UNIVERSITY OF ILLINOIS
DIGITAL COMPUTER

Aux.
LIBRARY ROUTINE Q5-237
By James Shelly

TITLE Complete Circuit Analyzer

TYPE Complete Program

NUMBER OF WORDS Main program 324
Function compiler 117 Print routine 13
$E(I_k)$ routine $18 + 2n + a + f + k$ where n is the number of nodes of the circuit, a is the number of auxiliary functions, f is the number of disallowed states used, and k is the total number of integers and plus signs used to describe the circuit.

TEMPORARY STORAGE 0-19 working storage
20-35 storage for printout
500-500+n+a-1 in $E(I_k)$ routine.

DURATION Depends on circuit to be analyzed.

LIMITATIONS The number of nodes n must satisfy $n \leq 78$. The numbers n,k,a,f defined above must satisfy

$$3n + 2a + f + k \leq 516$$

The number of states S possible at any time must satisfy

$$S \leq 853.$$

I. PURPOSE This routine analyzes a complete circuit for speed independence, where the circuit is defined by the logical equations for each decision element and the specification of the initial state. The user also supplies a stop control and print out information.

This routine is an extension of Illiac library routine Q-3, but enables the user to analyze larger circuits in a more convenient fashion than the earlier routine could, subject to the limitations listed above. The time taken to analyze a circuit is quite indeterminate, but is mostly due to a search on the magnetic drum. The length of search is proportional to the number of states in the new list, while the number of searches is proportional to the number of excited nodes, so that the time taken on the longest search is roughly proportional to the square of the greatest number of simultaneously excited nodes.

II. INTRODUCTION A decision element is an element with an arbitrary number of input lines and one output line where the signals on all lines are taken only to possess two values, 0 and 1. To each decision element there corresponds a Boolean function of the input signals and possibly the output signal itself. We specify the state of the element by listing a set of signals on the input and output lines. The element is then said to be in equilibrium if the computed value of the function equals the signal assigned to the output line; otherwise it is said to be excited. An element is always said to act in such a way as to place itself in equilibrium and will only act if excited; but no restriction is placed on the time it takes to act.

An interconnection of decision elements at points called nodes has the property that each node has at most one output line feeding it and feeds at least one input line. Since no assumptions are made for the relative speeds of the decision elements, we cannot consider clocked or synchronous circuits. Hence, the above interconnection will only apply to asynchronous circuits.

A complete circuit is an asynchronous circuit such that every node has one and only one decision element feeding it. Any asynchronous circuit can be considered to be a complete circuit by attaching fed-back delay elements (elements whose input equals its output) to those nodes which have no decision elements feeding them (see Section I). Since every node in a complete circuit is fed by a decision element we shall label each node by the decision element feeding it.

An immediate state of a complete circuit is defined by listing the value of the signal (either 0 or 1) at each node in the circuit. Since every node is fed by one decision element the immediate state (abbreviated as I-state) determines uniquely those nodes which are excited. An I-state S is an equilibrium state if every decision element is in equilibrium. An I-state B is said to directly follow an I-state A if B results from A by no more than one decision element acting and if that decision element was excited when the circuit was in I-state A. B is said to follow A if there exists a sequence of states $A = A_1, A_2, A_3, \dots A_k = B$ such that each state $A_{i+1} (i = 1 \text{ to } k - 1)$ directly follows the state A_i. It is important to note that there might be states A and B where neither A follows B nor B follows A.

A *cumulative state* (abbreviated C-state) is defined with respect to some initial I-state, I_0, by listing the number of changes that have occurred at each node since the circuit was last placed in I_0.

A complete circuit is said to be *speed independent with respect to an I-state S* if for every I-state which follows S no decision element passes from an excited state to an equilibrium state without acting.

If now we extend the definitions of follows and directly follows to the C-states and use "follows" as a partial ordering relation, then it can be shown that the set of C-states of a speed-independent circuit forms a semi-modular lattice [cf. 1]. Using this fact it then can be shown that there are only two sub-classes of speed independent circuits defined by the types of lattices of the C-states.

A speed independent circuit is said to be *totally sequential with respect to an initial state S* if for any two C-states A and B which follow S, either A follows B or B follows A holds. This is equivalent to saying that for any I-state T which follows S at most one decision element is excited; hence at most one I-state may directly follow T.

A speed independent circuit is said to be *distributive with respect to an initial state S* if for any three C-states A, B and C which follow S with C directly following A and B, then there exists a C-state D which follows S such that both A and B directly follow D. Since a C-state may directly follow itself, it is clear that a totally sequential circuit is distributive. If one tries to apply this definition to the I-states then in general it will not work, but the cases where it fails are those circuits which eventually break up into at least two independent parts. That is the circuit behavior at a particular set of the nodes is independent of the circuit behavior at the remaining nodes. We shall define the three classes as *totally sequential*, *distributive but not totally sequential*, and *semi modular but not totally sequential or distributive*.

III. METHOD

Given a complete circuit with n nodes labeled from 1 to n then an I-state I_k will be denoted by a vector

$$I_k = (i_k^1, i_k^2, i_k^3, \dots i_k^n)$$

where the i^j_k are the values of the signals either 0 or 1 at the nodes. The set of excited decision elements $E(I_k)$ will be denoted by a vector

$$E(I_k) = (e^1_k, e^2_k, e^3_k, \dots e^n_k)$$

where $e^j_k = 1$ if the j^{th} node is excited, otherwise $e^j_k = 0$. The notation $E(I_k)$ formalizes the fact that the set of excited nodes is uniquely determined by the I-state I_k. These numbers are stored in the machine as words with the j^{th} component in the $(2^{-j})^{th}$ position, their sign always being positive.

The memory is divided into two sections A and B: one containing the list of current states and excited nodes during a cycle, the other containing a list of states which directly follow the current states along with their excited nodes. In addition these lists contain a third word for each state which is used for testing distributivity and will be described later. The routine begins by selecting a state I_k in the current list and its excited nodes $E(I_k)$. It then forms a new state by selecting an excited node appearing in $(e^1_k\ e^2_k\ \dots\ e^n_k)$, say it is e^j_k. It forms a new state I^*_k by allowing this node to pass to equilibrium hence

$$I^*_k = (i^1_k, i^2_k, \dots i^{j-1}_k, \overline{i^j_k}, i^{j+1}_k \dots i^n_k)$$

It then examines the list of directly following states to see if I^*_k has already been generated. If it had not been generated it plants it in the new list and enters the function routine to calculate $E(I^*_k)$ and stores the excited nodes corresponding to I^*_k. (If it was in the list it would already have done so). Next it forms $E(I_k)^*$ which is

$$E(I_k)^* = (e^1_k, e^2_k, \dots e^{j-1}_k, 0, e^{j+1}_k, \dots e^n_k)$$

It now is able to make a test for speed independence which is that no other node in $E(I_k)^*$ except possibly the j^{th} node passes into equilibrium. This is equivalent mathematically to:

$$E(I_k)^* \subseteq E(I_k{}^*) \quad \text{(componentwise)}$$

If $E(I_k)^* \not\subseteq E(I_k{}^*)$ the circuit has violated speed independence and the routine jumps to a failure stop (see "non-programmed stops") otherwise it adjusts the third word for $I_k{}^*$ (see distributive test) and returns to I_k and picks the next excited node in the sequence e_k^{j+1} e_k^{j+2} ... e_k^n. It continues this process until all the states which directly follow I_k have been exhausted. It then goes to the next state in the current list, repeats the process on this state and continuing in this manner it finally exhausts the current list. An overwrite test is made to see if the directly following list overwrites its memory capacity.

