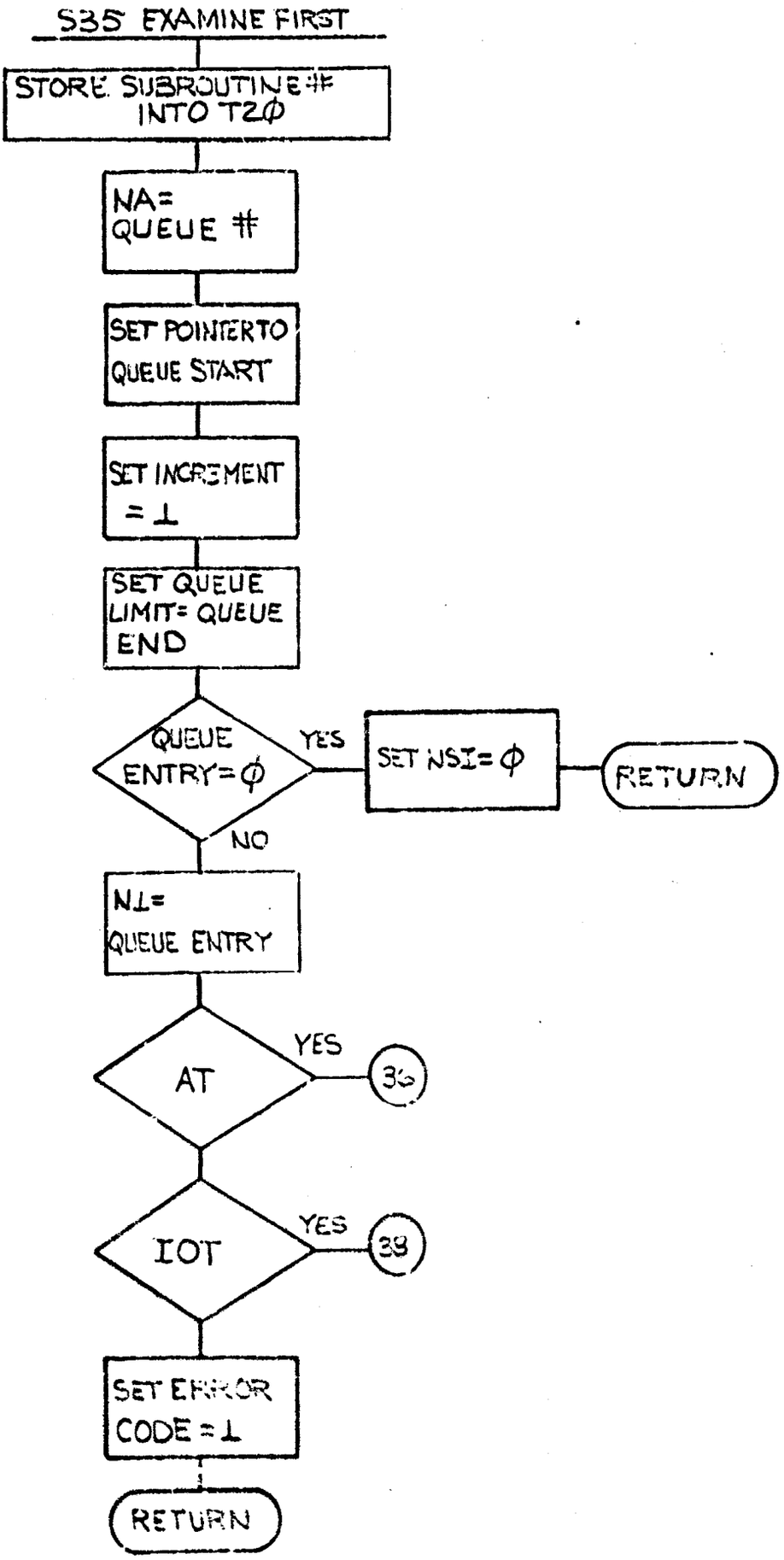

STORE SUBROUTINE# INTO T2Ø
CALL S21Ø
N1= MEMORY OPERAND-1
N2=MEMORY OPERAND-2
ADD 1 TO PACK COUNTER FOR ALL MEMORIES
N3=LOW PAGE IN SPAN
N4=HIGH PAGE IN SPAN
N5= POINTER-1
N6= POINTER-2
8Ø


8Ø
POINTER-1 PAGE FREE
YES
12Ø
NO
9Ø
ADD 1 TO POINTER-1
END OF RANGE
YES
2ØØ
NO

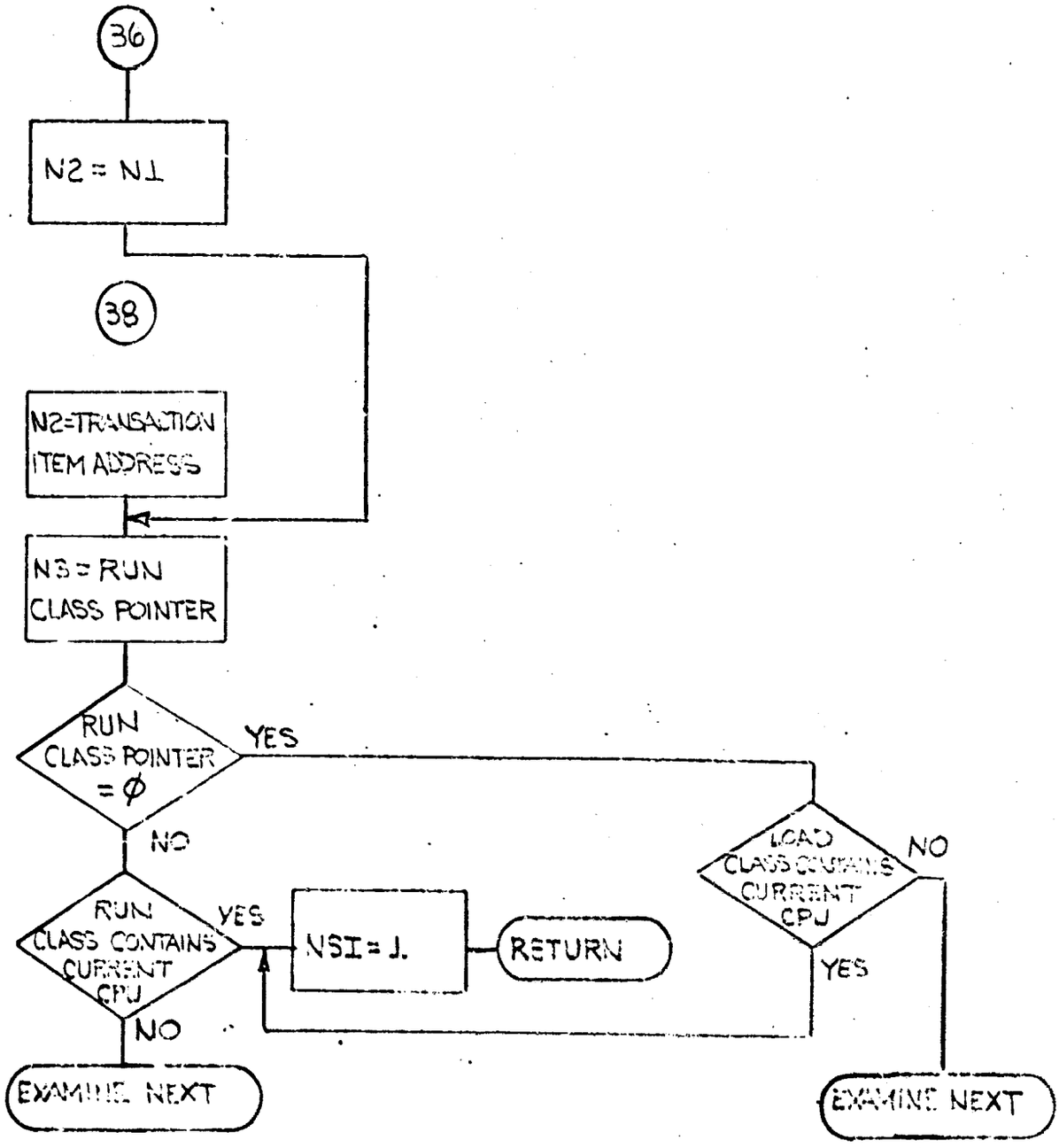

12Ø
N6= N5+1
125
POINTER-2 PAGE BUSY
YES
16Ø
NO
ADD 1 TO POINTER-2
END OF RANGE
YES
2ØØ

S34 PACK (CONTINUED)

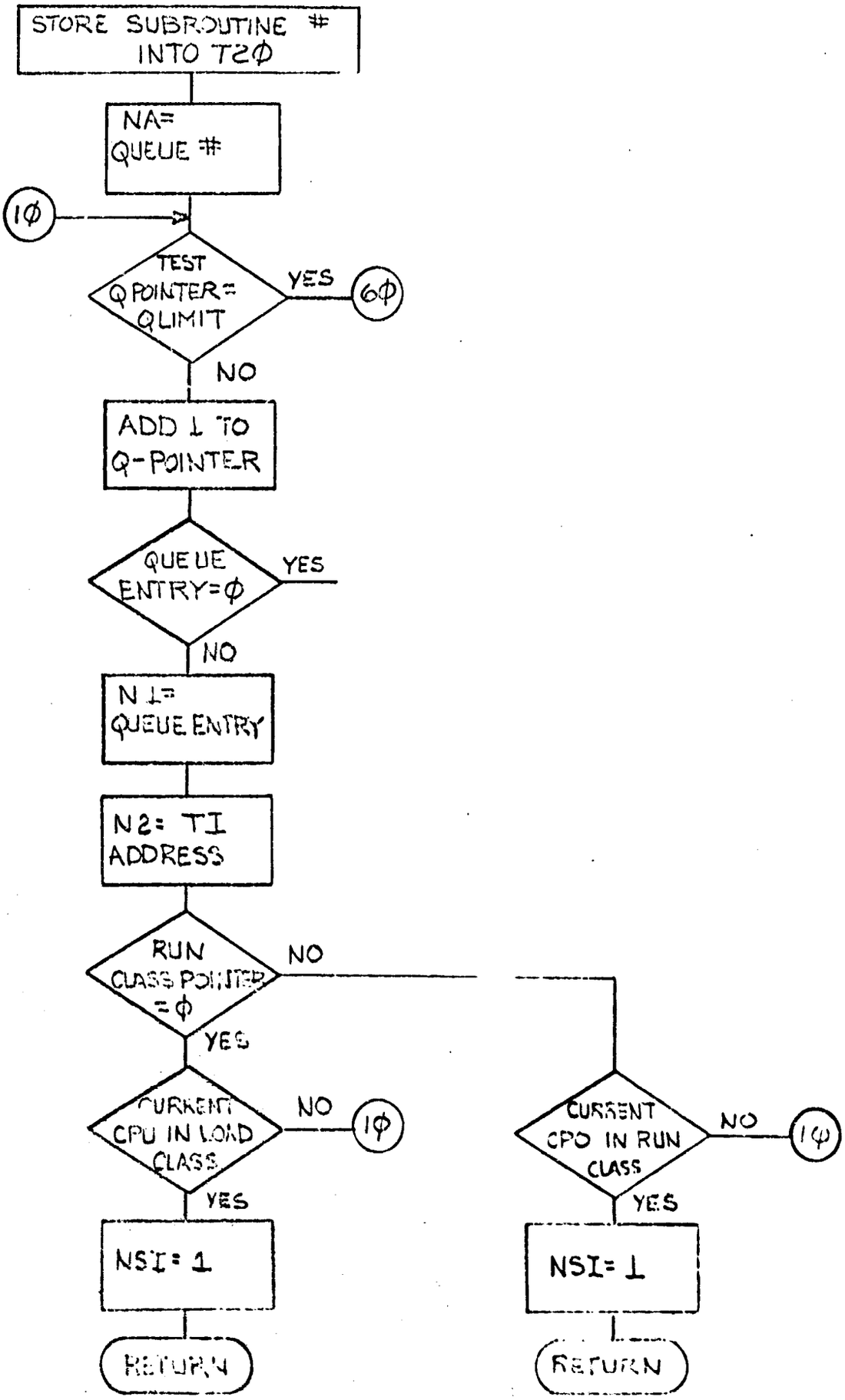


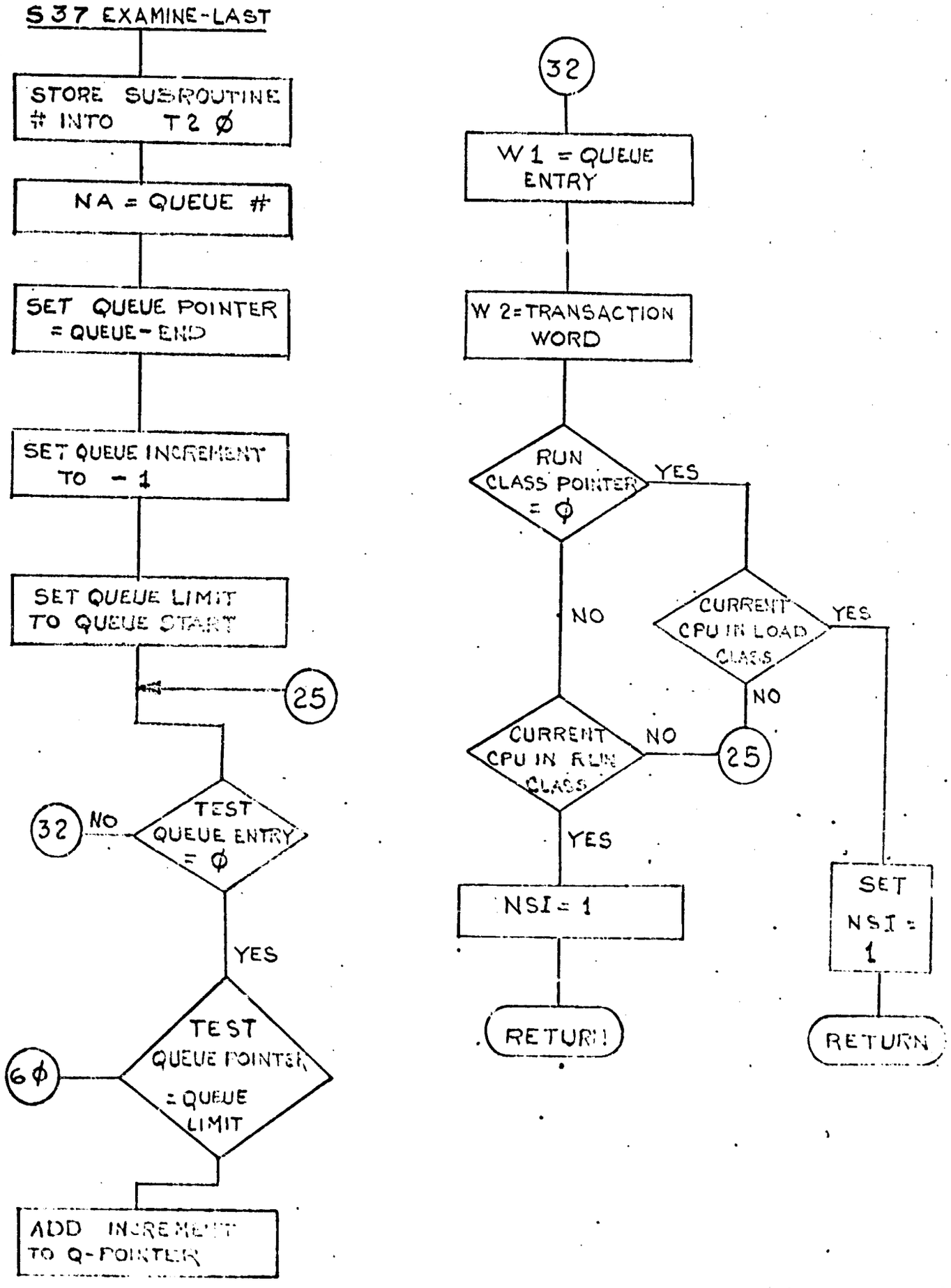

S35 EXAMINE FIRST
STORE SUBROUTINE # INTO T2Ø
NA= QUEUE #
SET POINTER TO QUEUE START
SET INCREMENT = 1
SET QUEUE LIMIT= QUEUE END
QUEUE ENTRY=Ø
YES
SET NSI=Ø
RETURN
NO
NT= QUEUE ENTRY
AT
YES
36
IOT
YES
38
SET ERROR CODE = 1
RETURN

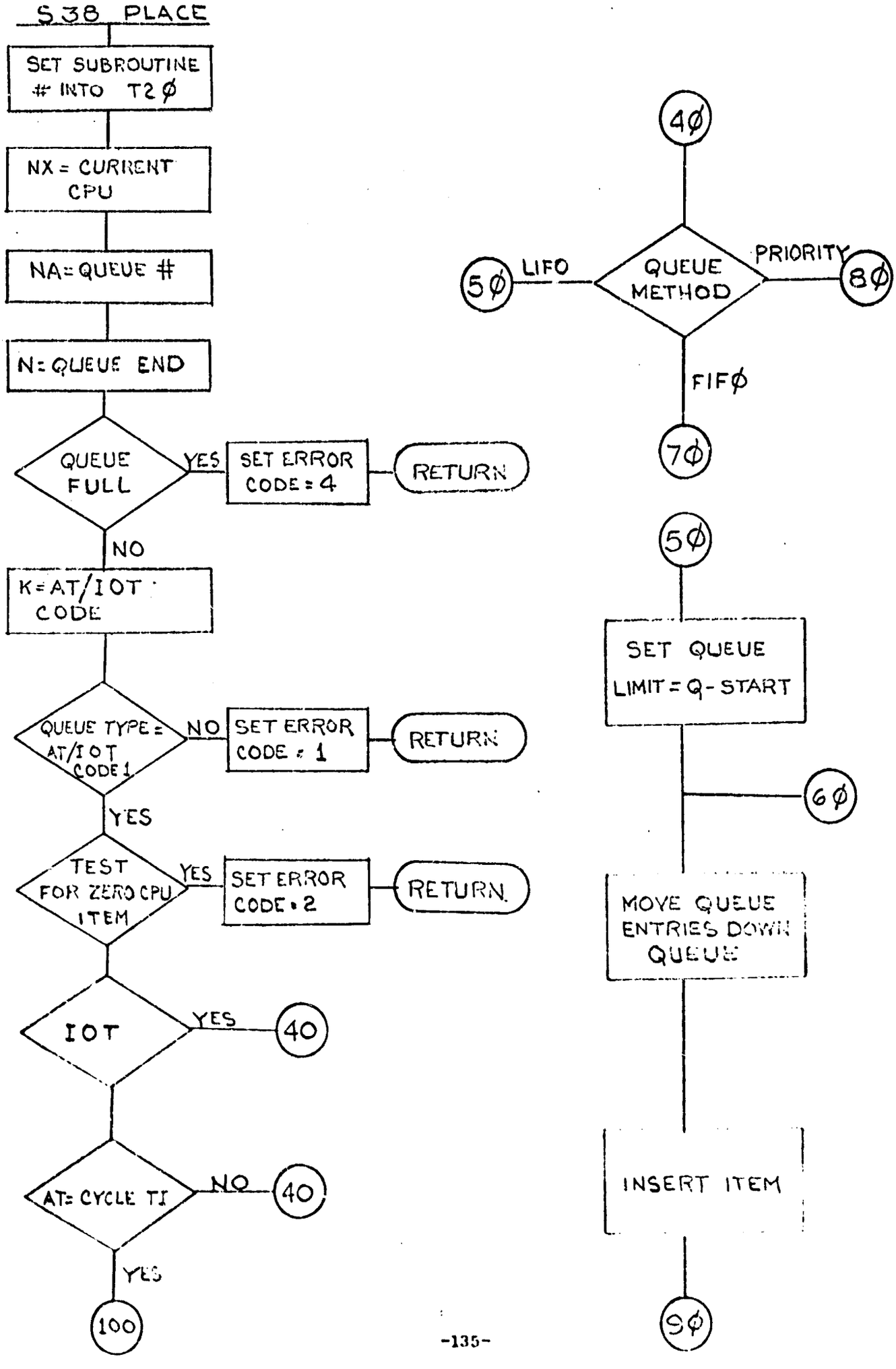
S35


36
N2 = N1
38
N2=TRANSACTION ITEM ADDRESS
N3 = RUN CLASS POINTER
RUN CLASS POINTER = ∅
YES
NO
RUN CLASS CONTAINS CURRENT CPU
YES
NSI = J.
RETURN
NO
EXAMINE NEXT
LOAD CLASS CONTAINS CURRENT CPU
NO
YES
EXAMINE NEXT

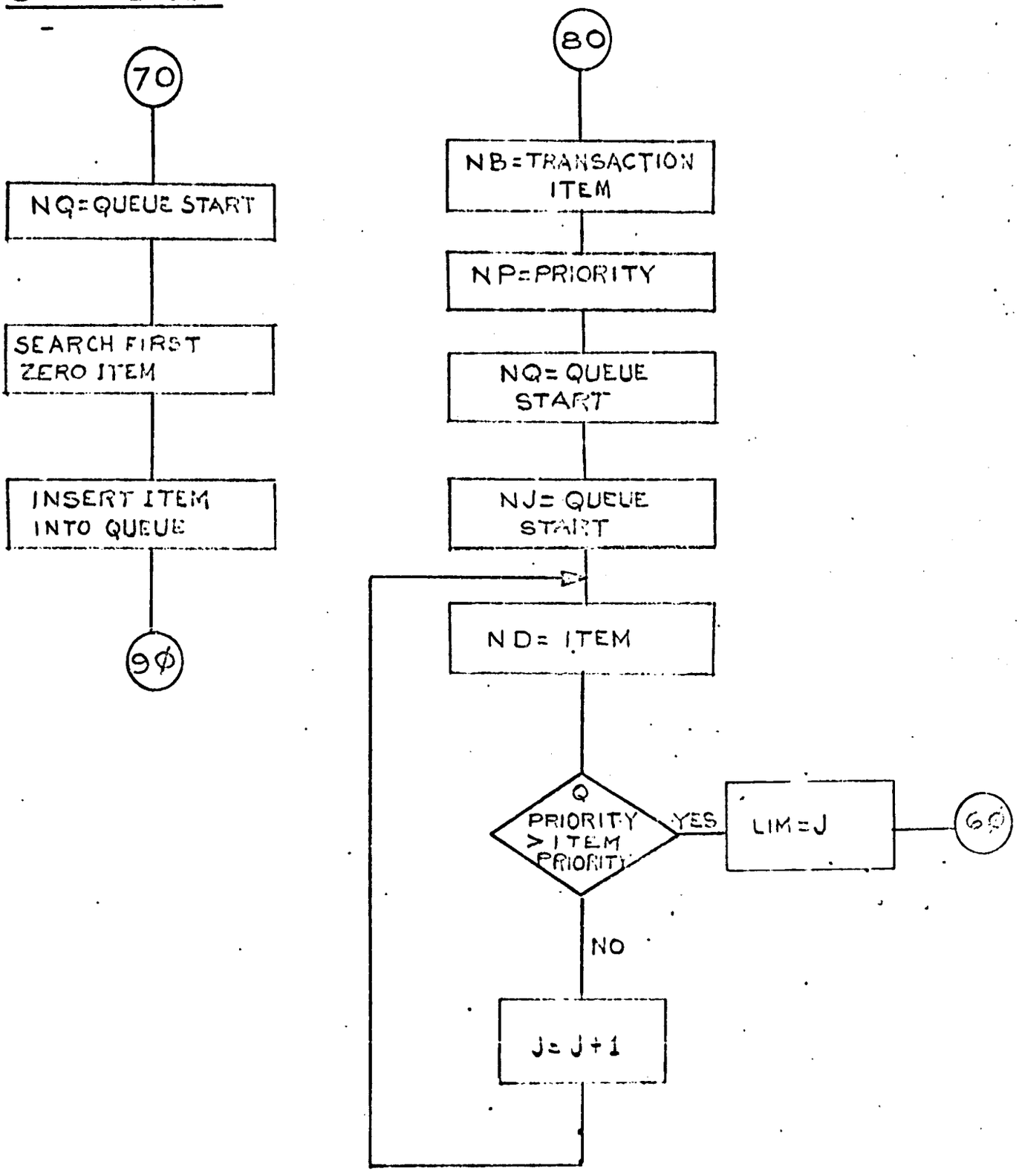
S36 EXAMINE NEXT



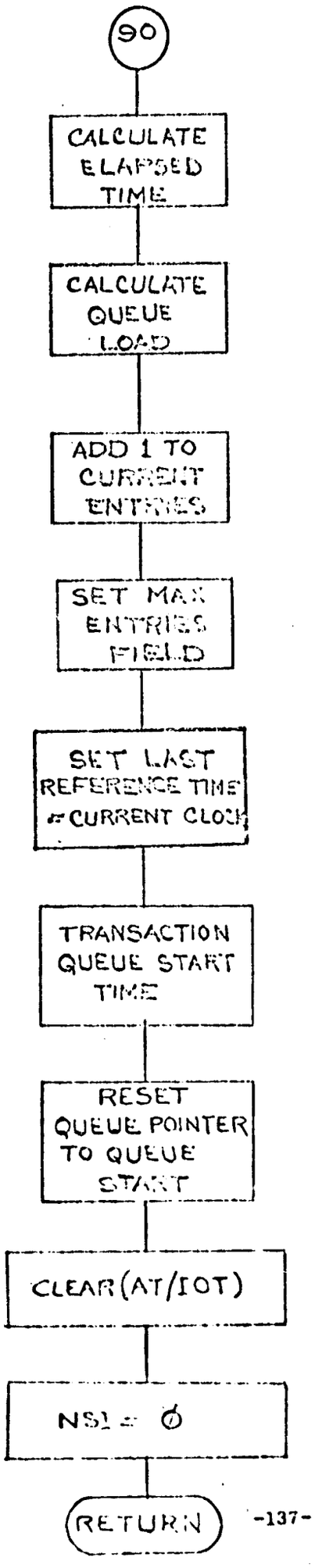

S37 EXAMINE-LAST
STORE SUBROUTINE # INTO T2 Ø
NA = QUEUE #
SET QUEUE POINTER = QUEUE-END
SET QUEUE INCREMENT TO -1
SET QUEUE LIMIT TO QUEUE START
25
32
NO
TEST QUEUE ENTRY = Ø
YES
TEST QUEUE POINTER = QUEUE LIMIT
6Ø
ADD INCREMENT TO Q-POINTER
32
W1 = QUEUE ENTRY
W2=TRANSACTION WORD
RUN CLASS POINTER = Ø
YES
NO
CURRENT CPU IN LOAD CLASS
YES
NO
CURRENT CPU IN RUN CLASS
NO
25
YES
NSI = 1
RETURN
SET NSI = 1
RETURN

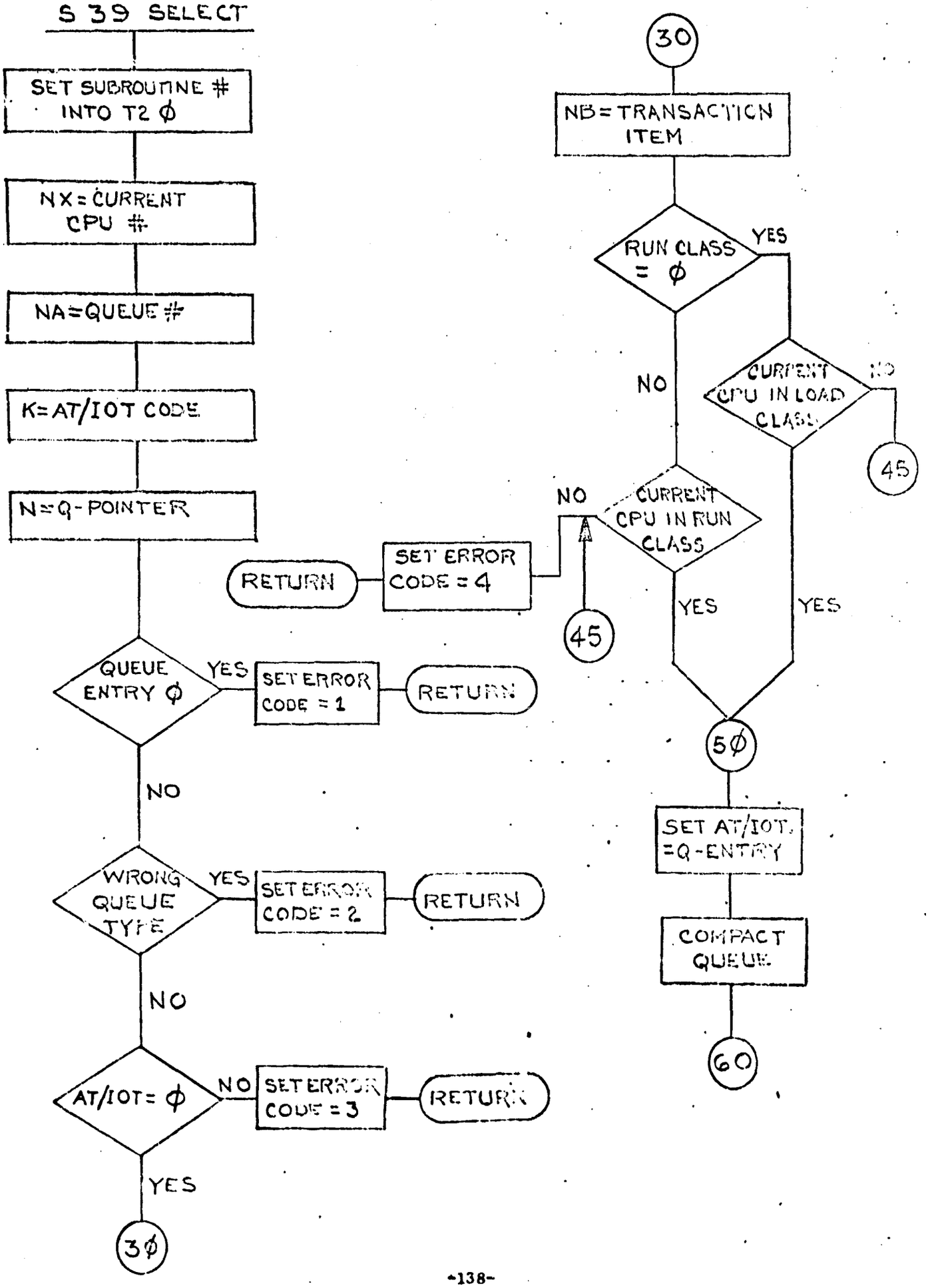

S38 PLACE
SET SUBROUTINE # INTO T2Ø
NX = CURRENT CPU
NA=QUEUE #
N=QUEUE END
QUEUE FULL
YES
SET ERROR CODE=4
RETURN
NO
K=AT/IOT CODE
QUEUE TYPE = AT/IOT CODE 1
NO
SET ERROR CODE = 1
RETURN
YES
TEST FOR ZERO CPU ITEM
YES
SET ERROR CODE=2
RETURN.
IOT
YES
40
AT= CYCLE TI
NO
40
YES
100
4Ø
QUEUE METHOD
LIFO
5Ø
PRIORITY
8Ø
FIFØ
7Ø
5Ø
SET QUEUE LIMIT = Q-START
6Ø
MOVE QUEUE ENTRIES DOWN QUEUE
INSERT ITEM
9Ø

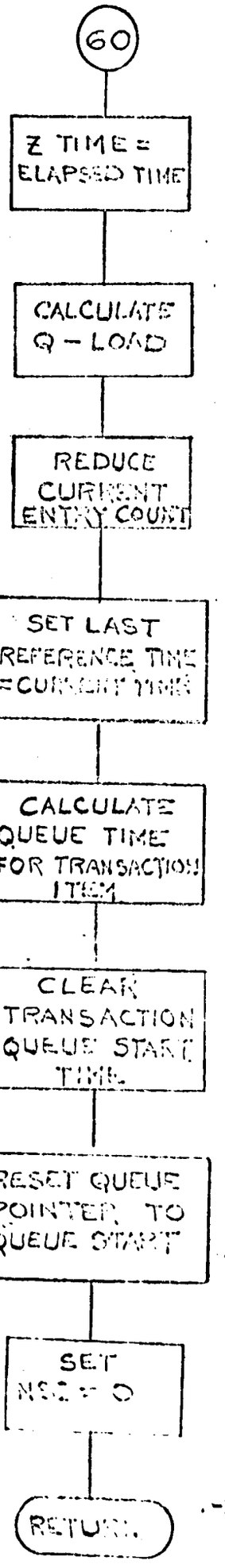
S 38 PLACE

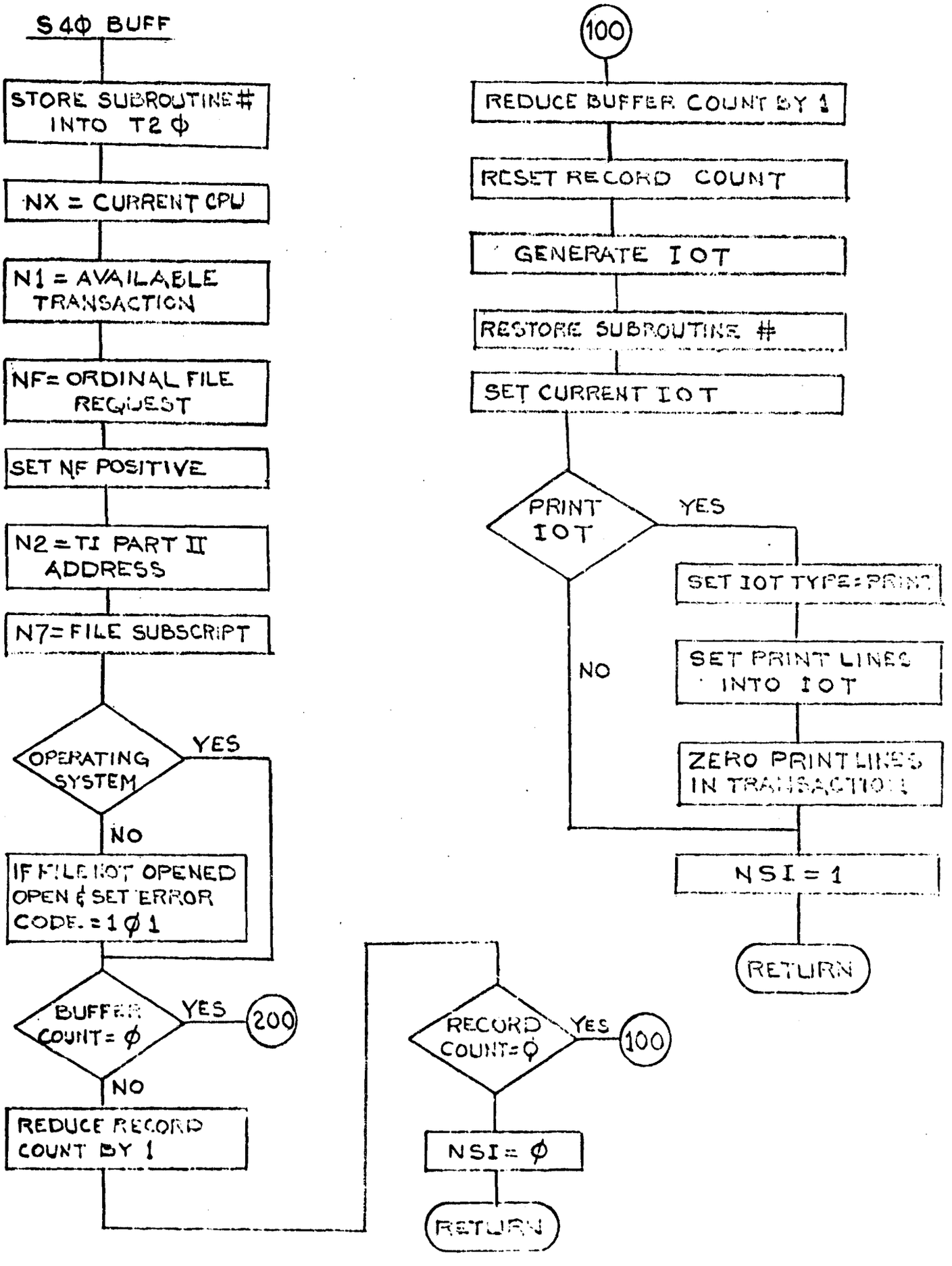

70
NQ=QUEUE START
SEARCH FIRST ZERO ITEM
INSERT ITEM INTO QUEUE
90
80
NB=TRANSACTION ITEM
NP=PRIORITY
NQ=QUEUE START
NJ=QUEUE START
ND=ITEM
Q PRIORITY > ITEM PRIORITY
YES
LIM=J
60
NO
J=J+1

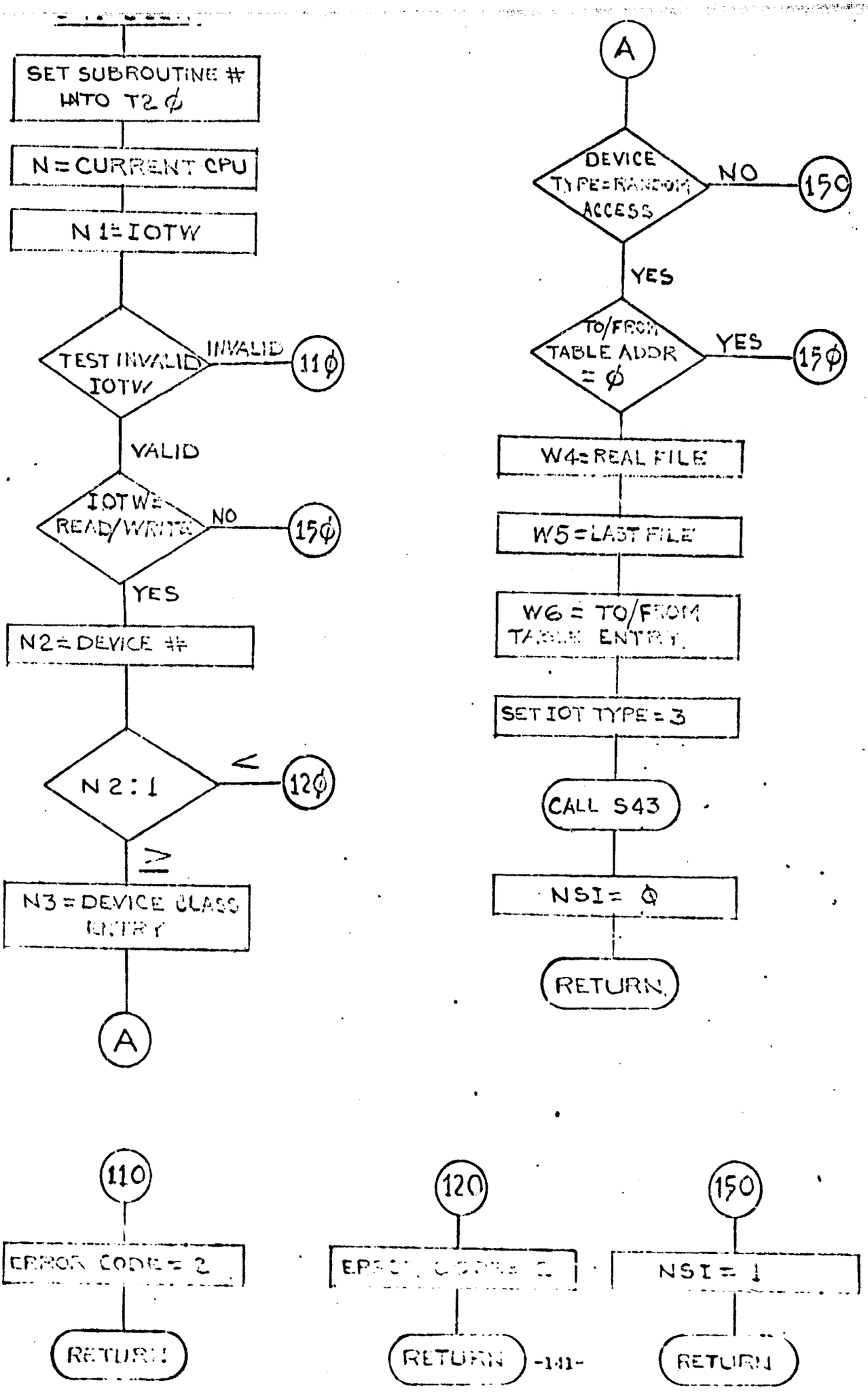
S 38 PLACE

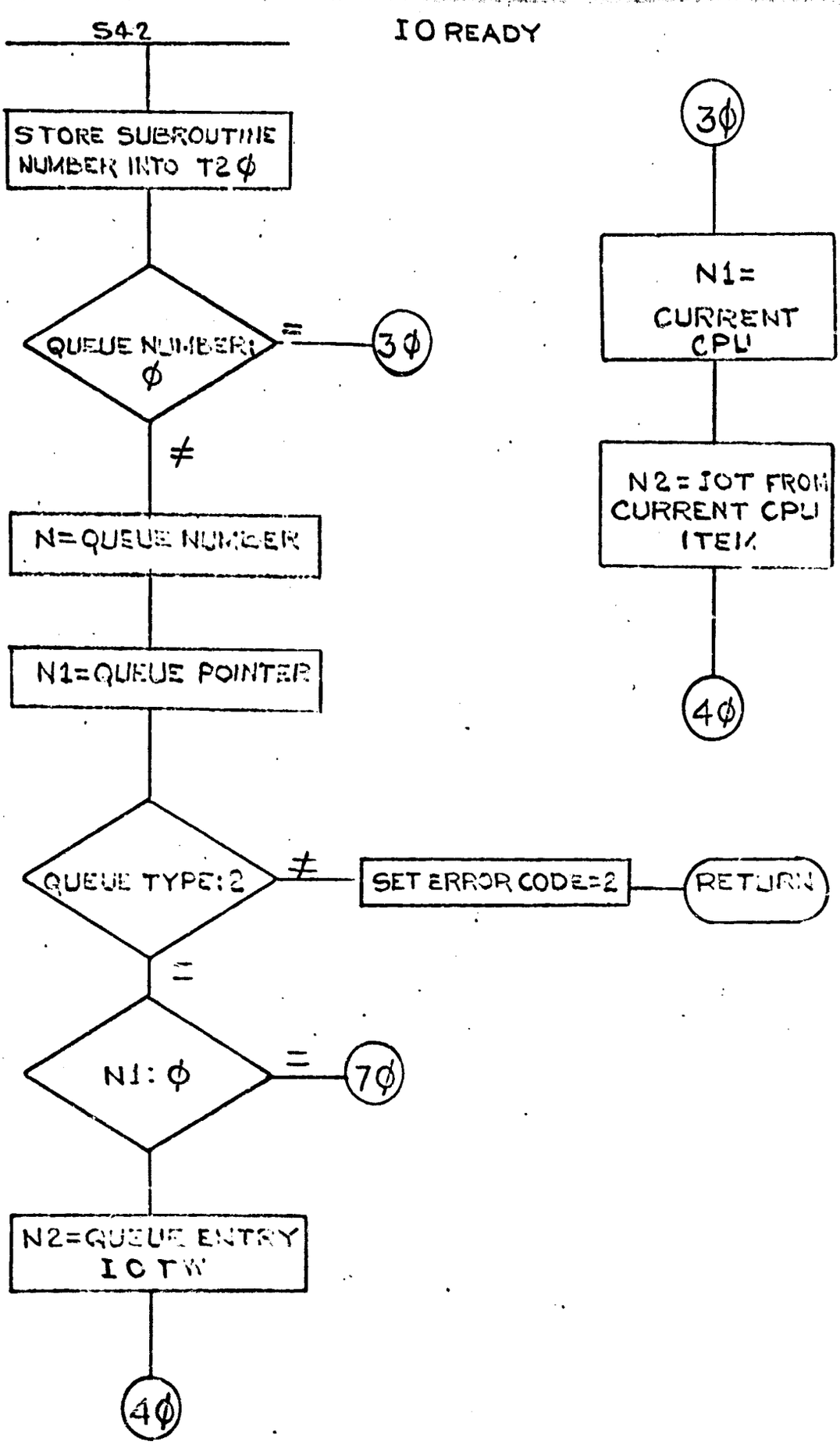

90
CALCULATE ELAPSED TIME
CALCULATE QUEUE LOAD
ADD 1 TO CURRENT ENTRIES
SET MAX ENTRIES FIELD
SET LAST REFERENCE TIME = CURRENT CLOCK
TRANSACTION QUEUE START TIME
RESET QUEUE POINTER TO QUEUE START
CLEAR (AT/IOT)
NSI = 0
RETURN

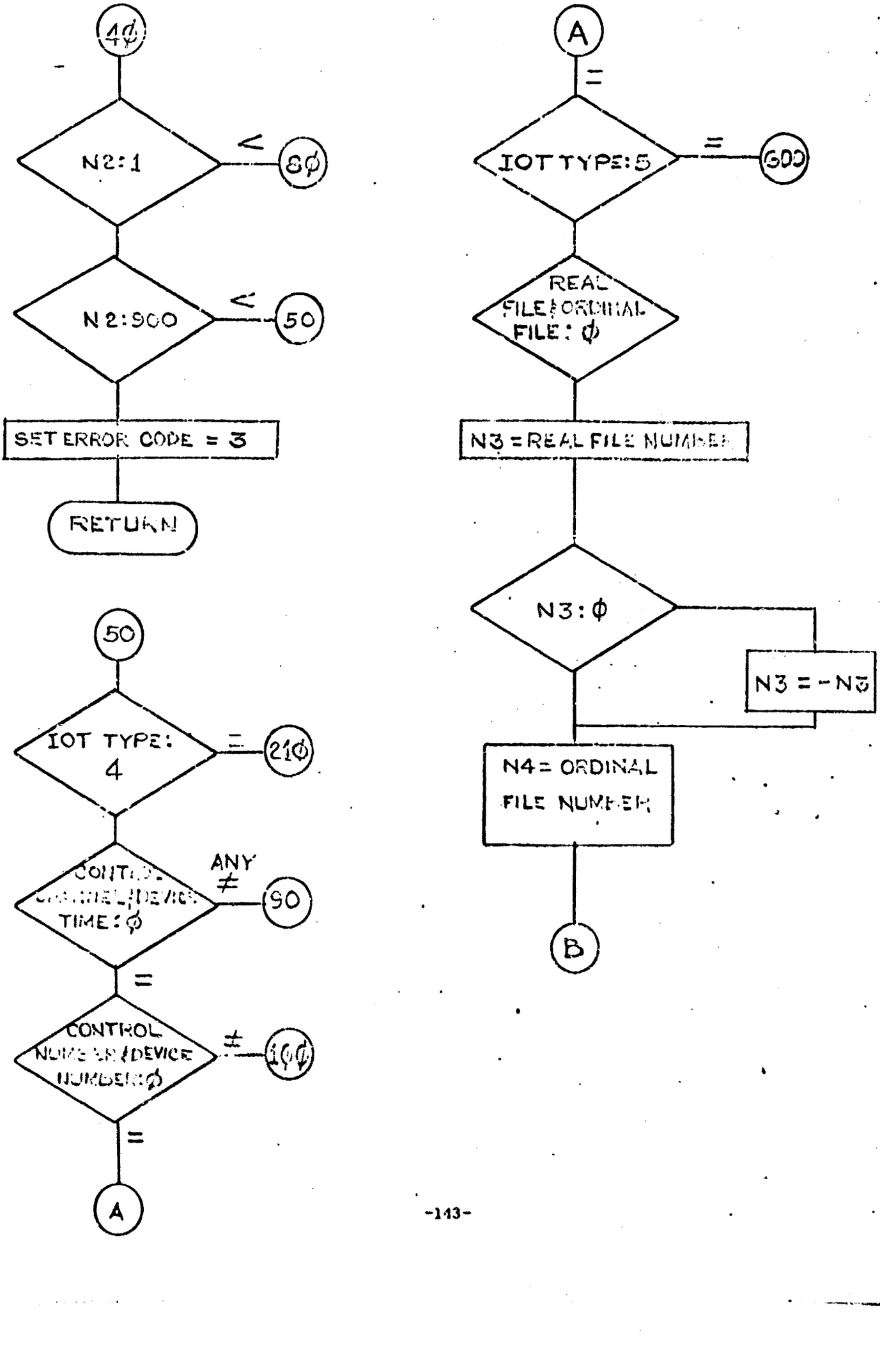

S 39 SELECT
SET SUBROUTINE # INTO T2 Ø
NX = CURRENT CPU #
NA = QUEUE #
K = AT/IOT CODE
N = Q-POINTER
QUEUE ENTRY Ø
YES
SET ERROR CODE = 1
RETURN
NO
WRONG QUEUE TYPE
YES
SET ERROR CODE = 2
RETURN
NO
AT/IOT = Ø
NO
SET ERROR CODE = 3
RETURN
YES
3Ø
30
NB = TRANSACTION ITEM
RUN CLASS = Ø
YES
NO
CURRENT CPU IN LOAD CLASS
NO
45
NO
CURRENT CPU IN RUN CLASS
SET ERROR CODE = 4
RETURN
45
YES
YES
5Ø
SET AT/IOT = Q-ENTRY
COMPACT QUEUE
6Ø