IV. SPECIAL TESTS Initially a switch had been set placing the circuit in the totally sequential class. If after exhausting any list (including the first list of just the initial state itself) it did not generate more than one state in the list that directly follows the current list, then the circuit remains totally sequential and no distributive test need be made. However, the first time more than one state is generated the circuit is no longer totally sequential, and a distributive test must be made on the next set of directly following states. It need not make the test when it first branches since trivially all those states arose from one state.

We now define a set of causation nodes for a directly following state $C(I_k^*)$:

$$C(I_k^*) = (c_k^1, c_k^2, c_k^3 \dots c_k^n)$$

Let $$I_k^* = (i_k^1 \; i_k^2 \; i_k^3 \dots i_k^n)$$

then $c_k^j = 1$ if there exists a state I_m in the current list where

$$I_m = (i_m^1 \, i_m^2 \, i_m^3 \dots i_m^{j-1} \; \overline{i_m^j} \; i_m^{j+1} \dots i_m^n)$$

and where I_k^* directly follows I_m.

Now assume we have exhausted the current list and that a distributive test must be made. The routine then works with the directly following list but instead of using the excited nodes to obtain new states it uses the causation nodes. Since we are using $C(I_p)$ instead of $E(I_p)$ then the set of states I_p^* (which directly follow I_p using $C(I_p)$ as the excited nodes) will always be found in the list of current states. This results directly from the definition of $C(I_p)$. The routine tests for distributivity by testing for speed independence with respect to the causation nodes. That is if

$$C(I_p)^* \not\subseteq C(I_p^*)$$

the circuit is no longer distributive (provided also that the circuit is composed of not more than one independent circuit). Hence, if it fails the test the routine reclassifies the circuit as semi modular and no longer needs to make the distributive test. If not it will continue until the list of directly following states is exhausted. In either case the routine then relabels the lists by calling the present directly following list the new current list and comes to a black switch stop (see non-programmed stops).

V. PROGRAMMED AND NON-PROGRAMMED STOPS The program accepts two types of programmed stop specifications.

(1) State count stop. This stops after a specified number of new states have been generated.

(2) Node change stop. This stops after a selected node has undergone a specified number of changes.

Type (2) stop is the most frequently used.

Of the six types of non-programmed stops, the most important is the cycling stop. Any circuit must either come to equilibrium, or continue to repeat some portion of its earlier behavior. The cycling test compares the A lists for the same number of words and the same cyclic check sum. If these two agree, the circuit has cycled. The routine punches "cycling" and looks for new circuit. If not, then the old check sum is replaced by the new one if the number of times in the A-list is a power of two. This is done to catch cycles which have any period whatsoever.

The other five types of non-programmed stops are as follows.

(2) Failure stop. This stops when the circuit fails the speed independence test, and punches "failure", followed by the current state, its excited nodes, the directly following state and its excited nodes.

(3) Overflow stop. This stops when the list of directly following states exceeds drum memory capacity. This prevents erroneous failure indications from occurring. The word "overflow" is punched out.

(4) Equilibrium stop. If a state passes into equilibrium, the word "equilibrium" and the equilibrium state are punched out.

(5) Black switch stop. Normally the circuit is run with the black switch down. If the switch is set to obey, the program will stop at the end of the list through which it is now passing. A white switch bypass leads to the final stop, where the circuit class is punched.

(6) Disallowed state stop. If a certain state or states may cause faulty operation of the circuit, the logical sum of these states may be punched on the circuit tape, followed by an F. When one of these states is reached, the program stops, and punches "disallowed", followed by the current state, its excited nodes and the directly following (disallowed) state.

All stops except the failure stop punch the class of the circuit, "sequential", "distributive", or "semimodular". The routine then reads the next circuit or a new initial state.

VI. OUTPUT CONTROL The user may select a set of nodes such that when any node of this set changes, an output occurs giving the current state, its excited nodes, the directly following state and its excited nodes.

VII. DATA TAPE PREPARATION This routine has two possible input modes, new circuit read-in or new initial state. These are distinguished by having a character consisting of a single fifth-hole only delay character under the reader when a new circuit is to be analyzed, and any different 5th hole character under the reader if a new initial state is to be read.

The input routine is very flexible, due to the availability of auxiliary functions a_ℓ, so that the nodal equations $z_i = f_i \ (z_i, \ldots z_k)$ are written in the form

$$z_i = \bigvee_{j=1}^{m} g_{ij} \text{ , where } g_{ij} \text{ is a product of the}$$

z_i's and/or of the a_e, with each factor independently complemented or uncomplemented. For example, the function

$$z_5 = (z_1 \bar{z}_2)\ (z_5 + z_7 \bar{z}_{10})$$

may be expanded into sums of products as

$$z_5 = z_1 \bar{z}_2 \bar{z}_5 z_7 \bar{z}_{10} \vee z_1 \bar{z}_2 z_5 \bar{z}_7 \vee z_1 \bar{z}_2 z_5 z_{10} ..$$

If we use an auxiliary function a_1

$$a_1 = \bar{z}_5 z_7 \bar{z}_{10} \vee z_5 \bar{z}_7 \vee z_5 z_{10} ,$$

we may write

$$z_5 = z_1 \bar{z}_2 a_1$$

Using two auxiliary functions a_1, a_2

$$a_1 = z_7 \bar{z}_{10}$$

$$a_2 = \bar{z}_5 a_1 \vee z_5 \bar{a}_1$$

and

$$z_5 = z_1 \bar{z}_2 a_2$$

After writing down the logical equations for the nodes and auxiliary functions used to describe the circuit, the data tape is prepared as follows.

(1) The number a of auxiliary functions is punched as a decimal integer inside parentheses.

(2) The number n of nodes is punched as a decimal integer, followed by any 5th hole character or space.

(3) The a auxiliary functions are punched in order, followed by the n nodal equations. These functions are subject to the following conventions.

   (a) The left hand sides ("z=") of the equations are omitted.

   (b) The term z_i is designated by the decimal integer i terminated by a space.

   (c) The term $\bar{z}_i$ is designated by the decimal integer i terminated by a prime.

(d) The auxiliary function a_j is designated by the decimal integer j in parenthesis followed by a space.

(e) The auxiliary function $\bar{a}_j$ is designated by the decimal integer j in parenthesis, followed by a prime.

(f) No auxiliary function may refer to another auxiliary function which follows it in the list.

(g) The operation $\vee$ is designated by + (K).

(h) Each nodal or auxiliary function is terminated by an N.

(i) Any number of 5th hole characters may precede an integer, but no 5th hole character may separate the digits of an integer or the integer and its terminations.

(j) Any number of 5th hole characters may precede or follow a + or N.

(k) A disallowed state is punched as above, and may appear anywhere inside the list of functions. That is, it may not follow the last nodal equation. The disallowed states function is terminated by an F.

The first function illustrated above may be punched as

(i) 1 2'5'7 10' +1 2'5 7'+1 2'5 10 N or

(ii) 1 2"5'7 '10''' +' '1 2'5 7'+1 2''5 '10 N

but not as

(iii) 1 2'5'7 1 0'+... , in which a space separates the digits of 10.

(3) Following the nodal equations the initial state is punched using $[\frac{n}{4}] + 1$ sexadecimal digits, where [k] is the greatest integer contained in k. If we had 5 nodes, an initial state of (10111) would be punched as S8, the digits being grouped by fours from left to right.

The output during the running of the program will have the same number of characters and significance of grouping.

(4) The stop information is next punched. If the state count stop is desired, then the number of new states is punched as a decimal integer terminated by a 5th hole character, followed by an N. If a node change count stop is required,

then the node number, terminated by a 5th hole character, is punched, followed by the number of changes and a 5th hole character. If neither of them is desired, the symbol N should be punched and the cycling test will be used for the normal stop.

(5) Following the stop control, the print control constant is punched. This has the significance that wherever a one appears, output will occur when the corresponding node changes. The grouping corresponds to that for the initial state. A hand punch may be used, if needed, to obtain more output. For example, if in a 5 node circuit the user is only interested in changes on node 2, he would punch 40. If he later wished to investigate nodes 3 and 5, he would punch 68.

(6) If the punch control is followed by a fifth-hole delay, a new circuit will be read in. If followed by any other 5th hole character, a new initial state will be input. All circuits or initial states on a given tape will be read automatically

The last circuit or initial state on the tape should be terminated by a fifth hole delay and a 0, to cause the program to proceed to the OF order.

If the black switch is set to obey on readin of a circuit, the machine will stop, enabling the user to bypass any number of initial states which he wishes.

The important things to remember are:

(A) No 5th hole character may separate the digits of an integer, and all integers are terminated by 5th hole character.

(B) All integers for auxiliary functions are enclosed in parentheses.

(C) Every nodal equation and auxiliary function is terminated by N, every integer representing a node or auxiliary function is terminated by either a space or prime, and every disallowed state is terminated by an F.

(D) The list of equations is preceded by the number of auxiliary functions in parentheses, and by the number of nodes followed by a 5th hole character.

(E) A fifth-hole delay must be under the reader to read in a new circuit. Any other 5th hole character reads in a new initial state, stop control and punch control.

(F) The number of sexadecimal characters for initial state and punch control is the number required to represent n bits, (where n is the number of nodes).

(G) Stop constants.

a - N   stops after a states   (a decimal integer)

a   b   stops after node a changes b times, (a, b both decimal integers)

N   cycling (normal) stop

VIII. USE The main tape is read into a cleared memory. If the sum check fails, it stops on an FF order. The data tape is placed in the reader, and after read in of equations, initial state, stop and end constants, one throw of the black switch will enable the circuit to be tested through one cycle. Normally the circuit is run in stop-disable. The routine will continue on to the next circuit or state until the end of tape is reached, coming to OF for a circuit with no nodes. If desired, a new data tape may be inserted at any black switch stop after the last print constant on the previous tape has been read.

IX. COMPILING ROUTINE This is a special input routine used to translate the equations and auxiliary functions into a subroutine for calculating $E(I_k)$. The subroutine calculates $E(I_k)$ by the following method.

(1) First n numbers are stored, such that if z_i is 1 the number is negative and if z_i is 0, the number is positive.

(2) The auxiliary and nodal functions are calculated in order by a series of conditional transfers, where if the function is 1, a 1 is added in the corresponding digital position.

```
L5  (z1)              36 (*)
F1  (a2)              36 (*)
36                    26 (**)
L5  (z4)              36 (***)    *
L5  (z6)              36 (***)
19  (node no)         00 1        **
L4  (marker)          40 (marker) .
```

The node number and marker addresses are adjusted in the subroutine to handle nodes numbered higher than 39.

Function forming is similar for auxiliary functions, except that no digital marker is formed. Instead, a negative word is stored if the function is 1, a positive word if the function is 0.

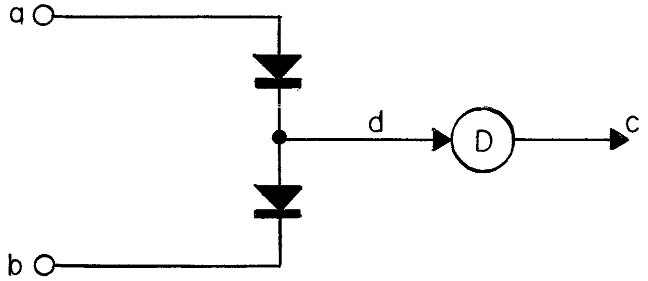
(3) At the end of the routine, the function $E(I_k) = I_k +$ (equation values) (digitwise) is formed, and control is transfered to the main program.

(4) In addition to compiling the $E(I_k)$ routine, the compiler plants appropriate test-constants and addresses in the main routine.

The function routine is generated by the compiler during the input of the equations. Upon entry to the function routine, a set of markers is generated so that if the i^{th} node has a one, the i^{th} marker is negative. Orders of the form L5 36 or F1 36 are chosen depending upon whether the corresponding node number in the input was followed by a space or prime, signifying that the variable or its complement respectively was required. The addresses for the L5 and F1 orders are set to pick up the desired marker. The 36 orders are set to transfer to the next function or to the next term, whichever comes next. The + or "or" connective transfers directly to the end of the function.

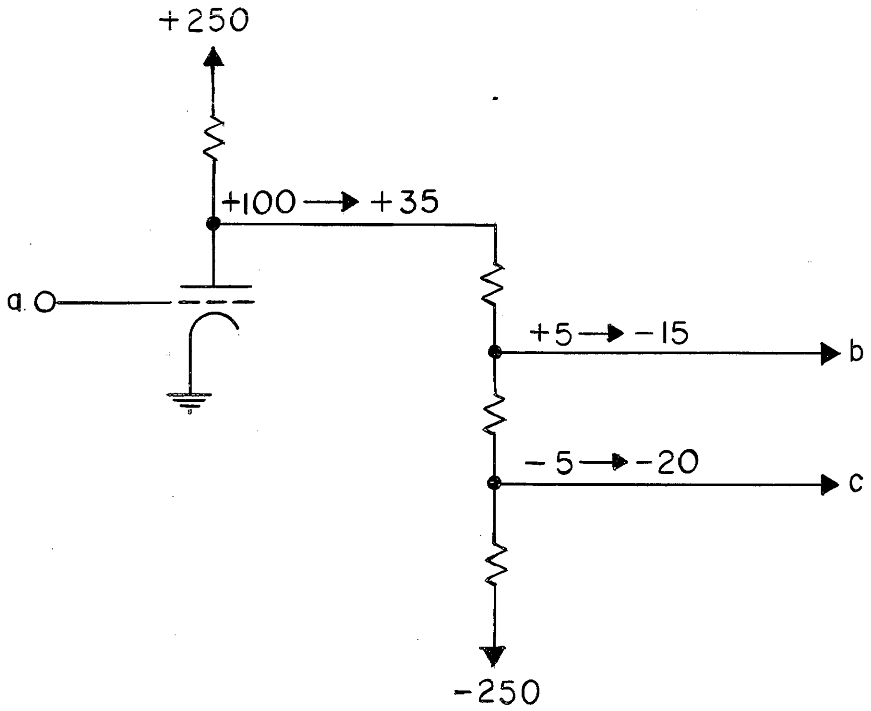
The end of function word pair generates a marker corresponding to the number of the node being generated. This marker is added to the sum of the set of markers previously generated, and restored. After all nodes have been evaluated, we have the corresponding (eventual) new state. The excited nodes are then extracted by the "exclusive or" operation in the last eight words of the function routine, and the control is then transfered to the main routine.

X. MISCELLANEOUS ASPECTS A situation where the disallowed state stop would be useful can be illustrated by considering the circuit shown below. If positive logic (positive voltage = 1) is used, the point d is not defined for the input combination a = 1, b = 0.



This may be taken care of by using the function ab'F, the disallowed state.