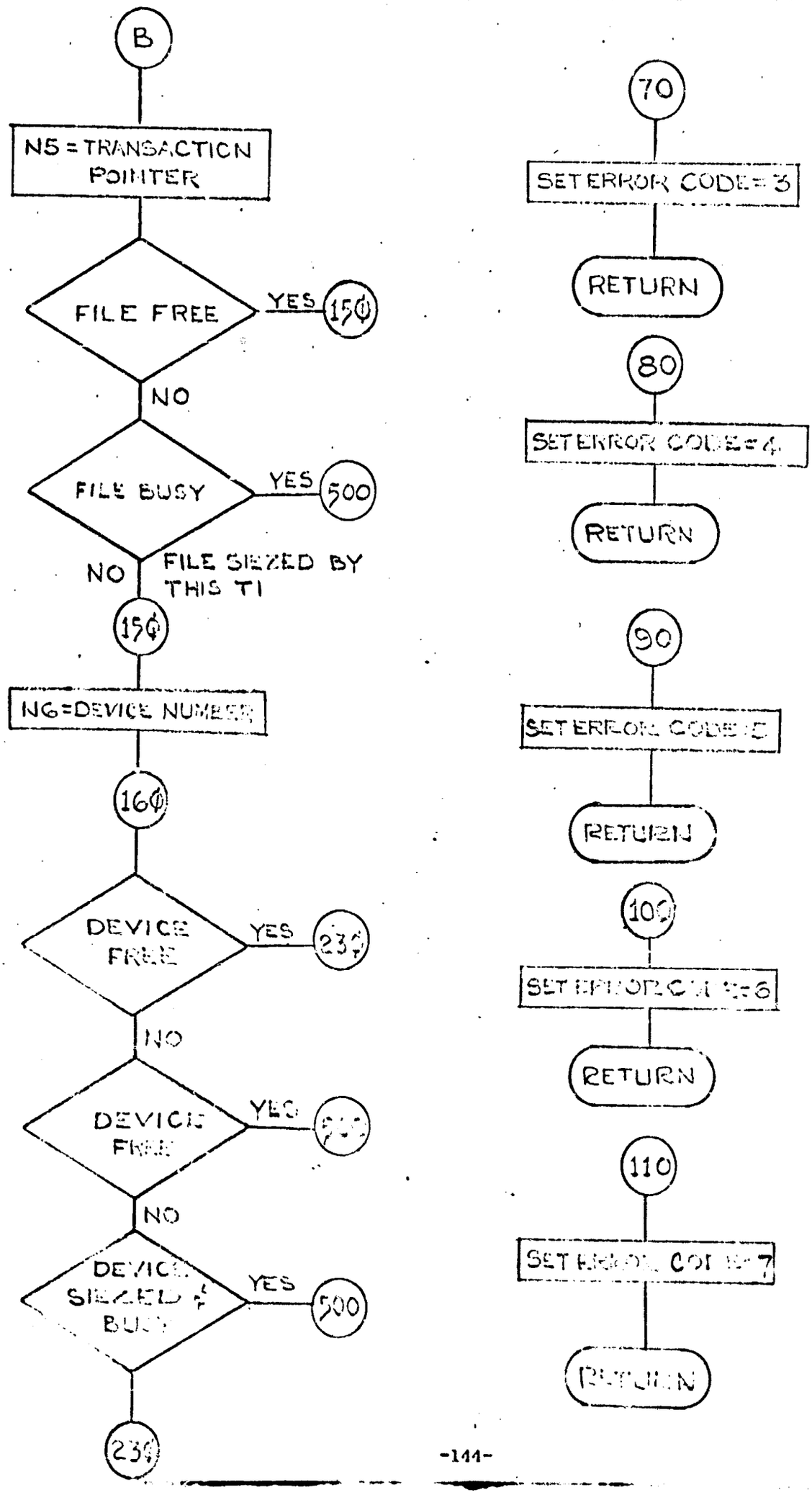

60
Z TIME = ELAPSED TIME
CALCULATE Q - LOAD
REDUCE CURRENT ENTRY COUNT
SET LAST REFERENCE TIME = CURRENT TIME
CALCULATE QUEUE TIME FOR TRANSACTION ITEM
CLEAR TRANSACTION QUEUE START TIME
RESET QUEUE POINTER TO QUEUE START
SET
RETURN

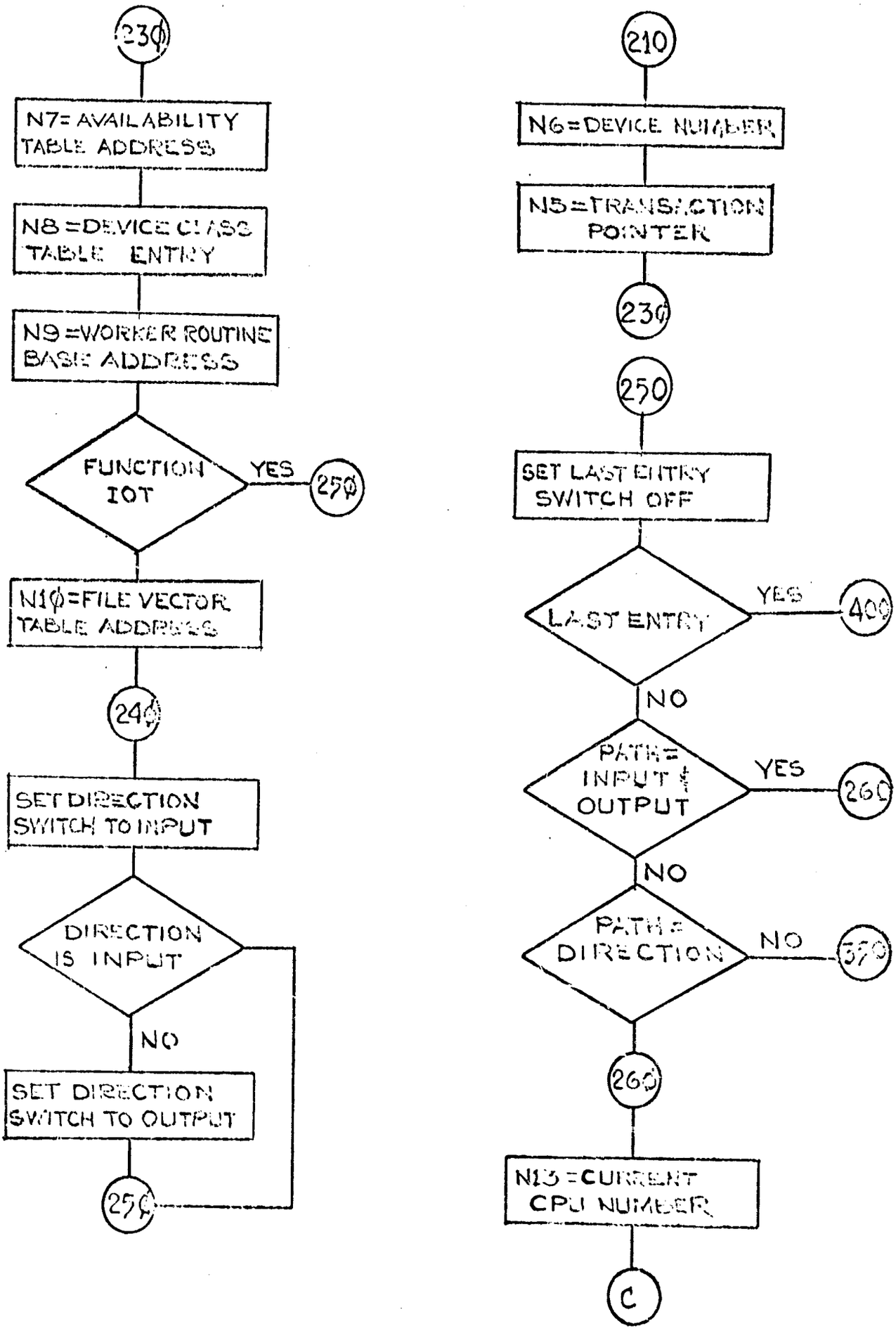

S4Φ BUFF
STORE SUBROUTINE # INTO T2 Φ
NX = CURRENT CPU
N1 = AVAILABLE TRANSACTION
NF= ORDINAL FILE REQUEST
SET NF POSITIVE
N2 = TI PART II ADDRESS
N7= FILE SUBSCRIPT
OPERATING SYSTEM
YES
NO
IF FILE NOT OPENED OPEN & SET ERROR CODE = 1Φ1
BUFFER COUNT = Φ
YES
200
NO
REDUCE RECORD COUNT BY 1
100
REDUCE BUFFER COUNT BY 1
RESET RECORD COUNT
GENERATE IOT
RESTORE SUBROUTINE #
SET CURRENT IOT
PRINT IOT
YES
NO
SET IOT TYPE=PRINT
SET PRINT LINES INTO IOT
ZERO PRINT LINES IN TRANSACTION
NSI = 1
RETURN
RECORD COUNT=Φ
YES
100
NSI = Φ
RETURN

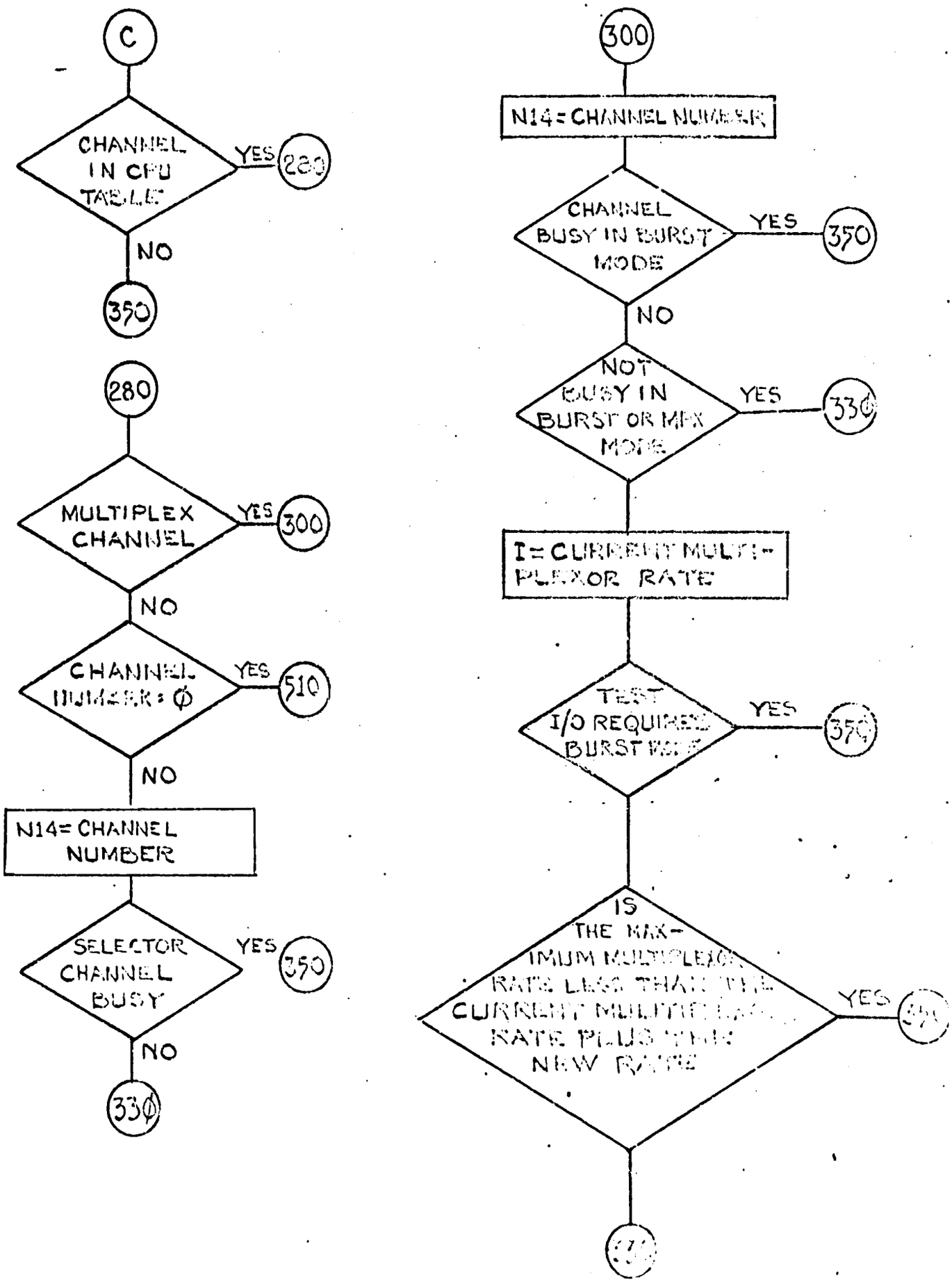

SET SUBROUTINE # INTO T2 ϕ
N = CURRENT CPU
N1 = IOTW
TEST INVALID IOTW
INVALID
11ϕ
VALID
IOTW = READ/WRITE
NO
15ϕ
YES
N2 = DEVICE #
N2 : 1
<
12ϕ
≥
N3 = DEVICE CLASS ENTRY
A
A
DEVICE TYPE = RANDOM ACCESS
NO
15C
YES
TO/FROM TABLE ADDR = ϕ
YES
15ϕ
W4 = REAL FILE
W5 = LAST FILE
W6 = TO/FROM TABLE ENTRY
SET IOT TYPE = 3
CALL S43
NSI = ϕ
RETURN
110
ERROR CODE = 2
RETURN
120
RETURN
150
NSI = 1
RETURN

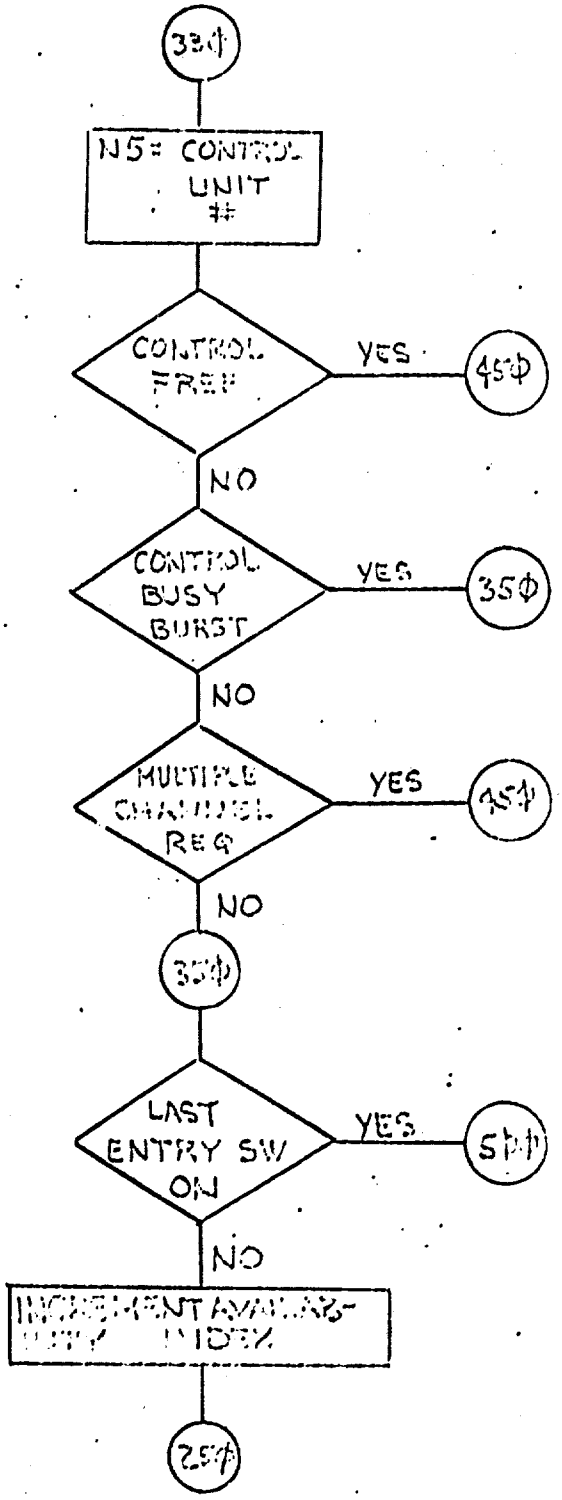

S42
IO READY
STORE SUBROUTINE NUMBER INTO T2Ø
QUEUE NUMBER: Ø
=
3Ø
≠
N=QUEUE NUMBER
N1=QUEUE POINTER
QUEUE TYPE: 2
≠
SET ERROR CODE=2
RETURN
=
N1: Ø
=
7Ø
N2=QUEUE ENTRY IOTW
4Ø
3Ø
N1= CURRENT CPU
N2 = IOT FROM CURRENT CPU ITEM
4Ø

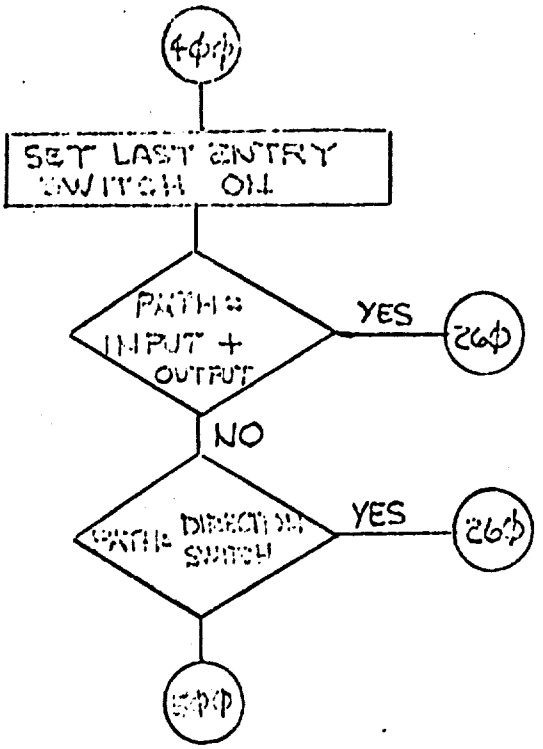

40
N2:1
<
80
N2:900
<
50
SET ERROR CODE = 3
RETURN
50
IOT TYPE: 4
=
210
CONTROL CHANNEL/DEVICE TIME: ϕ
ANY
≠
90
=
CONTROL NUMBER/DEVICE NUMBER: ϕ
≠
100
=
A
A
=
IOT TYPE: 5
=
600
REAL FILE/ORDINAL FILE: ϕ
N3 = REAL FILE NUMBER
N3: ϕ
N3 = -N3
N4 = ORDINAL FILE NUMBER
B

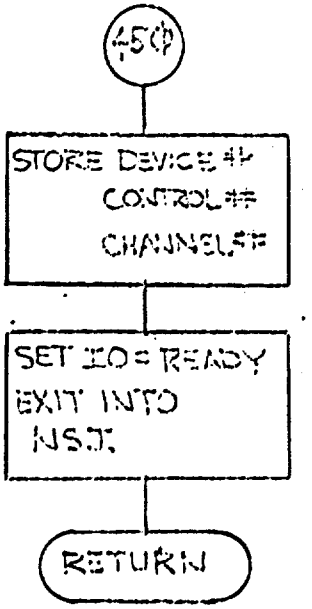

B
N5 = TRANSACTION POINTER
FILE FREE
YES
150
NO
FILE BUSY
YES
500
NO
FILE SIZED BY THIS TI
150
NG=DEVICE NUMBER
160
DEVICE FREE
YES
230
NO
DEVICE FREE
YES
NO
DEVICE SIZED BY
YES
500
230
70
SET ERROR CODE=3
RETURN
80
SET ERROR CODE=4
RETURN
90
SET ERROR CODE=5
RETURN
100
SET ERROR CODE=6
RETURN
110
SET ERROR CODE=7
RETURN

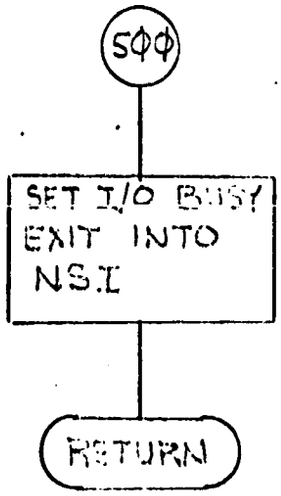

230
N7=AVAILABILITY TABLE ADDRESS
N8=DEVICE CLASS TABLE ENTRY
N9=WORKER ROUTINE BASE ADDRESS
FUNCTION IOT
YES
250
N10=FILE VECTOR TABLE ADDRESS
240
SET DIRECTION SWITCH TO INPUT
DIRECTION IS INPUT
NO
SET DIRECTION SWITCH TO OUTPUT
250
210
N6=DEVICE NUMBER
N5=TRANSACTION POINTER
230
250
SET LAST ENTRY SWITCH OFF
LAST ENTRY
YES
400
NO
PATH=INPUT & OUTPUT
YES
260
NO
PATH=DIRECTION
NO
260
N13=CURRENT CPU NUMBER
C

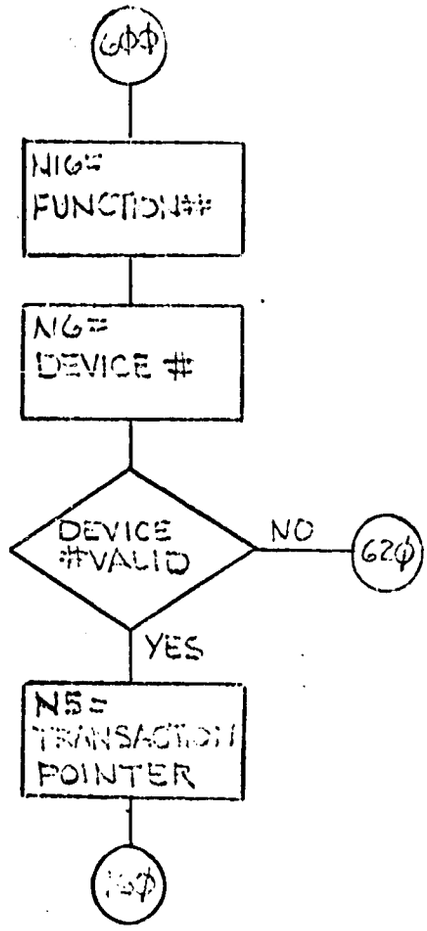

C
CHANNEL IN CPU TABLE
YES
NO
350
280
MULTIPLEX CHANNEL
YES
300
NO
CHANNEL NUMBER = Ø
YES
510
NO
N14 = CHANNEL NUMBER
SELECTOR CHANNEL BUSY
YES
350
NO
330
300
N14 = CHANNEL NUMBER
CHANNEL BUSY IN BURST MODE
YES
350
NO
NOT BUSY IN BURST OR MPX MODE
YES
330
I = CURRENT MULTI-PLEXOR RATE
TEST I/O REQUIRES BURST MODE
YES
370
IS THE MAX-IMUM MULTIPLEXOR RATE LESS THAN THE CURRENT MULTIPLEXOR RATE PLUS THE NEW RATE
YES

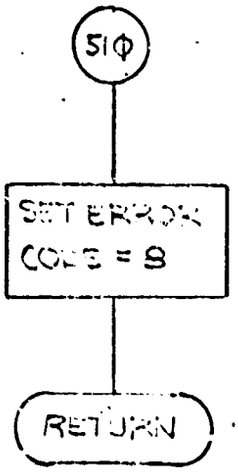

33Φ
N5= CONTROL UNIT #
CONTROL FREE
YES
45Φ
NO
CONTROL BUSY BURST
YES
35Φ
NO
MULTIPLE CHANNEL REQ
YES
45Φ
NO
35Φ
LAST ENTRY SW ON
YES
5ΦΦ
NO
INCREMENT AVAILABILITY INDEX
25Φ

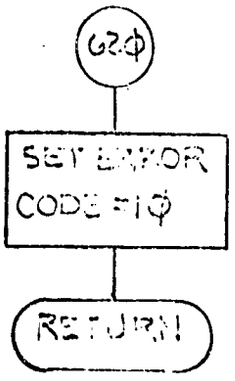

4ΦΦ
SET LAST ENTRY SWITCH ON
PATH= INPUT + OUTPUT
YES
26Φ
NO
PATH= DIRECTION SWITCH
YES
26Φ

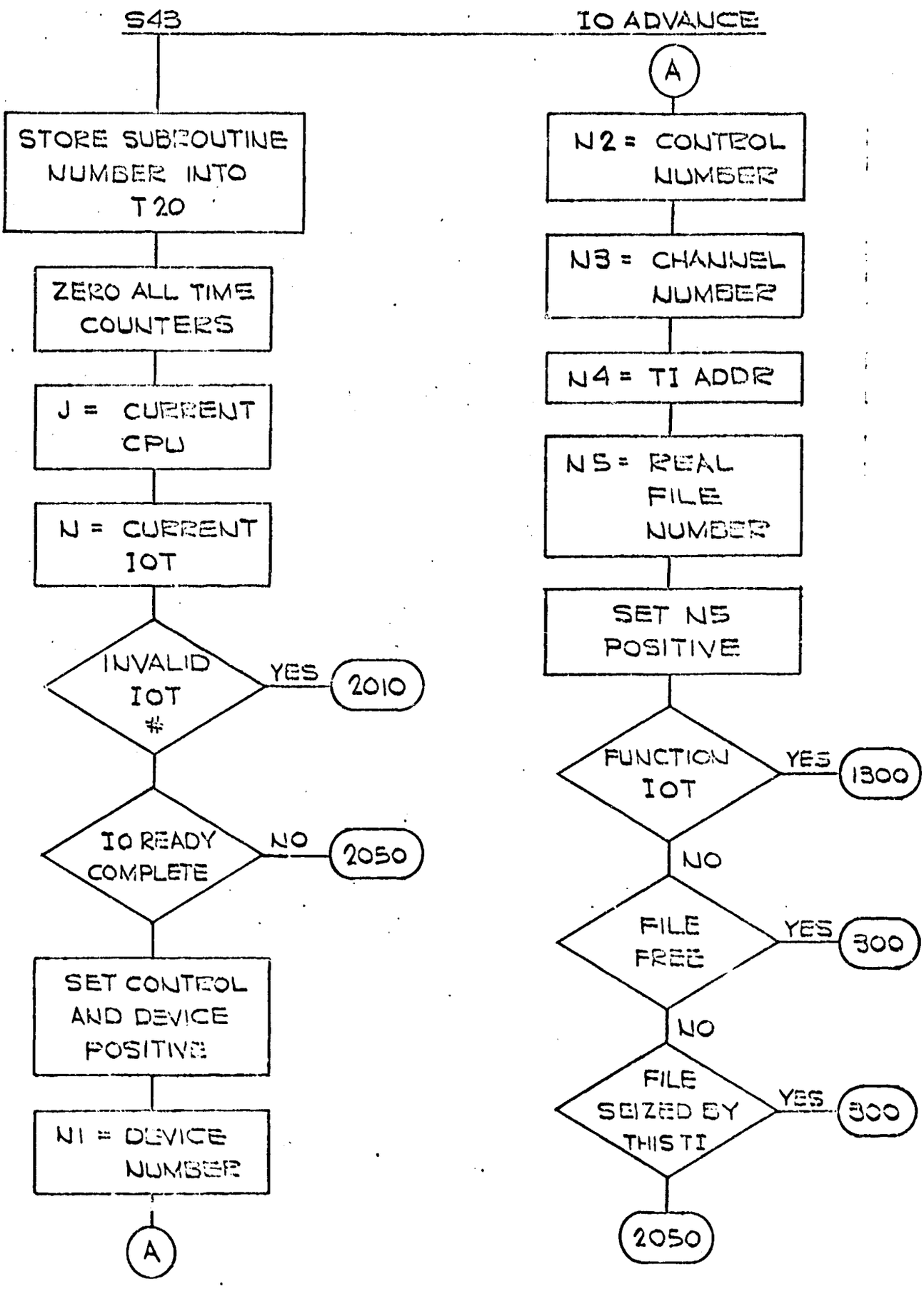

45Φ
STORE DEVICE #
CONTROL #
CHANNEL #
SET IO= READY
EXIT INTO NSJ.
RETURN


5ΦΦ
SET I/O BUSY
EXIT INTO
N.S.I
RETURN

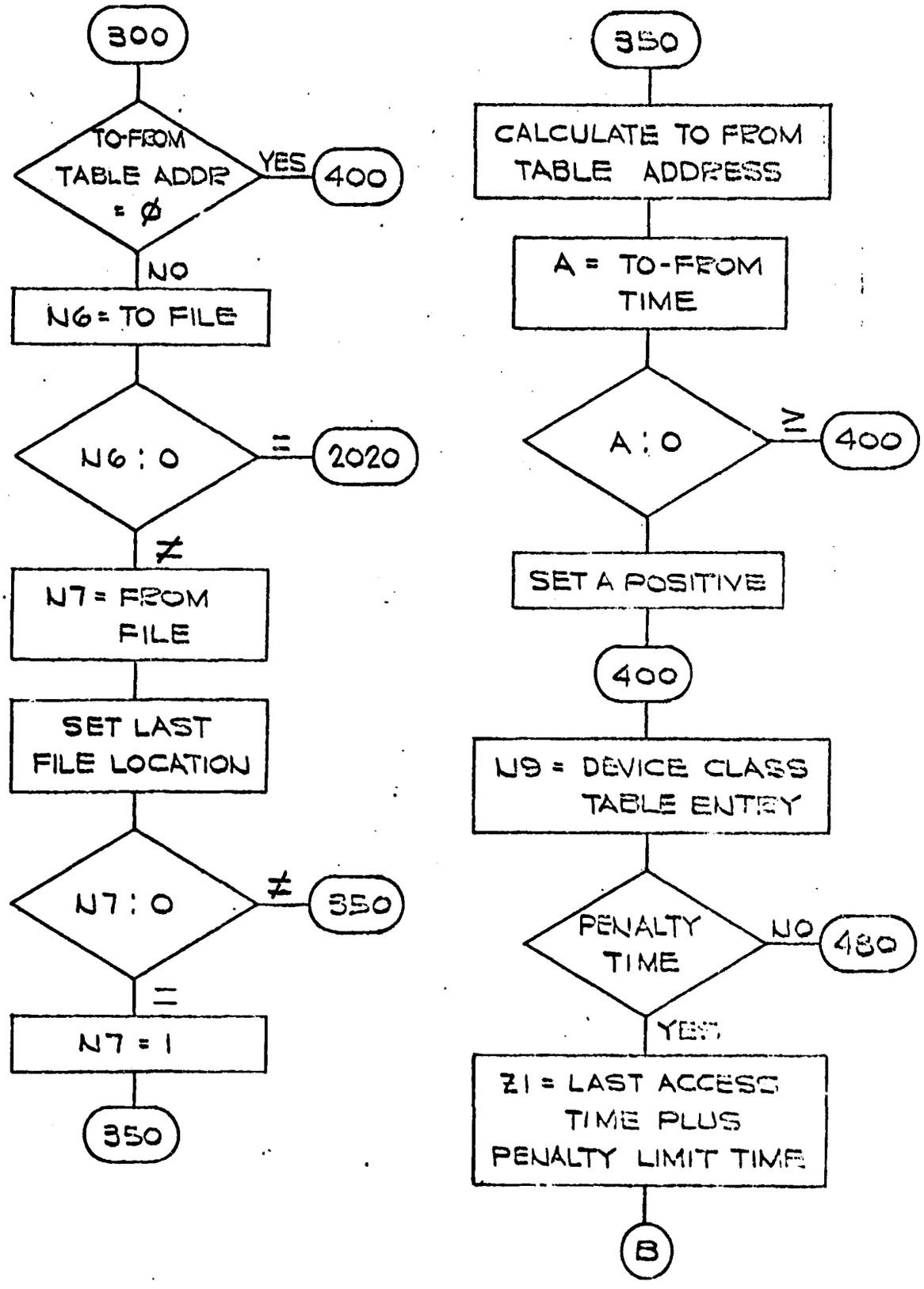

6ΦΦ
N16=
FUNCTION#
N6=
DEVICE #
DEVICE
#VALID
NO
62Φ
YES
N5=
TRANSACTION
POINTER
1ΦΦ


51Φ
SET ERROR
CODE = 8
RETURN

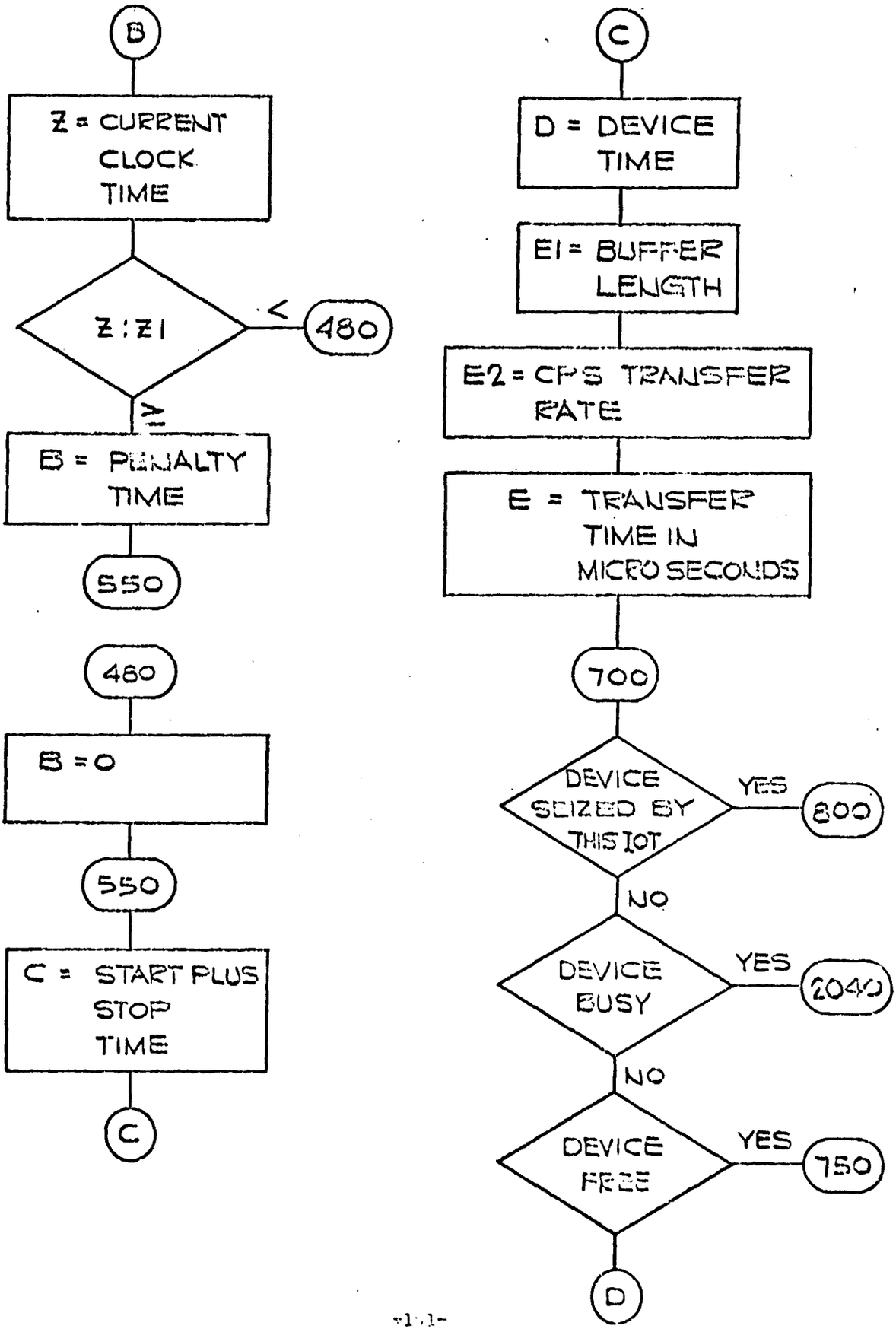

62Φ
SET ERROR
CODE = 1Φ
RETURN


S43
IO ADVANCE
STORE SUBROUTINE NUMBER INTO T20
ZERO ALL TIME COUNTERS
J = CURRENT CPU
N = CURRENT IOT
INVALID IOT #
YES
2010
IO READY COMPLETE
NO
2050
SET CONTROL AND DEVICE POSITIVE
N1 = DEVICE NUMBER
A
A
N2 = CONTROL NUMBER
N3 = CHANNEL NUMBER
N4 = TI ADDR
N5 = REAL FILE NUMBER
SET N5 POSITIVE
FUNCTION IOT
YES
1300
NO
FILE FREE
YES
300
NO
FILE SEIZED BY THIS TI
YES
300
2050

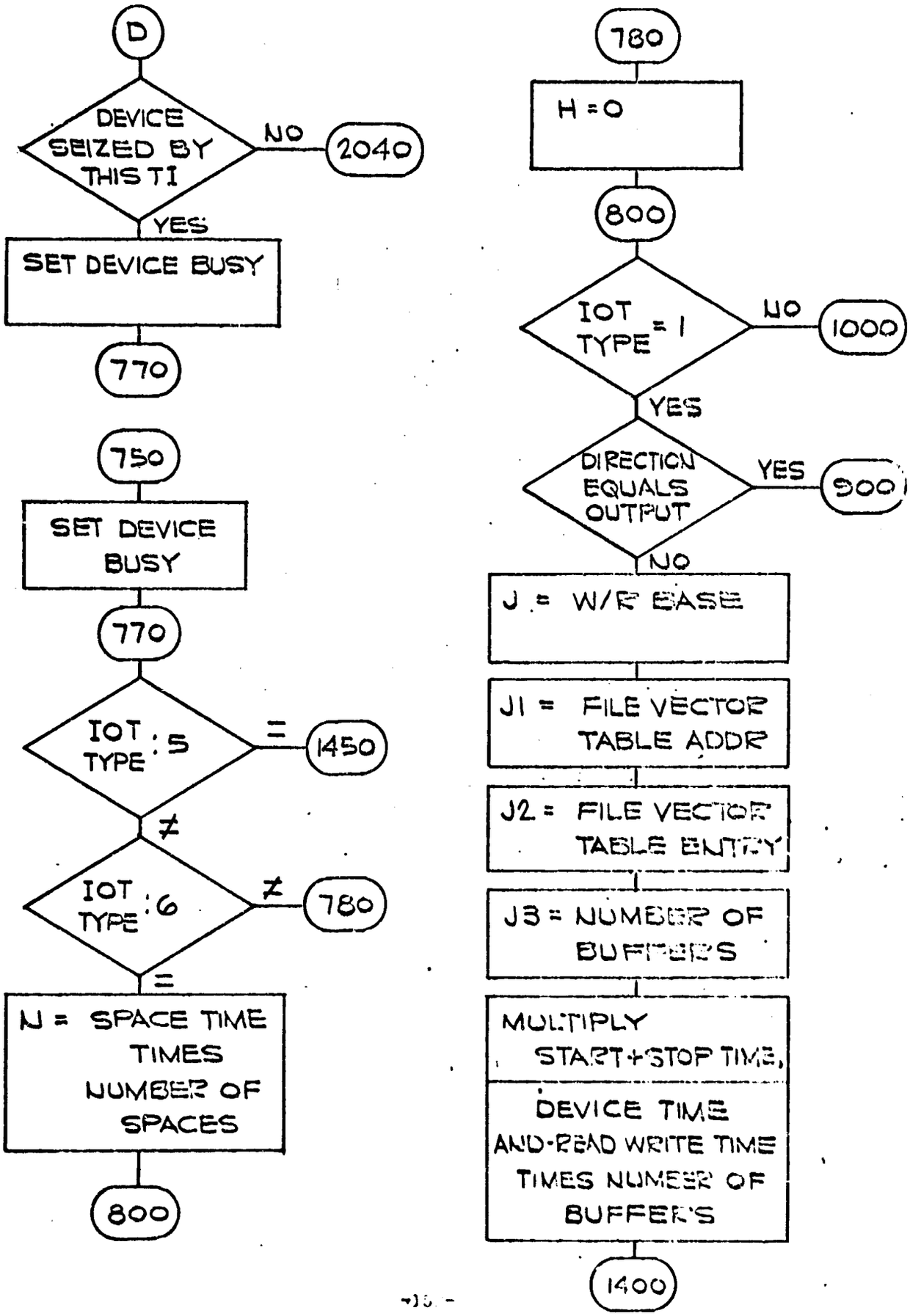
IO ADVANCE (CONTINUED)



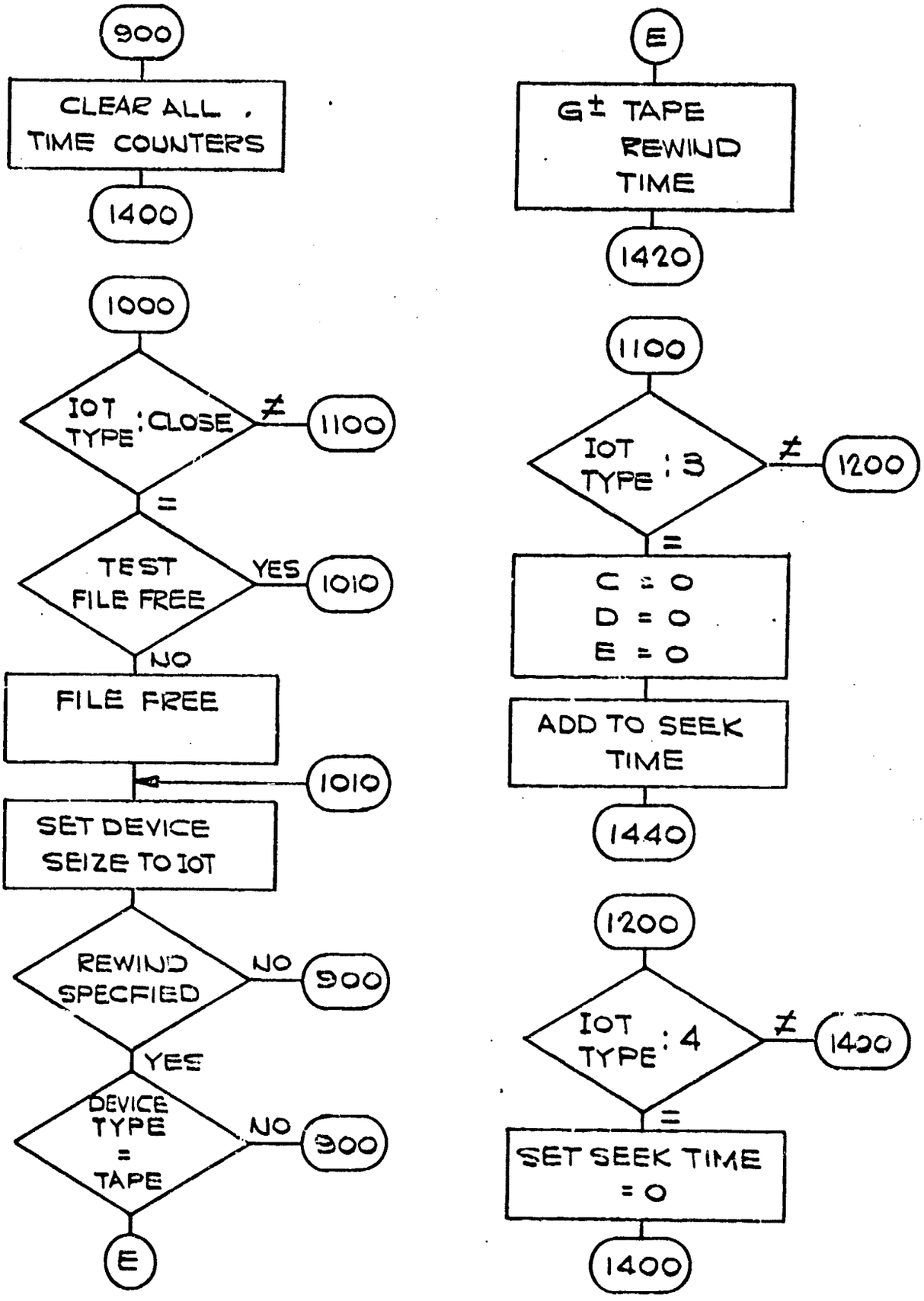
IO ADVANCE (CONTINUED)