Sometimes it is convenient to represent divider chains in a circuit, for consider the following circuit:

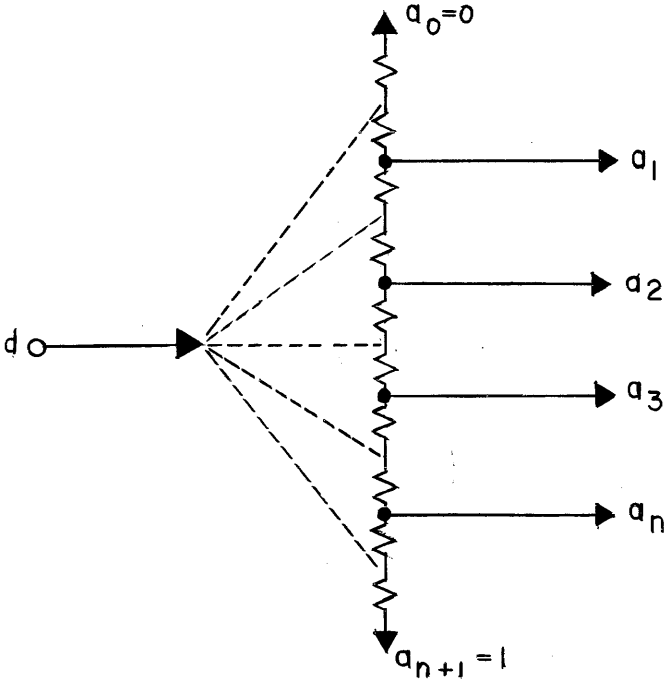


where both lines b and c may be shunted by arbitrary capacitors. Thus it is no longer possible to say b = c as would be done normally. This "bleeder chain has the property that at no time can b have the value 1 and c have the value 0 in negative logic. Thus b can only take on the value $\bar{a}$ when c is 1. The logic for lines b and c would then be

$$b = \overline{a}c$$

$$c = (\overline{a} \vee b)$$

The general divider chain with outputs $a_1\ a_2\ \ldots\ a_n$ pictured below



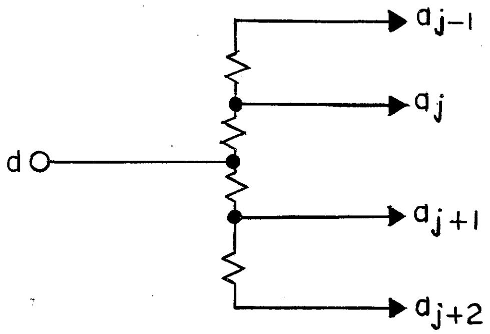
would have as the system of equations

$$a_1 = da_2 \vee a_0$$

$$a_2 = da_3 \vee a_1$$

$$a_n = da_{n+1} \vee a_{n-1}$$

where d is the driving signal. d has the property that if it is situated between a_j and a_{j+1}, as shown in the figure,



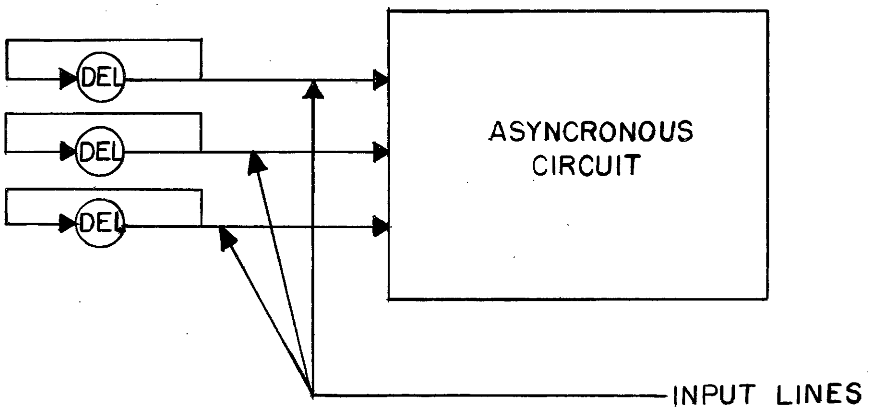
then the lowest voltage value of d = 1 must be greater than the highest voltage value at $a_{j+2} = 0$. Otherwise, it would be possible for $a_{j+1} = 1$ and $a_{j+2} = 0$. Also, the highest voltage value of d = 0 must be less than the lowest voltage value at $a_{j-1} = 1$, for if not ~~it~~ would be possible for $a_{j-1} = 1$ and $a_j = 0$. The divider using positive logic becomes $a_0 = 1$, $a_{n+1} = 0$,

$$a_1 = da_0 \vee a_2$$

$$a_2 = da_1 \vee a_3$$

$$a_n = da_{n-1} \vee a_{n+1}$$

A general asynchronous circuit can be handled by feeding all nodes having no decision elements attached to them with fed back delay elements. This then allows the user to attach a signal to what normally would be called the input lines.

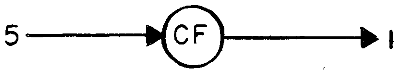
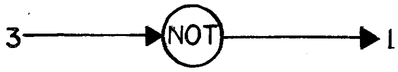
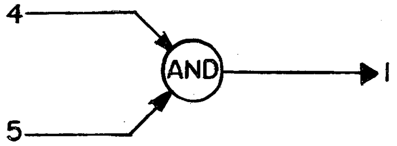
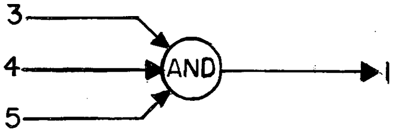
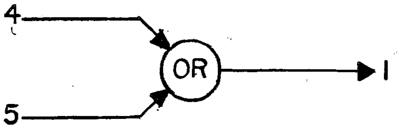


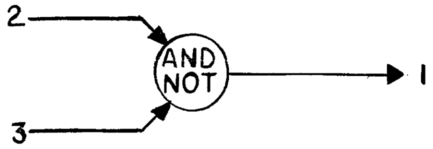
This method requires that the inputs remain fixed during the analysis. If one wishes one of them to change from a 0 to 1 then that line could be tied to a "one" elemenc (i.e. $z_1 = z_1 \vee \bar{z}_1$) which was initially placed in the 0 state.

XI. REPRESENTATION OF VARIOUS LOGICAL ELEMENTS

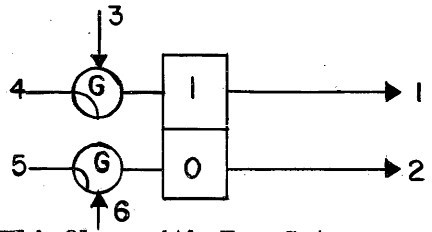
Several examples of data tape preparations for Illiac type logical elements and special logical elements are given below. For construction of these elements the reader is referred to Reference 2 for Illiac types and References 3, 4, 5 for several of the special types. These elements are written with integers specifying input and output lines to indicate how the set of nodal equations would be typed on the tape. All outputs are designated by node number 1 and if needed node number 2.

ILLIAC TYPES

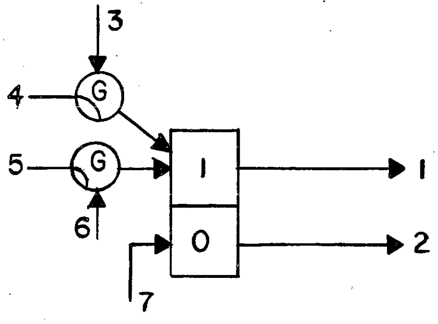
Circuit Designation	The First and Possibly the Second Nodal Equation would be Punches as:
 Cathode Follower or Delay:	5 N
 Not:	3'N
 And:	4 5 N
 3 Input And:	3 4 5 N
 Or:	4 +5 N



And Not: 2'+3'N

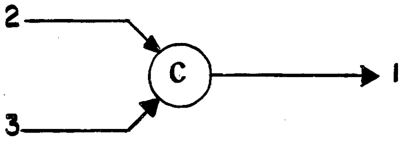


Flipflop with Two Gates: 3'4 +2'N
5 6'+1'N

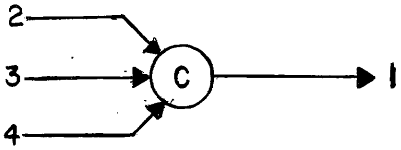


Flipflop with Clear and Two Gates: 3'4 +5 6'+2'N
7 +1'N

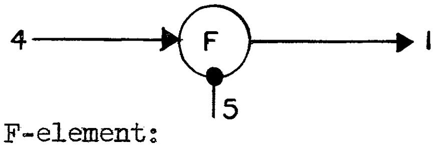
ADDITIONAL TYPES



C-element: 2 3 +1 2 +1 3 N

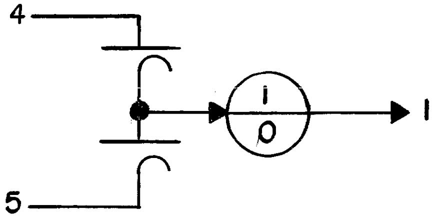


Three Input C-element: 2 3 4 +1 2 +1 3 +1 4 N



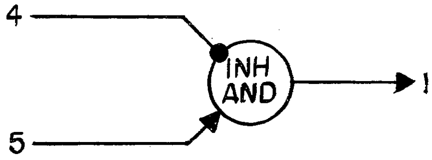
F-element:

4 5 +1 5'N



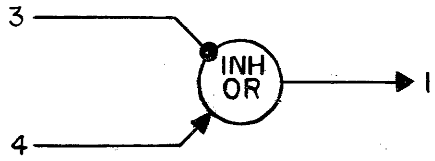
Single Output Flipflop
with Two Diode Gates:
(negative logic)

4 5 +1 4 5'+1'4'5 N



Inhibit And:

4'5 N

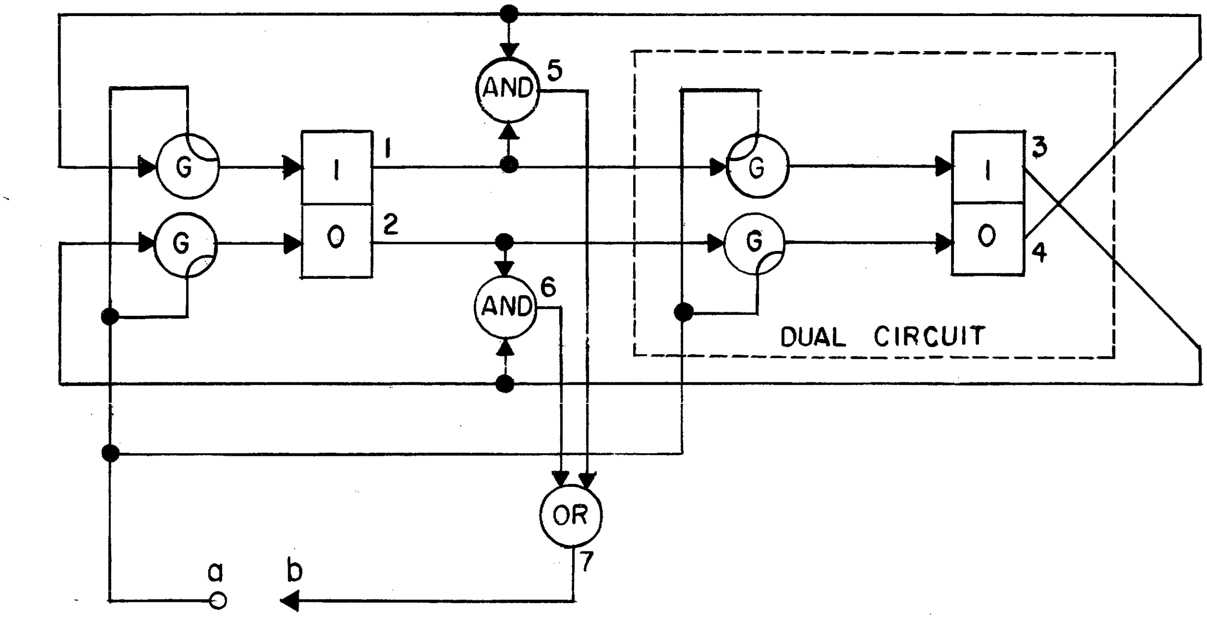


Inhibit-or:

3'+4 N

XII. A SPEED-INDEPENDENT COUNTER

A one state binary counter is diagrammed on Pg. 19. The node numbering is indicated at the output of each element and the flipflop and gates are included in two logical equations. The dual flipflop has the dual equations. The point a is the enabling signal and the point b is the complement of the completion signal. These were tied together to cause the counter to cycle indefinitely.



The equations for this circuit were typed as:

7	Number of Nodes
4'7 +2'N 3'7 +1'N	ordinary flipflop with gates
1'4'+7 4'N 2'3'+7 3'N	dual flipflop with gates
1 4 N	"and" element 5
2 3 N	"and" element 6
5 +6 N	"or" element 7
+ +	initial state
7 4	Node 7 to change 4 time
0 2	Punch when node 7 changes

The Output that followed was:		
	EXPLANATION	
	Flipflops	Node 7 change
+2 02 +0 20	1011	1 → 0
98 02 9+ 40	1001	0 → 1
52 02 50 10	0101	1 → 0
64 02 66 80	0110	0 → 1
SEQUENTIAL	Totally Sequential.	

The columns in the output are respectively: current state, excited nodes for current state, directly following state, and excited nodes for directly following state. Due to the print out selected the two states per output are the state where node number 7 is excited and the state that directly follows that state when node number 7 changes. The final line of the output indicates that the circuit is surprisingly enough, totally sequential.

XIII. REFERENCES

1. D. E. Muller, "Theory of Asynchronous Circuits," Internal Report No. 66, University of Illinois Digital Computer Laboratory (Dec. 6, 1955).