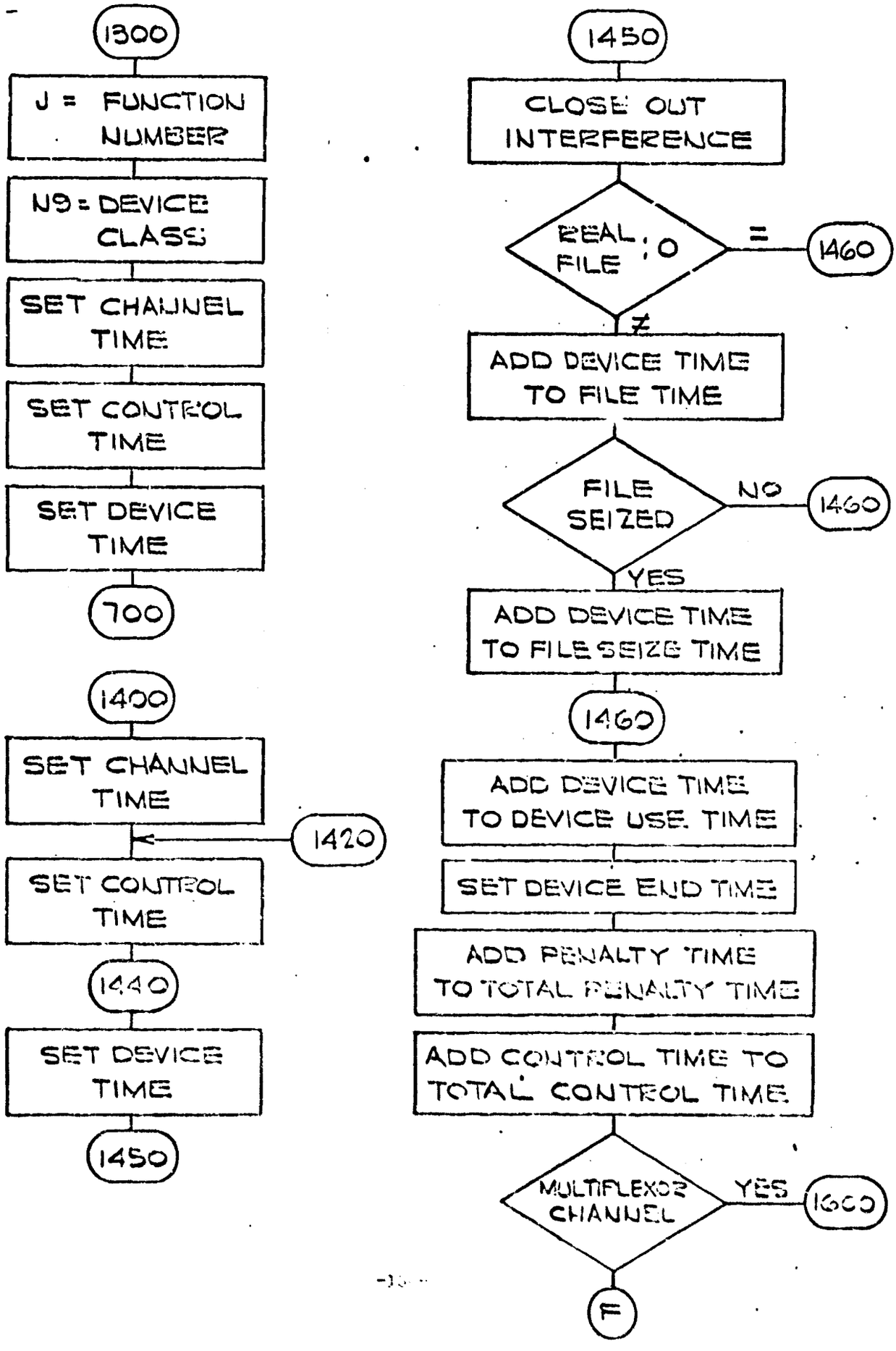
IO ADVANCE (CONTINUED)



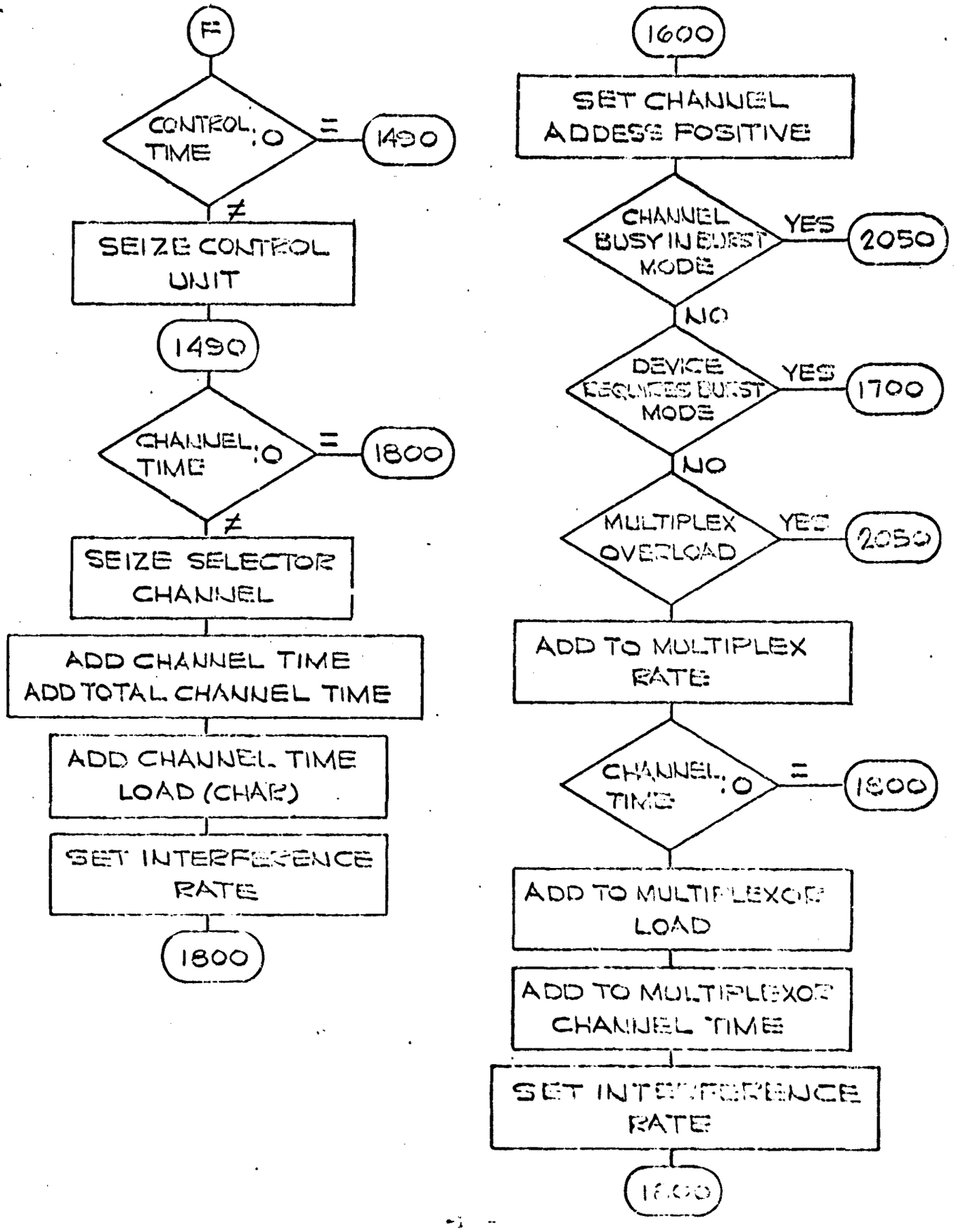
IO ADVANCE (CONTINUED)



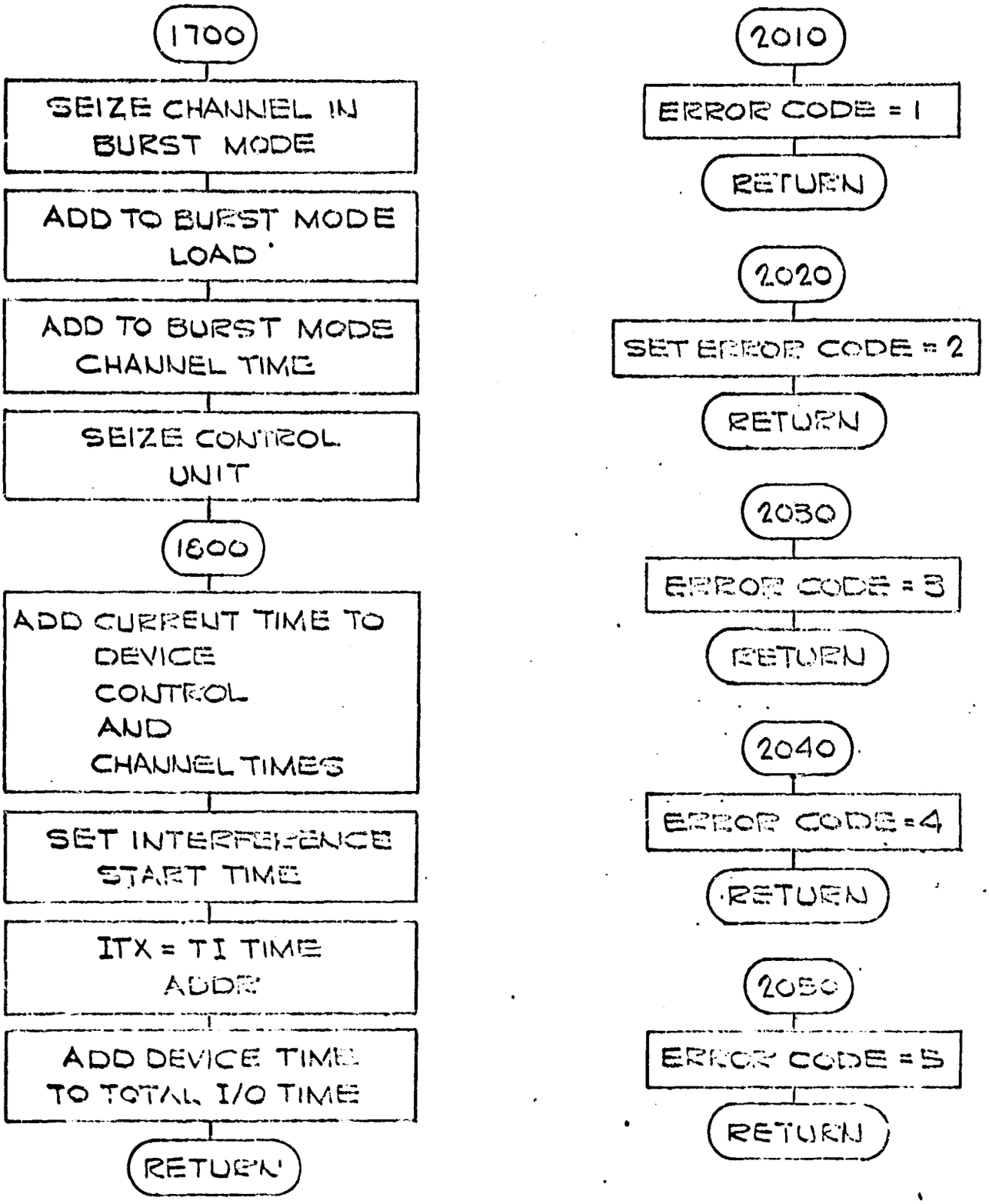
IO ADVANCE (COUTINUED)

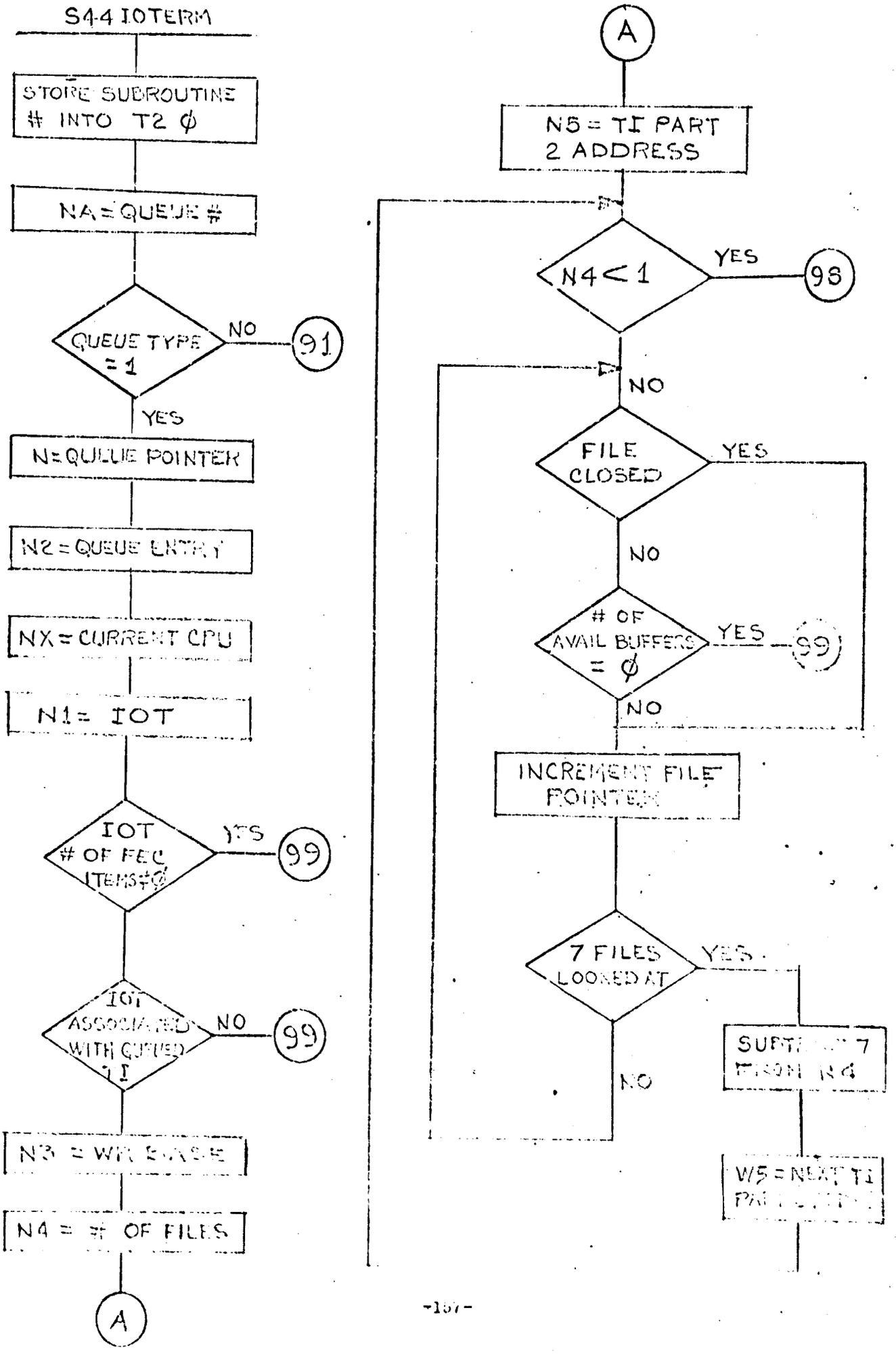


IO ADVANCE (CONTINUED)

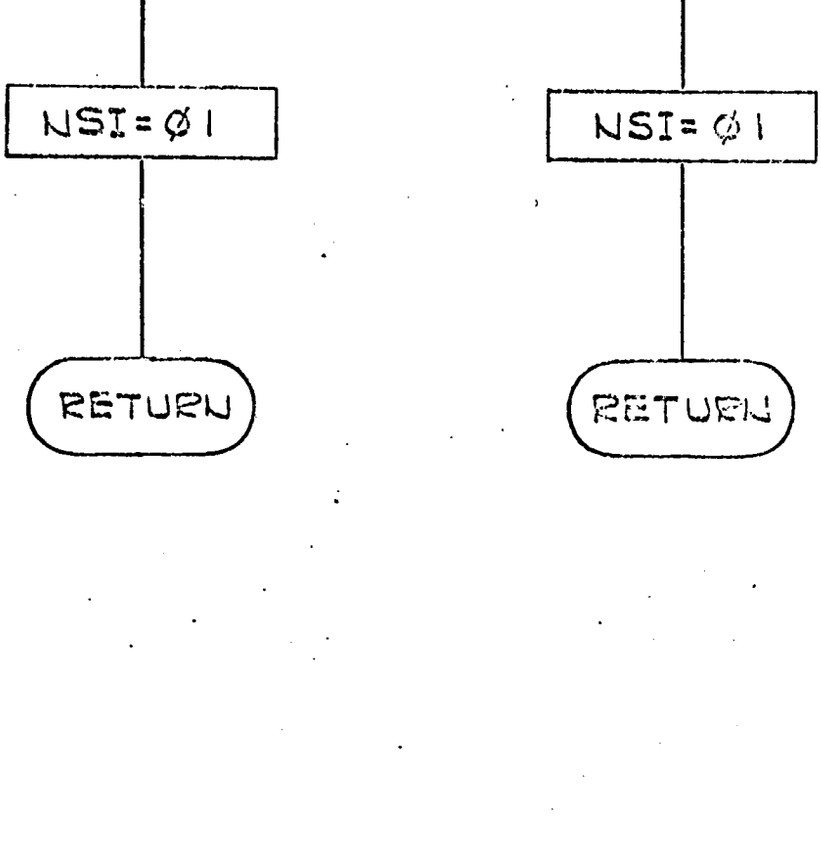


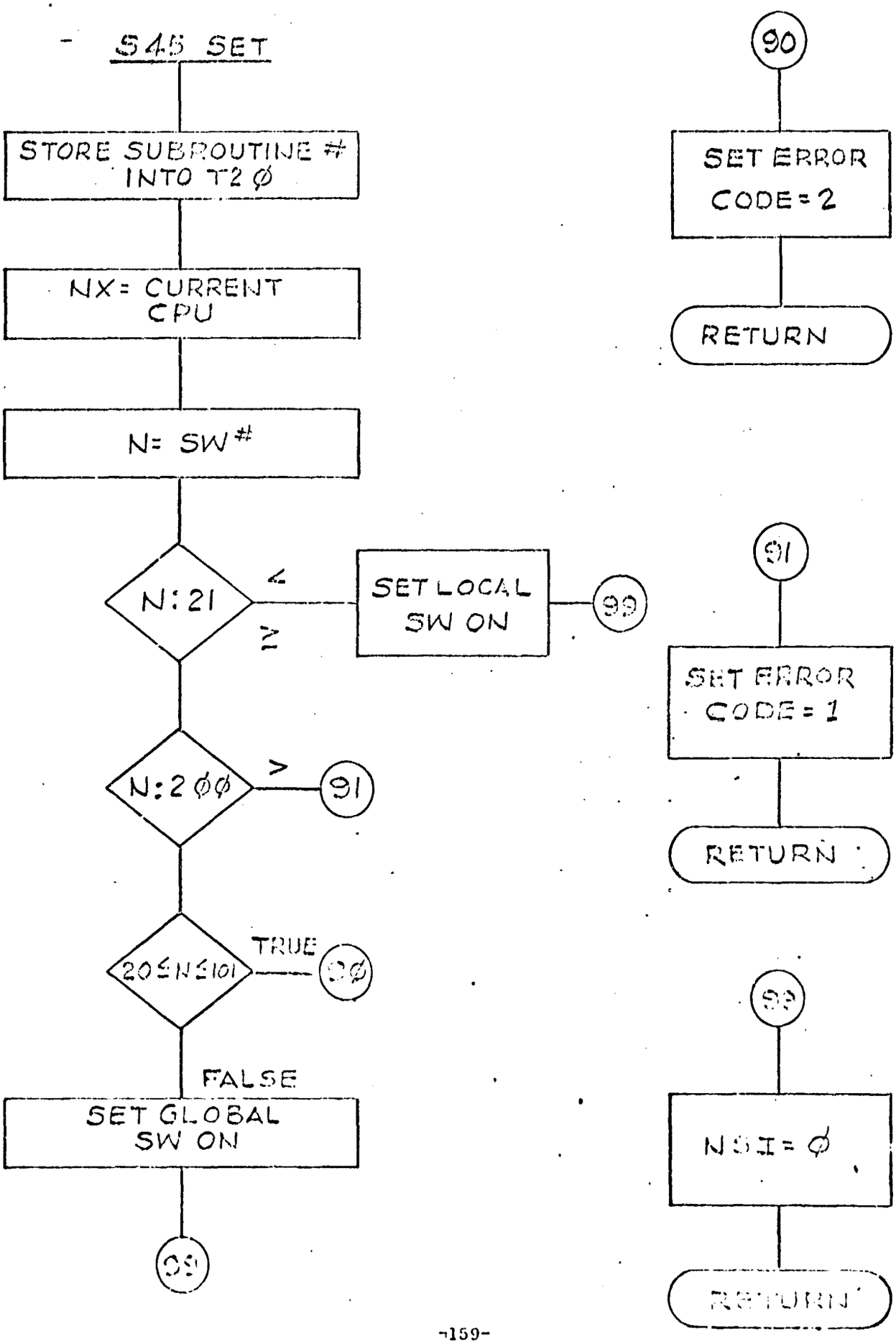
IO ADVANCE (CONTINUED)

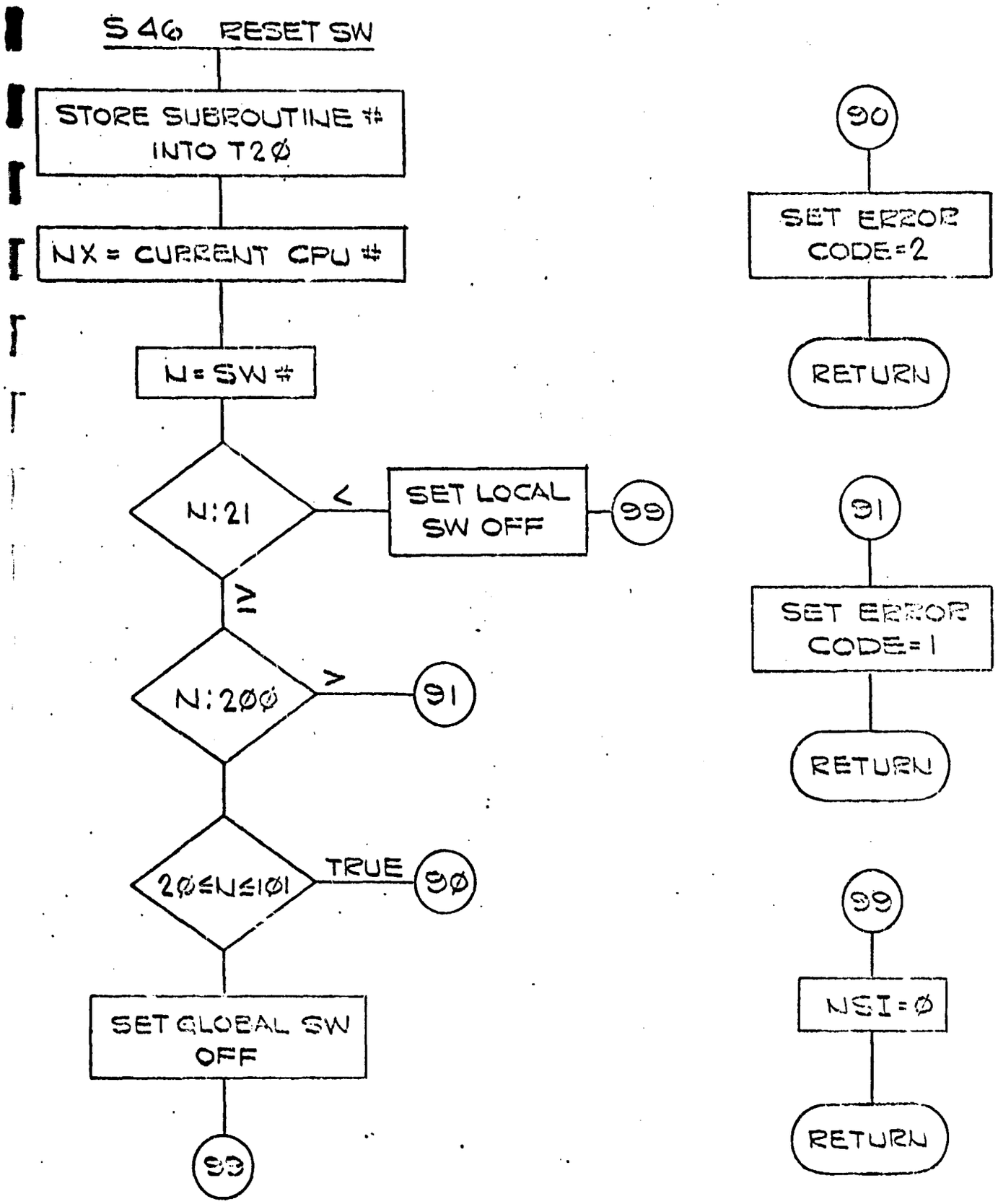


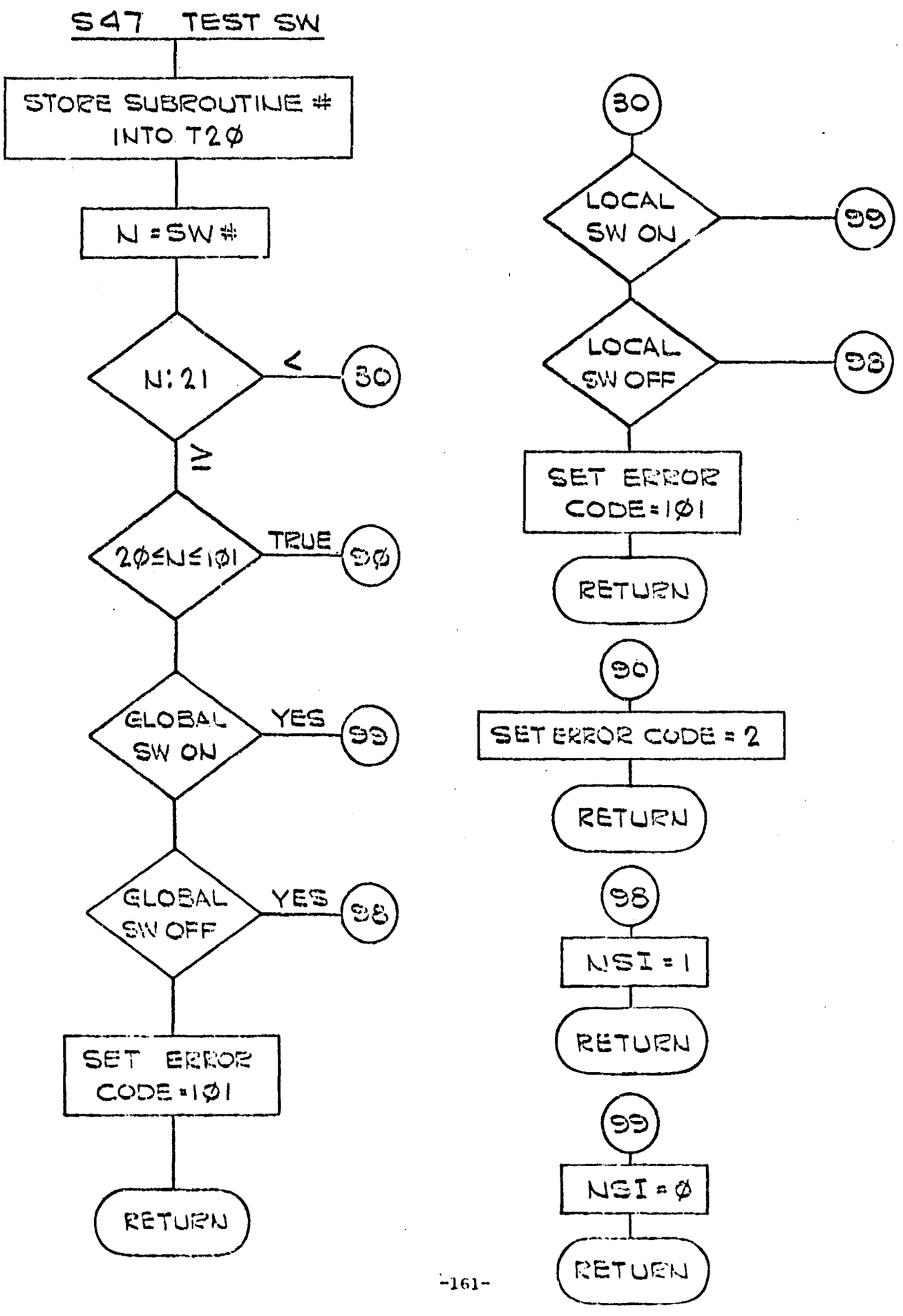

S4-4 IOTERM
STORE SUBROUTINE # INTO T2 ϕ
N4 = QUEUE #
QUEUE TYPE = 1
NO
91
YES
N = QUEUE POINTER
N2 = QUEUE ENTRY
NX = CURRENT CPU
N1 = IOT
IOT # OF REC ITEMS ≠ ϕ
YES
99
IOT ASSOCIATED WITH QUEUED I
NO
99
N3 = WR BASE
N4 = # OF FILES
A
A
N5 = TI PART 2 ADDRESS
N4 < 1
YES
98
NO
FILE CLOSED
YES
NO
OF AVAIL BUFFERS = ϕ
YES
99
NO
INCREMENT FILE POINTER
7 FILES LOOKED AT
YES
NO
W5 = NEXT TI

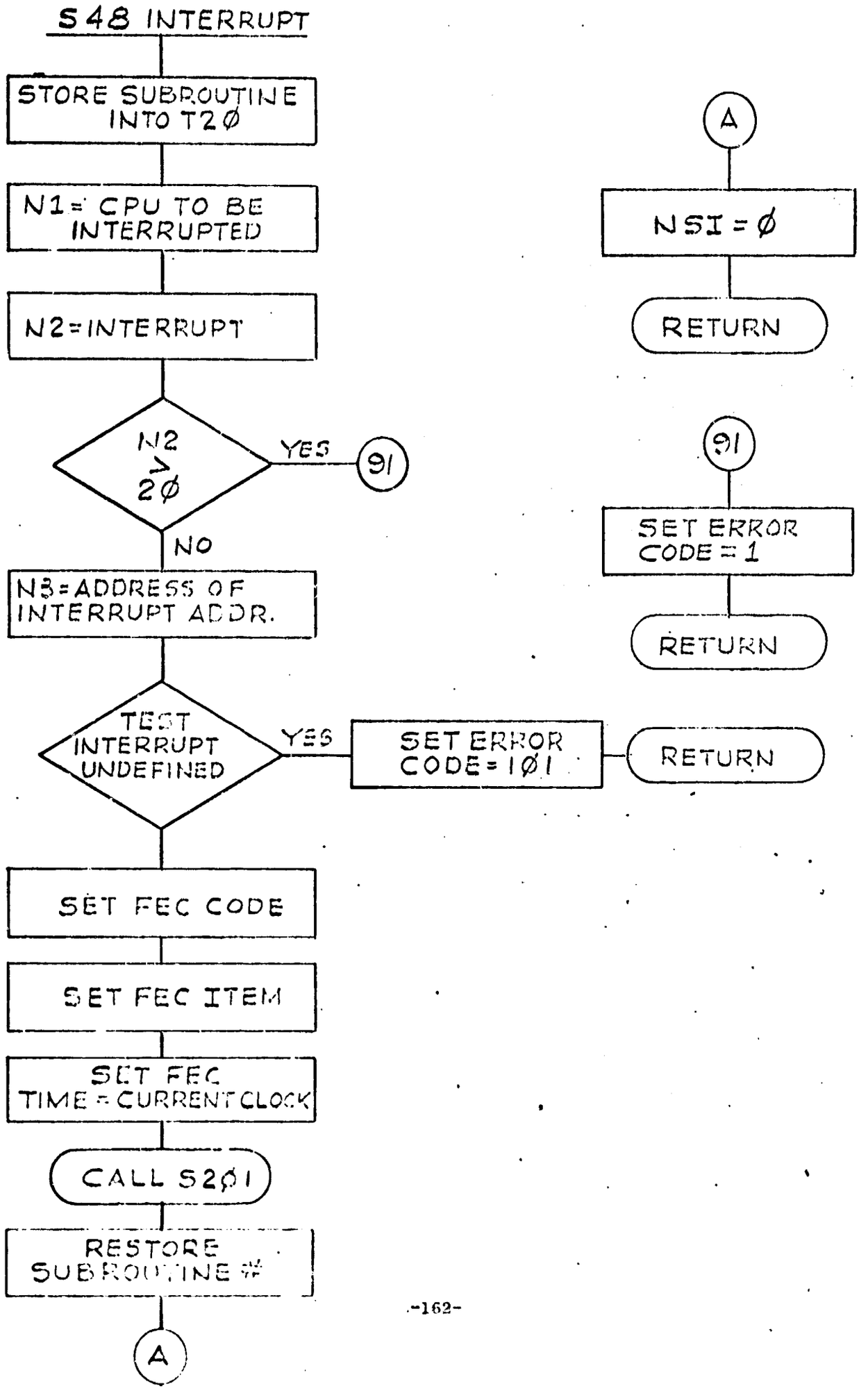
S 44

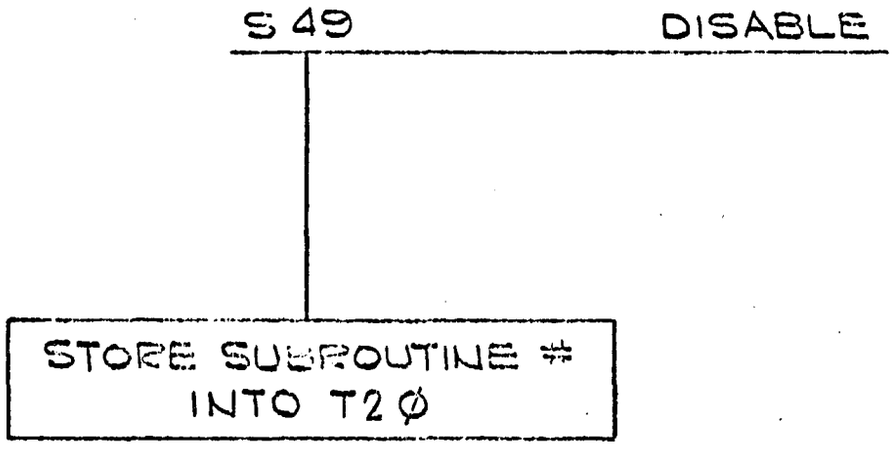


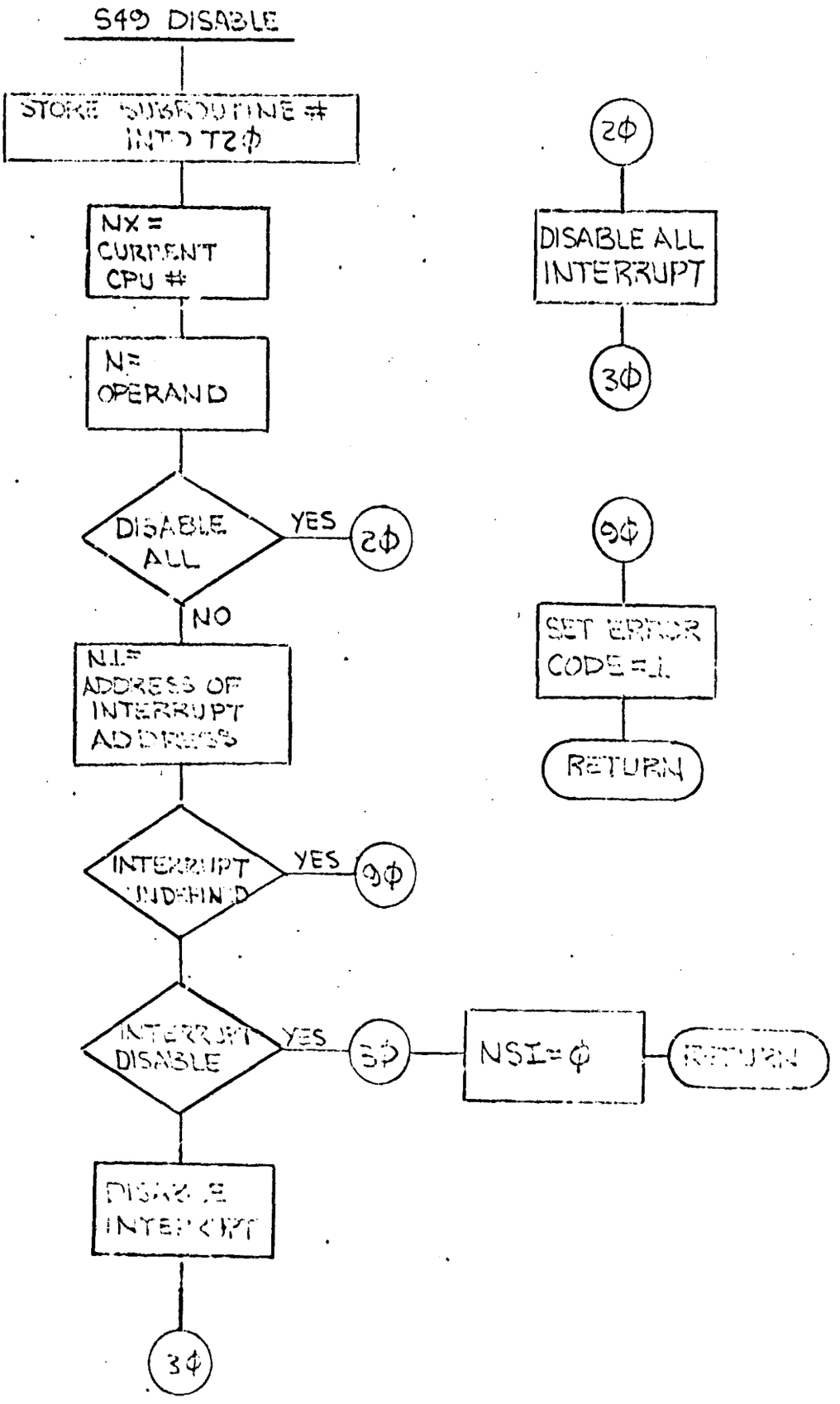

S45 SET
STORE SUBROUTINE # INTO T2 Ø
NX= CURRENT CPU
N= SW#
N:21
<
≥
SET LOCAL SW ON
99
N:2ØØ
>
91
20≤N≤101
TRUE
Ø9
FALSE
SET GLOBAL SW ON
Ø9
90
SET ERROR CODE=2
RETURN
91
SET ERROR CODE=1
RETURN
99
NSI=Ø
RETURN

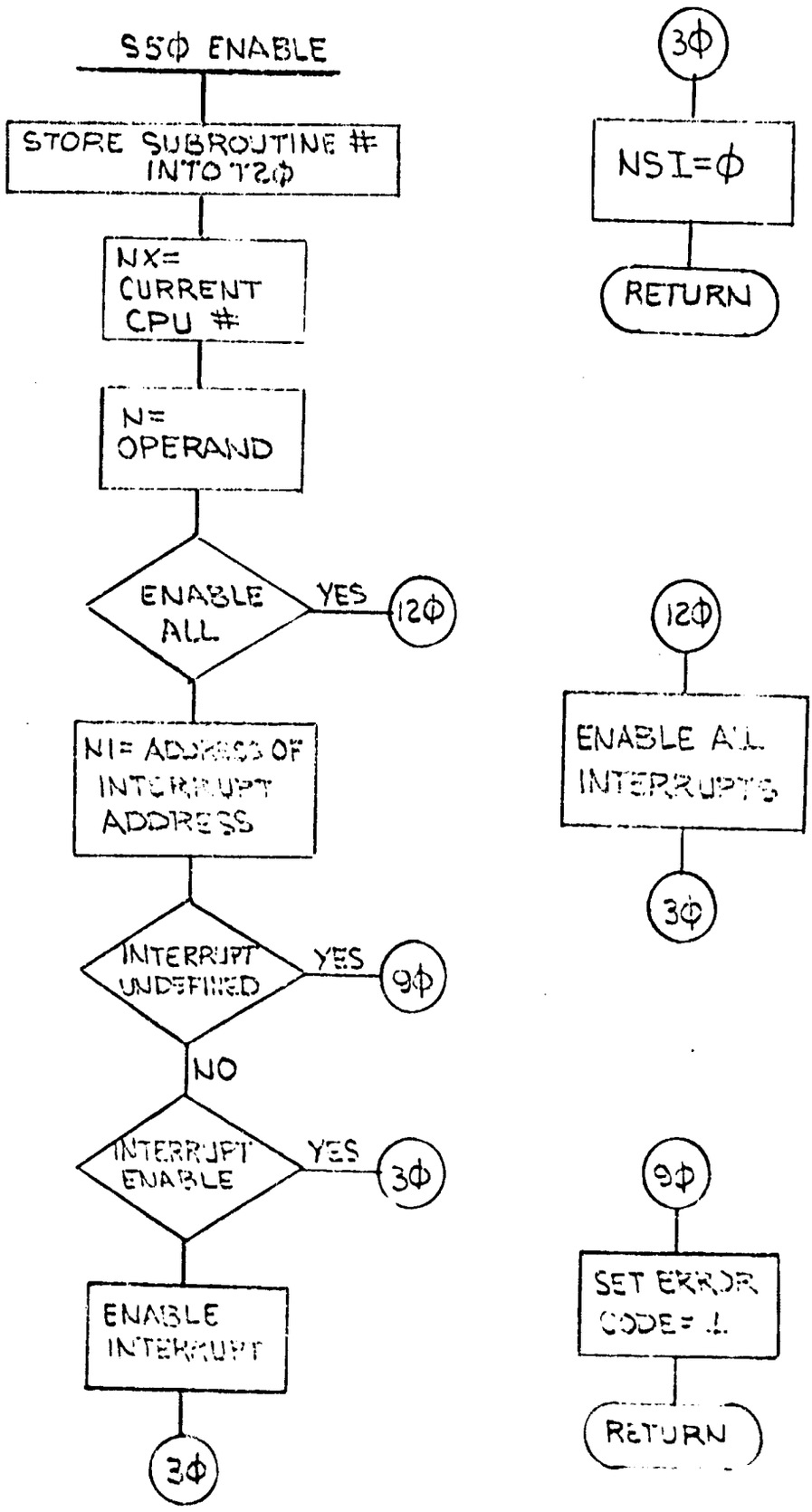

S 46 RESET SW
STORE SUBROUTINE # INTO T2Ø
NX = CURRENT CPU #
N = SW #
N:21
<
SET LOCAL SW OFF
99
≥
N:2ØØ
>
91
2Ø≤N≤1Ø1
TRUE
9Ø
SET GLOBAL SW OFF
93
90
SET ERROR CODE=2
RETURN
91
SET ERROR CODE=1
RETURN
99
NSI=Ø
RETURN

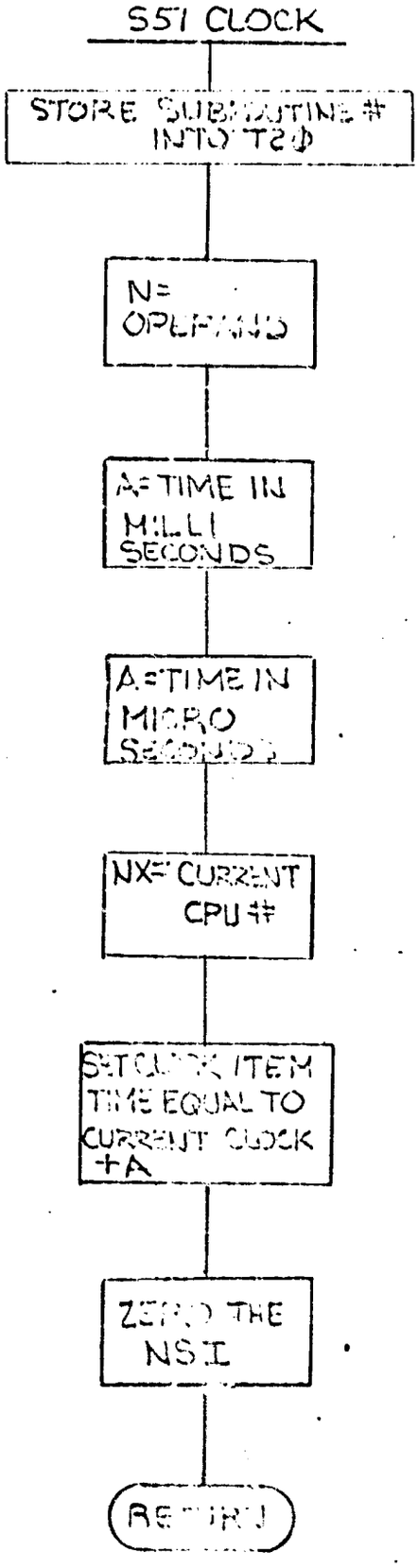

S47 TEST SW
STORE SUBROUTINE # INTO T2Ø
N = SW #
N: 21
<
30
≥
2Ø≤N≤1Ø1
TRUE
9Ø
GLOBAL SW ON
YES
99
GLOBAL SW OFF
YES
98
SET ERROR CODE=1Ø1
RETURN
30
LOCAL SW ON
99
LOCAL SW OFF
98
SET ERROR CODE=1Ø1
RETURN
9Ø
SET ERROR CODE = 2
RETURN
98
NSI = 1
RETURN
99
NSI = Ø
RETURN

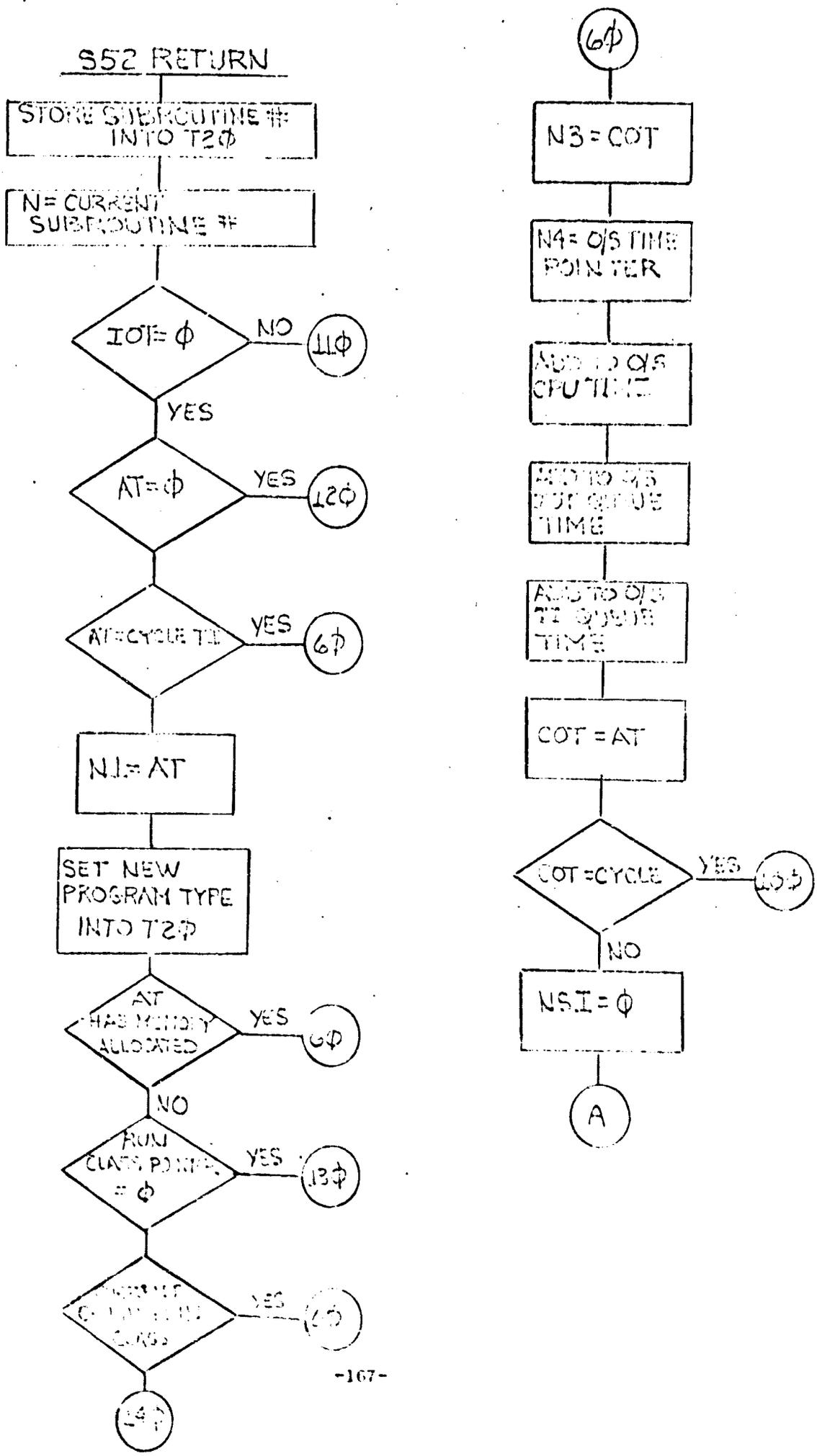

S48 INTERRUPT
STORE SUBROUTINE INTO T2Ø
N1 = CPU TO BE INTERRUPTED
N2 = INTERRUPT
N2 > 2Ø
YES
91
NO
N3 = ADDRESS OF INTERRUPT ADDR.
TEST INTERRUPT UNDEFINED
YES
SET ERROR CODE = 1Ø1
RETURN
SET FEC CODE
SET FEC ITEM
SET FEC TIME = CURRENT CLOCK
CALL S2Ø1
RESTORE SUBROUTINE #
A
A
NSI = Ø
RETURN
91
SET ERROR CODE = 1
RETURN