2. Staff of the Computer Laboratory, Ordvac Manual, University of Illinois Digital Computer Laboratory (Jan. 24, 1955).

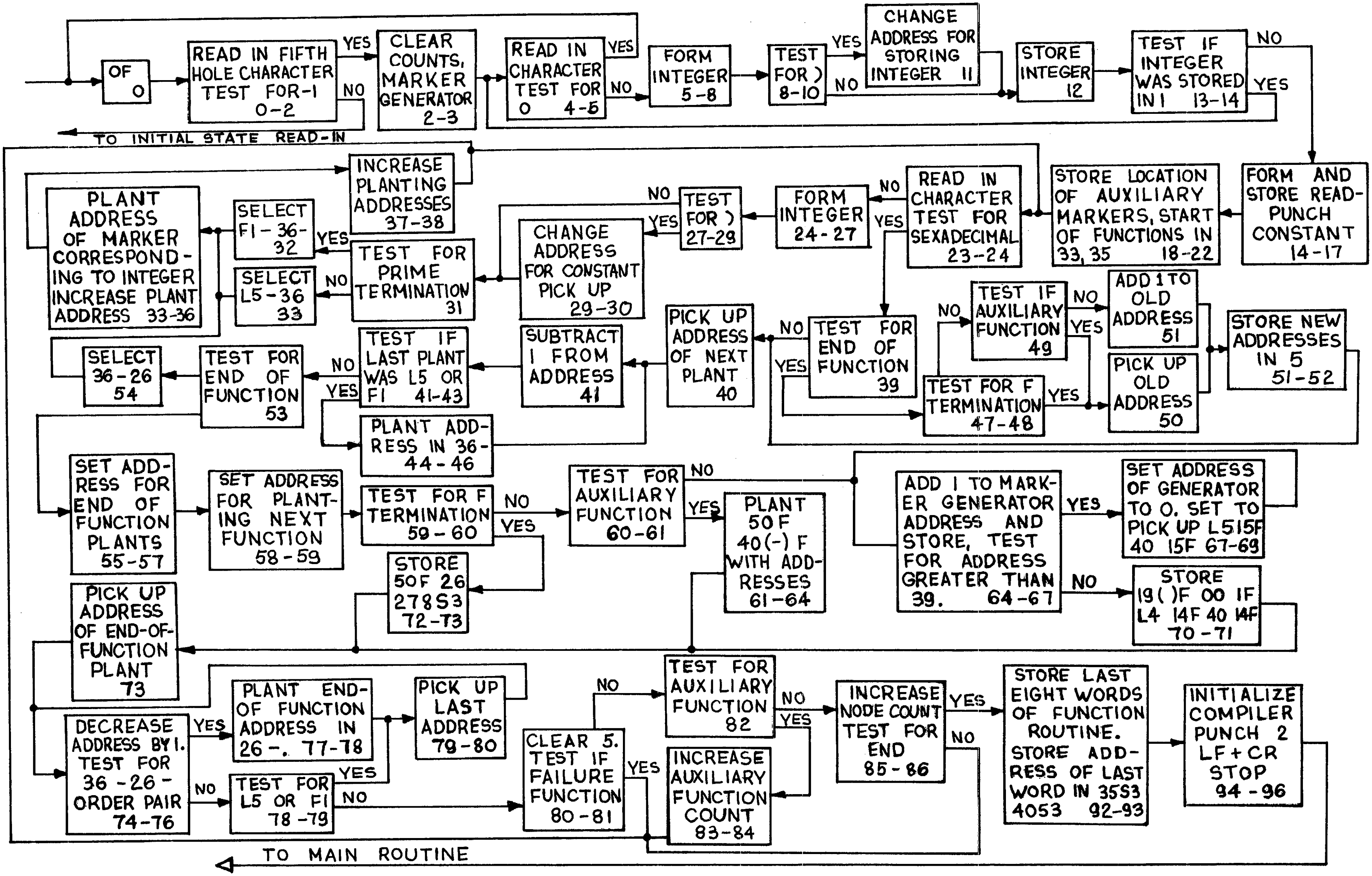
3. W. Poppelbaum, "A Fast Junction Transistor Flipflop with Stabilized Output Levels," University of Illinois Digital Computer Laboratory (Jan. 24, 1955)

4. J. M. Wier, "Some Direct Coupled Computer Circuits Utilizing NPN and PNP Transistors in Combination," Internal Report No. 65, University of Illinois Digital Computer Laboratory (Aug. 31, 1955).

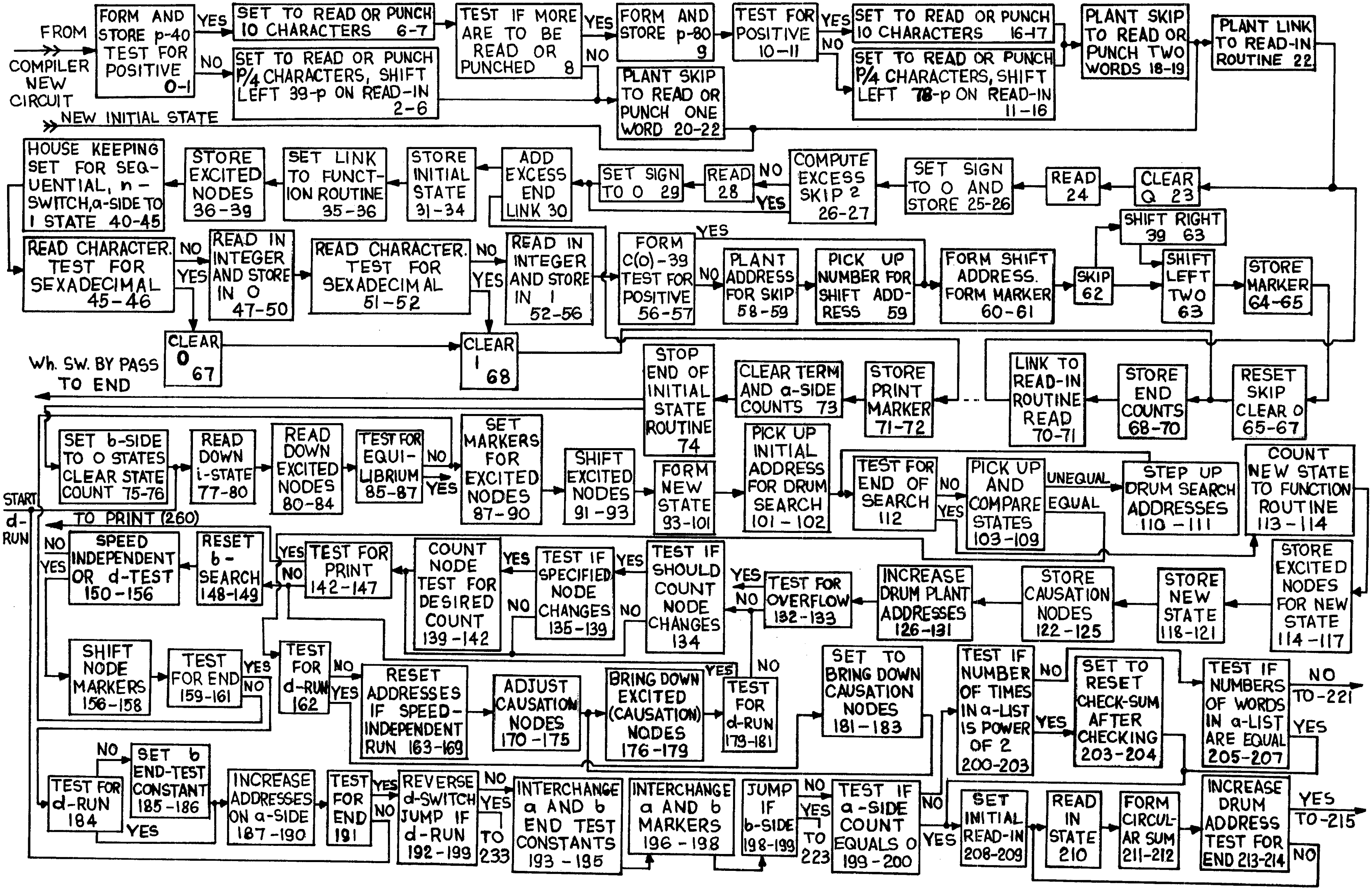
5. Staff of the Computer Laboratory, "Trance Circuits," University of Illinois Digital Computer Laboratory (May 11, 1956).