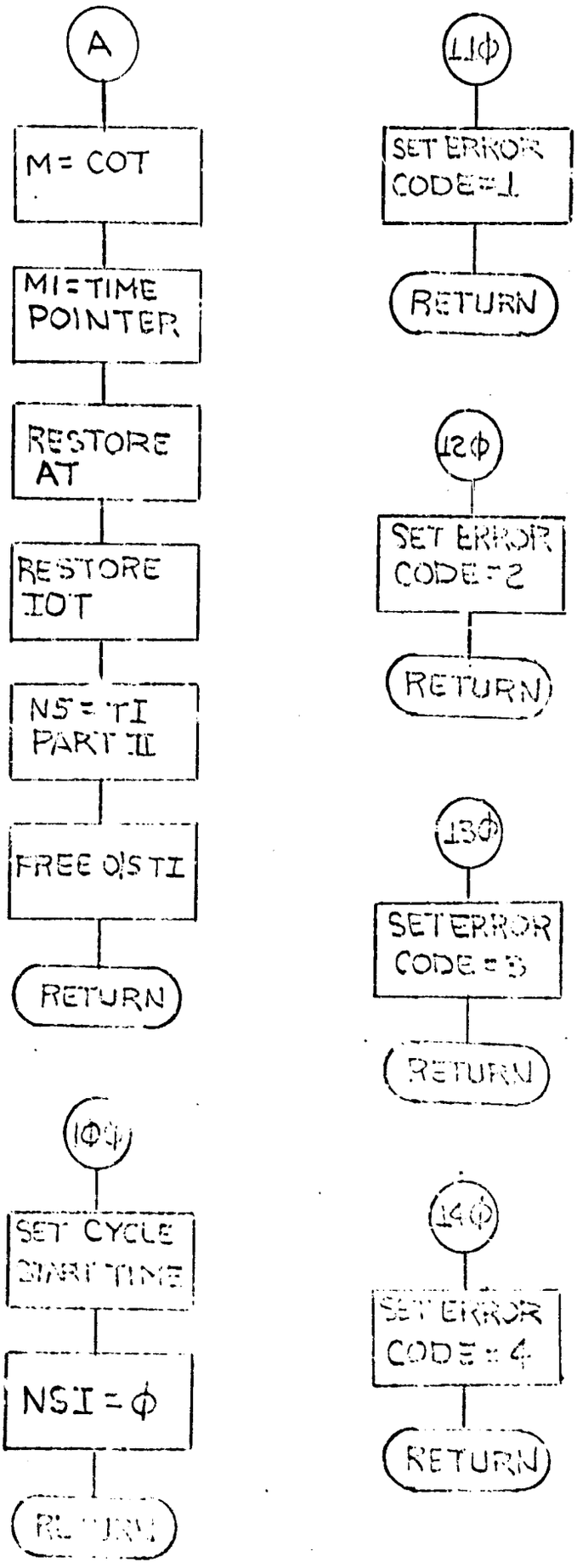

S 49
DISABLE
STORE SUBROUTINE #
INTO T2 Ø

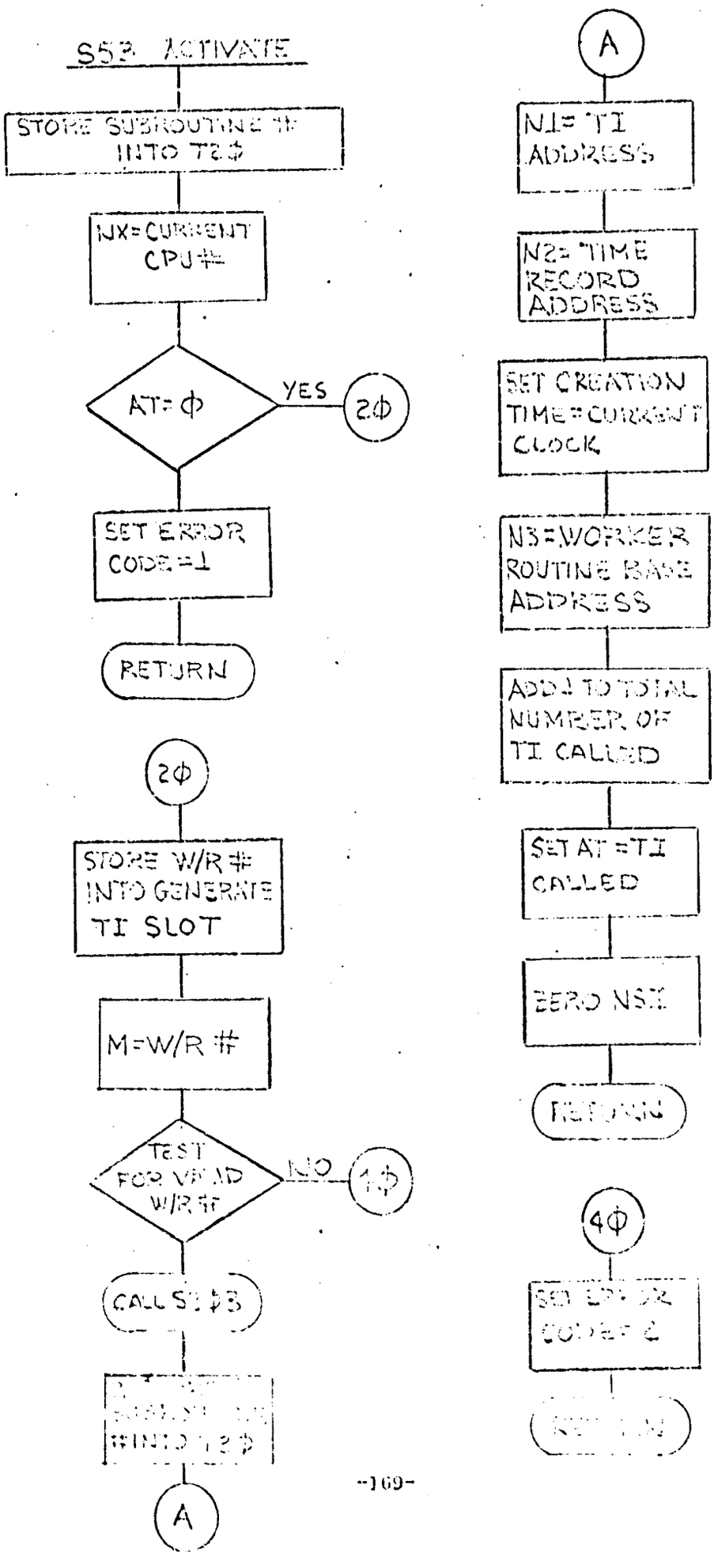

S49 DISABLE
STORE SUBROUTINE #
INTO T2Φ
NX =
CURRENT
CPU #
N=
OPERAND
DISABLE
ALL
YES
2Φ
NO
N.I.=
ADDRESS OF
INTERRUPT
ADDRESS
INTERRUPT
UNDEFINED
YES
9Φ
INTERRUPT
DISABLE
YES
3Φ
NSI=Φ
RETURN
DISABLE
INTERRUPT
3Φ
2Φ
DISABLE ALL
INTERRUPT
3Φ
9Φ
SET ERROR
CODE=1
RETURN

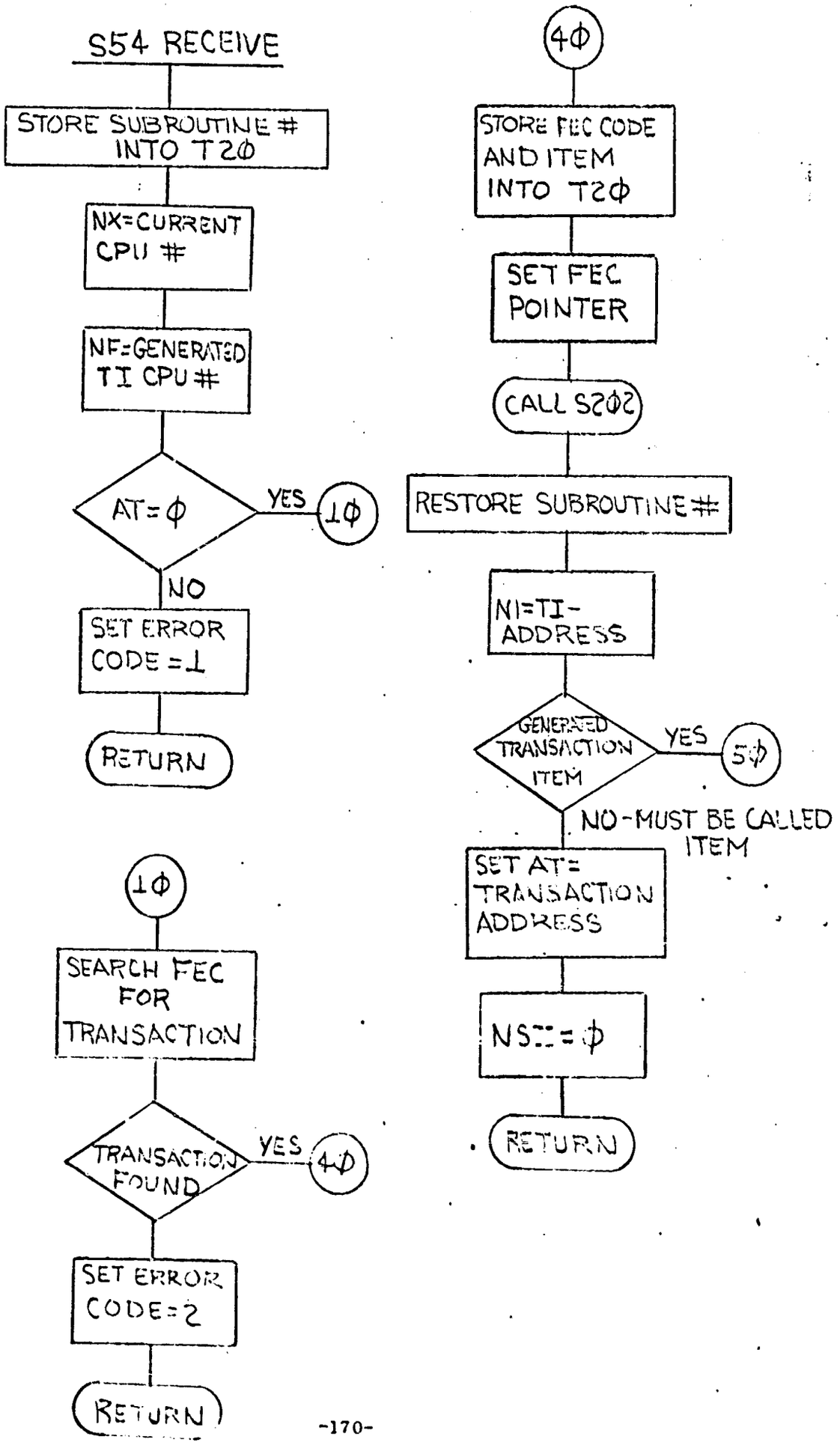

55Φ ENABLE
STORE SUBROUTINE # INTO 72Φ
NX= CURRENT CPU #
N= OPERAND
ENABLE ALL
YES
12Φ
N1= ADDRESS OF INTERRUPT ADDRESS
INTERRUPT UNDEFINED
YES
9Φ
NO
INTERRUPT ENABLE
YES
3Φ
ENABLE INTERRUPT
3Φ
3Φ
NSI=Φ
RETURN
12Φ
ENABLE ALL INTERRUPTS
3Φ
9Φ
SET ERROR CODE= 1
RETURN

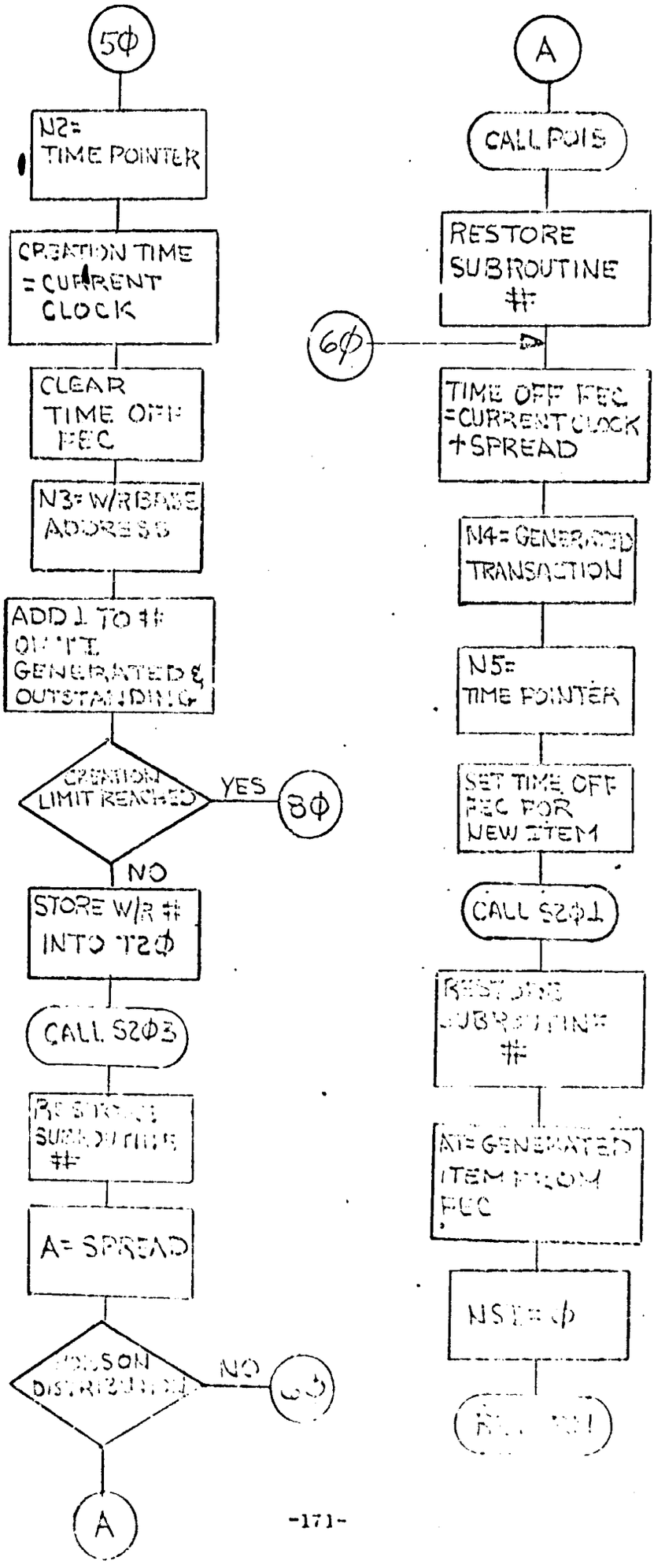

SST CLOCK
STORE SUBROUTINE # INTO T2Ø
N= OPERAND
A=TIME IN MILLI SECONDS
A=TIME IN MICRO SECONDS
NX=CURRENT CPU #
SET CLOCK ITEM TIME EQUAL TO CURRENT CLOCK +A
ZERO THE NSI
RETURN

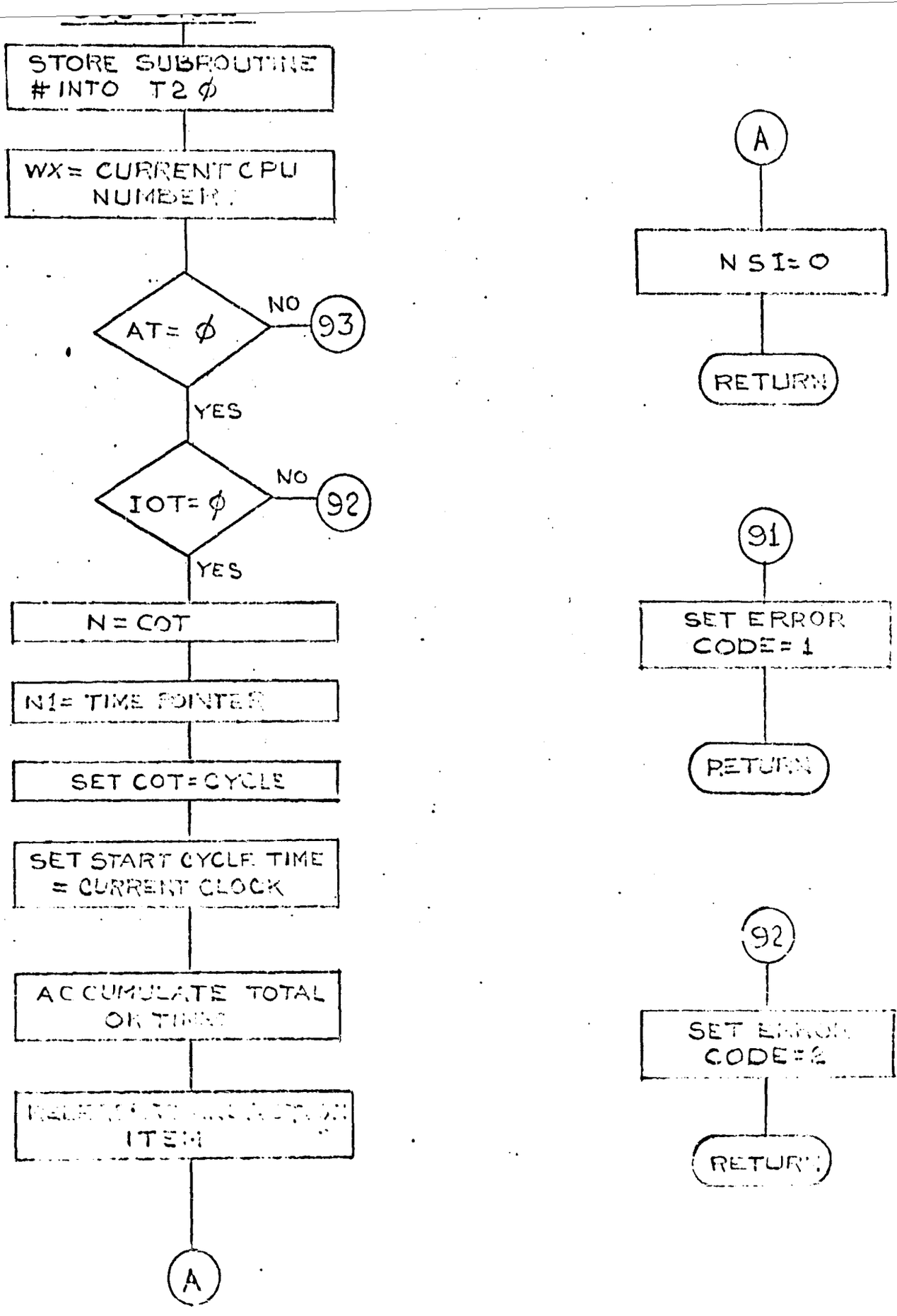

S52 RETURN
STORE SUBROUTINE # INTO T2Φ
N = CURRENT SUBROUTINE #
IOT = Φ
NO
11Φ
YES
AT = Φ
YES
12Φ
AT = CYCLE TI
YES
6Φ
N1 = AT
SET NEW PROGRAM TYPE INTO T2Φ
AT HAS MEMORY ALLOCATED
YES
6Φ
NO
RUN CLASS POINTER = Φ
YES
13Φ
YES
14Φ
6Φ
N3 = COT
N4 = O/S TIME POINTER
ADD TO O/S CPU TIME
ADD TO O/S QUEUE TIME
ADD TO O/S TI QUEUE TIME
COT = AT
COT = CYCLE
YES
15Φ
NO
NSI = Φ
A

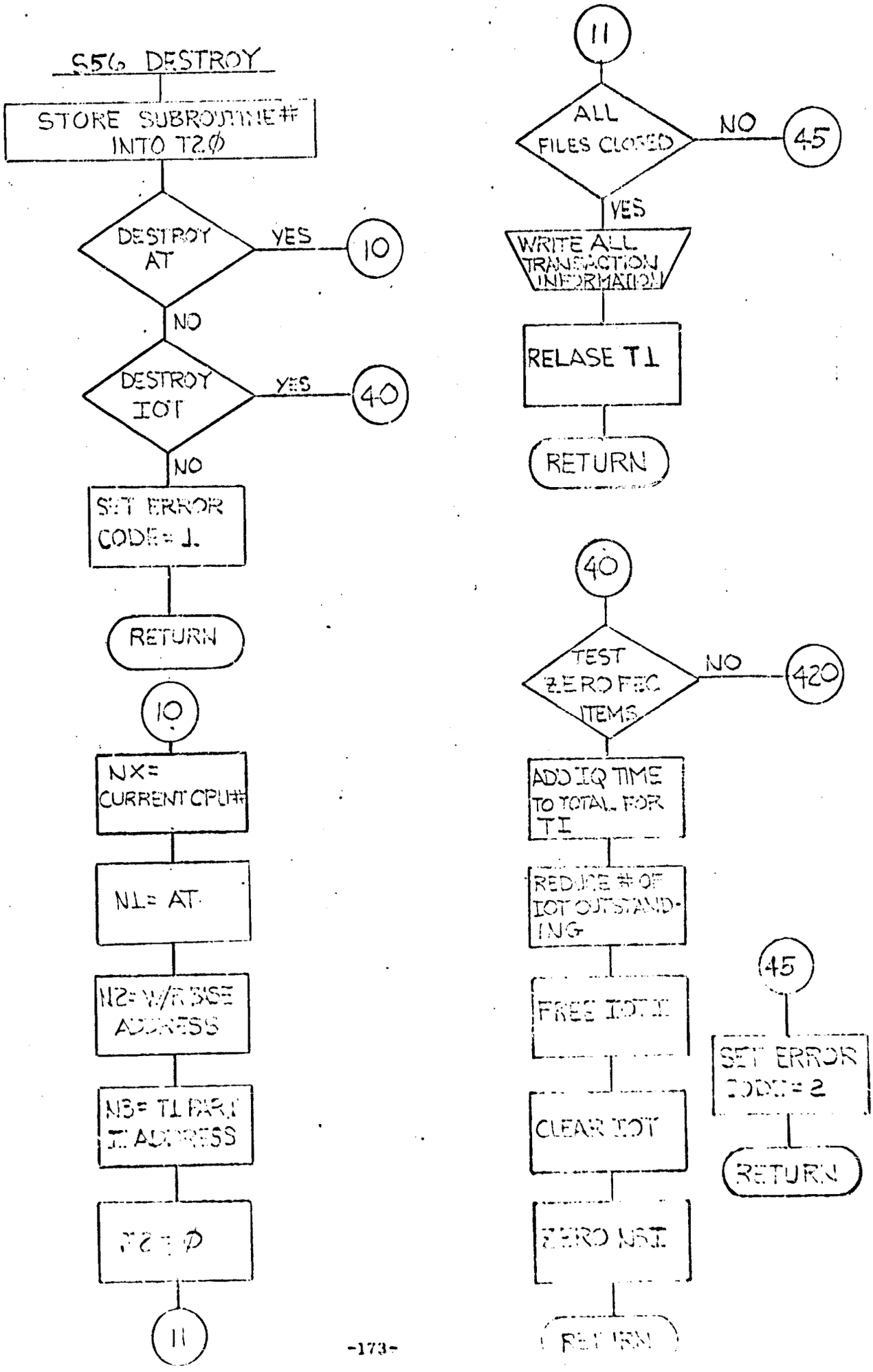

A
M = COT
MI = TIME POINTER
RESTORE AT
RESTORE IOT
NS = TI PART II
FREE DIS TI
RETURN
10φ
SET CYCLE START TIME
NSI = φ
RETURN
11φ
SET ERROR CODE = 1
RETURN
12φ
SET ERROR CODE = 2
RETURN
13φ
SET ERROR CODE = 3
RETURN
14φ
SET ERROR CODE = 4
RETURN

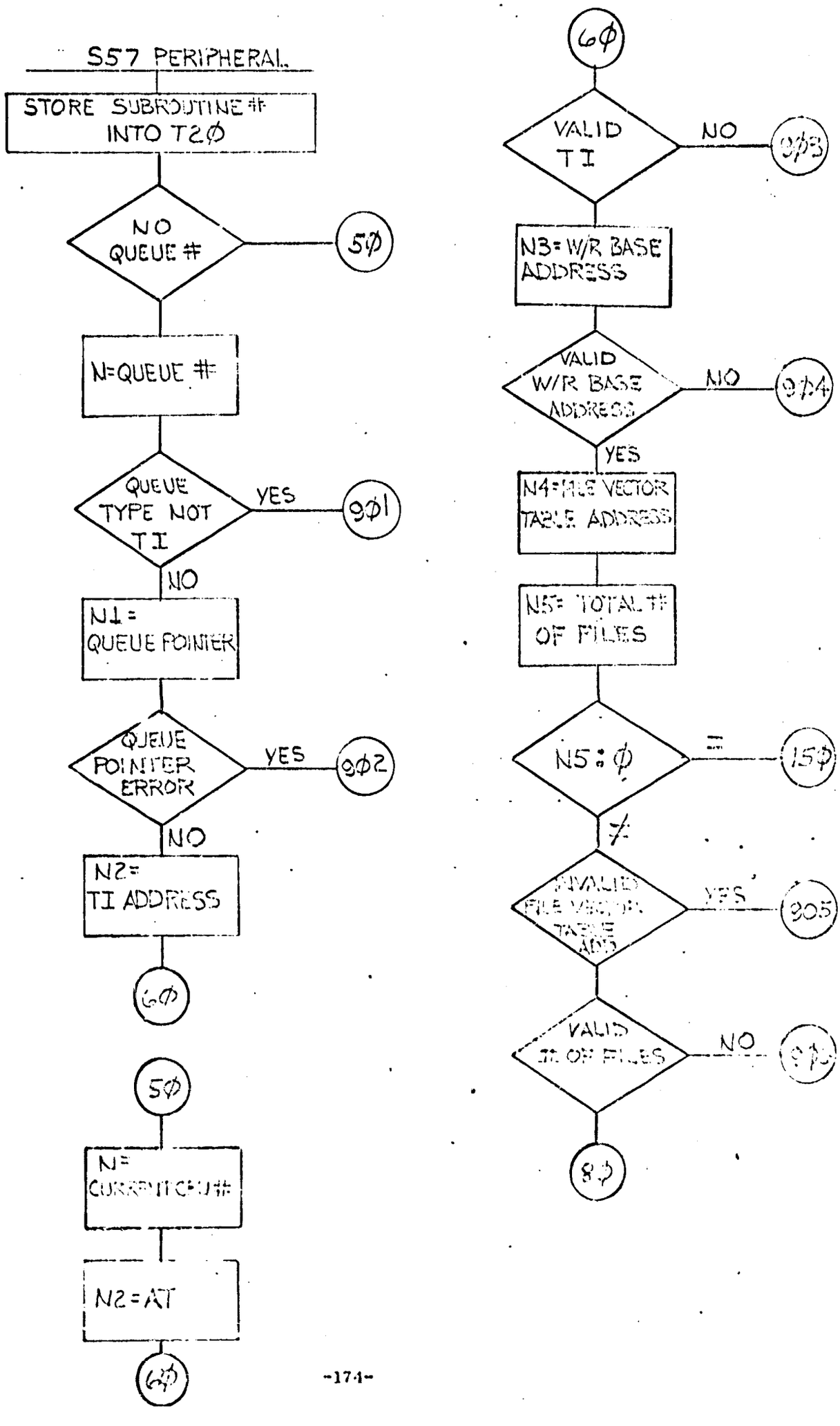

SSP ACTIVATE
STORE SUBROUTINE # INTO T2$
NX=CURRENT CPU#
AT=Φ
YES
2Φ
SET ERROR CODE=1
RETURN
2Φ
STORE W/R# INTO GENERATE TI SLOT
M=W/R #
TEST FOR VALID W/R #
NO
1Φ
CALL SSP3
A
A
N1=TI ADDRESS
N2=TIME RECORD ADDRESS
SET CREATION TIME=CURRENT CLOCK
N3=WORKER ROUTINE BASE ADDRESS
ADD 1 TO TOTAL NUMBER OF TI CALLED
SET AT=TI CALLED
ZERO NSI
RETURN
4Φ

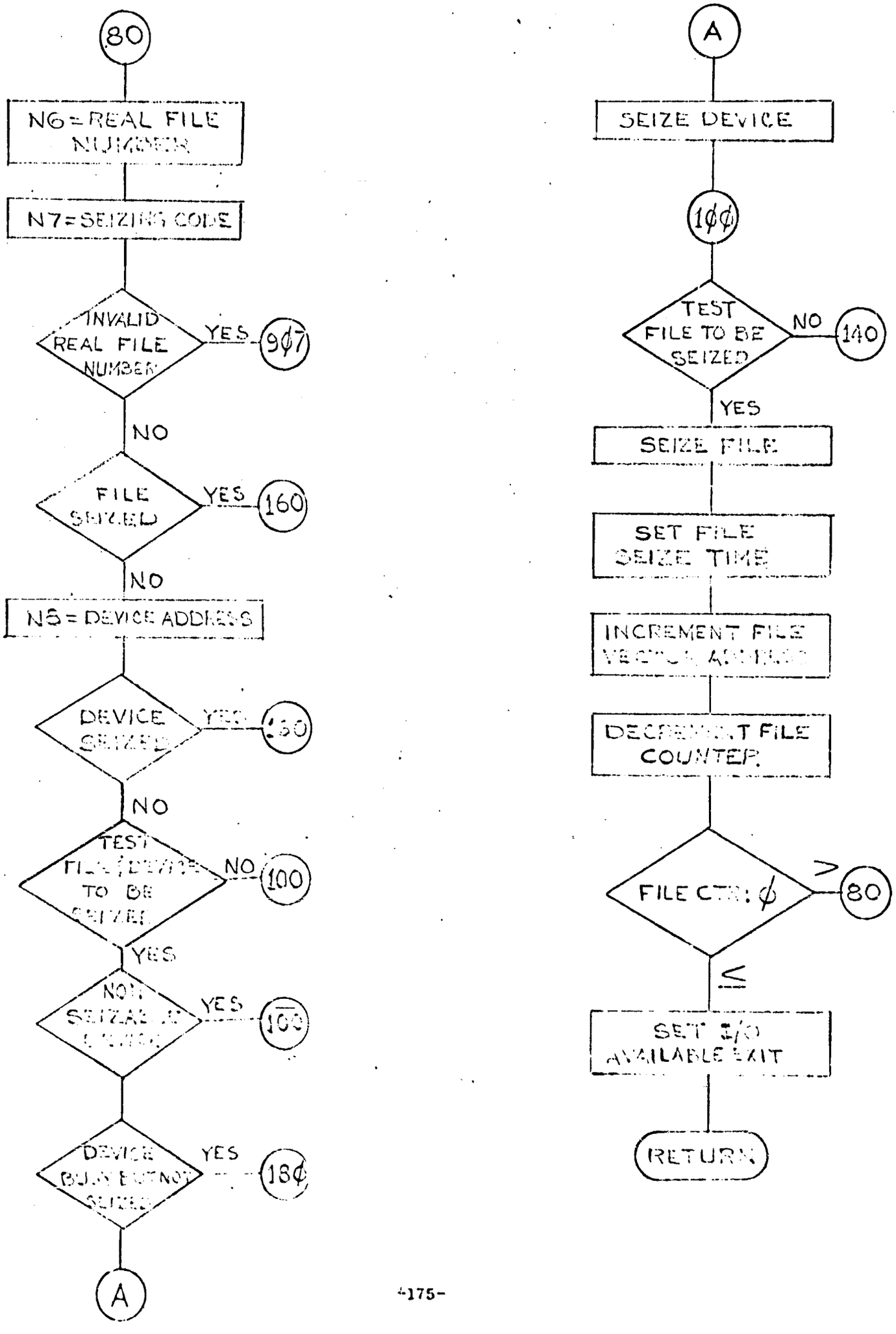

S54 RECEIVE
STORE SUBROUTINE # INTO T2Φ
NX=CURRENT CPU #
NF=GENERATED TI CPU #
AT=Φ
YES
1Φ
NO
SET ERROR CODE=1
RETURN
1Φ
SEARCH FEC FOR TRANSACTION
TRANSACTION FOUND
YES
4Φ
SET ERROR CODE=2
RETURN
4Φ
STORE FEC CODE AND ITEM INTO T2Φ
SET FEC POINTER
CALL S2Φ2
RESTORE SUBROUTINE #
NI=TI-ADDRESS
GENERATED TRANSACTION ITEM
YES
5Φ
NO-MUST BE CALLED ITEM
SET AT= TRANSACTION ADDRESS
NSI=Φ
RETURN

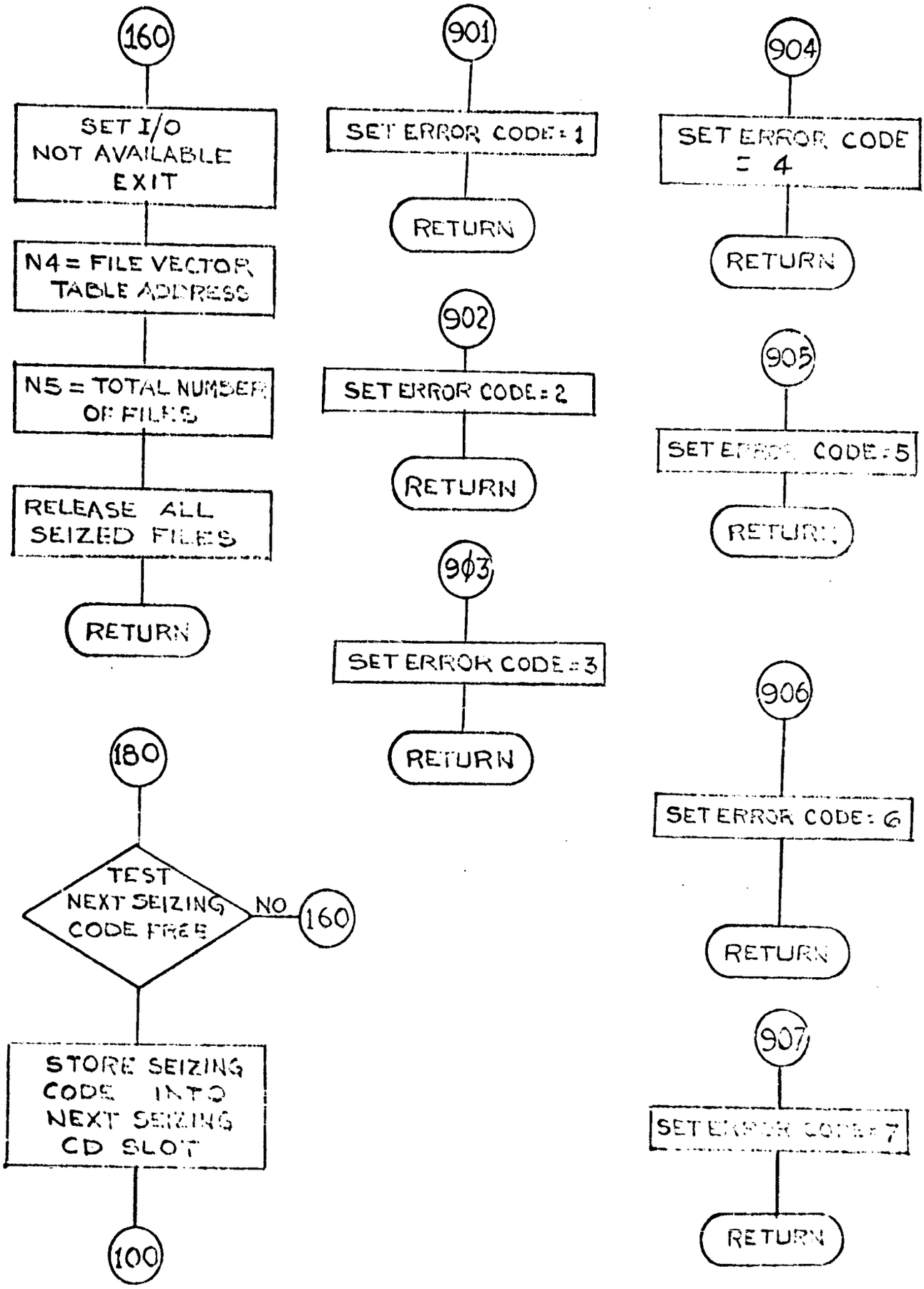

5φ
N2=
TIME POINTER
CREATION TIME
=CURRENT
CLOCK
CLEAR
TIME OFF
FEC
N3=W/R BASE
ADDRESS
ADD 1 TO #
ON IT
GENERATED &
OUTSTANDING
CREATION
LIMIT REACHED
YES
8φ
NO
STORE W/R #
INTO T2φ
CALL S2φ3
RESTORE
SUBROUTINE
#
A=SPREAD
POISSON
DISTRIBUTION
NO
3φ
A
A
CALL POIS
RESTORE
SUBROUTINE
#
6φ
TIME OFF FEC
=CURRENT CLOCK
+SPREAD
N4=GENERATED
TRANSACTION
N5=
TIME POINTER
SET TIME OFF
FEC FOR
NEW ITEM
CALL S2φ1
RESTORE
SUBROUTINE
#
N5=GENERATED
ITEM FROM
FEC
N5=φ

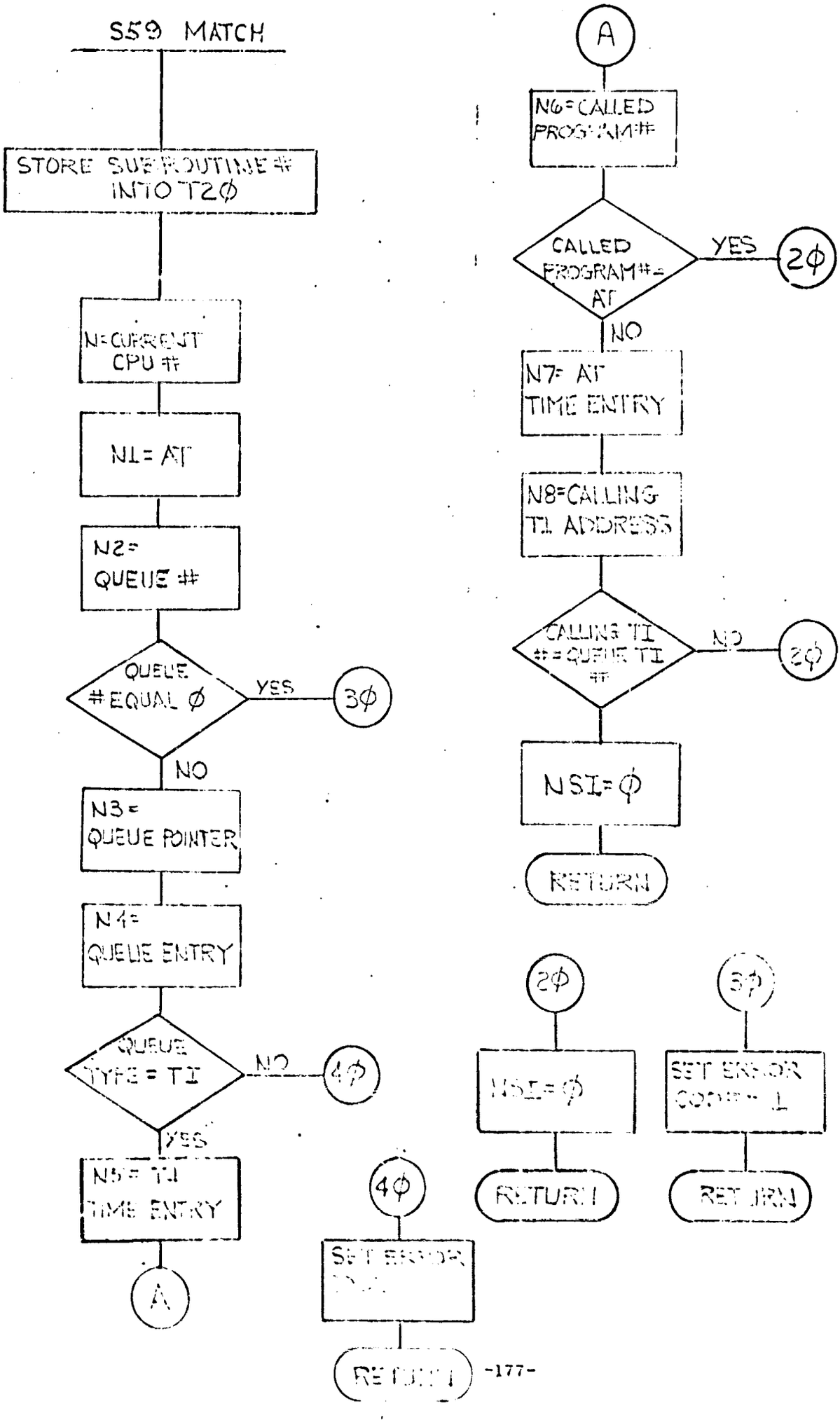

STORE SUBROUTINE # INTO T2Ø
WX= CURRENT CPU NUMBER
ATØ
NO
93
YES
IOTØ
NO
92
YES
N= COT
N1= TIME POINTER
SET COT= CYCLE
SET START CYCLE TIME = CURRENT CLOCK
ACCUMULATE TOTAL
ITEM
A
A
NSI=0
RETURN
91
SET ERROR CODE=1
RETURN
92
SET ERROR CODE=2
RETURN

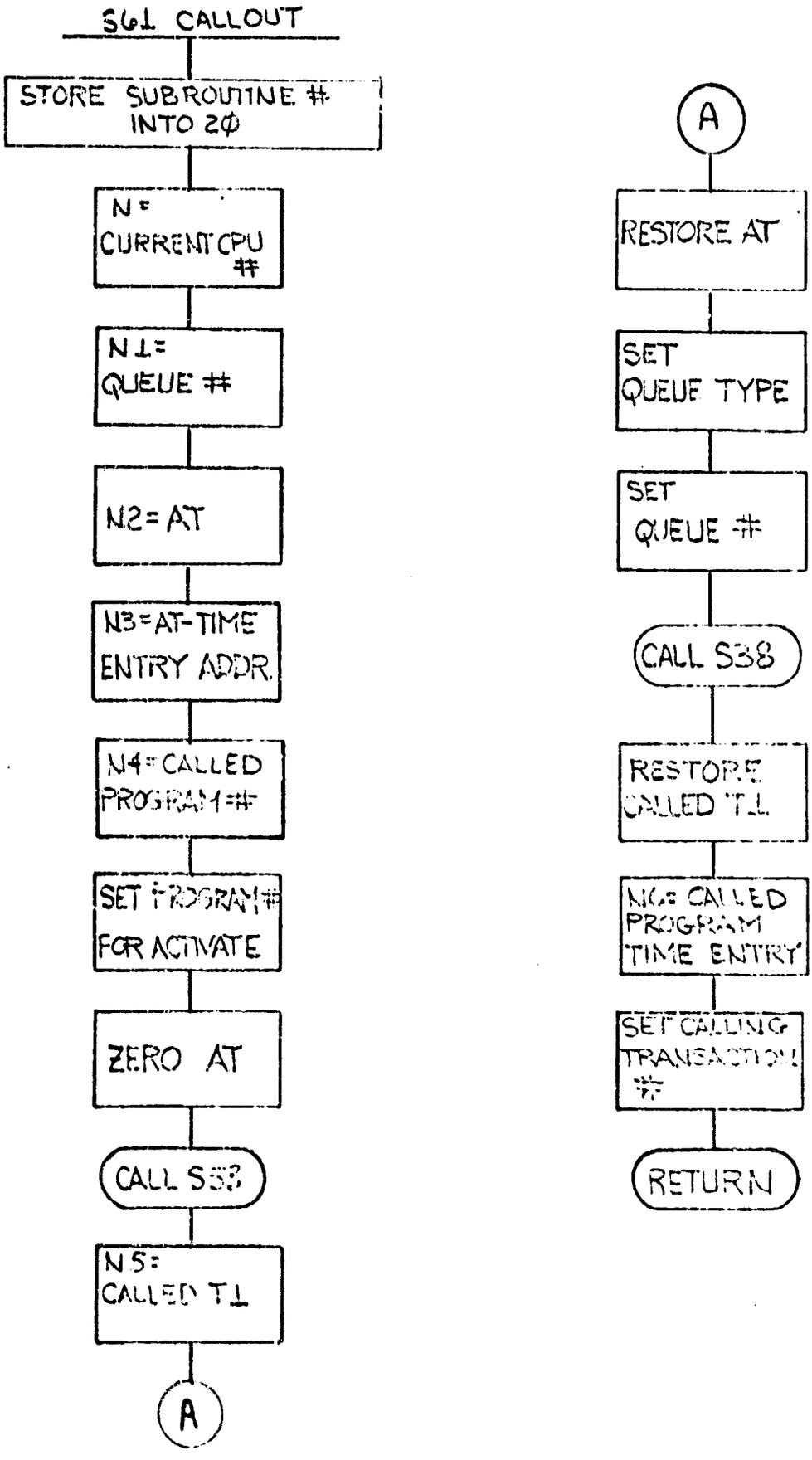

556 DESTROY
STORE SUBROUTINE# INTO T2.Ø
DESTROY AT
YES
10
NO
DESTROY IOT
YES
40
NO
SET ERROR CODE = 1.
RETURN
10
NX= CURRENT CPU#
N1= AT
N2= W/R BASE ADDRESS
N3= T1 PART I ADDRESS
T2 = Ø
11
11
ALL FILES CLOSED
NO
45
YES
WRITE ALL TRANSACTION INFORMATION
RELASE T1
RETURN
40
TEST ZERO FEC ITEMS
NO
420
ADD IQ TIME TO TOTAL FOR TI
REDUCE # OF IOT OUTSTANDING
FREE IOT I
CLEAR IOT
ZERO N3 I
RETURN
45
SET ERROR CODE = 2
RETURN