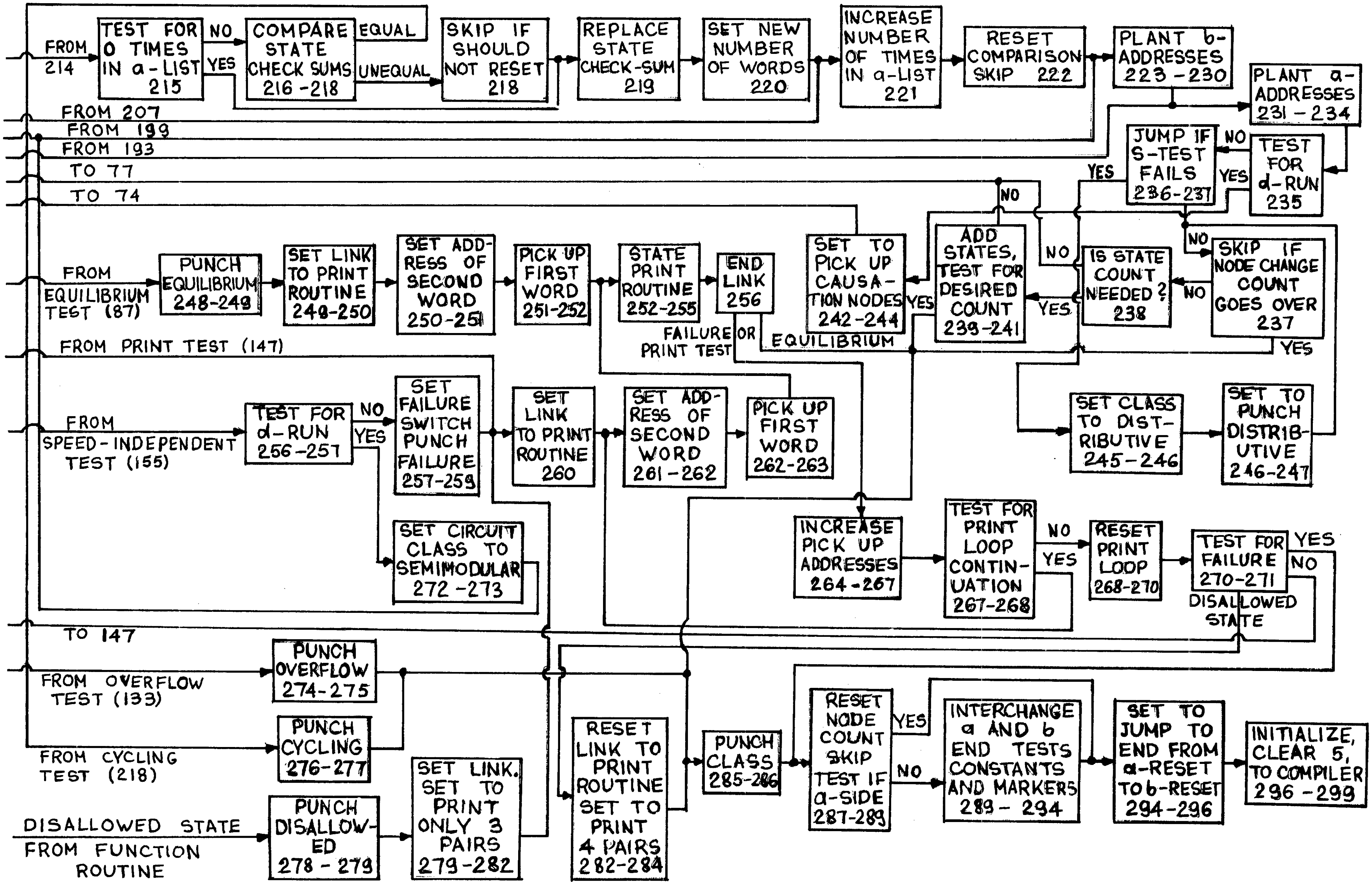
6. W. S. Bartky, "Complete Circuit Analyzer," Library Routine Q-3, University of Illinois Digital Computer Laboratory (June 14, 1956).

DATE July 22, 1957
PROGRAMMED BY James Shelly
APPROVED BY D E, Muller


OF 0
READ IN FIFTH HOLE CHARACTER TEST FOR-1 0-2
YES
NO
TO INITIAL STATE READ-IN
CLEAR COUNTS, MARKER GENERATOR 2-3
READ IN CHARACTER TEST FOR 0 4-5
YES
NO
FORM INTEGER 5-8
TEST FOR) 8-10
YES
NO
CHANGE ADDRESS FOR STORING INTEGER 11
STORE INTEGER 12
TEST IF INTEGER WAS STORED IN 1 13-14
NO
YES
FORM AND STORE READ-PUNCH CONSTANT 14-17
STORE LOCATION OF AUXILIARY MARKERS, START OF FUNCTIONS IN 33, 35 18-22
READ IN CHARACTER TEST FOR SEXADECIMAL 23-24
NO
YES
FORM INTEGER 24-27
TEST FOR) 27-29
NO
YES
CHANGE ADDRESS FOR CONSTANT PICK UP 29-30
INCREASE PLANTING ADDRESSES 37-38
TEST FOR PRIME TERMINATION 31
YES
NO
SELECT F1-36-32
SELECT L5-36 33
PLANT ADDRESS OF MARKER CORRESPONDING TO INTEGER INCREASE PLANT ADDRESS 33-36
TEST FOR END OF FUNCTION 39
NO
YES
TEST IF AUXILIARY FUNCTION 49
NO
YES
ADD 1 TO OLD ADDRESS 51
PICK UP OLD ADDRESS 50
TEST FOR F TERMINATION 47-48
YES
STORE NEW ADDRESSES IN 5 51-52
PICK UP ADDRESS OF NEXT PLANT 40
SUBTRACT 1 FROM ADDRESS 41
TEST IF LAST PLANT WAS L5 OR F1 41-43
NO
YES
PLANT ADDRESS IN 36-44-46
TEST FOR END OF FUNCTION 53
SELECT 36-26 54
SET ADDRESS FOR END OF FUNCTION PLANTS 55-57
SET ADDRESS FOR PLANTING NEXT FUNCTION 58-59
TEST FOR F TERMINATION 59-60
NO
YES
STORE 50F 26 278S3 72-73
TEST FOR AUXILIARY FUNCTION 60-61
NO
YES
PLANT 50F 40(-) F WITH ADDRESSES 61-64
ADD 1 TO MARKER GENERATOR ADDRESS AND STORE, TEST FOR ADDRESS GREATER THAN 39. 64-67
YES
NO
SET ADDRESS OF GENERATOR TO 0. SET TO PICK UP L515F 40 15F 67-69
STORE 19( )F 00 IF L4 14F 40 14F 70-71
PICK UP ADDRESS OF END-OF-FUNCTION PLANT 73
DECREASE ADDRESS BY 1. TEST FOR 36-26-ORDER PAIR 74-76
YES
NO
PLANT END-OF FUNCTION ADDRESS IN 26-. 77-78
PICK UP LAST ADDRESS 79-80
TEST FOR L5 OR F1 78-79
YES
NO
CLEAR 5. TEST IF FAILURE FUNCTION 80-81
NO
YES
TEST FOR AUXILIARY FUNCTION 82
NO
YES
INCREASE AUXILIARY FUNCTION COUNT 83-84
INCREASE NODE COUNT TEST FOR END 85-86
YES
NO
STORE LAST EIGHT WORDS OF FUNCTION ROUTINE. STORE ADDRESS OF LAST WORD IN 35S3 4053 92-93
INITIALIZE COMPILER PUNCH 2 LF+CR STOP 94-96
TO MAIN ROUTINE