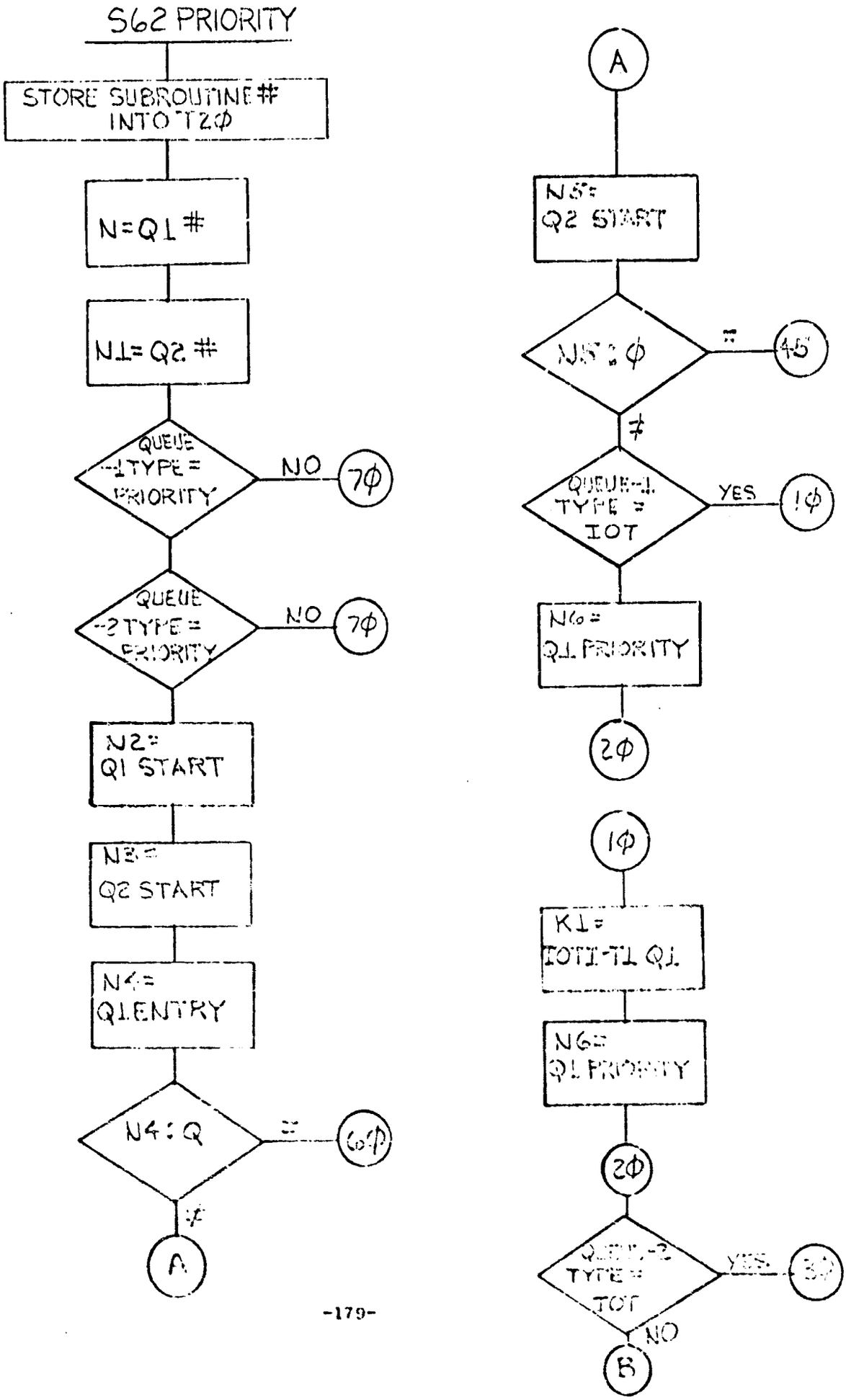

S57 PERIPHERAL
STORE SUBROUTINE # INTO T2Ø
NO QUEUE #
5Ø
N=QUEUE #
QUEUE TYPE NOT TI
YES
ØØ1
NO
N1 = QUEUE POINTER
QUEUE POINTER ERROR
YES
ØØ2
NO
N2= TI ADDRESS
6Ø
5Ø
N= CURRENT CHNL #
N2=AT
6Ø
6Ø
VALID TI
NO
ØØ3
N3= W/R BASE ADDRESS
VALID W/R BASE ADDRESS
NO
ØØ4
YES
N4= FILE VECTOR TABLE ADDRESS
N5= TOTAL # OF FILES
N5 : Ø
=
15Ø
≠
INVALID FILE VECTOR TABLE ADD
YES
2Ø5
VALID # OF FILES
NO
8Ø

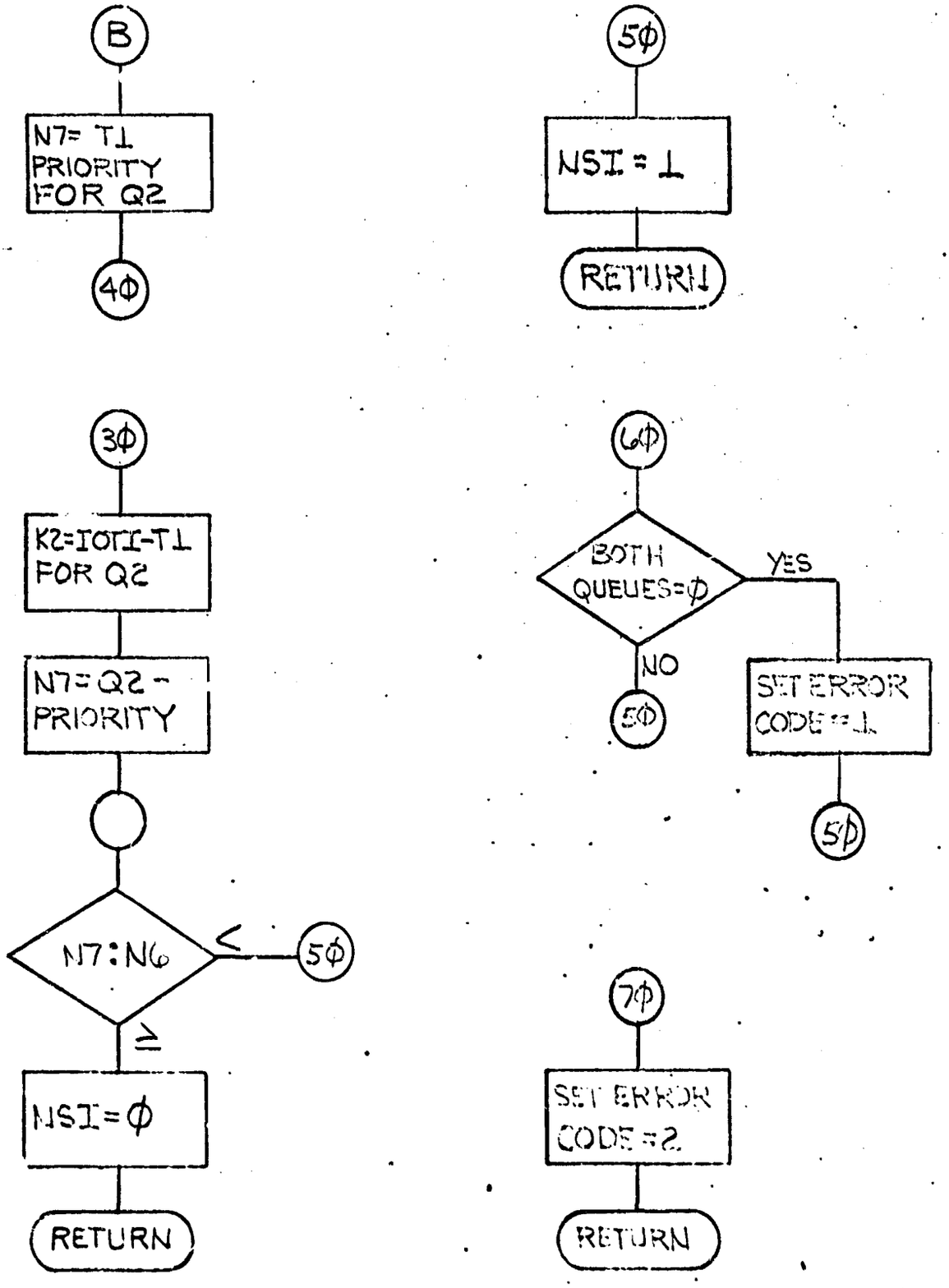

80
N6=REAL FILE NUMBER
N7=SEIZING CODE
INVALID REAL FILE NUMBER
YES
90
NO
FILE SEIZED
YES
160
NO
N8=DEVICE ADDRESS
DEVICE SEIZED
YES
130
NO
TEST FILE/DEVICE TO BE SEIZED
NO
100
YES
NOT SEIZABLE
YES
100
DEVICE BUSY BUT NOT SEIZED
YES
180
A
A
SEIZE DEVICE
100
TEST FILE TO BE SEIZED
NO
140
YES
SEIZE FILE
SET FILE SEIZE TIME
INCREMENT FILE
DECREMENT FILE COUNTER
FILE CTR: 0
>
80
≤
SET I/O AVAILABLE EXIT
RETURN

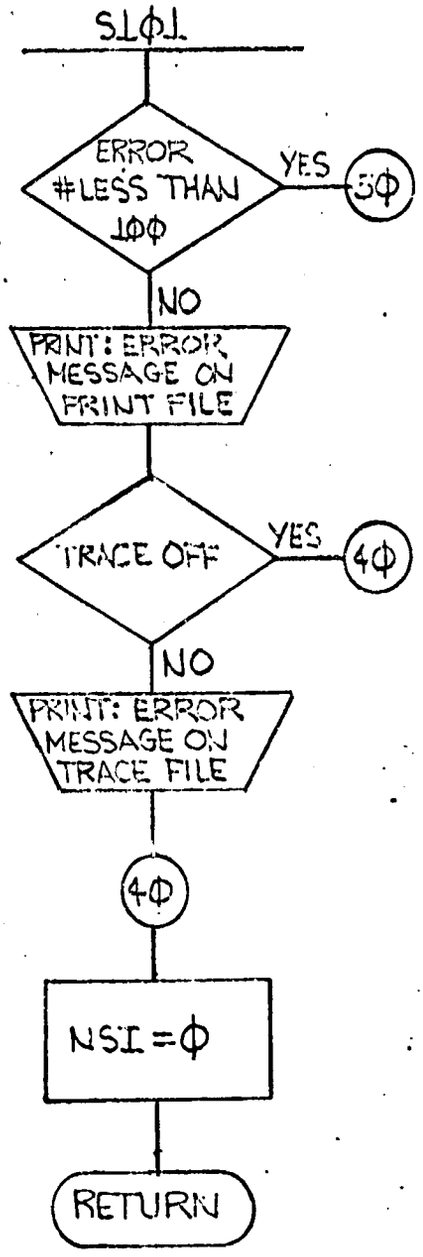

160
SET I/O NOT AVAILABLE EXIT
N4 = FILE VECTOR TABLE ADDRESS
N5 = TOTAL NUMBER OF FILES
RELEASE ALL SEIZED FILES
RETURN
901
SET ERROR CODE = 1
RETURN
902
SET ERROR CODE = 2
RETURN
903
SET ERROR CODE = 3
RETURN
904
SET ERROR CODE = 4
RETURN
905
SET ERROR CODE = 5
RETURN
906
SET ERROR CODE = 6
RETURN
907
SET ERROR CODE = 7
RETURN
180
TEST NEXT SEIZING CODE FREE
NO
160
STORE SEIZING CODE INTO NEXT SEIZING CD SLOT
100

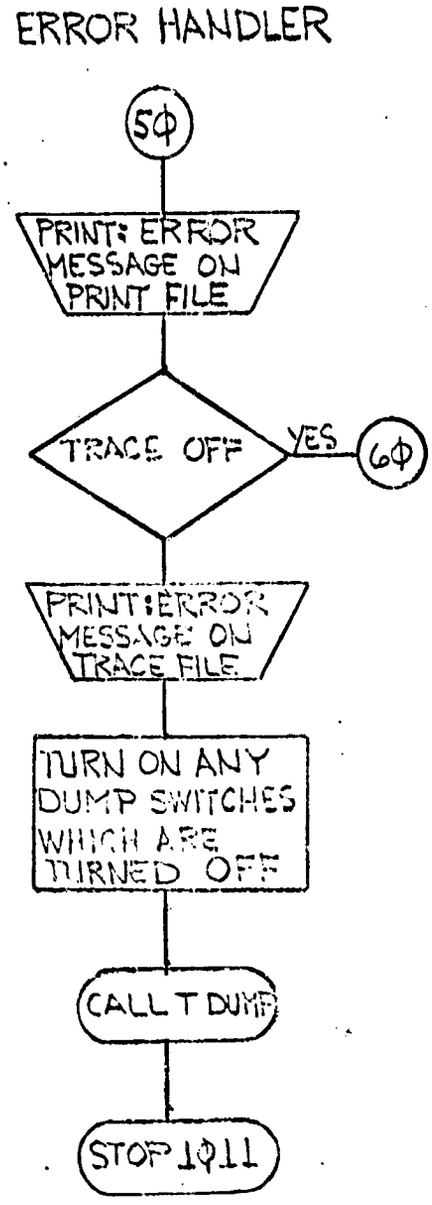

S59 MATCH
STORE SUBROUTINE # INTO T2Ø
N=CURRENT CPU #
N1= AT
N2= QUEUE #
QUEUE # EQUAL Ø
YES
3Ø
NO
N3= QUEUE POINTER
N4= QUEUE ENTRY
QUEUE TYPE = TI
NO
4Ø
YES
N5= TI TIME ENTRY
A
A
N6= CALLED PROGRAM #
CALLED PROGRAM #= AT
YES
2Ø
NO
N7= AT TIME ENTRY
N8=CALLING TI ADDRESS
CALLING TI # = QUEUE TI #
NO
2Ø
NSI=Ø
RETURN
2Ø
NSI=Ø
RETURN
3Ø
SET ERROR CODE = 1
RETURN
4Ø
SET ERROR
RETURN

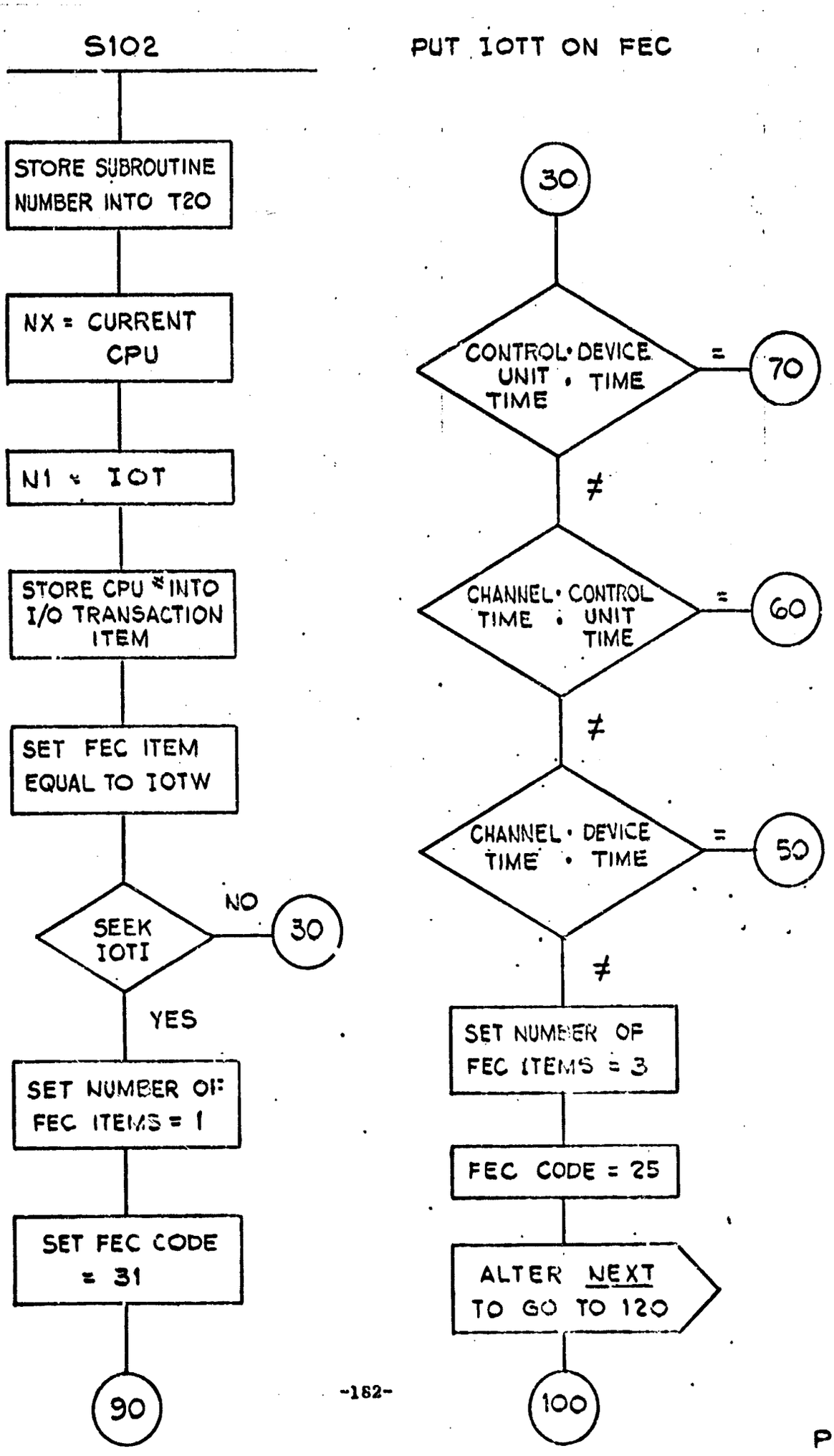

S61 CALLOUT
STORE SUBROUTINE # INTO 2Φ
N= CURRENT CPU #
N1= QUEUE #
N2= AT
N3=AT-TIME ENTRY ADDR.
N4= CALLED PROGRAM #
SET PROGRAM # FOR ACTIVATE
ZERO AT
CALL S38
N5= CALLED T1
A
A
RESTORE AT
SET QUEUE TYPE
SET QUEUE #
CALL S38
RESTORE CALLED T1
N6= CALLED PROGRAM TIME ENTRY
SET CALLING TRANSACTION #
RETURN


S62 PRIORITY
STORE SUBROUTINE # INTO T2Φ
N=Q1 #
N1=Q2 #
QUEUE -1 TYPE = PRIORITY
NO
7Φ
QUEUE -2 TYPE = PRIORITY
NO
7Φ
N2= Q1 START
N3= Q2 START
N4= Q1 ENTRY
N4 : Q
=
6Φ
≠
A
A
N5= Q2 START
N5 : Φ
=
45
≠
QUEUE -1 TYPE = IOT
YES
1Φ
N6= Q1 PRIORITY
2Φ
1Φ
K1= IOT1-T1 Q1
N6= Q1 PRIORITY
2Φ
QUEUE -2 TYPE = IOT
YES
3Φ
NO
B

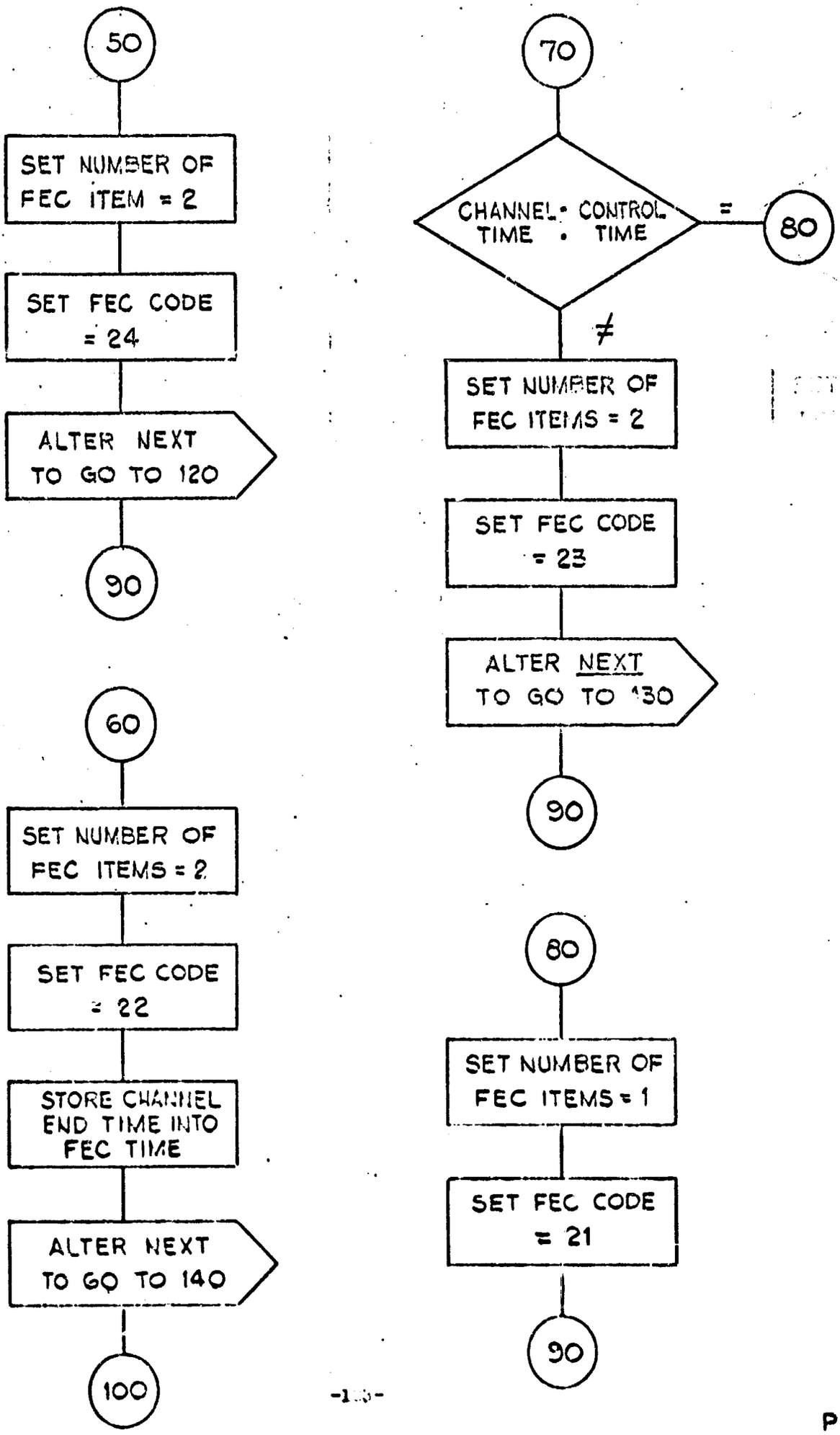

B
N7= T1
PRIORITY
FOR Q2
4Φ
5Φ
NSI = 1
RETURN
3Φ
K2=IOTI-T1
FOR Q2
N7= Q2 -
PRIORITY
N7 : N6
<
5Φ
≥
NSI=Φ
RETURN
6Φ
BOTH
QUEUES=Φ
YES
NO
5Φ
SET ERROR
CODE = 1
5Φ
7Φ
SET ERROR
CODE = 2
RETURN

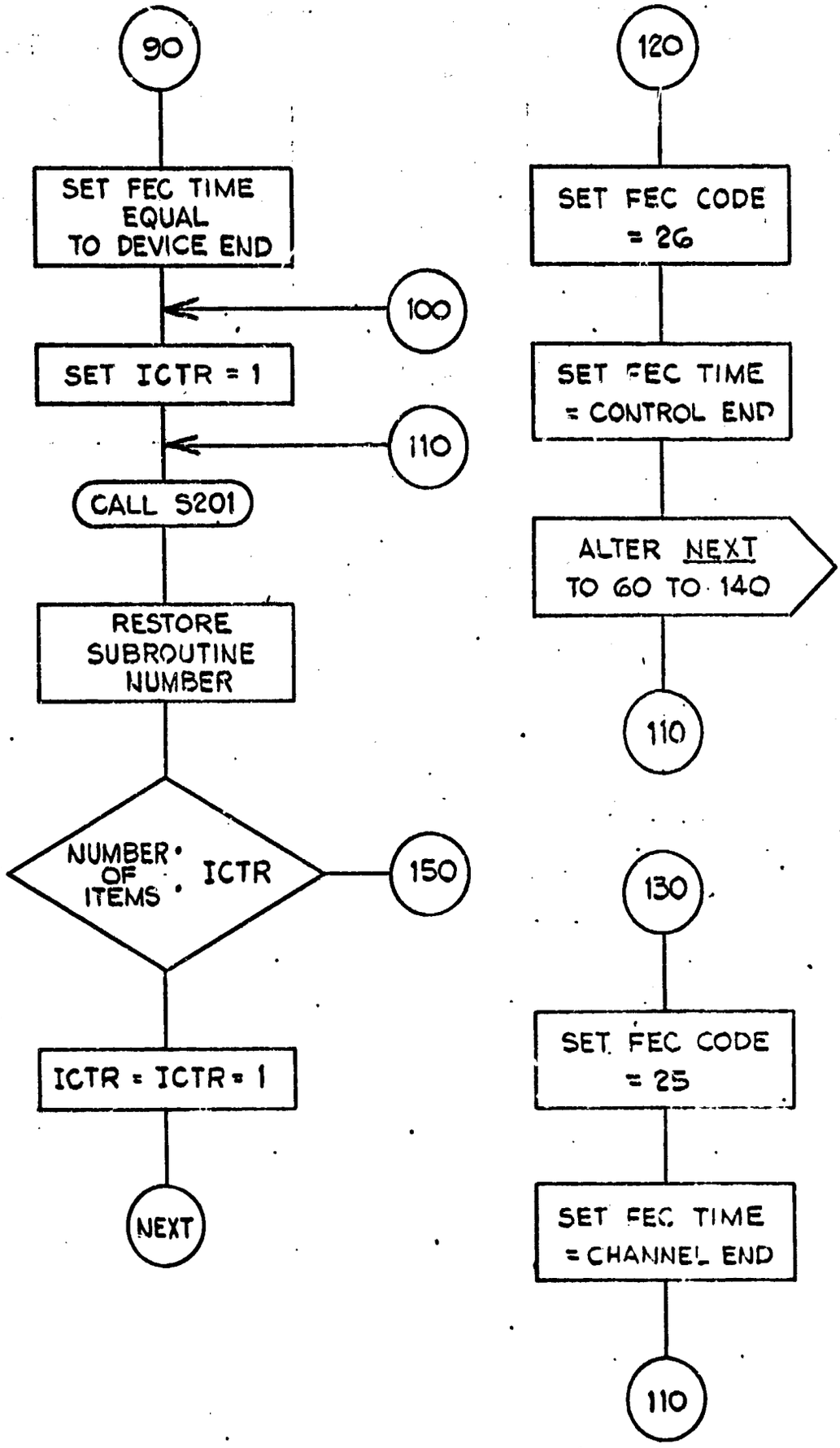

S1Φ1
ERROR #LESS THAN 1ΦΦ
YES
5Φ
NO
PRINT: ERROR MESSAGE ON PRINT FILE
TRACE OFF
YES
4Φ
NO
PRINT: ERROR MESSAGE ON TRACE FILE
4Φ
NSI=Φ
RETURN

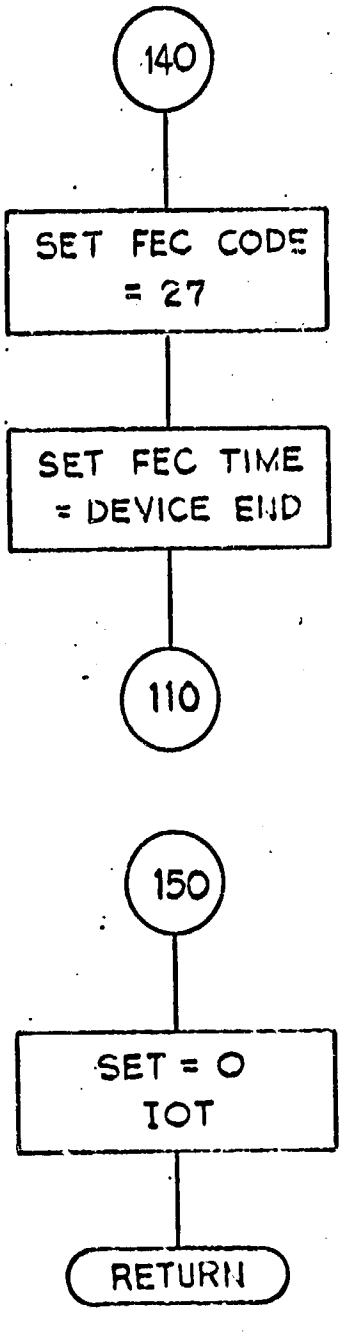

ERROR HANDLER
5Φ
PRINT: ERROR MESSAGE ON PRINT FILE
TRACE OFF
YES
6Φ
PRINT: ERROR MESSAGE ON TRACE FILE
TURN ON ANY DUMP SWITCHES WHICH ARE TURNED OFF
CALL T DUMP
STOP 1Φ11


S102
PUT IOTT ON FEC
STORE SUBROUTINE NUMBER INTO T20
NX = CURRENT CPU
N1 = IOT
STORE CPU # INTO I/O TRANSACTION ITEM
SET FEC ITEM EQUAL TO IOTW
SEEK IOTI
NO
30
YES
SET NUMBER OF FEC ITEMS = 1
SET FEC CODE = 31
90
30
CONTROL • DEVICE UNIT • TIME TIME
=
70
≠
CHANNEL • CONTROL TIME • UNIT TIME
=
60
≠
CHANNEL • DEVICE TIME • TIME
=
50
≠
SET NUMBER OF FEC ITEMS = 3
FEC CODE = 25
ALTER NEXT TO GO TO 120
100

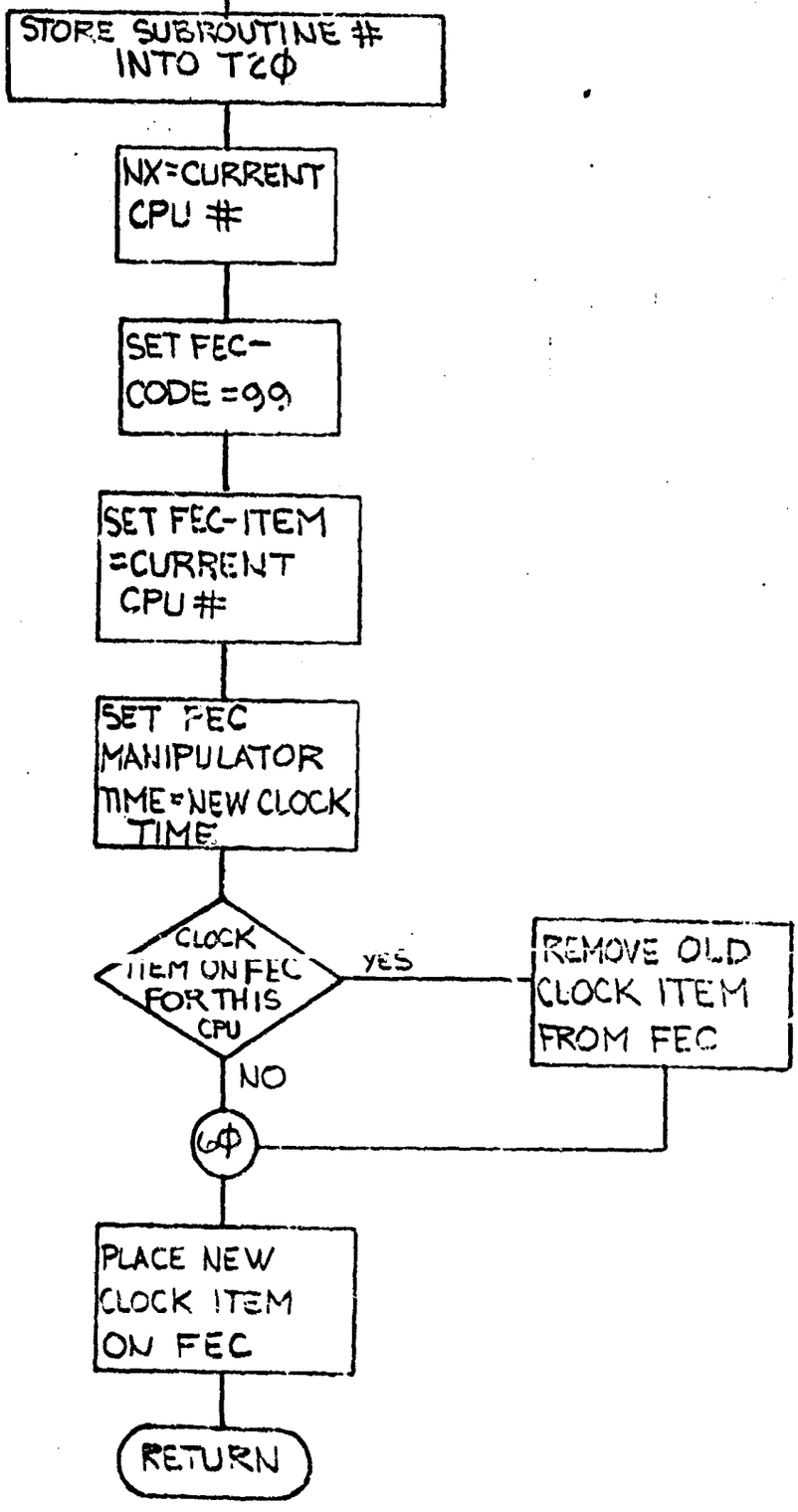
PG 1


50
SET NUMBER OF FEC ITEM = 2
SET FEC CODE = 24
ALTER NEXT TO GO TO 120
90
60
SET NUMBER OF FEC ITEMS = 2
SET FEC CODE = 22
STORE CHANNEL END TIME INTO FEC TIME
ALTER NEXT TO GO TO 140
100
70
CHANNEL CONTROL TIME . TIME
=
80
≠
SET NUMBER OF FEC ITEMS = 2
SET FEC CODE = 23
ALTER NEXT TO GO TO 130
90
80
SET NUMBER OF FEC ITEMS = 1
SET FEC CODE = 21
90


90
SET FEC TIME EQUAL TO DEVICE END
100
SET ICTR = 1
110
CALL S201
RESTORE SUBROUTINE NUMBER
NUMBER OF ITEMS
ICTR
150
ICTR = ICTR + 1
NEXT
120
SET FEC CODE = 26
SET FEC TIME = CONTROL END
ALTER NEXT TO GO TO 140
110
130
SET FEC CODE = 25
SET FEC TIME = CHANNEL END
110


140
SET FEC CODE
= 27
SET FEC TIME
= DEVICE END
110
150
SET = O
IOT
RETURN

S103 PUT CLOCK ITEM ON FEC



S104 PLACE IT ON FEC

STORE SUBROUTINE NUMBER INTO T2Ø

NX = CURRENT CPU NUMBER

FEC-CODE = CURRENT CPU NUMBER

FEC-ITEM = COT

NA = COT

NB = COT - TIME POINTER

TIME OFF FEC = CURRENT CLOCK & ADVANCE TIME

SET FEC MANIPULATOR TIME = TIME OFF FEC

SET CPU NUMBER INTO TRANSACTION ITEM

CALL S201

RESTORE SUBROUTINE NUMBER

(A)

(A)

SET COT = ZERO

N = 1

(40)

ENTRY S105 CYCLE

STORE SUBROUTINE NUMBER INTO T2Ø

N = 1

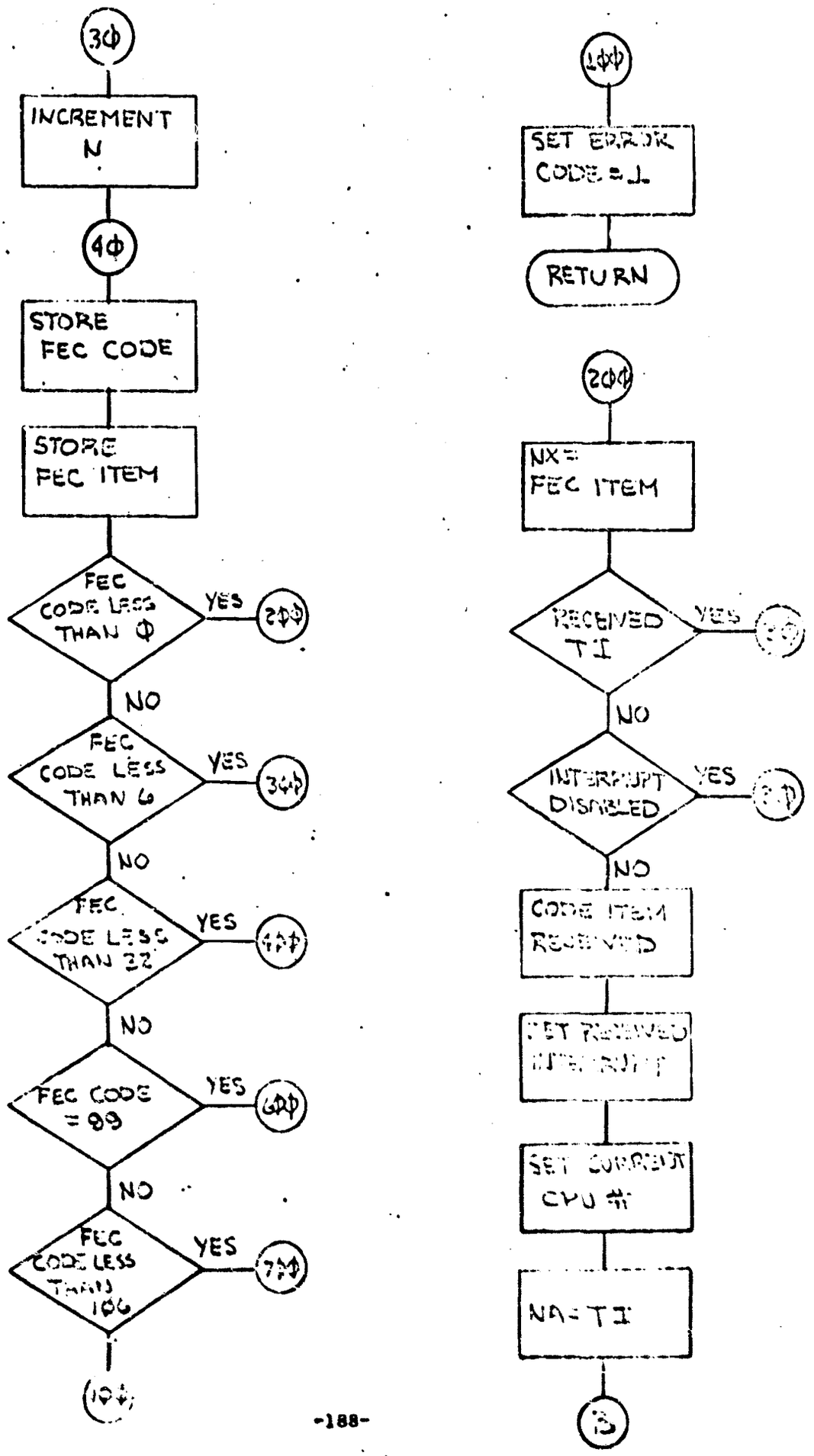
TEST CHAIN EMPTY

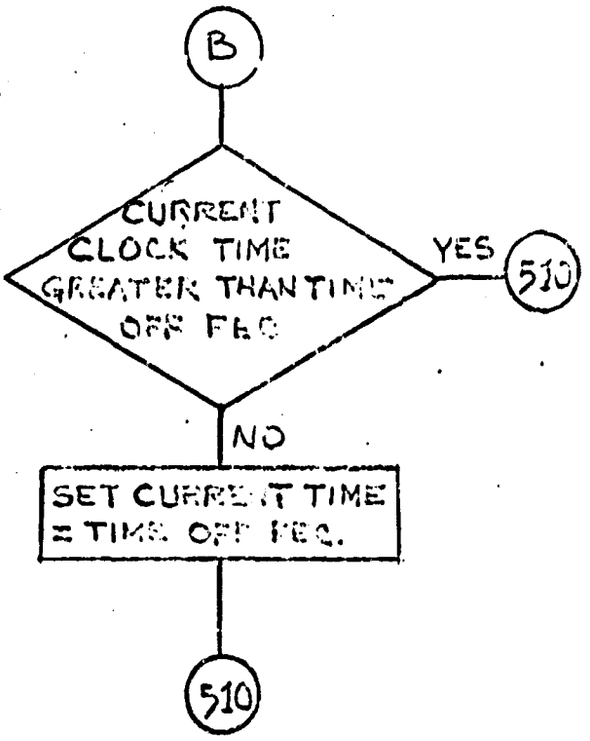
NO → (40)

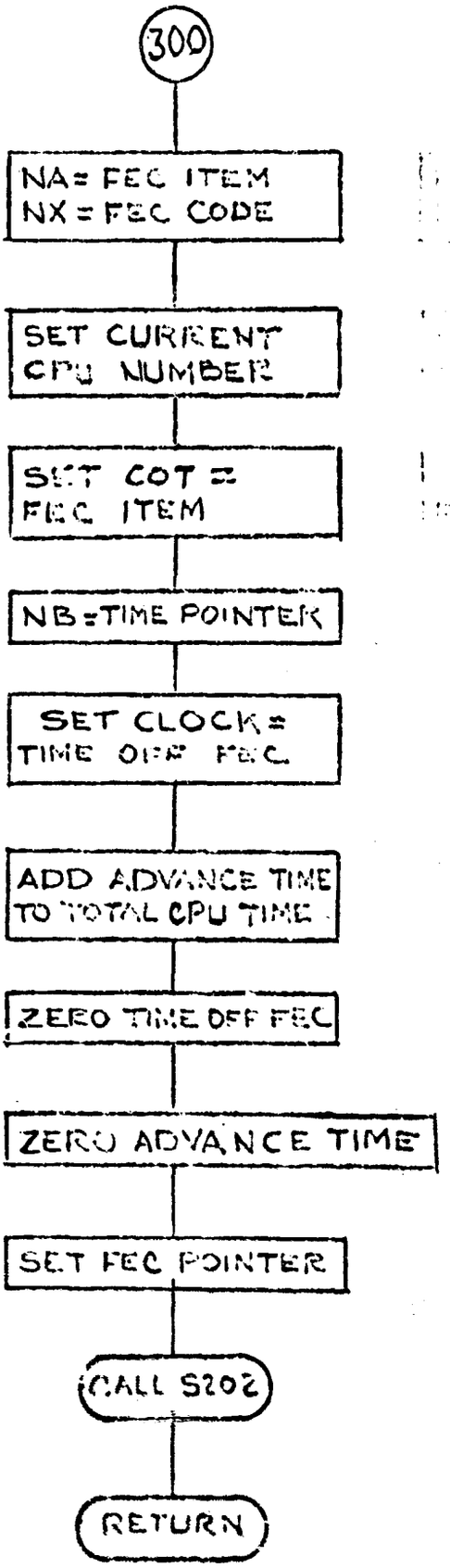
YES

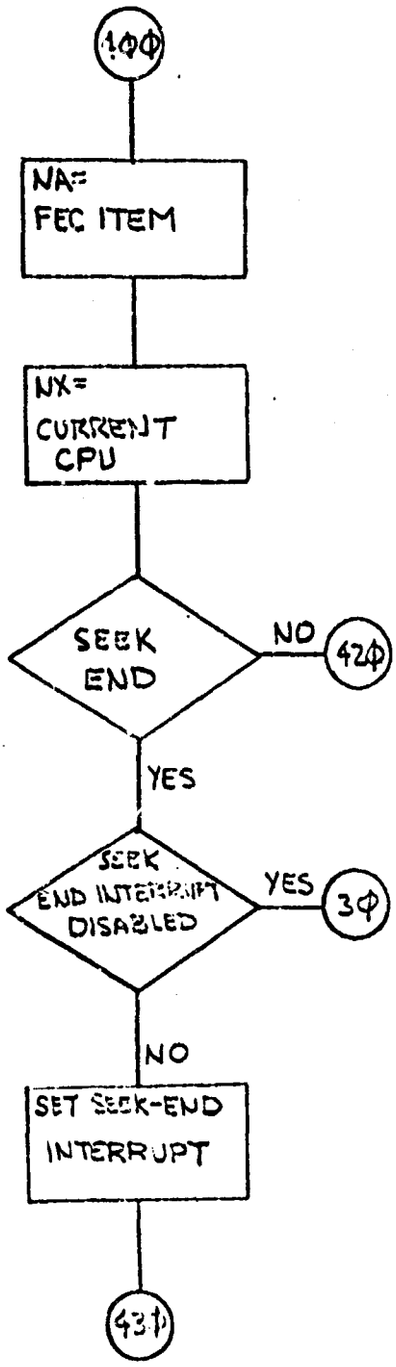
SET ERROR CODE 1

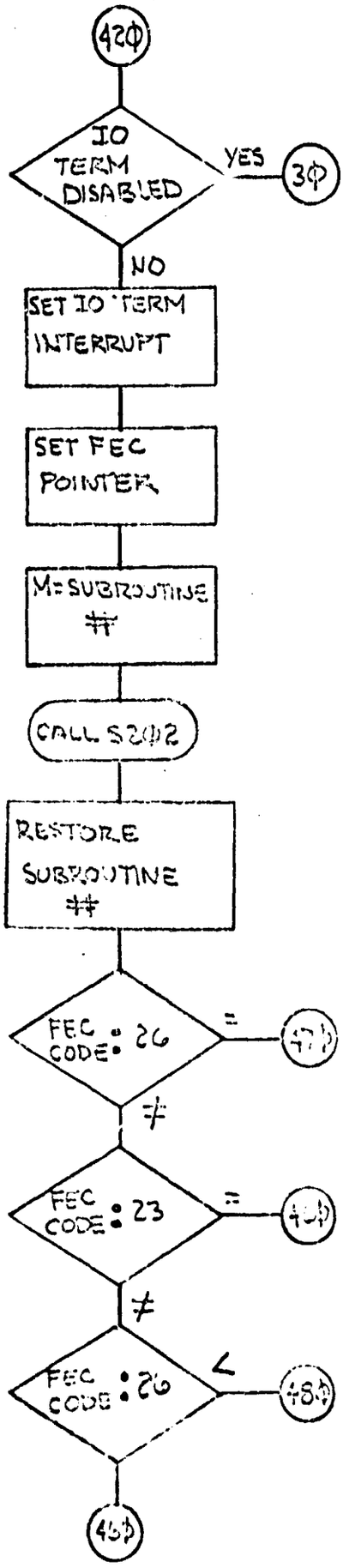
RETURN

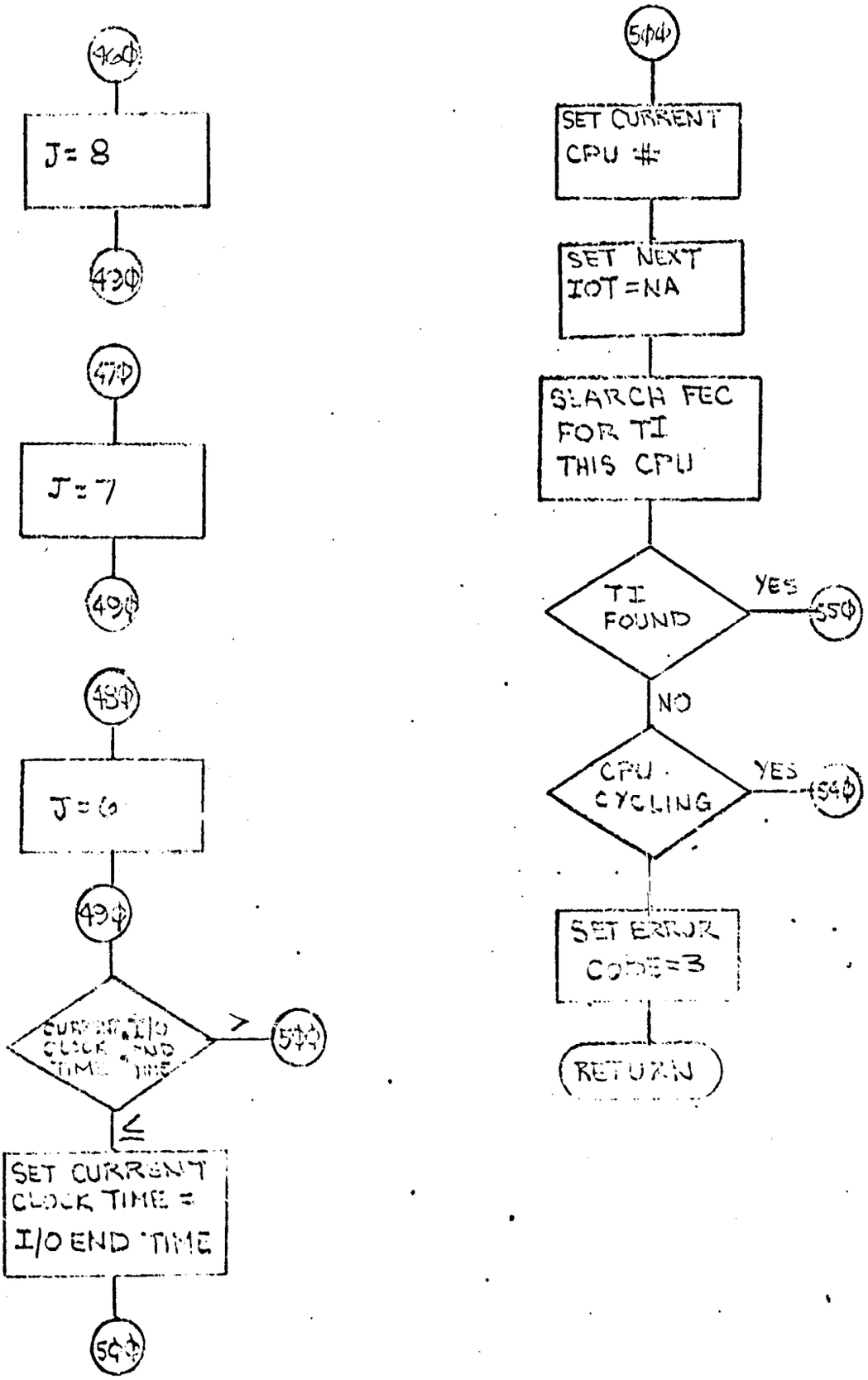

3Φ
INCREMENT N
4Φ
STORE FEC CODE
STORE FEC ITEM
FEC CODE LESS THAN Φ
YES
2ΦΦ
NO
FEC CODE LESS THAN 6
YES
34Φ
NO
FEC CODE LESS THAN 32
YES
NO
FEC CODE = 89
YES
6ΦΦ
NO
FEC CODE LESS THAN 106
YES
7ΦΦ
10Φ
1ΦΦ
SET ERROR CODE = 1
RETURN
2ΦΦ
NX= FEC ITEM
RECEIVED TI
YES
NO
INTERRUPT DISABLED
YES
NO
CODE ITEM RECEIVED
SET RECEIVED INTERRUPT
SET CURRENT CPU #
NA= TI
13

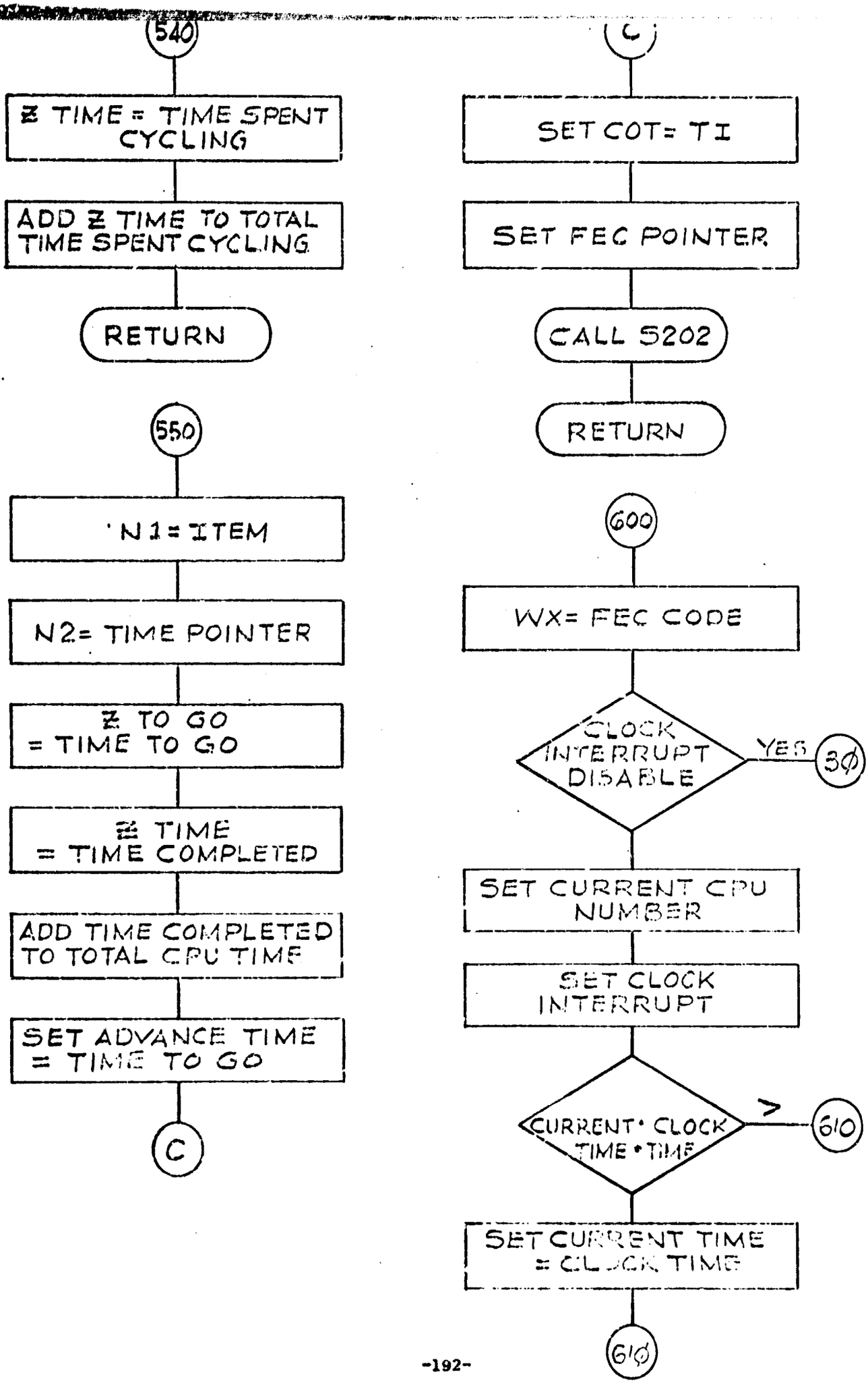

B
CURRENT CLOCK TIME GREATER THAN TIME OFF FEC
YES
510
NO
SET CURRENT TIME = TIME OFF FEC.
510

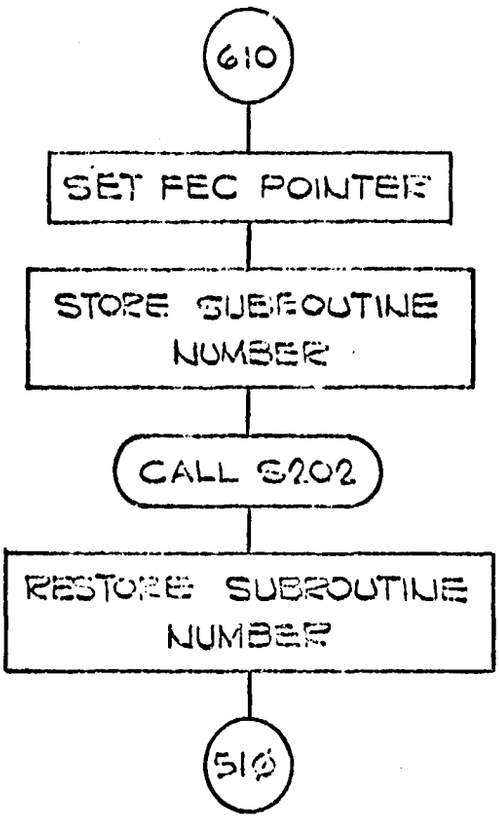

300
NA = FEC ITEM
NX = FEC CODE
SET CURRENT CPU NUMBER
SET COT = FEC ITEM
NB = TIME POINTER
SET CLOCK = TIME OFF FEC.
ADD ADVANCE TIME TO TOTAL CPU TIME
ZERO TIME OFF FEC
ZERO ADVANCE TIME
SET FEC POINTER
CALL S202
RETURN

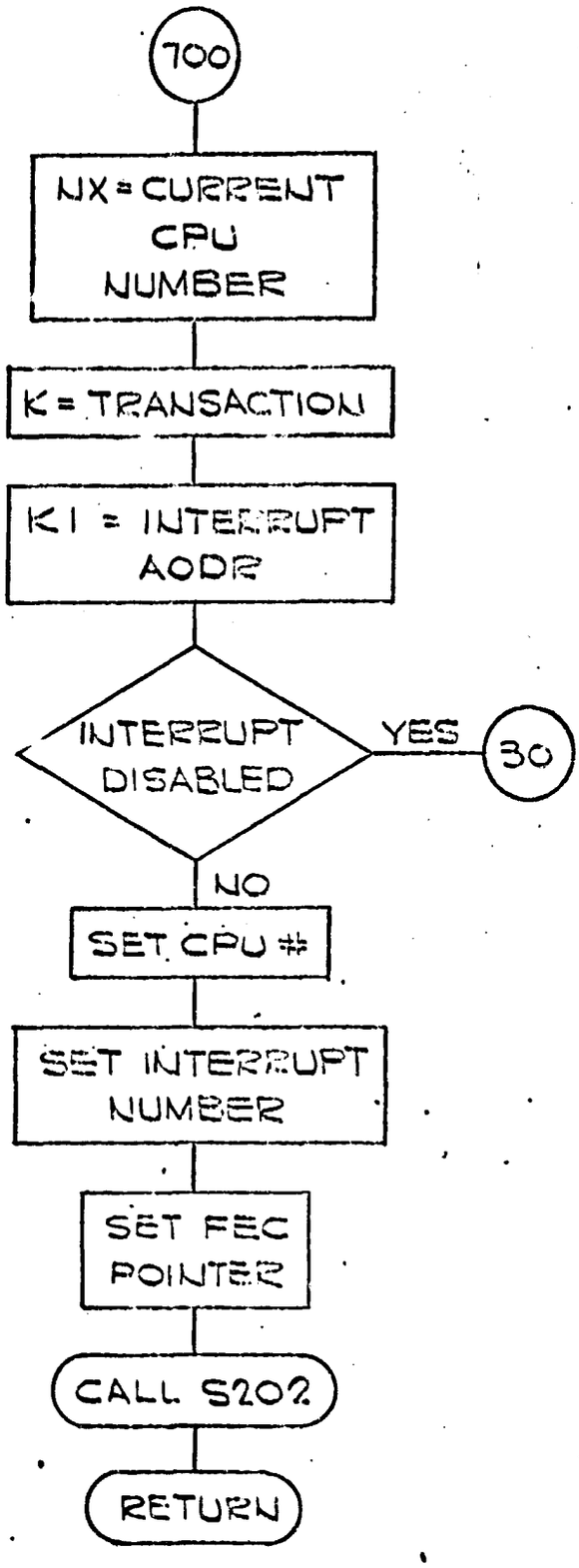

10Φ
NA=
FEC ITEM
NX=
CURRENT
CPU
SEEK
END
NO
42Φ
YES
SEEK
END INTERRUPT
DISABLED
YES
3Φ
NO
SET SEEK-END
INTERRUPT
43Φ


42Φ
IO
TERM
DISABLED
YES
3Φ
NO
SET IO TERM
INTERRUPT
SET FEC
POINTER
M= SUBROUTINE
##
CALL S202
RESTORE
SUBROUTINE
##
FEC
CODE = 26
=
47Φ
≠
FEC
CODE = 23
=
4-Φ
≠
FEC
CODE = 26
<
48Φ
45Φ

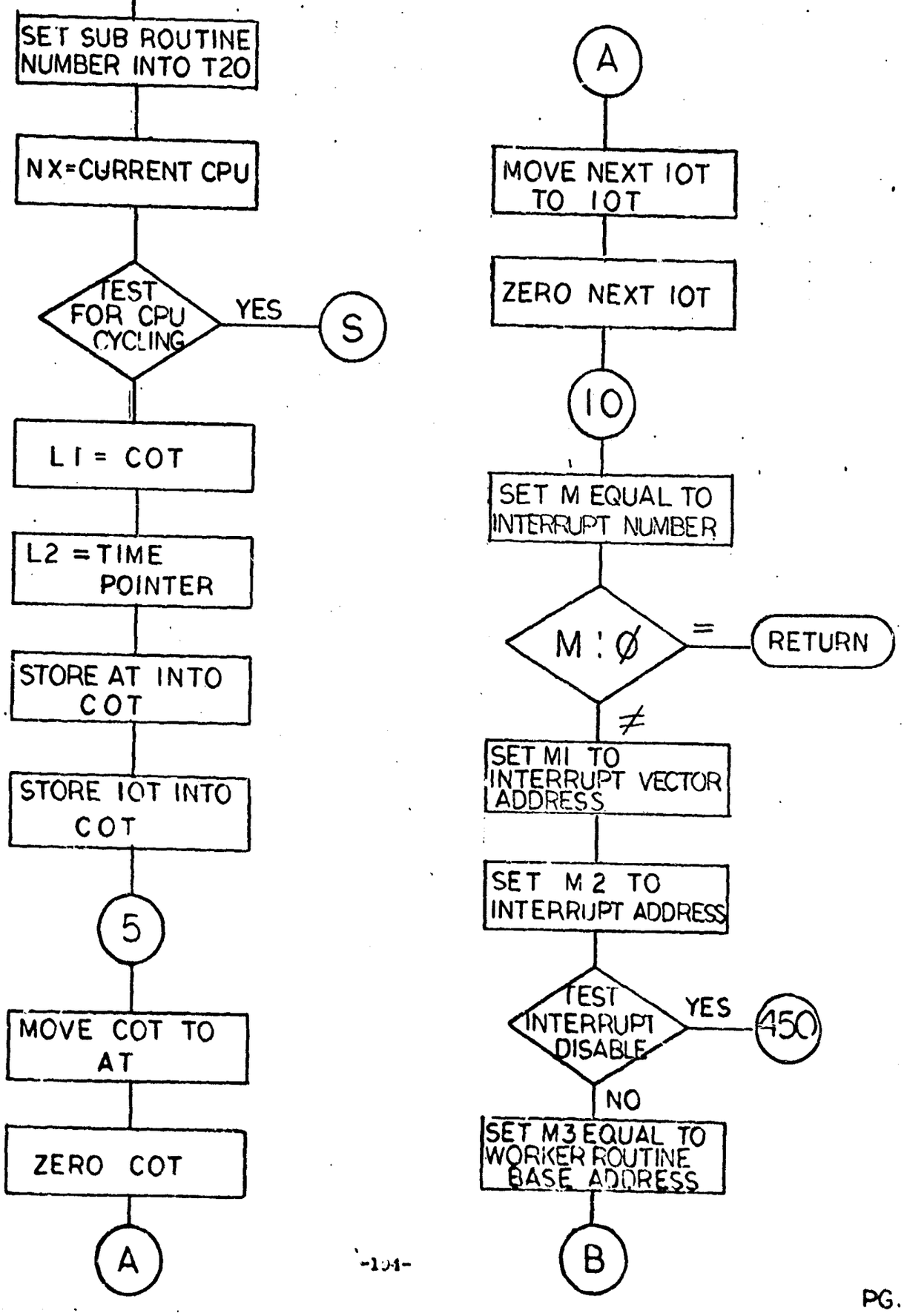

420
J = 8
430
470
J = 7
490
480
J = 6
490
CURRENT I/O CLOCK END TIME
>
510
≤
SET CURRENT CLOCK TIME = I/O END TIME
500
504
SET CURRENT CPU #
SET NEXT IOT = NA
SEARCH FEC FOR TI THIS CPU
TI FOUND
YES
550
NO
CPU CYCLING
YES
540
SET ERROR CODE = 3
RETURN


540
Σ TIME = TIME SPENT CYCLING
ADD Σ TIME TO TOTAL TIME SPENT CYCLING
RETURN
550
N1 = ITEM
N2 = TIME POINTER
Σ TO GO = TIME TO GO
Σ TIME = TIME COMPLETED
ADD TIME COMPLETED TO TOTAL CPU TIME
SET ADVANCE TIME = TIME TO GO
C
C
SET COT = TI
SET FEC POINTER
CALL 5202
RETURN
600
WX = FEC CODE
CLOCK INTERRUPT DISABLE
YES
30
SET CURRENT CPU NUMBER
SET CLOCK INTERRUPT
CURRENT CLOCK TIME > TIME
610
SET CURRENT TIME = CLOCK TIME
610

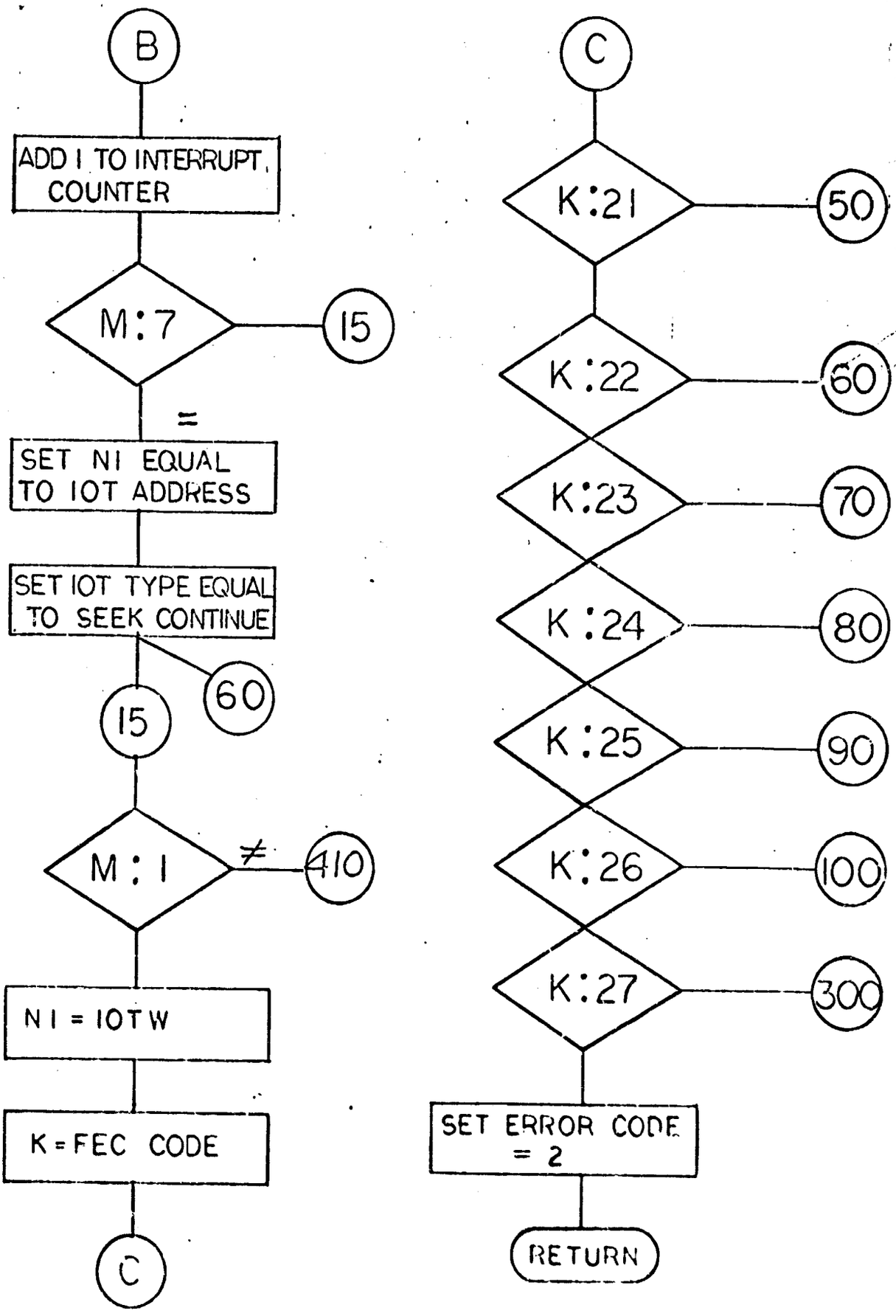

610
SET FEC POINTER
STORE SUBROUTINE NUMBER
CALL S202
RESTORE SUBROUTINE NUMBER
510


700
NX = CURRENT CPU NUMBER
K = TRANSACTION
K1 = INTERRUPT AODR
INTERRUPT DISABLED
YES
30
NO
SET CPU #
SET INTERRUPT NUMBER
SET FEC POINTER
CALL S202
RETURN

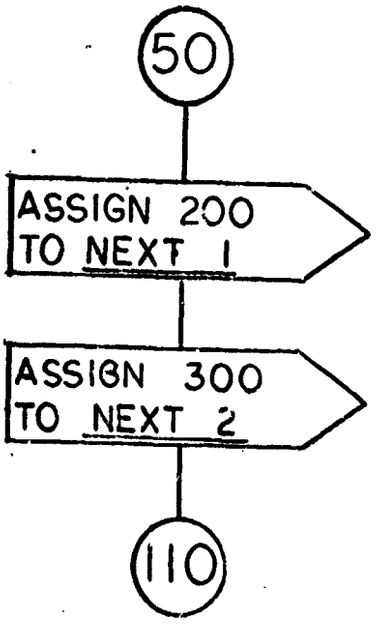
S110 INTERRUPT HANDLER

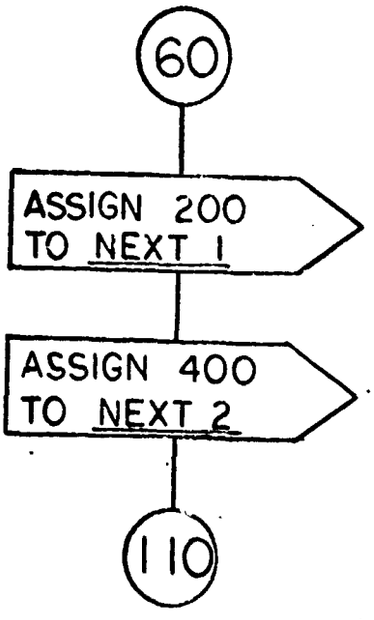


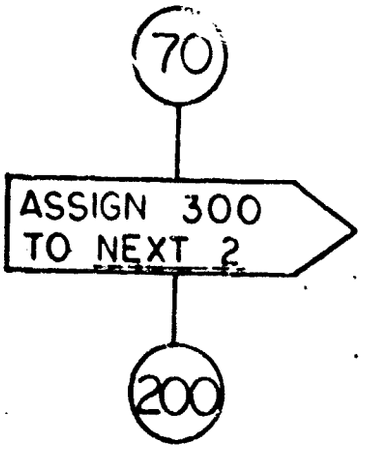
PG.1

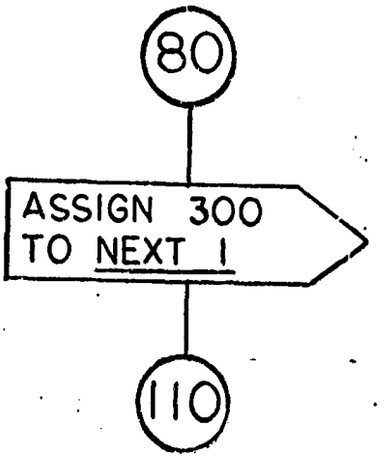

B
ADD 1 TO INTERRUPT COUNTER
M:7
15
=
SET N1 EQUAL TO IOT ADDRESS
SET IOT TYPE EQUAL TO SEEK CONTINUE
60
15
M:1
≠
410
N1 = IOTW
K = FEC CODE
C
C
K:21
50
K:22
60
K:23
70
K:24
80
K:25
90
K:26
100
K:27
300
SET ERROR CODE = 2
RETURN

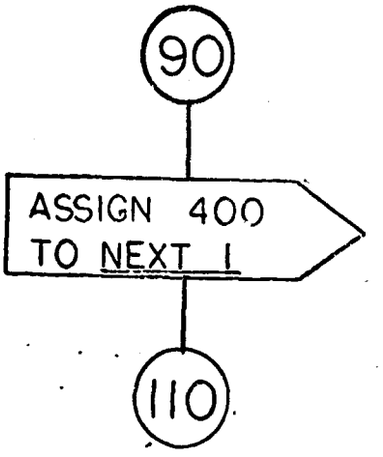
PG 2

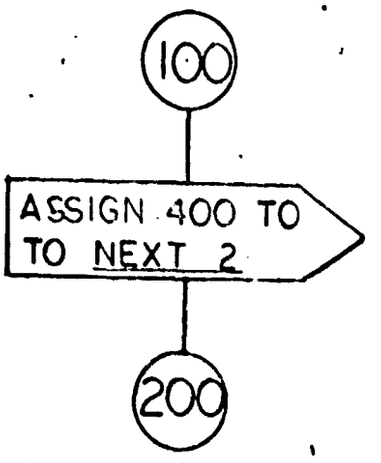

50
ASSIGN 200
TO NEXT 1
ASSIGN 300
TO NEXT 2
110

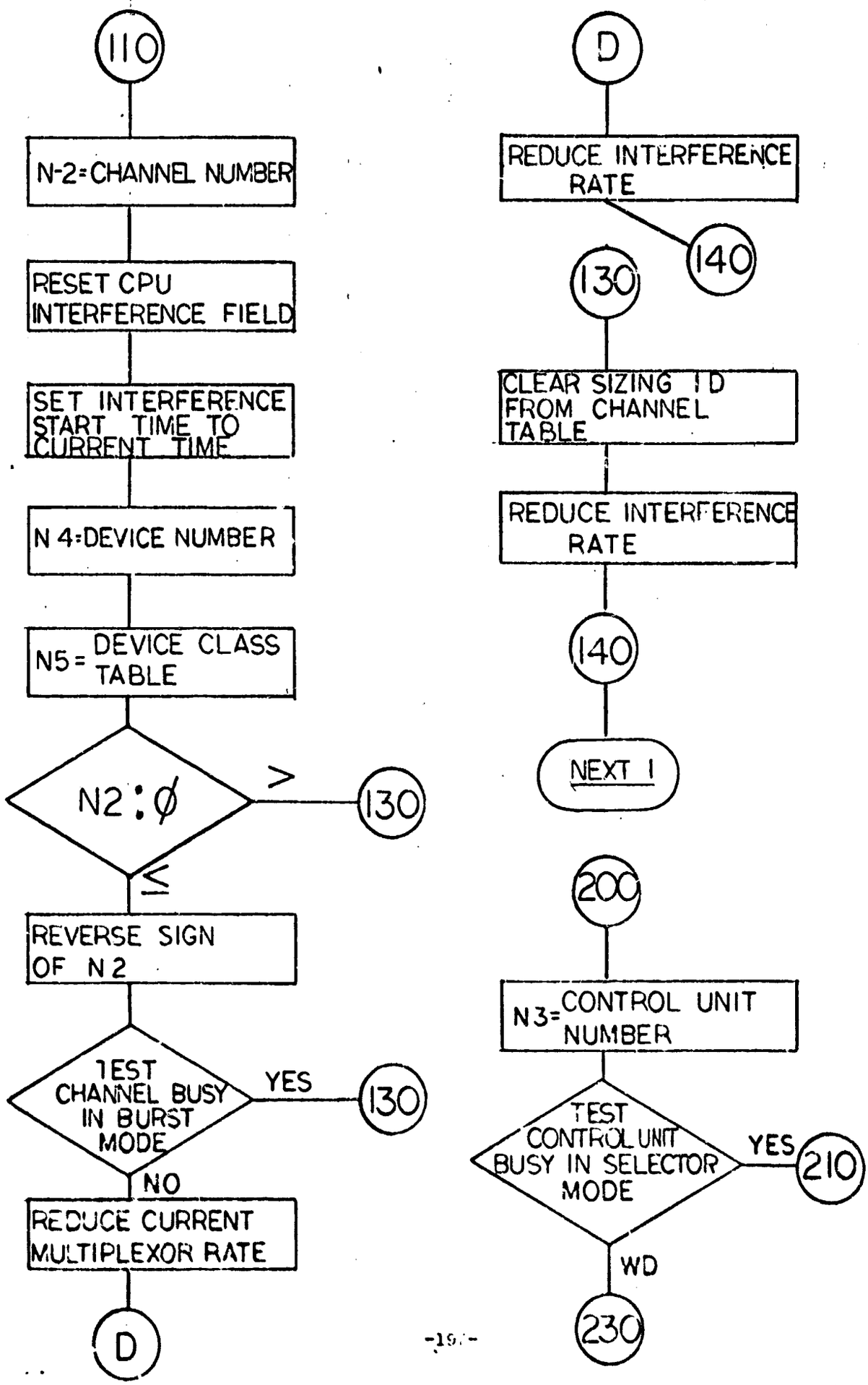

60
ASSIGN 200
TO NEXT 1
ASSIGN 400
TO NEXT 2
110

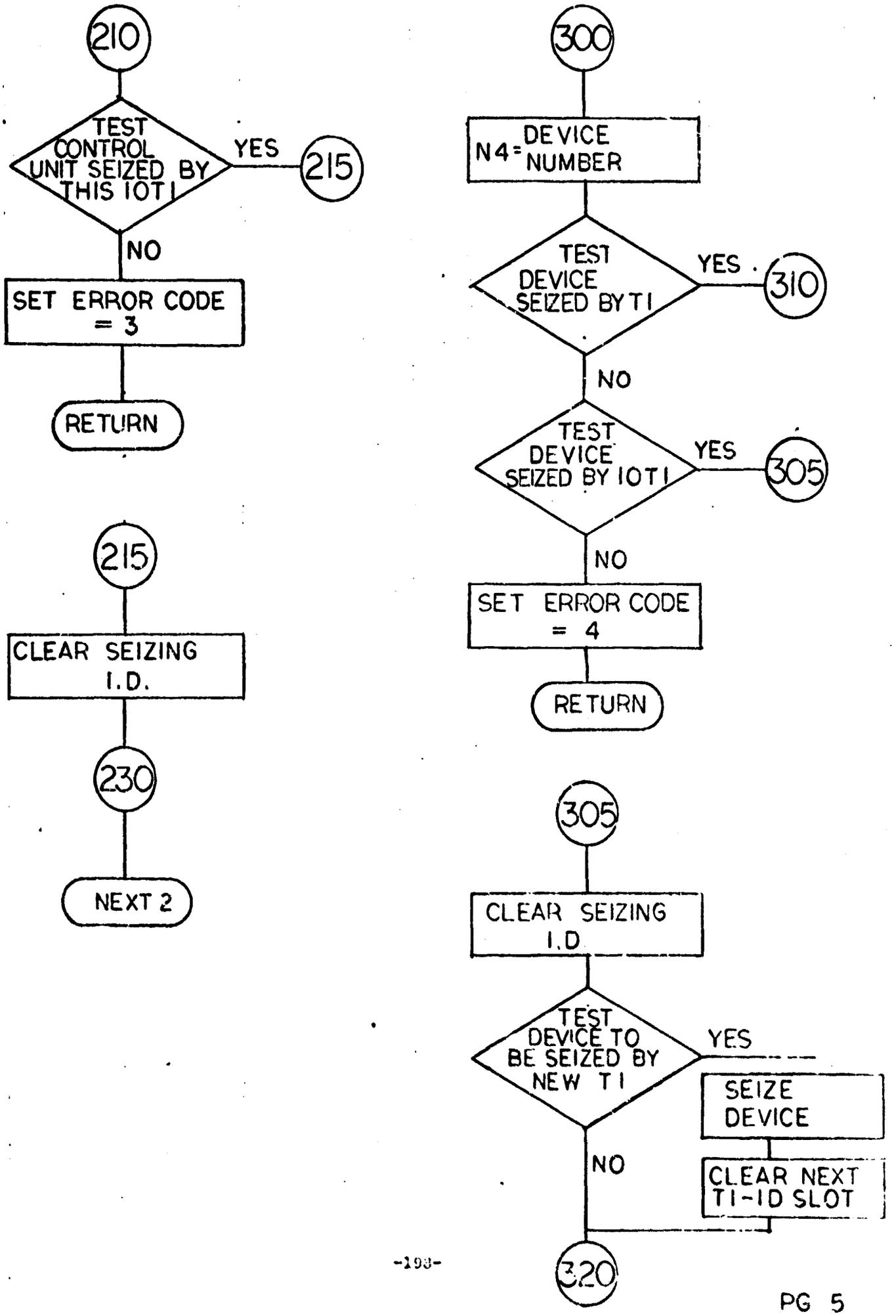

70
ASSIGN 300
TO NEXT 2
200

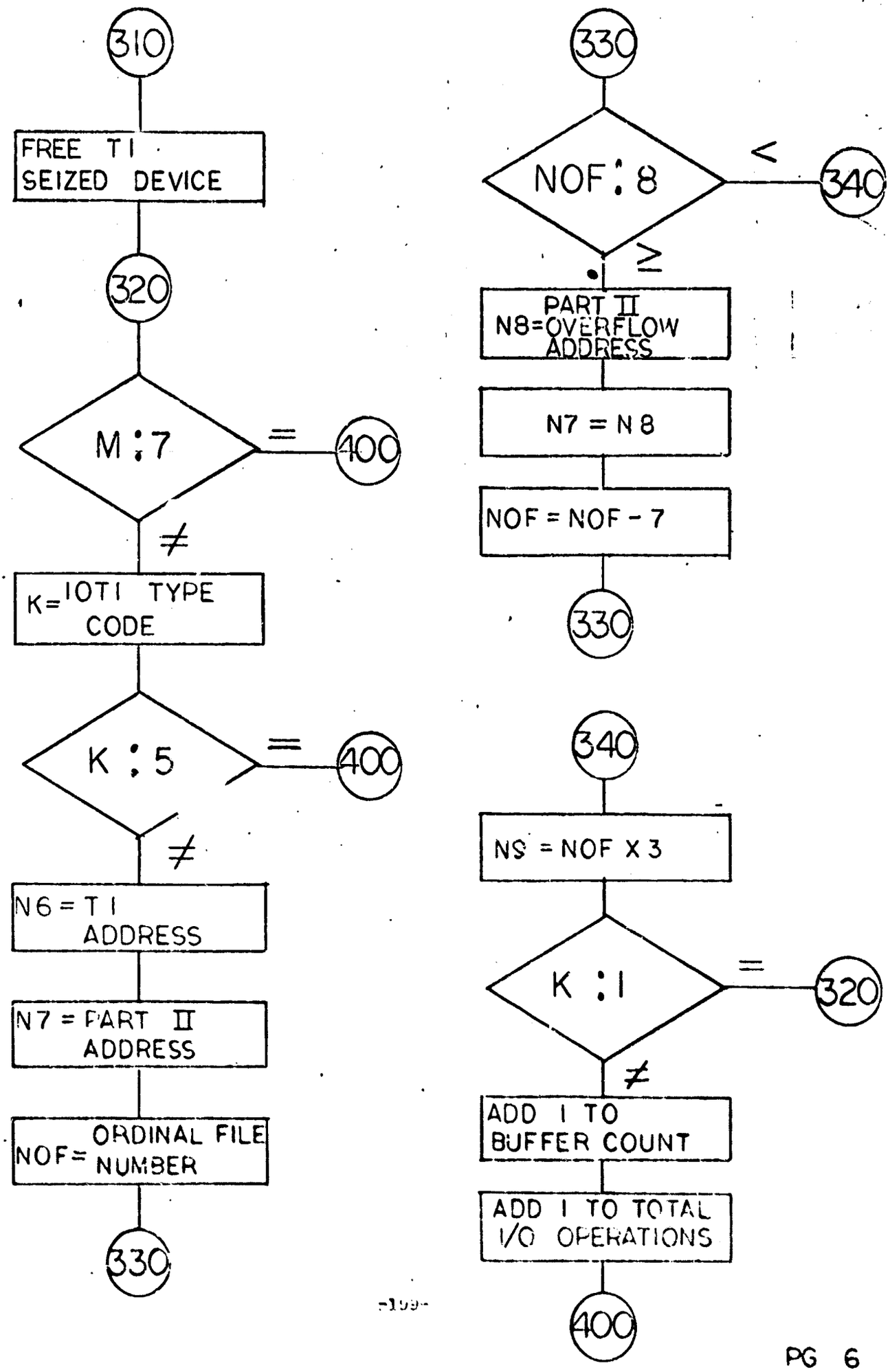

80
ASSIGN 300
TO NEXT 1
110

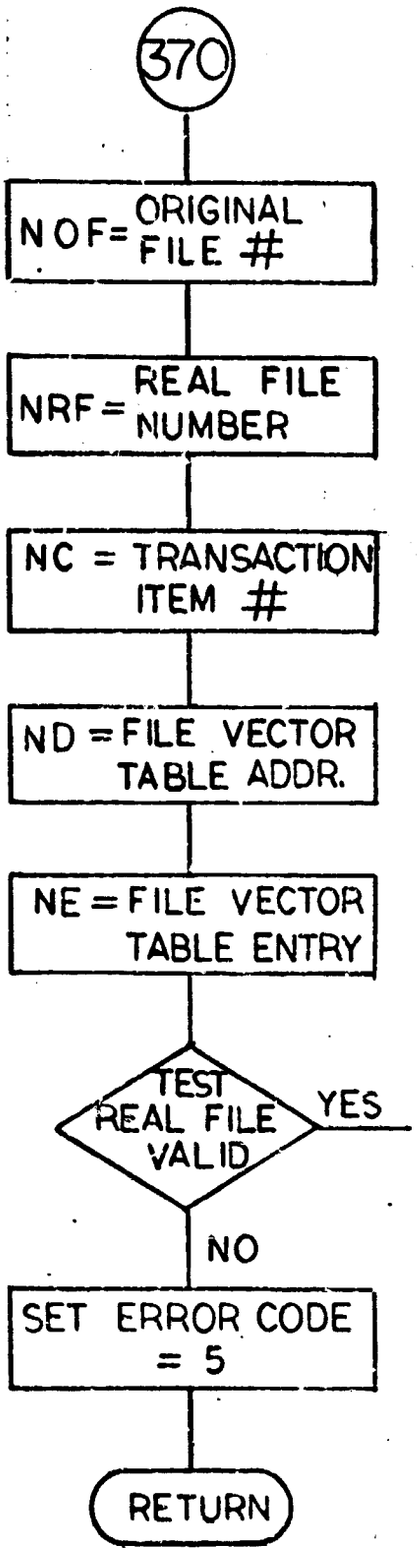

90
ASSIGN 400
TO NEXT 1
110

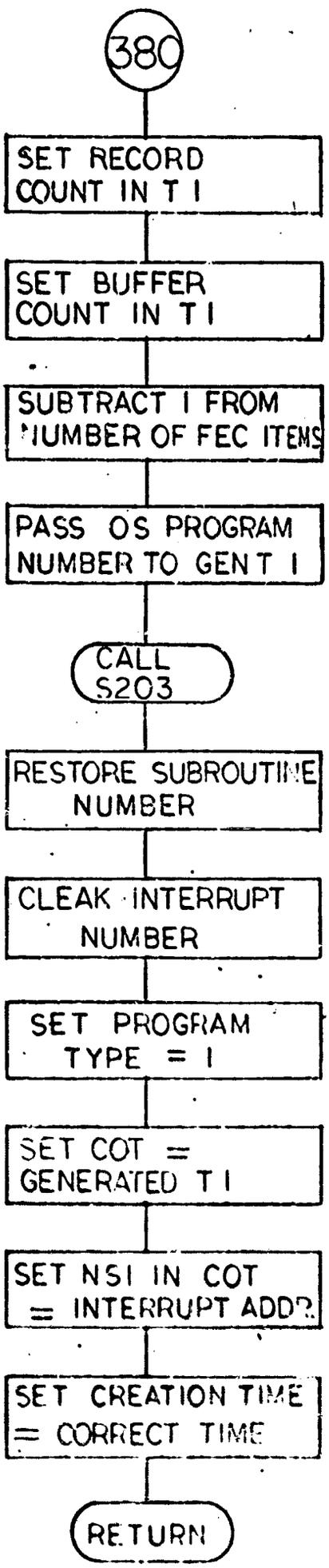

100
ASSIGN 400 TO
TO NEXT 2
200


110
N-2=CHANNEL NUMBER
RESET CPU INTERFERENCE FIELD
SET INTERFERENCE START TIME TO CURRENT TIME
N 4=DEVICE NUMBER
N5 = DEVICE CLASS TABLE
N2 : Ø
>
130
≤
REVERSE SIGN OF N 2
TEST CHANNEL BUSY IN BURST MODE
YES
130
NO
REDUCE CURRENT MULTIPLEXOR RATE
D
D
REDUCE INTERFERENCE RATE
140
130
CLEAR SIZING TD FROM CHANNEL TABLE
REDUCE INTERFERENCE RATE
140
NEXT I
200
N3=CONTROL UNIT NUMBER
TEST CONTROL UNIT BUSY IN SELECTOR MODE
YES
210
WD
230

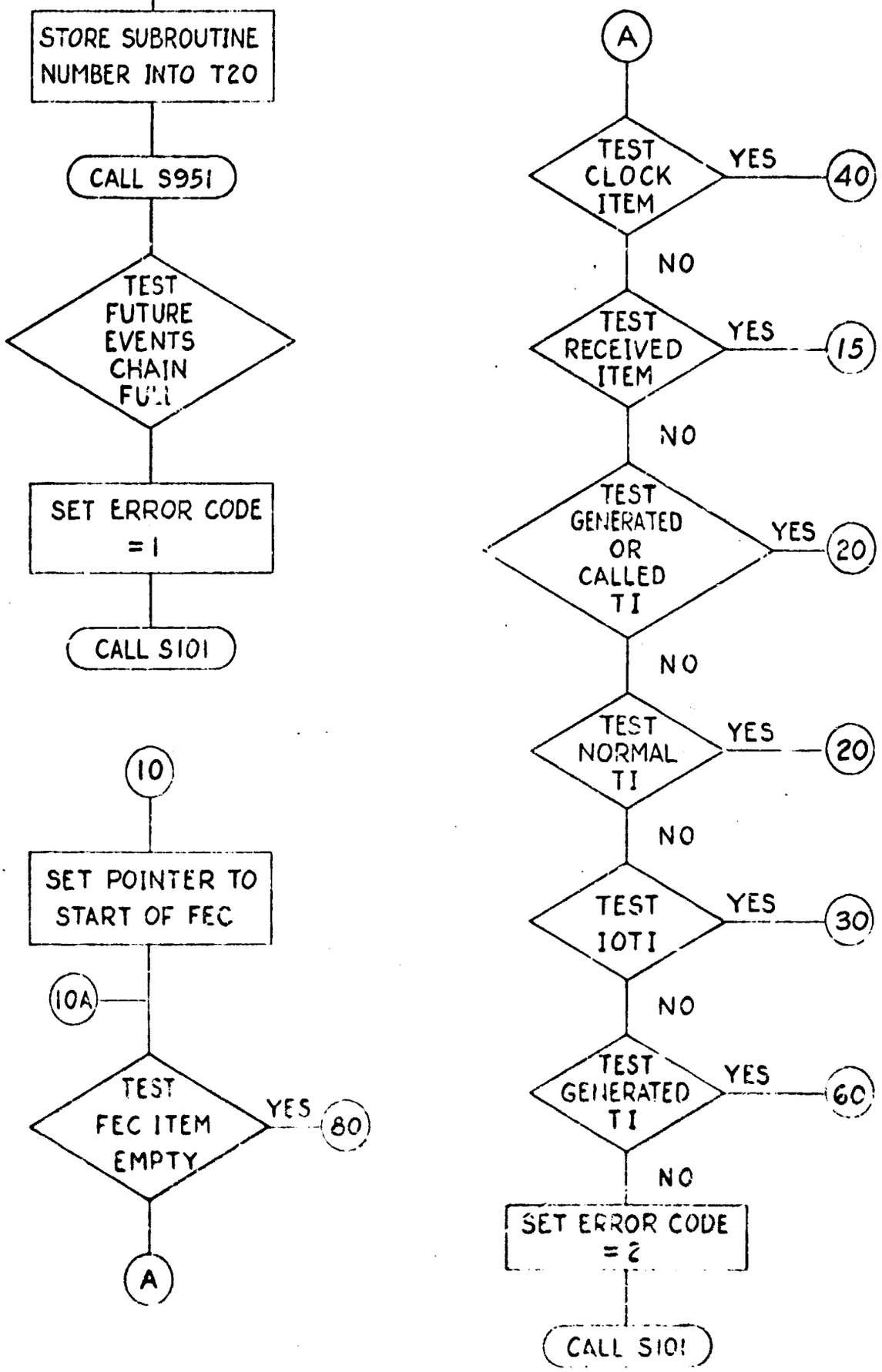

210
TEST CONTROL UNIT SEIZED BY THIS IOTI
YES
215
NO
SET ERROR CODE = 3
RETURN
215
CLEAR SEIZING I.D.
230
NEXT 2
300
N4 = DEVICE NUMBER
TEST DEVICE SEIZED BY TI
YES
310
NO
TEST DEVICE SEIZED BY IOTI
YES
305
NO
SET ERROR CODE = 4
RETURN
305
CLEAR SEIZING I.D
TEST DEVICE TO BE SEIZED BY NEW TI
YES
SEIZE DEVICE
NO
CLEAR NEXT TI-ID SLOT
320

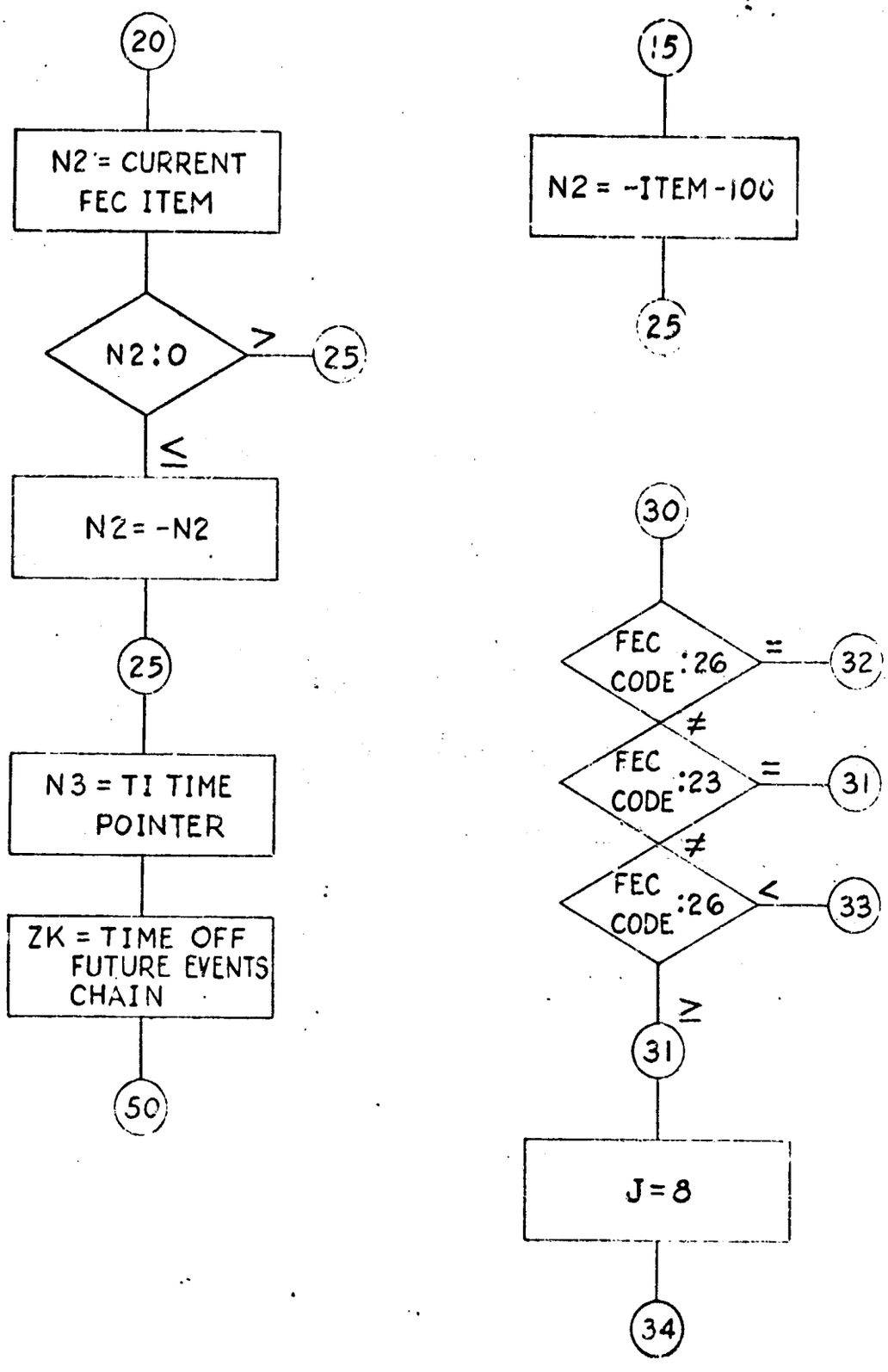

310
FREE TI
SEIZED DEVICE
320
M : 7
=
400
≠
K = IOTI TYPE CODE
K : 5
=
400
≠
N6 = TI ADDRESS
N7 = PART II ADDRESS
NOF = ORDINAL FILE NUMBER
330
330
NOF : 8
<
340
≥
PART II N8=OVERFLOW ADDRESS
N7 = N8
NOF = NOF - 7
330
340
NS = NOF X 3
K : 1
=
320
≠
ADD 1 TO BUFFER COUNT
ADD 1 TO TOTAL I/O OPERATIONS
400

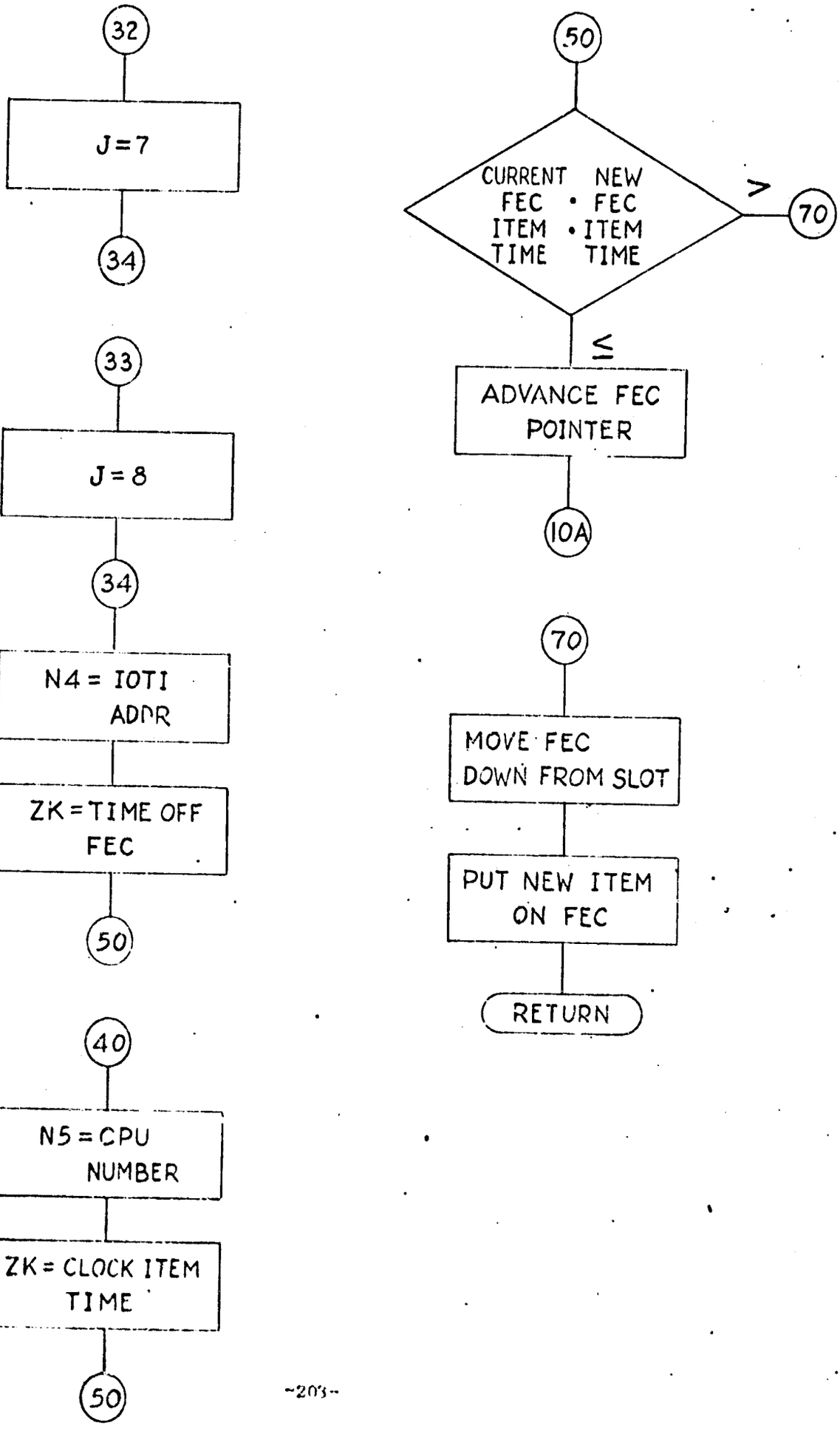

370
NOF= ORIGINAL FILE #
NRF= REAL FILE NUMBER
NC = TRANSACTION ITEM #
ND = FILE VECTOR TABLE ADDR.
NE = FILE VECTOR TABLE ENTRY
TEST REAL FILE VALID
YES
NO
SET ERROR CODE = 5
RETURN


380
SET RECORD COUNT IN T I
SET BUFFER COUNT IN T I
SUBTRACT I FROM NUMBER OF FEC ITEMS
PASS OS PROGRAM NUMBER TO GEN T I
CALL S203
RESTORE SUBROUTINE NUMBER
CLEAK INTERRUPT NUMBER
SET PROGRAM TYPE = I
SET COT = GENERATED T I
SET NSI IN COT = INTERRUPT ADDR
SET CREATION TIME = CORRECT TIME
RETURN

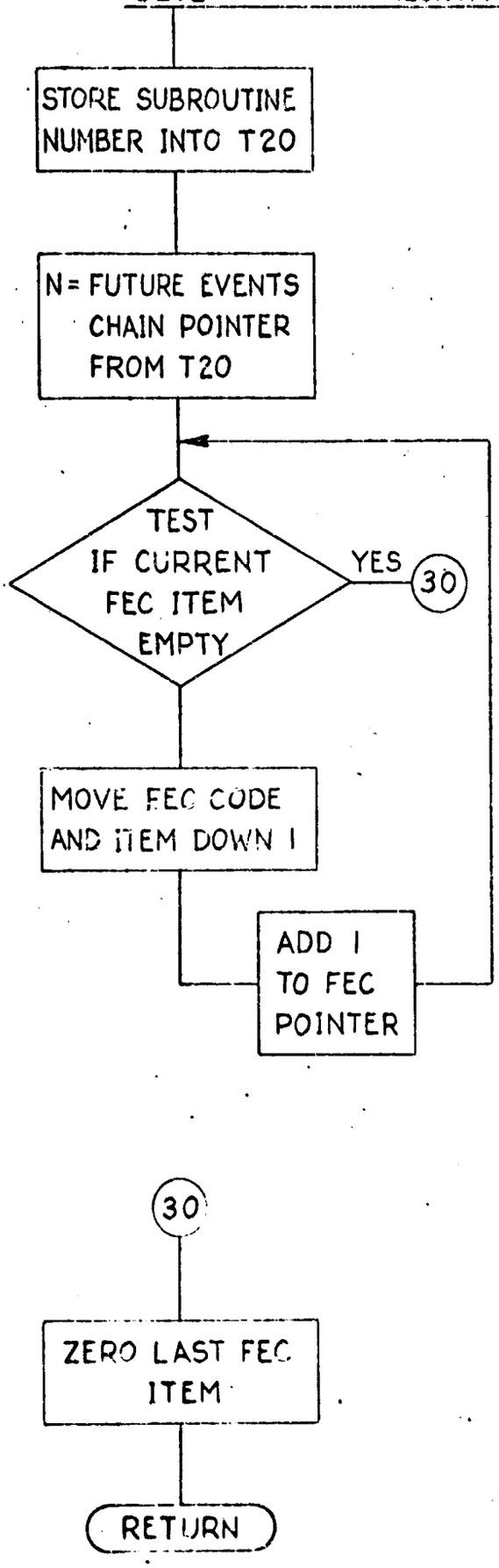
S201 PLACE ITEM ON FUTURE EVENTS CHAIN

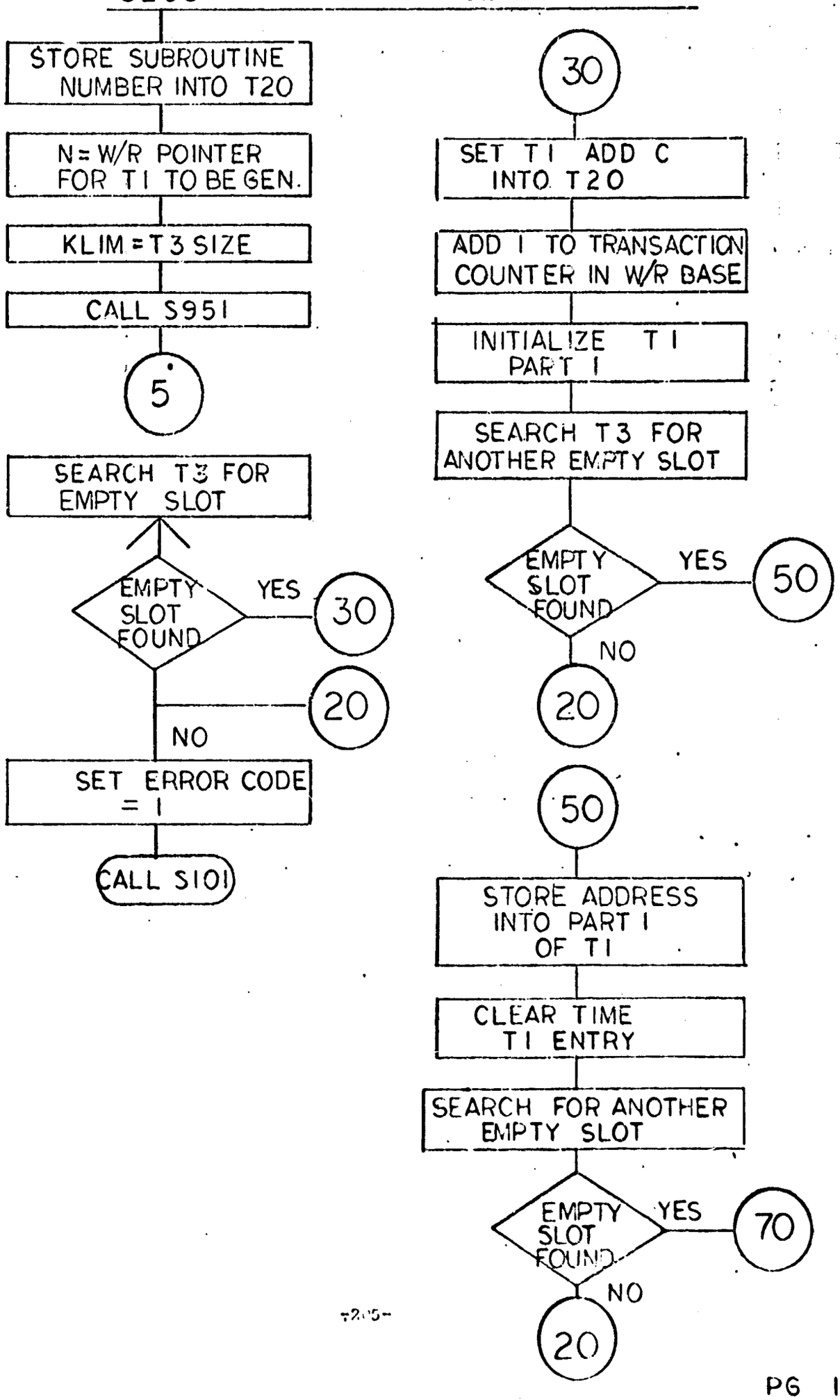


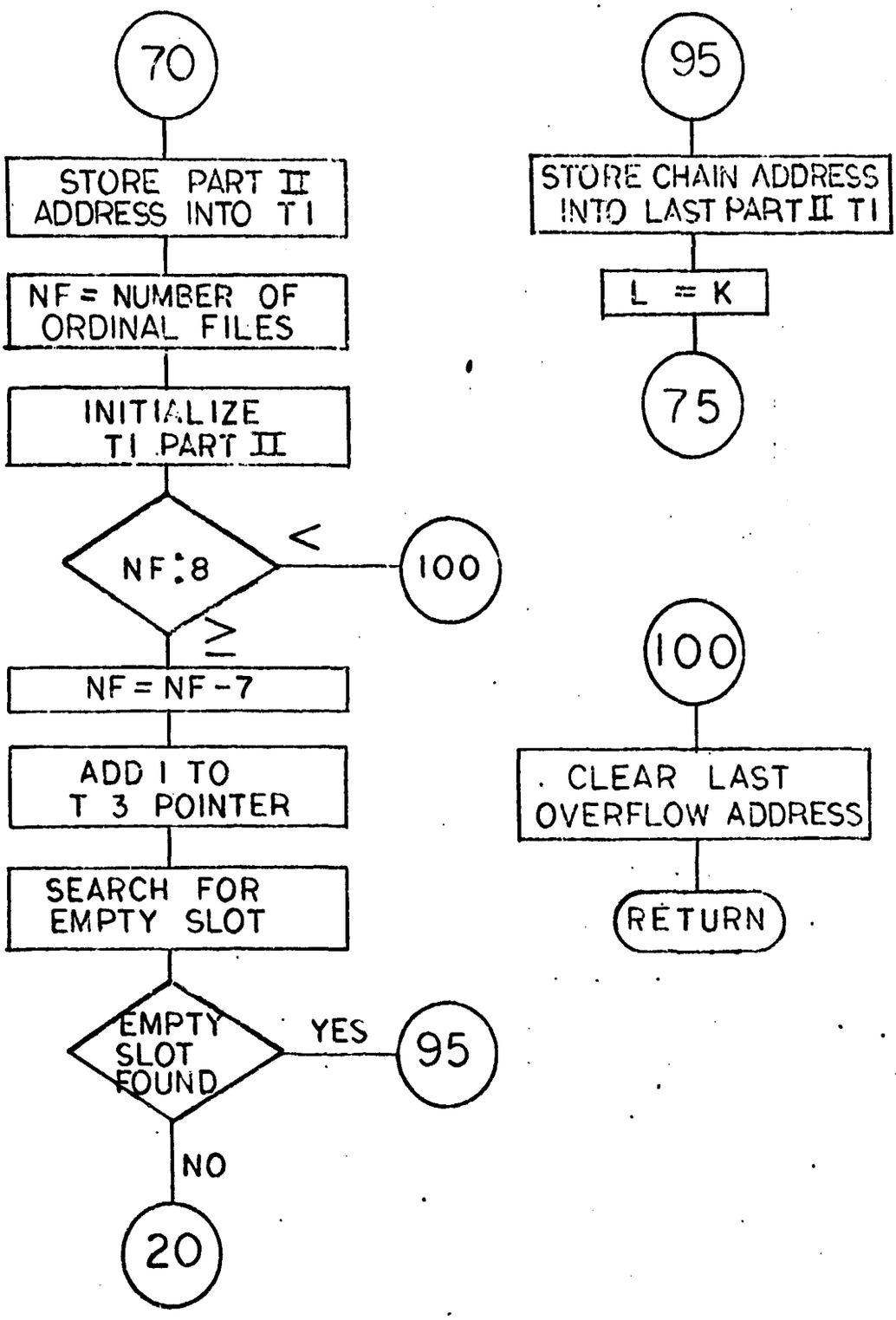

20
N2 = CURRENT FEC ITEM
N2 : 0
>
25
≤
N2 = -N2
25
N3 = TI TIME POINTER
ZK = TIME OFF FUTURE EVENTS CHAIN
50
15
N2 = -ITEM - 100
25
30
FEC CODE : 26
=
32
≠
FEC CODE : 23
=
31
≠
FEC CODE : 26
<
33
≥
31
J = 8
34

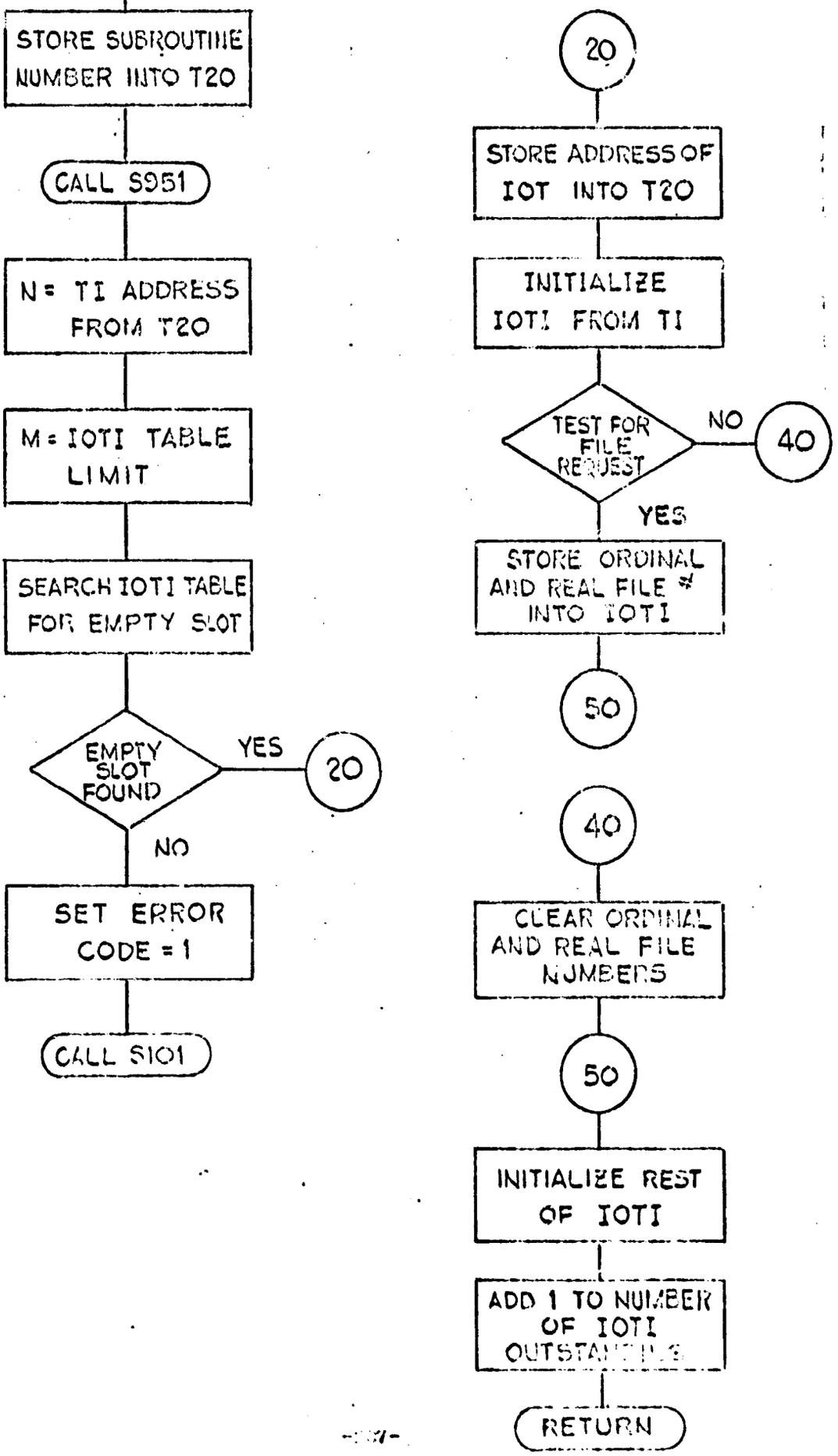

32
J=7
34
33
J=8
34
N4 = IOTI ADDR
ZK = TIME OFF FEC
50
40
N5 = CPU NUMBER
ZK = CLOCK ITEM TIME
50
50
CURRENT FEC ITEM TIME
NEW FEC ITEM TIME
>
70
≤
ADVANCE FEC POINTER
10A
70
MOVE FEC DOWN FROM SLOT
PUT NEW ITEM ON FEC
RETURN

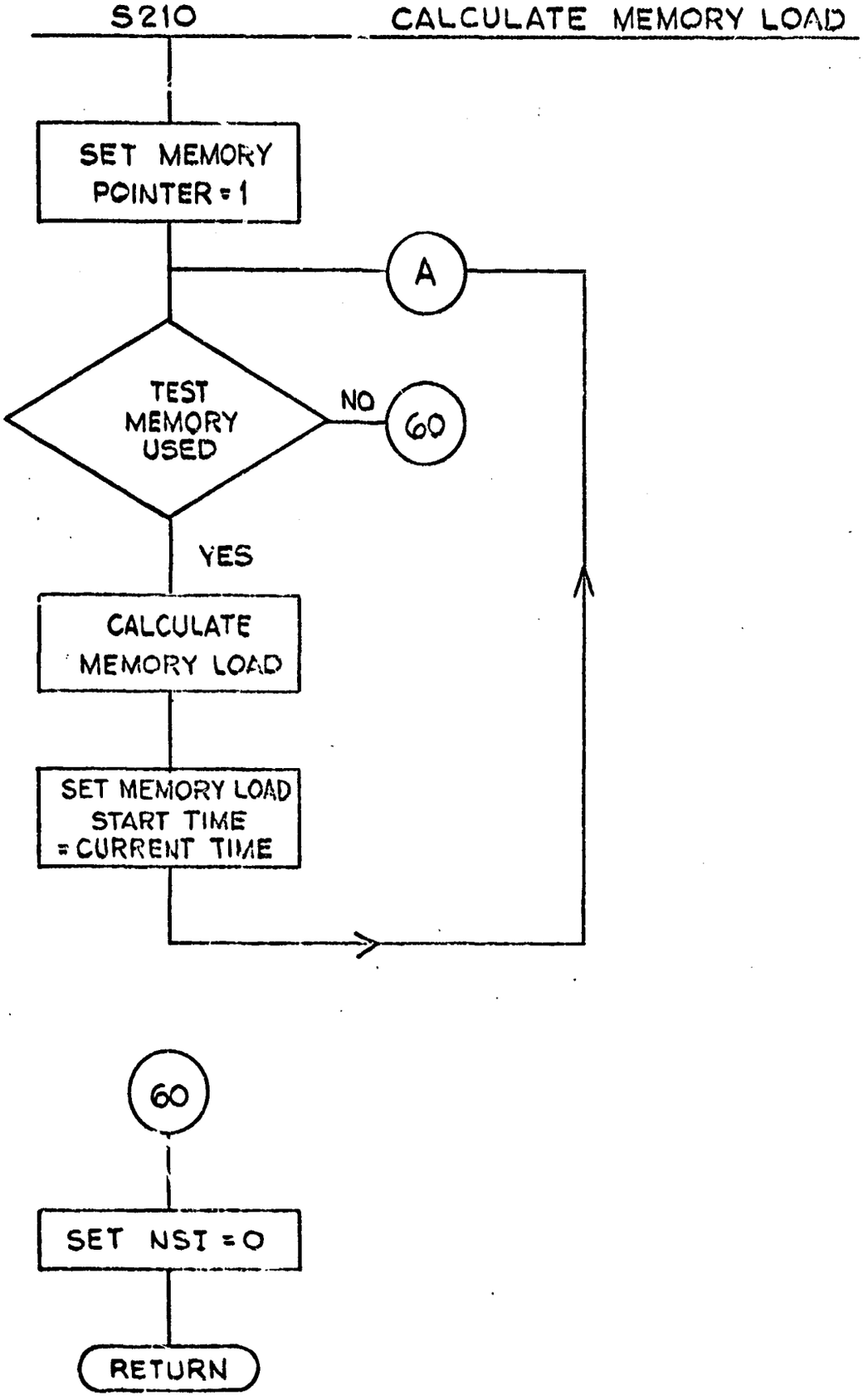
S202 REORGANIZE FUTURE EVENTS CHAIN



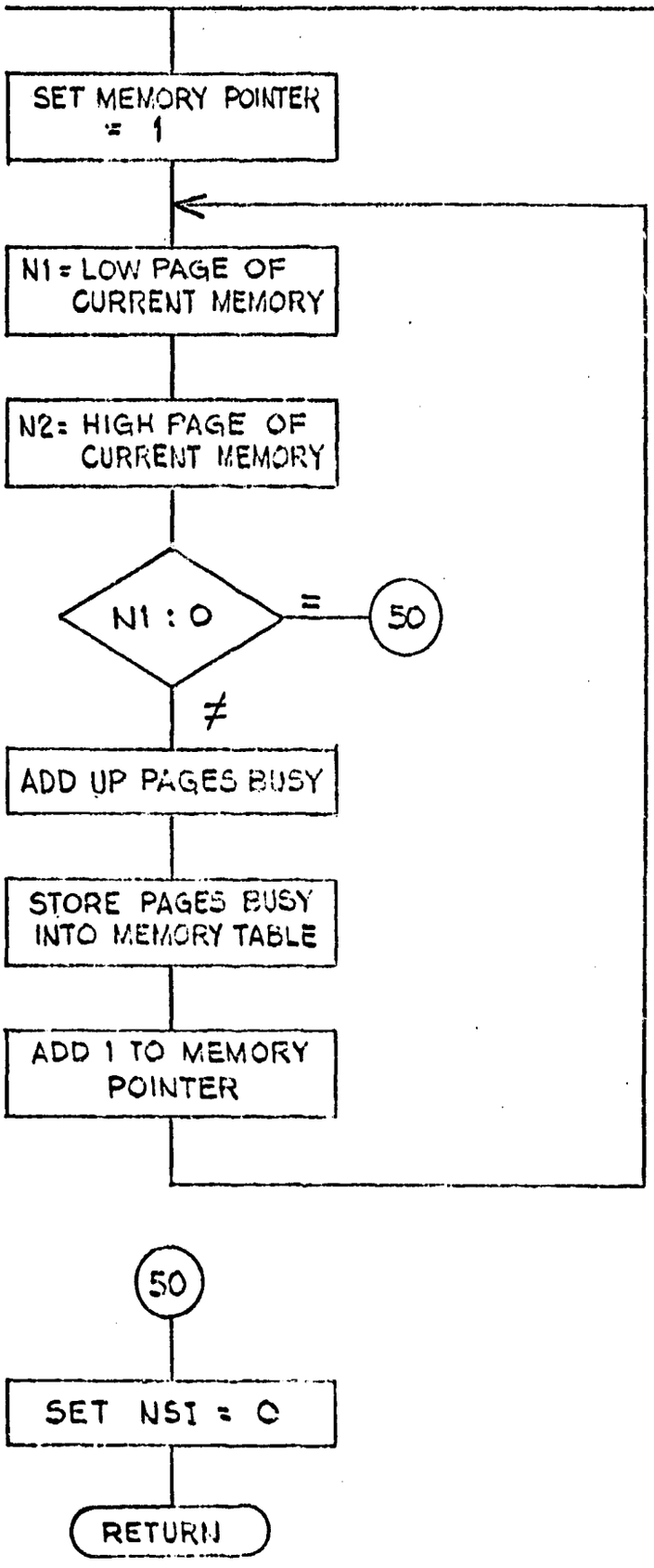

STORE SUBROUTINE NUMBER INTO T20
N = W/R POINTER FOR T I TO BE GEN.
KLIM = T3 SIZE
CALL S951
5
SEARCH T3 FOR EMPTY SLOT
EMPTY SLOT FOUND
YES
30
20
NO
SET ERROR CODE = 1
CALL S101
30
SET T I ADD C INTO T20
ADD I TO TRANSACTION COUNTER IN W/R BASE
INITIALIZE T I PART I
SEARCH T3 FOR ANOTHER EMPTY SLOT
EMPTY SLOT FOUND
YES
50
NO
20
50
STORE ADDRESS INTO PART I OF T I
CLEAR TIME T I ENTRY
SEARCH FOR ANOTHER EMPTY SLOT
EMPTY SLOT FOUND
YES
70
NO
20


70
STORE PART II ADDRESS INTO T I
NF = NUMBER OF ORDINAL FILES
INITIALIZE T I PART II
NF : 8
<
100
≥
NF = NF - 7
ADD I TO T 3 POINTER
SEARCH FOR EMPTY SLOT
EMPTY SLOT FOUND
YES
95
NO
20
95
STORE CHAIN ADDRESS INTO LAST PART II T I
L = K
75
100
CLEAR LAST OVERFLOW ADDRESS
RETURN

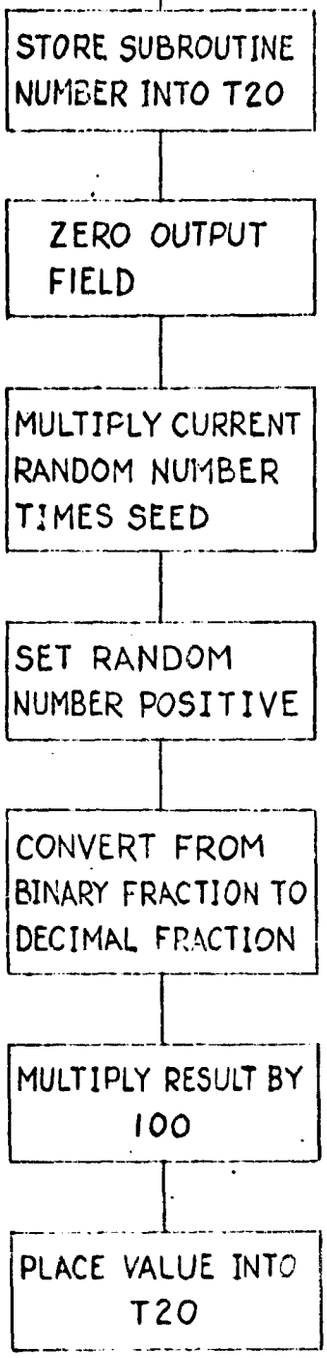

STORE SUBROUTINE NUMBER INTO T20
CALL S951
N = TI ADDRESS FROM T20
M = IOTI TABLE LIMIT
SEARCH IOTI TABLE FOR EMPTY SLOT
EMPTY SLOT FOUND
YES
20
NO
SET ERROR CODE = 1
CALL S101
20
STORE ADDRESS OF IOT INTO T20
INITIALIZE IOTI FROM TI
TEST FOR FILE REQUEST
NO
40
YES
STORE ORDINAL AND REAL FILE # INTO IOTI
50
40
CLEAR ORDINAL AND REAL FILE NUMBERS
50
INITIALIZE REST OF IOTI
ADD 1 TO NUMBER OF IOTI OUTSTANDING
RETURN


S210
CALCULATE MEMORY LOAD
SET MEMORY POINTER = 1
A
TEST MEMORY USED
NO
60
YES
CALCULATE MEMORY LOAD
SET MEMORY LOAD START TIME = CURRENT TIME
60
SET NST = 0
RETURN

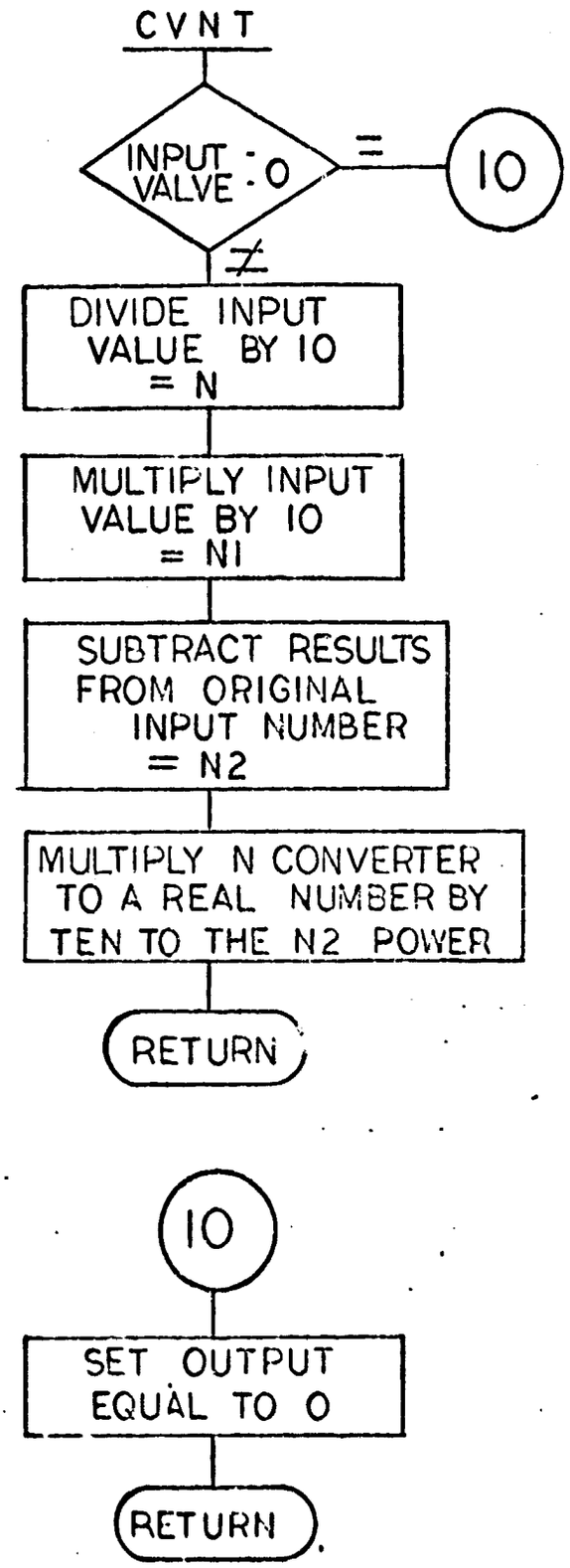
S211 SET MEMORY PAGES BUSY



S212 RANDOM NUMBER GENERATOR



CONVERT NUMBER - FROM DDDE TO FLOATING POINT



WORKING PAPER

SECTION III

S^3 TABLES AND TABLE DESCRIPTIONS

PREPARED FOR

DEPARTMENT OF THE ARMY

UNDER

DEFENSE SUPPLY SERVICE
CONTRACT
#DA 49-083-OSA-3306

BY

C-E-I-R, Incorporated
5272 River Road
Washington, D. C. 20016

WORKING PAPER

TABLE OF CONTENTS

TABLE DESCRIPTIONS	PAGE
FUTURE EVENTS CHAIN TABLE - T1	6
CPU ITEM TABLE - T2	9
TRANSACTION ITEM TABLE - T3	15
I/O TRANSACTION TABLE - T4	23
WORKER ROUTINE BASE TABLE - T5	27
WORKER ROUTINE FILE VECOTR TABLE - T6	30
MEMORY TABLE - T7	32
PAGE TABLE - T8	34
LOAD CLASS TABLE - T9	35
RUN CLASS TABLE - T10	36
QUEUE TABLE - T12	37
QUEUE ENTRY TABLE - T13	40
FILE SET TABLE - T14	41
DEVICE SET TABLE - T15	44
AVAILABILITY TABLE - T16	48
CHANNEL TABLE - T17	49
CONTROL UNIT TABLE - T18	53
DEVICE CLASS TABLE - T19	53
GENERAL SIMULATION TABLE - T20	55

TABLE DESCRIPTIONS	PAGE
FUNCTION TABLE - T21	68
STATEMENT TABLE T22	69
SWITCH TABLE T23	70
TO - FROM T24	71
MARK TIME TABLE T25	72

TABLE DESCRIPTIONS

Introduction

This chapter describes the FORTRAN tables constructed by the simulator machine and their application during the running of a simulation. Twenty different table types are utilized (table 1 through table 21; there is no table 11). Since S3 is written in FORTRAN, the construction of such tables is straightforward, and provides a simple and effective method for the S3 executive to obtain and interrogate data during the actual simulation.

Some of the tables described will be constructed by the pre-simulator from static input data such as configuration data and systems parameters. Other tables will be constructed dynamically, as needed, during the running of the simulation. For example, an IOTI table (T4) is constructed when an I/O is to be initiated, and will be maintained until the simulated I/O is complete.

The facility for maintaining addresses within

one table to point to another, relative table is a very real asset in executing some of the more complex decisions required during a simulation. For example, when the S3 executive attempts to initiate an I/O for a worker routine, it will, using this facility, access the following tables:

(T5) Worker Routine Base Table

(T6) File Vector Table

(T14) File Set Table

(T15) Device Set Table

(T16) Class Table

(T17) Channel Table

(T18) Control Unit Table

From these tables the S3 executive can quickly obtain the information required to determine if an I/O can be executed for a given worker routine, the peripherals required and transmission time for each of the peripherals.

The tables also provide a logical place to maintain raw statistical data. This allows utilization

values, running time, etc. to be associated with a particular device or facility and makes this information readily available to the statistics program.

CARD TYPE 29 DUMP CONTROL CARD

FIELD	TABLE #	TABLE DESCRIPTION
3-4	T1	FUTURE EVENTS CHAIN TABLE
5-6	T2	CPU ITEM TABLE
7-8	T3	TRANSACTION ITEM TABLE
9-10	T4	I/O TRANSACTION ITEM TABLE
11-12	T5	WORKER ROUTINE BASE TABLE
13-14	T6	FILE VECTOR TABLE
15-16	T7	MEMORY TABLE
17-18	T8	PAGE TABLE
19-20	T9	LOAD CLASS TABLE
21-22	T10	RUN CLASS TABLE
23-24	T11	NOT USED
25-26	T12	QUEUE TABLE
27-28	T13	QUEUE ENTRY TABLE
29-30	T14	FILE SET TABLE
31-32	T15	DEVICE SET TABLE
33-34	T16	AVAILABILITY TABLE
35-36	T17	CHANNEL TABLE

CARD TYPE 29 DUMP CONTROL CARD

FIELD	TABLE #	TABLE DESCRIPTION
37-38	T18	CONTROL UNIT TABLE
39-40	T19	DEVICE CLASS TABLE
41-42	T20	GENERAL SIMULATION TABLE
43-44	T21	FUNCTION TABLE
45-46	T22	STATEMENT TABLE
47-48	T23	SWITCH TABLE
49-50	T24	TO-FROM TABLE
51-52	T25	MARK TIME TABLE
63-64	—	SNAP ROUTINE
65-66	—	TRACE ROUTINE

FUTURE EVENTS CHAIN TABLE - T1

PT1(1,N) FEC CODE

The FEC CODE defines the type of item which occupies this entry in the FEC.

1 = Active transaction item for CPU-1

2 = " " " " -2

3 = " " " " -3

4 = " " " " -4

5 = " " " " -5

21 = I/O transaction for Channel, Control, & Device

22 = " " Channel & Control

23 = " " Device & Control

24 = " " Channel & Device

25 = " " Channel

26 = " " Control

27 = " " Device

31 = SEEK Interrupt

99 = CLOCK Interrupt

-1 = Generated or Called transaction for CPU-1

-2 = " " " -2

-3 = " " " -3

-4 = " " " -4

-5 = " " " -5

101-105 = Generated interrupt CPU 1 to 5

PT1(2,N) ITEM

This entry contains the item which will be taken off the FEC chain.

If the FEC CODE = 1 to 5

ITEM = TW

If the FEC CODE = 21 to 27

ITEM = IOTW

If the FEC CODE = 31

ITEM = IOTW

If the FEC CODE = 99

ITEM = CPU#

If the FEC CODE =-1 to -5

and the ITEM< Ø

ITEM = CALLED TI

If the FEC CODE =-1 to -5

and the ITEM> Ø

ITEM = generated TI

If the FEC CODE = 101 to 105

ITEM = Interrupt # for interrupt generated by INTERRUPT statement

If the FEC CODE = -101 to -105

ITEM = a generated TI for which a receiver interrupt has been generated, but which has not yet been taken off the FEC by the RECEIVE statement.

CPU ITEM TABLE - T2

IT2(1,N) COT

The COT is the field containing the address of the current operating transaction in T3.

IT2(2,N) AT

The AT is the field containing the address of the available transaction in T3. This field must be zero when the COT is a worker routine.

IT2(3,N) IOT

The IOT is the field containing the address of the current I/O transaction in T4. This field must be zero when the COT is a worker routine.

IT2(4,N) OS Program Number

This field contains the address of the entry in the worker routine base table T5 for the operating system controlling this CPU.

IT2(5,N) NOT USED

IT2(6,N) to IT2(15,N) MEMORIES

These fields contain the addresses of entries in the memory table T7 for each memory accessible by this CPU.

IT2(16,N) to IT2(35,N) CHANNELS

These fields contain the addresses of entries in the channel table T17 for each channel accessible by this CPU.

IT2(36,N) INTERRUPT 1 ADDRESS

This field contains the address in the statement table which will be accessed by the OS transaction generated for a number 1 interrupt.

IT2(37,N) INTERRUPT 1 COUNTER

This field contains a counter for the number of times an interrupt number 1 has occurred.

IT2(38,N) to IT2(75,N)

PAIRS OF INTERRUPT ADDRESSES AND COUNTERS.

IT2(76,N) to IT2(78,N)

IT2(79,N) CPU ID

This field contains the CPU identification number from the CPU definition card which defines this CPU.

IT2(80,N) LOGICAL DATA UNIT

This field contains the logical data unit size in bits, as used by this CPU.

IT2(81,N) NUMBER OF DECIMAL DIGITS

This field contains the number of decimal digits used to determine the decimal add, multiply, and divide

times.

AT2,(82,N) DECIMAL ADD TIME

This field contains the time in micro seconds to perform a decimal add upon two fields of the length given by IT2(81,N).

AT2(83,N) DECIMAL MULTIPLY TIME

This field contains the time in micro seconds to perform a decimal multiplication upon two fields of the length given by IT2(81,N).

AT2(84,N) DECIMAL DIVIDE TIME

This field contains the time in micro seconds to perform a division between two fields of the size given by IT2(81,N).

AT2(85,N) FIXED POINT ADD TIME

This field contains the time in micro seconds to perform a fixed point addition between two normal fixed point fields.

AT2(86,N) FIXED POINT MULTIPLY TIME

This field contains the time in micro seconds to perform a fixed point multiplication between two normal fixed point fields.

AT2(87,N) FIXED POINT DIVIDE TIME

This field contains the time in micro seconds to perform a fixed point division between two normal fixed point fields.

AT2(88,N) FLOATING POINT ADD TIME

This field contains the time in micro seconds to perform a floating point addition between two normal floating point fields.

AT2(89,N) FLOATING POINT MULTIPLY TIME

This field contains the time in micro seconds to multiply two normal floating point fields.

AT2(90,N) FLOATING POINT DIVIDE TIME

This field contains the time in micro seconds to divide two normal floating point fields.

IT2(91,N) INSTRUCTION LENGTH

This field contains the instruction length for this CPU in logical data units.

IT2(92,N) DECIMAL DIGITS PER LOGICAL DATA UNIT

This field contains the number of decimal digits contained in one logical data unit.

IT2(93,N) CHARACTERS PER LOGICAL DATA UNIT

This field contains the number of characters con-

tained in one logical data units.

AT2(94,N) MOVE TIME PER CHARACTER

This field contains the time in micro seconds to move one character from one location to another.

AT2(95,N) MOVE AND EDIT TIME PER CHARACTER

This field contains the time in micro seconds to move and edit one character from one location to another.

IT2(96,N) to IT2(103,N) FIXED POINT TABLE

These fields contain the fixed point tables described in the CPU definition description.

IT2(104,N) to IT2(109,N) FLOATING POINT TABLE

These fields contain the floating point tables described in the CPU definition description.

AT2(110,N) GIBSON TIME

This field contains the average instruction access and execution time expressed in micro seconds.

ZT2(56,N) CLOCK ITEM TIME

This field contains the time at which the current clock item (if any) for the CPU on the future events chain (FEC) is due to create a CLOCK interrupt.

ZT2(57,N) TOTAL OS TIME

This field contains the total amount of operating system CPU time supplied by the CPU at the current time expressed in micro seconds.

ZT2(58,N) TOTAL OS IOT QUEUE TIME

This field contains the total amount of time which I/O transaction items for the operating system have been queued.

ZT2(59,N) START CYCLE TIME

This field contains the last time a CYCLE statement was interpreted by this CPU.

ZT2(60,N) TOTAL CYCLE TIME

This field contains the total amount of time in micro seconds which this CPU has spent cycling.

ZT2(61,N) TOTAL OS TI QUEUE TIME

This field contains the total amount of time in micro seconds which operating systems transaction items have been queued.

TRANSACTION ITEM TABLE - T3

TRANSACTION ITEM PART I

IT3(1,N) WORKER - ROUTINE POINTER

-1 = ENTRY IS NOT BEING USED

This entry contains the address of the worker routine base in (T5) with which this transaction is associated.

IT3(2,N) CURRENT FILE REQUEST

This entry contains the ordinal file # of the current read - write request for this transaction #. This field should be filled by the READ and WRITE statements and set to zero by the BUFF statement.

IT3(3,N) TRANSACTION ITEM ID

This entry contains the identification number of this transaction item. This field is generated as follows PPNNNN. Where PP is the worker routine base # and NNNN is the number of this transaction.

IT3(4,N) RUN CLASS POINTER

This entry contains either a zero or the entry in the run class table (PT10) which lists the CPU's which can run this transaction.

This field is filled by the ALLOCATE subroutines. If the field is zero, this transaction has not been loaded and may not be run.

IT3(5,N) PRIORITY

This entry contains the priority of this transaction item. This entry is used by the PLACE subroutine.

IT3(6,N) NSI LOCATION

This entry contains the location of the next sequential statement to be interpreted in the statement table by the worker routine or operating system routine.

IT3(7,N) #IOT OUTSTANDING

This entry gives the total number of I/O transaction items associated with this transaction, which are currently active in the system.

IT3(8,N) PROGRAM TYPE

1 = operating system
2 = primary worker
3 = commercial worker
4 = scientific worker

IT3(9,N) CPU # FOR FEC

This entry gives the number of the CPU which placed this transaction item on the future events chain. This field is filled and used by the future events chain manipulator.

IT3(10,N) LOOP ADDRESS - 1

This field holds the address of the first LOOP statement encountered.

IT3(11,N) LOOP COUNTER - 1

This field holds the count from the first LOOP statement encountered.

IT3(12,N) TO IT3(15,N)

Additional loop addresses and counters.

IT3(16,N) SUBROUTINE EXIT - 1

The return address of the first subroutine called.

IT3(17,N) and IT3(18,N)

Additional subroutine return addresses.

IT3(19,N) PRINT LINES

Number of lines to be skipped as specified in last PRINT statement.

IT3(20,N) USE CODE

A -1 in this field indicates this entry is a Part I TI.

IT3(21,N) TIME POINTER

This entry contains the address of the time entry in the transaction item table (T3) associated with this transaction item.

IT3(22,N) PART - 2 - POINTER

This entry contains the address of the file description entry in the transaction table (T3) associated with this transaction item.

TIME ENTRY - T3

ZT3(1,N) CREATION TIME

This entry contains the time at which this transaction first reached the top of the future events chain, was created by an ACTIVATE statement or was called by a TERM statement.

ZT3(2,N) TIME DUE OFF FEC

This entry contains the time at which this transaction is due to come off of the future events chain. This field is used by the future events chain manipulator.

ZT3(3,N) ADVANCE TIME

This entry contains the amount of time this transaction item will spend on the future events chain.

ZT3(4,N) TOTAL CPU TIME

This entry contains the total of all advance times associated with this transaction item.

ZT3(5,N) TI QUEUE START TIME

This entry contains the time at which this transaction item was placed on a queue when it is on a queue. This field is set by the PLACE statement and set to zero

by the SELECT statement.

ZT3(6,N) TI TOTAL QUEUE TIME

This entry contains the total time this transaction item spent on all queues. This field is accumulated by the SELECT statement.

ZT3(7,N) IOT TOTAL QUEUE TIME

This entry contains the total time all IOT's associated with this transaction spent on queues. This time is accumulated to each time an IOT is DESTROYED.

ZT3(8,N) TOTAL I/O TIME USED

This field contains a total of all of the device times, for all devices accessed by this transaction, used up to the current time. This field is incremented by the IOADVANCE subroutine, and is used to determine dynamic priority.

IT3(19,N) AT SAVE

This field is used to store the current AT whenever a transaction (COT) is interrupted. Upon the execution of a RETURN statement, making the AT the COT again, the value of this field is moved to the AT for the current CPU, item, and this field is set to zero.

IT3(20,N) IOT SAVE

This field is used to store the current IOT whenever a transaction (COT) is interrupted. This field is handled exactly like the AT SAVE field above.

IT3(21,N) CALLING TI NUMBER

This entry contains the transaction number of the transaction which called this transaction. This entry is used only if this transaction was created by a CALL statement.

IT3(22,N) CALLED PROGRAM #

This entry contains the worker routine number of the last worker routine called by this transaction.

TRANSACTION ITEM FILE SECTION - PART II

IT3(1,N) BUFFER COUNT

This entry contains the number of buffers which are empty/full for this file. Each time this count is decremented by a BUFF statement, a new IOT is generated. Each time an IOT comes off of the future events chain for this file this field is incremented.

IT3(2,N) RECORD COUNT

This entry contains the number of records currently available in this file. This field is decremented by a BUFF statement until it is zero at which time the BUFFER count is decremented and the record count is reset from the FILE SET table (T14).

AT3(3,N) TOTAL I/O OPERATIONS

This entry contairs the total number of blocks of data contained in this file.

IT2(22,N) PART - 3 - POINTER

If there are more than seven files associated with this transaction item this field contains a chain address to the next entry.

I/O TRANSACTION TABLE - T4

IT4(1,N) TI POINTER

This entry contains the address of the transaction item associated with this I/O transaction.

IT4(2,N) # FEC ITEMS

This field is used if this I/O transaction requires multiple items to be placed on the future events chain. This field is used by the FEC manipulator.

IT4(3,N) TYPE CODE

This field defines what kind of I/O transaction this is.

0 = READ - WRITE
1 = OPEN
2 = CLOSE
3 = SEEK
4 = SEEK - CONTINUE
5 = FUNCTION
6 = PRINT

A READ - WRITE IOT is generated by the BUFF statement. An OPEN IOT is generated by the OPEN statement. A CLOSE IOT is generated by a CLOSE statement. A SEEK IOT is generated by a SEEK statement. A SEEK - CONTINUE IOT is generated by the SEEK interrupt routine upon the completion of a seek operation. A FUNCTION IOT is generated by a FUNCTION statement.

IT4(4,N) CPU # (FEC)

This field contains the number of the CPU which placed this IOT on the future events chain. This field is used by the future events chain manipulator.

IT4(5,N) CHANNEL #

This field is filled by the IO READY statement and contains the number of the channel which will be used by this IOT. If this field is positive the channel is a selector and if this field is negative this channel is a multiplexor. The selector or multiplexor must be released when the IOT comes off the FEC.

IT4(6,N) CONTROL UNIT #

This field is filled by the IO READY statement with the number of the CONTROL unit to be used by this I/O transaction. This fields is set negative by IO READY and positive by IO ADVANCE as a check that the IO READY has been performed. The control unit specified in this field must be released when this IOT comes off the FEC.

IT4(7,N) DEVICE #

This field is filled by the IO READY statement with the number of the DEVICE to be used by this IOT. The field is set negative by the IO READY and positive by the IO ADVANCE statement. This device must be released when

this IOT comes off of the FEC unless this is a SEEK IOT or the device has been assigned for the exlusive use of a worker routine.

IT4(8,N) REAL FILE #

This field is filled for all IOT except those generated by a FUNCTION statement. This field must be filled when an IOT is generated.

IT4(9,N) ORDINAL FILE #

This field is used just like the REAL FILE #.

IT4(10,N) FUNCTION #

This field is used by a FUNCTION IOT's. When used it will contain the entry in the function table to be used by this IOT.

for a CLOSE IOT a

1 = no rewind
2 = rewind

for PRINT IOT

= FORM ADVANCE TIMES

ZT4(6,N) CHANNEL END TIME

This field contains the time at which the channel should be released from this I/O operation. This time is calculated by the IO ADVANCE statement.

ZT4(7,N) CONTROL END TIME

This field contains the time at which the control unit should be released from this I/O operation. This time is calculated by the IO ADVANCE statement.

ZT4(8,N) DEVICE END TIME

This field contains the time at which the device should be released from this I/O operation. This time is calculated by the IO ADVANCE statement.

ZT4(9,N) QUEUE START TIME

This field contains the time at which this I/O transaction was placed on a queue, when it is on a queue. This field is set by the PLACE statement and set to zero by the SELECT statement.

ZT4(10,N) TOTAL QUEUE TIME

This field contains the total time this I/O transaction has spent on a queue. This field is added to the TI when the IOT is destroyed.

WORKER ROUTINE BASE TABLE - T5

AT5(1,N) SPREAD

The time interval between generation of transaction items for this worker routine. If this field is positive the spread is fixed, if this field is negative the spread is poisson.

IT5(2,N) LIMIT

The total number of transaction items to be generated.

AT5(3,N) TOTAL NUMBER GENERATED

The total number of transactions generated for this worker routine by the above two statements.

IT5(4,N) LOAD CLASS POINTER

The address of an entry in the LOAD CLASS TABLE (T9) which specifies the CPU's capable of loading this worker routine.

IT5(5,N) PRIORITY

The priority associated with this worker routine. This priority will govern all transaction items and I/O transaction items.

IT5(6,N) FIRST INSTRUCTION LOCATION

The entry in the statement table (T22) which contains the first executable instruction for this worker routine.

IT5(7,N) REENTRANT - NONREENTRANT CODE

A code specifying whether this worker routine is reentrant or not.

1 = nonreentrant
2 = reentrant

IT5(8,N) PROGRAM TYPE

A code specifying the instruction class available to this worker routine.

1 = Operating System
2 = Primary Worker Routine
3 = Commercial Worker
4 = Scientific Worker

IT5(9,N) NUMBER OF TRANSACTION ITEMS OUTSTANDING

A counter containing the number of transaction items for this routine currently in memory. This counter must be incremented by 1 when a TI is loaded and decremented when an AT is removed from memory.

IT5(10,N) RECEIVING CPU NUMBER

The number of the CPU which is to be interrupted when a transaction item referencing this routine first reaches the top of the future events chain.

IT5(11,N) NUMBER OF FILES

The total number of files associated with this routine.

IT5(12,N) FILE VECTOR TABLE ENTRY

The first entry in the file vector table for this routine. This entry corresponds to ordinal file number one.

AT5(13,N) INSTRUCTION STORAGE

The number of pages, in decimal form, of storage required for instructions in this program.

AT5(14,N) DATA STORAGE

The number of pages, in decimal form, required for data in this program.

AT5(15,N) I/O STORAGE

The number of pages, in decimal form, required for I/O in this program.

IT5(16,N) TOTAL NUMBER OF TI GENERATED

The total number of transactions generated for this routine by means of the ACTIVATE or TERM statement.

A-IT5(17,N) +2

A-IT5(18,N) THE TOTAL NUMBER OF TRANSACTIONS GENERATED FOR THIS WORKER ROUTINE

WORKER ROUTINE FILE VECOTR TABLE - T6

PT6(1,N) REAL FILE NUMBER

This entry contains the number of the real file associated with this ordinal file entry.

To obtain a real file from an ordinal file.

N = N1+IT5(12,N2)-1

N3 = PT6(1,N)

where

N1 = ordinal file number

N2 = worker routine base number

N = file vector total entry

N3 = real file number

PT6(2,N) NUMBER OF BUFFERS

This entry contains the number of buffers available for this ordinal file.

PT6(3,N) SEIZING CODE

This entry contains a code used by the PERIPHERAL subroutine to determine what files and devices (if any) should be seized.

1 = DON'T SEIZE ANYTHING

2 = SEIZE FILE

3 = SEIZE FILE AND DEVICE

PT6(4,N) NOT USED

MEMORY TABLE - T7

IT7(1,N) LOW PAGE

The number of the entry in the PAGE TABLE (T8) for the lowest page in this memory.

IT7(2,N) HIGH PAGE

The number of the entry in the PAGE TABLE (T8) for the highest page in this memory.

IT7(3,N) NUMBER OF PACKS

The total number of PACK operation performed upon this memory.

IT7(4,N) NUMBER OF NO MEMORY EXITS

The total number of times this memory was tested by a MEMORY subroutine and insufficient memory was found.

IT7(5,N) PAGES BUSY

The total number of pages in this memory which are currently busy.

IT7(6,N) NOT USED

ZT7(4,N) LAST REFERENCE TIME

This entry contains the last time this memory was referenced.

ZT7(5,N) MEMORY LOAD

This entry contains the total memory load expressed

in PAGE MICROSECONDS applied to this memory up to the last reference time.

PAGE TABLE - T8

PT8(N) PAGE ENTRY

This entry contains a value which indicates if this page of simulated memory is currently busy and if so, which transaction is utilizing this page.

0 = Page free

+N = Page used by TI #N

-N = Page used by W/R #N for reentrant code

LOAD CLASS TABLE - T9

PT9(1,N) to PT9(5,N) LOAD CLASS ENTRY

The entries in this table will contain the load class values as supplied by the S/P load class statement, and will specify the CPU's capable of loading a worker routine. The value of N may not exceed 15.

RUN CLASS TABLE - T10

PT10(1,N) to PT10(5,N) RUN CLASS ENTRY

The entries in this table will contain the run class values as supplied by the S/P run class statement and will specify the CPU's capable of executing a worker routine. The value of N may not exceed 5.

QUEUE TABLE - T12

IT12(1,N) QUEUE START

This entry contains the address of the first item in the Queue Entry table (T13) related to the specified queue (N).

IT12(2,N) QUEUE END

This entry contains the address of the last item in the Queue Entry table (T13) related to the specified queue (N).

IT12(3,N) QUEUE POINTER

This entry contains the address of the current item in the Queue Entry table (T13) for the specified queue (N). This pointer is set and modified by the EXAMINE statements and used by the SELECT statement. After Execution of PLACE and SELECT statements, the pointer is set equal to the queue start address.

IT12(4,N) ENTRY TYPE

This entry indicates whether the contents of this queue represent transaction itesm or I/O transaction items.

1 = TI

2 = IOTI

IT12(5,N) QUEUING METHOD

This entry indicates the contents of this queue are organized by program priority or on a first-in-first-out or last-in-first-out basis.

1 = program priority

2 = FIFO

3 = LIFO

IT12(6,N) MAXIMUM ENTRIES

This entry indicates the maximum number of entries on the specified queue during the current simulation. This entry is affected only by the PLACE statement.

IT12(7,N) CURRENT ENTRIES

This entry indicates the number of items currently present on the specified queue. This entry is affected by both the PLACE and SELECT statements.

IT12(8,N) NOT USED

ZT12(5,N) LAST REFERENCE TIME

This entry indicates the last time at which the contents of the specified queue were altered. This entry is used by the PLACE and SELECT statements to calculate the queue load.

ZT12(6,N) QUEUE LOAD

This entry is used to accumulate the total load on the specified queue. Each time the contents of the queue are altered by a PLACE or SELECT statement, the current load is calculated and added to the previous contents of this entry.

QUEUE ENTRY TABLE - T13

PT13(N) QUE ENTRY

This table is used to hold all TW's associated with queues. It can contain 2000 entries that may be divided among up to 30 queues as the user sees fit. The limits of a named queue's entries in this table are defined by that queue's relative Queue Start and Queue End entries in the Queue Table (T12).

FILE SET TABLE - T14

IT14(1,N) SEIZING ID

This field contains the identification of the transaction currently using this file. If this field is zero the file is free. If this field is negative it contains the number of the transaction item which seized this file by a FBRIPHERAL statement. If this field is positive it contains the number of the I/O transaction currently using this file.

IT14(2,N) DEVICE #

This field contains the identification of the device upon which this file is resident. This information comes from the system parameters.

IT14(3,N) RELATIVE LOCATION

This field contains the relative number of this file on the device. This information is used to access the TO-FROM table. If this entry is negative it means that this is the only file on the device. This information comes from system parameters.

IT14(4,N) BUFFER LENGTH

This field contains the length of each physical record on this file. This information comes from system parameters.

IT14(5,N) RECORDS PER BUFFER

This field is used by the BUFF statement to reset the number of records available when a buffer has been emptied/filled.

IT14(6,N) NUMBER OF BLOCKS IN FILE

This field is used by the EOF statement to determine whether this file is ended.

ZT14(4,N) SEIZE START TIME

This field contains the time at which this file was seized. This field is filled by the IO ADVANCE statement when an OPEN statement seizes a file. This field is reset to zero when the file is closed.

ZT14(5,N) TOTAL SEIZE TIME

This field contains the total amount of time this file was seized. This field time is accumulated each time the file is released by a CLOSE statement.

ZT14(6,N) I/O TIME

This field contains the total device time associated with using this file. This field is added to each

time an IO ADVANCE is executed for this file.

DEVICE SET TABLE - T15

IT15(1,N) SEIZING ID

This entry contains the identification of the TI or IOTI which has seized the device. If this field is zero the device is free. If this field is negative it contains the address of a transaction item which has seized this device in a PERIPHERAL statement. If this field is positive it contains the address of an I/O transaction item now on the FEC using this device.

IT15(2,N) AVAILABILITY TABLE ADDRESS

This entry contains the address of the first entry in the availability table (T16) indicating a channel control unit path to this device. The last entry in the availability table associated with this device will be minus in the first field.

IT15(3,N) DEVICE CLASS TABLE ADDRESS

This entry contains the address of this device class table (T19) which contains the fixed information about a specific device type.

IT15(4,N) TO-FROM TABLE ADDRESS

This entry contains the address of the first entry in the TO-FROM table (T24) associated with the device.

IT15(5,N) TO-FROM TABLE WIDTH

This entry contains width of the TO-FROM table associated with this device. To access on entry in the TO-FROM table multiply the number of the row to be accessed by the table width, add the number of the column and the table address from above. This gives the address of the entry to be accessed.

IT15(6,N) RELATIVE LOCATION OF LAST FILE

If there is more than one file on this device, each time a file on this device is referenced the IOADVANCE routine stores the relative location of the file being referenced into this field. When referencing the TO-FROM table, this is the FROM file and the new file to be accessed is the TO file.

IT15(7,N) SEIZABLE DEVICE CODE

This entry contains a code which tells the system if this device can be seized or not. If an OPEN statement attempts to seize a device which is classed an un-seizable, the file will seized rather than the device.

1 = seizable device

2 = nonseizable device

IT15(8,N) PERIPHERAL SEIZING ID

This field is used by the PERIPHERAL subroutines to indicate that a new worker routine wishes to seize this device, although at the time of the PERIPHERAL statement the device was busy.

Upon receipt of a device I/O termination the interrupt subroutine S110 checks this field and if it is non zero moves this field to the seizing ID.

ZT15(5,N) LAST END TIME

This entry contains the time at which the last I/O operation on this device ended. This time is set by the IOADVANCE routine and is used to determine penalty time.

ZT15(6,N) TOTAL SEEK TIME

This entry contains the total seek time used by the device. Normally this field should be zero for all devices other than random access. This field is used by the IOADVANCE statement.

ZT15(7,N) TOTAL NON-SEEK TIME

This field contains the amount of time this device was busy performing operations other than SEEKING. This field is used by the IOADVANCE statement.

ZT15(8,N) TOTAL PENALTY TIME

This field contains the total penalty time used by this device. This field is used by the IOADVANCE statement.

AVAILABILITY TABLE - T16

PT16(1,N) INPUT - OUTPUT - BOTH DIRECTION CODE

This entry contains a code which specifies the direction in which this channel - control unit path may be used.

1 = INPUT ONLY

2 = OUTPUT ONLY

3 = INPUT OR OUTPUT

PT16(2,N) CHANNEL ADDRESS

This entry contains the address of the channel in the channel table (T17) which is used by this available path. If this field is positive the channel is a selector channel and if this field is negative the channel is a multiplexor channel.

PT16(3,N) CONTROL UNIT ADDRESS

This entry contains the address of the control unit in the control unit table (T18) which is used by this available path.

PT16(4,N) NOT USED

CHANNEL TABLE - T17

CHANNEL TABLE ENTRY I

This entry is used for selector channels or the burst mode section of a multiplexor channel. A selector channel has only one entry in this table whereas a multiplexor channel has two entries.

IT17(1,N) SEIZING ID

This entry contains the address of the I/O transaction item currently using this channel. If this entry is zero the channel is free. If this is a multiplexor channel this entry must be zero if the channel is operating in multiplex mode, and the current multiplexor rate must be zero if the channel is operating in burst mode.

IT17(2,N) MAXIMUM CHANNEL RATE

This entry contains the maximum transfer rate in characters per second for a selector channel or for a multiplexor channel operating in burst mode.

IT17(3,N) % INTERFERENCE

This entry contains the % of CPU interference generated by this channel. This value is in % per 1000 CPS transfer rate.

IT17(4,N) NOT USED

ZT17(3,N) TOTAL TIME USED

This entry contains the total number of micro seconds this channel was active. For a multiplexor channel this field contains the time this channel was busy in burst mode.

ZT17(4,N) LOAD IN CHARACTERS TRANSMITTED

This entry contains the total number of characters transmitted over this channel. For a multiplexor channel this is the total number of characters transmitted in burst mode.

CHANNEL TABLE ENTRY II

This table entry is used for the multiplexor mode of a multiplexor channel.

IT17(1,N) CURRENT MULTIPLEXOR RATE

This field contains the current transfer rate on this multiplexor channel. If this field is zero the multiplexor channel is not being used. This field is added to by an IOADVANCE statement and subtracted from when an I/O termination occurs for this channel.

IT17(2,N) MAXIMUM MULTIPLEXOR RATE

This field contains the maximum transfer rate in characters per second for a multiplexor channel operating in multiplexor mode.

IT17(3,N) % MULTIPLEXOR INTERFERENCE

This entry contains the % of CPU interference generated by the multiplexor channel operating in multiplex mode. This value is in % per 1000 characters per second transfer rate.

IT17(4,N) NOT USED

ZT17(3 N) NOT USED

ZT17(4,N) LOAD IN CHARACTERS TRANSMITTED

This entry contains the total number of characters transmitted over this channel in multiplexor mode.

CONTROL UNIT TABLE - T18

IT18(1,N) SEIZING ID

The address of the IOTI which is currently using this control unit. If this entry is zero this control unit is free.

If negative, this field contains the # of MPX operations using this control unit.

IT18(2,N) NOT USED

ZT18(2,N) TOTAL CONTROL UNIT TIME

This entry contains the total amount of time in micro seconds for which this control unit has been busy.

DEVICE CLASS TABLE - T19

IT19(1,N) DEVICE TYPE CODE

This entry contains a code which specifies what type of device this entry defines.

1 = UNIT RECORD

2 = TAPE

3 = RANDOM ACCESS

4 = OTHER

AT19(2,N) TRANSFER RATE

This entry contains the rate at which this device is capable of accepting data. This rate is given in characters per second.

AT19(3,N) TRANSFER WIDTH IN BITS

This entry contains the width of the data being transfered in bits.

AT19(4,N) START & STOP TIME

This entry gives the time in micro seconds needed to start and stop this device. This time is added to the channel, control unit, and device time for an I/O operation by the IOADVANCE subroutines.

AT19(5,N) DEVICE TIME

This entry gives the time in micro seconds needed by the device before it becomes free after an I/O operation.

AT19(6,N) TIME LIMIT FOR PENALTY

This entry gives the time in micro seconds which is used to determine if a penalty time should be added to an I/O operation. A penalty time, if any, is added to an I/O operation if the sum of the time when the last I/O operation completed plus the time limit for a penalty is less than the current time. This test is made by the IOADVANCE subroutine.

AT19(7,N) PENALTY TIME

This field contains the amount of time in micro seconds to be added to device, control, and channel time if a penalty as described above is incurred.

AT19(8,N) REWIND TIME

This field contains the amount of time in micro seconds for which this device will be busy if a CLOSE statement which specifies REWIND is created. This field is normally used only for tape devices.

GENERAL SIMULATION TABLE - T20

IT20(1) CURRENT CPU

This entry contains the number of the CPU in the simulation model which is currently executing instructions. This field is altered by the interrupt handler S110.

IT20(2) CURRENT PROGRAM TYPE

This entry contains the current program type.

1 = OS

2 = PRIMARY WORKER

3 = COMMERCIAL WORKER

4 = SCIENTIFIC WORKER

IT20(3) CURRENT OP CODE

This entry contains the current operation code being executed.

IT20(4) to IT20(8) OPERANDS

This entry contains the operands from 1 to 5 associated with the current operation code.

IT20(9) SUBROUTINE ERROR CODE

This entry contains a zero until some subroutines detects an error in the simulator. When a sub routine detects an error, the error number and the subroutines Number are picked up from T20 and printed. Subroutine error codes below 100 terminate the simulation. Subroutine error codes above 100 are of a warning nature and do not terminate the simulator.

IT20(10) NSI CODE

This entry contains the number of statements to be skipped by SM2 before executing the next instruction. If the only exit line from a subroutine is the next sequential instruction this field is left zero. If the next instruction is to be skipped this field is set to 1 and so on.

ZT20(6) CURRENT SIMULATOR TIME

This entry contains the current simulator time as a double precision number accurate to 18 decimal places. This value is kept to one micro seconds precision or better.

IT20(13) SUBROUTINE NUMBER

This entry contains the subroutine number of the current subroutine being executed by SM2. This field is set by the first instruction in each subroutine.

IT20(14) FEC CODE

This entry contains the FEC code to be placed on the future events chain by the PUT FEC subroutine.

IT20(15) FEC ITEM

This entry contains the fec item to be placed on the future events chain by the PUT FEC subroutine.

IT20(16) STORAGE CONTIGUITY CODE

This entry contains a code from 1 to 5 which specifies the manner in which memory is to be allocated.

1 = contiguous instruction pages, non-contiguous data and I/O pages.
2 = non-contiguous instruction pages, contiguous data and I/O pages.
3 = contiguous instruction pages, contiguous but separate data and I/O pages.
4 = contiguous instruction, data, and I/O pages.
5 = non-contiguous instruction, data, and I/O pages.

IT20(17) MEMORY OPERAND -1

This entry contains the first operand of the last MEMORY statement executed. This field is used to insure that a MEMORY statement has been executed before an ALLOCATE statement.

IT20(18) MEMORY OPERAND -2

This entry contains the second operand of the last MEMORY statement executed. This field is used to insure that a MEMORY statement has been executed before an ALLOCATE statement.

IT20(19) LARGEST CONTIGUOUS STORAGE AVAILABLE

This entry contains a count of the largest number of contiguous pages available for the next ALLOCATE statement. It is set by the MEMORY statement and reset to zero by the ALLOCATE statement.

IT20(20) LARGEST CONTIGUOUS STORAGE REQUIRED

This entry contains a count of the largest number of contiguous pages required by the last MEMORY statement. This field is set by the MEMORY subroutine and reset by the ALLOCATE subroutine.

IT20(21) SECOND LARGEST CONTIGUOUS STORAGE AVAILABLE

This entry contains the same information as IT20(19) except for the second largest contiguous storage.

IT20(22) SECOND LARGEST CONTIGUOUS STORAGE REQUIRED

This entry contains the same information at IT20(20) except for the second largest contiguous storage.

IT20(23) NON CONTIGUOUS STORAGE REQUIRED

This entry contains a count of the number of pages of noncontiguous memory required by the last memory statement.

IT20(24) MEMORY TI

This entry contains the transaction number of the last transaction for which a MEMORY subroutine was executed. This field is used to insure that a MEMORY statement has been executed for a transaction before an ALLOCATE statement is executed. The ALLOCATE subroutine resets this field to zero.

IT20(25) PACK CODE

This entry contains a zero until the PACK REQUIRED exit line of a MEMORY statement is taken. A PACK statement resets this field to zero. An ALLOCATE subroutine will issue a diagnostic if a PACK was required by the last MEMORY test but was not performed.

IT20(26) INTERRUPT CODE

This entry contains the number of the current interrupt pending (if any) this field is used by the interrupt handler S110 and reset to zero.

ZT20(14) FEC MANIPULATOR TIME

This entry contains the time at which the current item being placed upon the future events chain is due to come off the chain.

IT20(29) FEC POINTER

This entry contains a pointer to the slot in the future events chain where the PUT ITEM subroutine is to place the FEC CODE (IT20(14)) and the FEC ITEM (IT20(15)).

IT20(30) PAGE SIZE

This entry contains the page size in bits for use by SM1 when determining the instruction, data, and I/O storage required for each transaction.

IT20(32) GENERATE TI WORKER ROUTINE NUMBER

This entry contains the worker routine number from 1 to 99 for which a transaction item is to be generated when calling the generate transaction item subroutine. Upon return from the subroutine this entry contains the address of the generated transaction item.

IT20(33) GENERATE IOTI TRANSACTION NUMBER

This entry contains the transaction item address from 1 to N for which an I/O transaction item is to be generated when calling the generate IOTI subroutine.

Upon return from this subroutine this entry contains the address of the generated transaction item.

IT20(34) STANDARD PRIORITY

This entry contains the standard priority as set by the system parameters for a worker routine which enters the system without a priority.

ZT20(18) STATISTICS INTERVAL

This entry gives the interval in micro seconds between statistics.

ZT20(19) NEXT STATISTIC TIME

This entry contains the next statistic time expressed in micro seconds.

IT20(39) TOTAL NUMBER OF STATISTICS

This entry contains the total number of statistics to be performed.

IT20(40) NUMBER OF STATISTICS COMPLETED

This entry contains the total number of statistics already completed.

IT20(41)

IT20(42) INTERFERENCE

This entry contains the interference rate currently existing in the system.

ZT20(22) INTERFERENCE START TIME

This entry contains the time expressed in micro seconds at which the interference rate last changed.

ZT20(23) INTERFERENCE MICRO SECONDS

This field gives the total percent - micro seconds of interference accumulated to date.

IT20(47) QUEUE LIMIT

This field gives the limiting entry for the queue currently being examined. If the queue is being examined from the top to the bottom this value is equal to the QUEUE-END field in the queue table for this queue. If the queue is being examined from the bottom to the top, this value is equal to the QUEUE-START field in the queue table for this queue.

IT20(48) QUEUE INCREMENT

This field gives the increment which will be added to the QUEUE-POINTER field for the queue being referenced by the EXAMINE-NEXT statement. If the queue is being referenced from top to bottom, this field contains a +1. If the queue is being referenced from bottom to top, this field contains a -1.

IT20(49) RANDOM NUMBER USED

This field contains the random number from 1 to 99 as generated by the random number generator.

IT20(50) RANDOM NUMBER SEED

This field contains the random number seed as set by SM1.

IT20(51) INTERNAL RANDOM NUMBER

This field is initialized to equal the random number seed as set by SM1. Each time the random number generator is referenced IT20(50) = IT20(49) * IT20(50).

IT20(52) NOT USED

ZT20(27) AVAILABLE SIMULATOR TIME

This field contains the amount of time available in all CPU's by this simulator for statistics.

IT20(81) to IT20(100) DISTRIBUTION TABLE

These fields contain the values to be used for the distribution of compute times by the COMPUTE subroutine.

ENTRY	RANDOM NUMBER RANGE
1	0-4
2	5-9
3	10-14
4	15-19
5	20-24
6	25-29
7	30-34
8	35-39
9	40-44
10	45-49
11	50-54
12	55-59
13	60-64
14	65-69
15	70-74
16	75-79
17	80-84
18	85-89
19	90-94
20	95-99

IT20(101) O/S PROGRAM NUMBER CPU 1

This entry contains the number of thw worker routines to be used as an operating system by CPU #2.

IT20(102) MEMORY # FOR CPU #1 O/S

This entry contains the number of the memory from which the space for the operating system for CPU number one should be allocated.

IT20(103) to IT20(110)

These entries contain the same information for CPU's 2 through 5 as the two entries above.

IT20(121) to IT20(127) STATISTIC COUNTERS

These fields contain the counter for each of the statistics subroutines ST1 through ST7. If any of these fields are set negative by a control card, the corresponding statistical print out will be ignored.

IT20(172) SNAP COUNTER

This entry contains a counter of the number of times the snap subroutines has been called. If this field is set to less than zero by control card the snap subroutine will not print.

IT20(173) TRACE LINE COUNTER

This entry contains a line counter for the trace subroutine which allows the trace subroutine to skip to a new page after N lines. This field is set to exceed the page limit by the SNAP subroutine.

IT20(174) TRACE COUNTER

This entry contains a counter of the number of times the trace subroutines has been called. If this field has been set to less than zero by a control card the trace subroutine will not print.

IT20(175) to IT20(200) DUMP SUBROUTINE COUNTERS

These fields provide a count of the number of times each dump subroutine was called. If any of these fields are set to less than zero by the dump control

card the corresponding DUMP subroutine is not executed.

FUNCTION TABLE - T21

AT21(1,N) CHANNEL TIME

This entry specifies the amount of time in micro seconds for which the channel is to be considered busy with this function I/O.

AT21(2,N) CONTROL TIME

This entry specifies the amount of time in micro seconds for which the control unit is to be considered busy with this function I/O.

AT21(3,N) DEVICE TIME

This entry specifies the amount of time in micro seconds for which this device is to be considered busy with this function I/O.

IT21(4,N) DEVICE NUMBER

This entry specifies the device number for which this function I/O is to apply.

STATEMENT TABLE T22

PT22(N)

Each entry in this table contains an operation code or an operand associated with an operation code which will control the execution of subroutines in the simulator.

SWITCH TABLE T23

The switch table contains 200 switches which can be set (turned-on) or reset (turned off). For each of the 5 CPU's there are 20 local switches.

CPU #1	SW 1 - 20
CPU #2	SW 21 - 40
CPU #3	SW 41 - 60
CPU #4	SW 61 - 80
CPU #5	SW 81 - 100

Whic are referenced as SW 1 - 20. Therefore, it is impossible to reference any switch with a number greater than 20 or less than 101.

In addition there are 100 switches numbered 101 to 200 which are global across all CPU's.

PT23(N)

Each entry contains a -1 for a switch which is off or a +1 for a switch which is on. All switches are initially turned off.

TO - FROM T24

AT24(N)

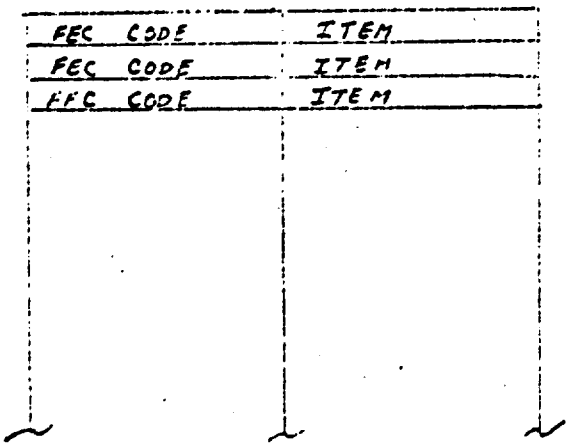
Each entry in this table contains a time in microseconds as entered from the system parameters to construct a TO - FROM Table.

MARK TIME TABLE T25

ZT25(N)

Each entry in this table contains the sum of all the compute times executed with a mark # equal to N.

FUTURE EVENTS CHAIN



T^

T 2

CPU ITEM

1 COT
AT
TCT
OS PROG #
MEMORY #
CHANNEL # 1
2
3
4
24
18
19
20
IOTEM. INTERRUPT ADDR
INTERRUPT CTR
READ/WRITE
FUNCTION
CLOCK
RECEIVE
ITEM
END SECT
49
50 OPEN/

NOT USED
NOT USED
NOT USED
CPU ID
LOGICAL DATA UNIT
#DEC DIGITS
DEC ADD A
DEC MULT A
DEC DIVIDE A
FIX ADD A
MULT A
DIV A
FLOAT ADD A
MULT A
DIV A
INS LEN
DEC DIGITS
CHAR/UNIT
MOVE TIME/HU A
MOVE + EDIT A
FIX E1
E1
E2
E2
E3

101 E3
102 E4
E4
FLOAT E1
E1
E2
E2
E3
E3
GIBSON TIME
CLOCK ITEM
TIME
TOTAL O/S TIME
TOTAL O/S IOT QUEUE TIME
START CYCLE TIME
TOTAL CYCLE TIME
120
121 TOTAL O/S
122 TQ QUEUE TIME

NOT REPRODUCIBLE

TRANSACTION ITEM
PART - 1

1	R/W POINTER	-1 = FREE
2	CURRENT FILE REQUEST ORIGINAL DISC #	+=INPT, -=OUTPT
3	I/O	
4	RUN CLASS POINTER F=NOT ASSIGNED	
5	PRIORITY	
6	NO. I LOC	
7	# IOT OUT	
8	PROGRAM TYPE	
9	CPU # FEC	
10	LOOP ADDR 1	
11	LOOP CTR-1	
12	LOOP ADDR 2	
13	LOOP CTR 2	
14	LOOP ADDR 3	
15	LOOP CTR 3	
16	SUB-EXIT-1	
17	SUB-EXIT-2	
18	SUB-EXIT-3	
19	PRINT LINES	
20	-1	
21	TIME - POINTER	
22	PART-2- POINTER	

TRANSACTION ITEM
TIMES

1	CREATION TIME
2	TIME OUT FEC
3	SERVICE TIME
4	TOTAL CPU TIME
5	TI QUEUE START TIME
6	TI TOTAL QUEUE TIME
7	IOT TOTAL QUEUE TIME
15	
16	
17	
18	
19	AT SAVE
20	IOT SAVE
21	CALLING TI #
22	CALLED UP #

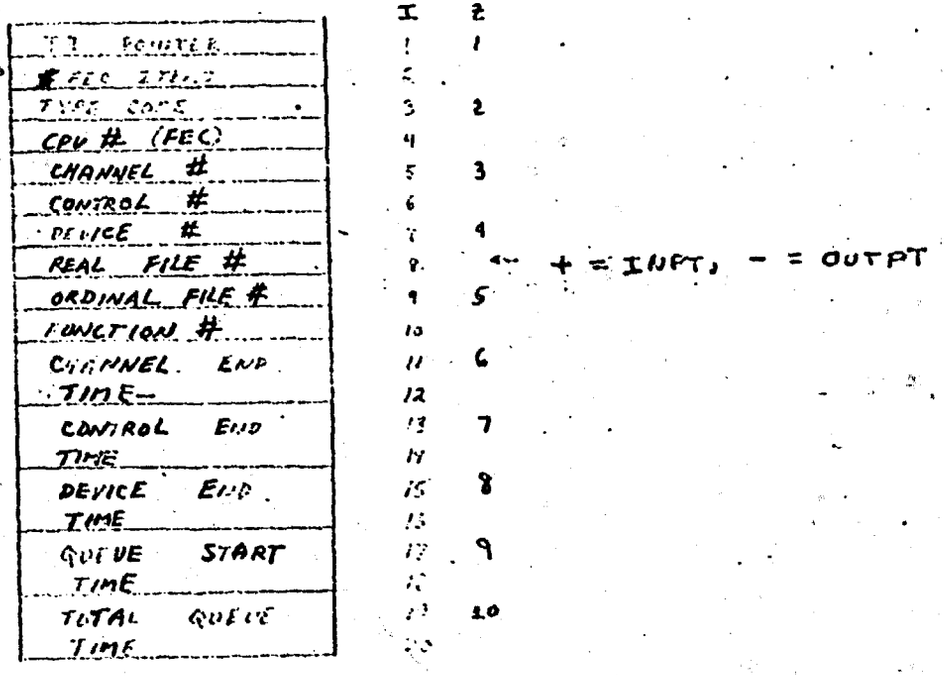
TRANSACTION ITEM
PART - 2

BUFFER COUNT	FILE # 1
RECORD COUNT	
TOTAL I/O OPERATIONS	
	FILE # 2
	FILE # 3
	FILE # 4
	FILE # 5
	FILE # 6
	FILE # 7
PART-3- POINTER	

TRANSACTION ITEM TABLE

T 3

NOT REPRODUCIBLE



I/O TRANSACTION ITEM

T 4-

NOT REPRODUCIBLE

SPREAD 0=FIXED 1=POISSON	A	1
LIMIT	I	2
TOTAL # GENERATED	A	3
LOAD CLASS POINTER	I	4
PRIORITY	I	5
FIRST INST LOC	I	6
1= NON-REENTRANT 2= REENTRANT	I	7
1=C/C 3=COMM 2=P/W 4=SCI	I	8
# TT OUTSTANDING	I	9
RECEIVING CPU #	I	10
# OF FILES	I	11
FILE VECTOR TABLE	I	12
INST STORAGE	A	13
DATA STORAGE	A	14
I/O STORAGE	A	15
TOTAL TT CALLED	I	16
NOT USED -1=NOT USED	A-I	17
TT COUNTER	A-I	18

-1 W/R BASE NOT USED
+1 PROG DICT CARD FN
+2 W/R STATEMENTS RECEIVED

WORKER ROUTINE BASE

T 5

NOT REPRODUCIBLE

WORKER ROUTINE
FILE VECTOR TABLE

T 6

	1/2	1/2		
1	REAL FILE#	# BUFFERS	SEIZING CD.	NOT USED
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				

MEMORY TABLE T7

LOGL PAGE	HIGH PAGE # PACKS	# IN MEM	PRES BODY	NOT USED	LAST REF TIME	MEMORY LOAD

PAGE TABLE T8

SEGMENT ID.

NOT REPRODUCIBLE

T 9

LOAD CLASS TABLE

1/2

CLASS	CPU #	CPU #	CPU #	CPU #	CPU #
1					
2					
3					
4					
5					
6					
7					
8					
↓					
15					

T 10

RUN CLASS TABLE

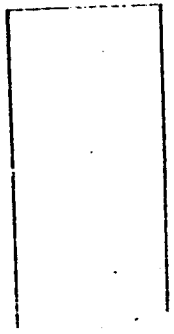
1/2

CPU #	CPU #	CPU #	CPU #	CPU #	CPU #
1					
2					
3					
4					
5					

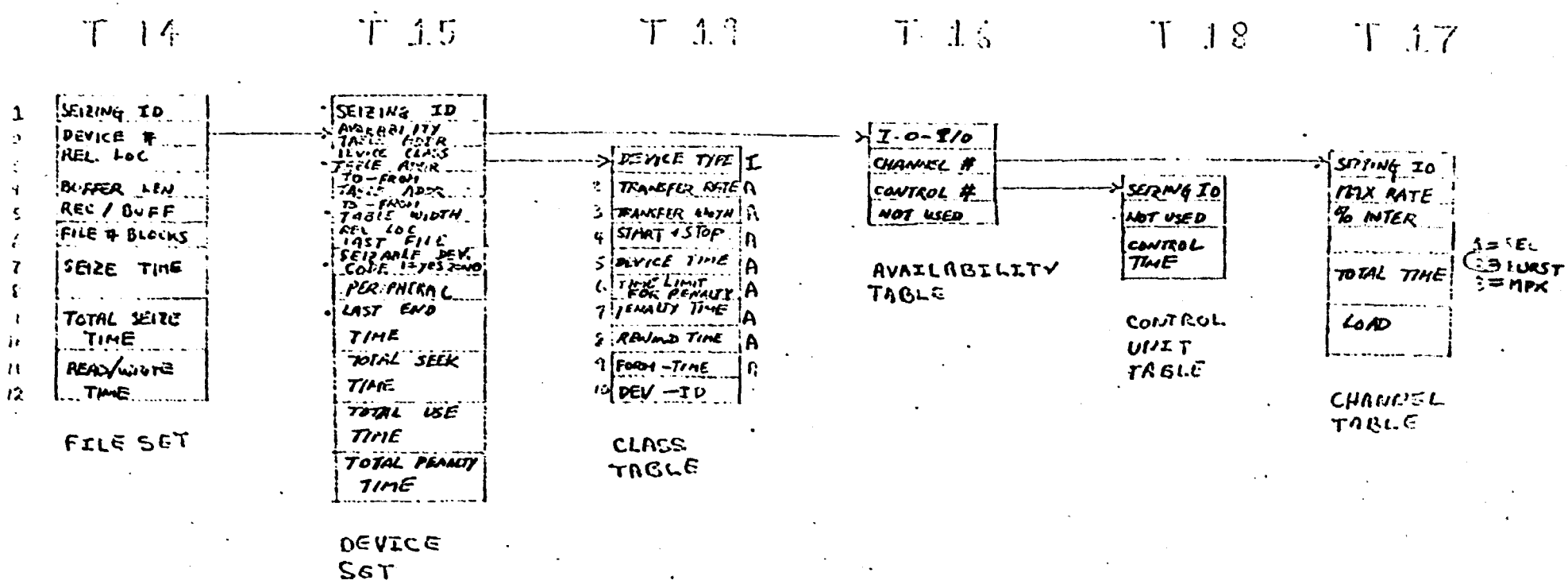
QUEUE TABLE (T12)

QUEUE #	QUEUE START	QUEUE END	QUEUE TYPE	ENTRY TYPE	QUEUING DISCIPLINE	MAXIMUM # ENTRIES	CURRENT # ENTRIES	NOT USED	LAST REFERENCE TIME	QUEUE LOCK
1				1= TI	1= PRIORITY					
2				2= ROTI	2= FIFO					
3					3= LIFO					
4										
5										

QUEUE ENTRIES (T13) P



NOT REPRODUCIBLE



NOT REPRODUCIBLE

GENERAL SIMULATION TABLE

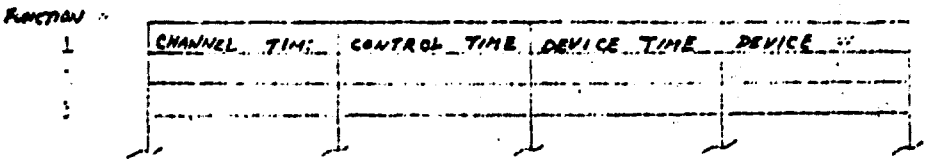
1	CURRENT CPU	✓	41	TI ADC→S 41		P1		121	Q-STAT CTR	S11	161		
	CUR. INS. TYPE	✓	42	INTERFERFACE	(3)	P2		122	MEM-STAT CTR	S12	162		
	CUR OP. CD	✓	43	INT START TIME		P3		123	FILE-STAT CTR	S13	163		
	OPERAND 1	✓	44			P4		124	DEV STAT CTR	S14	164		
	2	✓	45	INT-SEC TOTAL		P5		125	CTL-STAT CTR	S15			
	3		46			P6		126	CH-STAT CTR	S16			
	4		47	Q LIMIT		P7		127	CPU-STAT-CTR				
	5		48	Q INCREMENT		P8							
	SUB. ERR. CODE		49	RANDOM #	= FOR SUB	P9							
	NSI CODE		50	RANDOM # SEED		P10							
11	CURRENT SIM TIME	2	51	INT RAND #		P11							
			52	AVAIL SIM TIME		P12						1	SNAP TRACE CTR
	SUB #		53			P13							TRACE-SW
	FEC CODE					P14							DT1 -CTR
	FEC ITEM					P15							DT2 -CTR
	STORAGE					P16							DT3 -CTR
	MEM OP 1					P17							DT4 -CTR
	MEM OP 2					P18							DT5 -CTR
	LCA					P19							DT6 -CTR
	LCR					P20							DT7 -CTR
	SCA					PROG #	— CPU 1						
	SCB					REAL #							
	NCR					PRIE	2					184	
	MEM TI					MEM	3					185	
	PACK CODE											186	
	INTERRUPT CODE						4					187	
27	FEC MANIPULATOR TIME	2										188	CTR
							5					189	DT15-CTR
	FEC PTR											190	DT16-CTR
	PAGE SIZE											191	DT17-CTR
	MEM. UNIT											192	DT18-CTR
	GEN TI PROG #											193	DT19-CTR
	GENTOTI. TI. #											194	DT20-CTR
	STANDARD PRI											195	DT21-CTR
	STATISTICS	2										196	DT22-CTR
	INTERVAL											197	DT23-CTR
37	NEXT STATISTICS	2										198	DT24-CTR
	TIME											199	DT25-CTR
	TOTAL # STATISTICS											200	DT26-CTR
	# STATISTICS DUMP												

NOT REPRODUCIBLE

T20

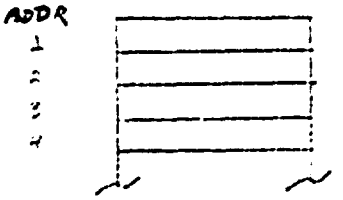
200 ENTRIES

FUNCTION TABLE



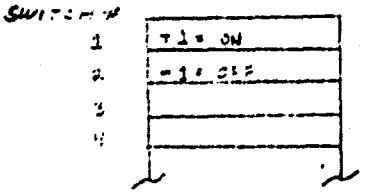
T 2 1

STATELIST TABLE



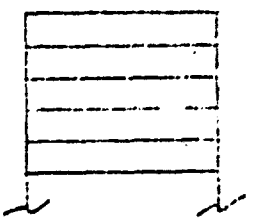
T 2 2

SWITCH TABLE



T 2 3

TO-FROM TABLE



T 2 4