Q5 COMPILER S5 49-144


FROM COMPILER NEW CIRCUIT
FORM AND STORE p-40 TEST FOR POSITIVE 0-1
YES
NO
SET TO READ OR PUNCH 10 CHARACTERS 6-7
SET TO READ OR PUNCH P/4 CHARACTERS, SHIFT LEFT 39-p ON READ-IN 2-6
TEST IF MORE ARE TO BE READ OR PUNCHED 8
YES
NO
FORM AND STORE p-80 9
PLANT SKIP TO READ OR PUNCH ONE WORD 20-22
TEST FOR POSITIVE 10-11
YES
NO
SET TO READ OR PUNCH 10 CHARACTERS 16-17
SET TO READ OR PUNCH P/4 CHARACTERS, SHIFT LEFT 78-p ON READ-IN 11-16
PLANT SKIP TO READ OR PUNCH TWO WORDS 18-19
PLANT LINK TO READ-IN ROUTINE 22
NEW INITIAL STATE
HOUSE KEEPING SET FOR SEQUENTIAL, n-SWITCH, a-SIDE TO 1 STATE 40-45
STORE EXCITED NODES 36-39
SET LINK TO FUNCTION ROUTINE 35-36
STORE INITIAL STATE 31-34
ADD EXCESS END LINK 30
SET SIGN TO 0 29
READ 28
NO
YES
COMPUTE EXCESS SKIP 2 26-27
SET SIGN TO 0 AND STORE 25-26
READ 24
CLEAR Q 23
READ CHARACTER. TEST FOR SEXADECIMAL 45-46
NO
YES
READ IN INTEGER AND STORE IN 0 47-50
READ CHARACTER. TEST FOR SEXADECIMAL 51-52
NO
YES
READ IN INTEGER AND STORE IN 1 52-56
FORM C(0)-39 TEST FOR POSITIVE 56-57
YES
NO
PLANT ADDRESS FOR SKIP 58-59
PICK UP NUMBER FOR SHIFT ADDRESS 59
FORM SHIFT ADDRESS. FORM MARKER 60-61
SKIP 62
SHIFT RIGHT 39 63
SHIFT LEFT TWO 63
STORE MARKER 64-65
CLEAR 0 67
CLEAR 1 68
Wh. SW. BY PASS TO END
STOP END OF INITIAL STATE ROUTINE 74
CLEAR TERM AND a-SIDE COUNTS 73
STORE PRINT MARKER 71-72
LINK TO READ-IN ROUTINE READ 70-71
STORE END COUNTS 68-70
RESET SKIP CLEAR 0 65-67
START d-RUN
SET b-SIDE TO 0 STATES CLEAR STATE COUNT 75-76
READ DOWN i-STATE 77-80
READ DOWN EXCITED NODES 80-84
TEST FOR EQUILIBRIUM 85-87
NO
YES
SET MARKERS FOR EXCITED NODES 87-90
SHIFT EXCITED NODES 91-93
FORM NEW STATE 93-101
PICK UP INITIAL ADDRESS FOR DRUM SEARCH 101-102
TEST FOR END OF SEARCH 112
NO
YES
PICK UP AND COMPARE STATES 103-109
UNEQUAL
EQUAL
STEP UP DRUM SEARCH ADDRESSES 110-111
COUNT NEW STATE TO FUNCTION ROUTINE 113-114
TO PRINT (260)
NO
YES
SPEED INDEPENDENT OR d-TEST 150-156
RESET b-SEARCH 148-149
YES
NO
TEST FOR PRINT 142-147
COUNT NODE TEST FOR DESIRED COUNT 139-142
YES
NO
TEST IF SPECIFIED NODE CHANGES 135-139
YES
NO
TEST IF SHOULD COUNT NODE CHANGES 134
YES
NO
TEST FOR OVERFLOW 132-133
INCREASE DRUM PLANT ADDRESSES 126-131
STORE CAUSATION NODES 122-125
STORE NEW STATE 118-121
STORE EXCITED NODES FOR NEW STATE 114-117
SHIFT NODE MARKERS 156-158
TEST FOR END 159-161
YES
NO
TEST FOR d-RUN 162
NO
YES
RESET ADDRESSES IF SPEED-INDEPENDENT RUN 163-169
ADJUST CAUSATION NODES 170-175
BRING DOWN EXCITED (CAUSATION) NODES 176-179
YES
NO
TEST FOR d-RUN 179-181
SET TO BRING DOWN CAUSATION NODES 181-183
TEST IF NUMBER OF TIMES IN a-LIST IS POWER OF 2 200-203
NO
YES
SET TO RESET CHECK-SUM AFTER CHECKING 203-204
TEST IF NUMBERS OF WORDS IN a-LIST ARE EQUAL 205-207
NO
TO-221
YES
NO
TEST FOR d-RUN 184
YES
SET b END-TEST CONSTANT 185-186
INCREASE ADDRESSES ON a-SIDE 187-190
TEST FOR END 191
YES
NO
REVERSE d-SWITCH JUMP IF d-RUN 192-199
NO
YES
TO 233
INTERCHANGE a AND b END TEST CONSTANTS 193-195
INTERCHANGE a AND b MARKERS 196-198
JUMP IF b-SIDE 198-199
NO
YES
TO 223
TEST IF a-SIDE COUNT EQUALS 0 199-200
NO
YES
SET INITIAL READ-IN 208-209
READ IN STATE 210
FORM CIRCULAR SUM 211-212
INCREASE DRUM ADDRESS TEST FOR END 213-214
YES
TO-215
NO


FROM 214
TEST FOR 0 TIMES IN a-LIST 215
NO
YES
COMPARE STATE CHECK SUMS 216-218
EQUAL
UNEQUAL
SKIP IF SHOULD NOT RESET 218
REPLACE STATE CHECK-SUM 219
SET NEW NUMBER OF WORDS 220
INCREASE NUMBER OF TIMES IN a-LIST 221
RESET COMPARISON SKIP 222
PLANT b-ADDRESSES 223-230
PLANT a-ADDRESSES 231-234
FROM 207
FROM 199
FROM 193
TO 77
TO 74
JUMP IF S-TEST FAILS 236-237
NO
YES
TEST FOR d-RUN 235
YES
NO
NO
ADD STATES, TEST FOR DESIRED COUNT 239-241
NO
IS STATE COUNT NEEDED? 238
NO
SKIP IF NODE CHANGE COUNT GOES OVER 237
YES
YES
SET TO PICK UP CAUSATION NODES 242-244
YES
FROM EQUILIBRIUM TEST (87)
PUNCH EQUILIBRIUM 248-249
SET LINK TO PRINT ROUTINE 249-250
SET ADDRESS OF SECOND WORD 250-251
PICK UP FIRST WORD 251-252
STATE PRINT ROUTINE 252-255
END LINK 256
FAILURE OR PRINT TEST
EQUILIBRIUM
FROM PRINT TEST (147)
SET CLASS TO DISTRIBUTIVE 245-246
SET TO PUNCH DISTRIBUTIVE 246-247
FROM SPEED-INDEPENDENT TEST (155)
TEST FOR d-RUN 256-257
NO
YES
SET FAILURE SWITCH PUNCH FAILURE 257-259
SET LINK TO PRINT ROUTINE 260
SET ADDRESS OF SECOND WORD 261-262
PICK UP FIRST WORD 262-263
SET CIRCUIT CLASS TO SEMIMODULAR 272-273
INCREASE PICK UP ADDRESSES 264-267
TEST FOR PRINT LOOP CONTINUATION 267-268
NO
YES
RESET PRINT LOOP 268-270
TEST FOR FAILURE 270-271
YES
NO
DISALLOWED STATE
TO 147
FROM OVERFLOW TEST (133)
PUNCH OVERFLOW 274-275
FROM CYCLING TEST (218)
PUNCH CYCLING 276-277
DISALLOWED STATE
FROM FUNCTION ROUTINE
PUNCH DISALLOWED 278-279
SET LINK. SET TO PRINT ONLY 3 PAIRS 279-282
RESET LINK TO PRINT ROUTINE SET TO PRINT 4 PAIRS 282-284
PUNCH CLASS 285-286
RESET NODE COUNT SKIP TEST IF a-SIDE 287-289
YES
NO
INTERCHANGE a AND b END TESTS CONSTANTS AND MARKERS 289-294
SET TO JUMP TO END FROM a-RESET TO b-RESET 294-296
INITIALIZE, CLEAR 5, TO COMPILER 296-299

LOCATION	ORDER	NOTES PAGE 1 Q 5
	S3	
0	L5 S7	Form and store
	L0 1S7	p-40
1	40 F	
	32 6L	Test
2	L5 S7	Set to read or punch p/4 characters
	L0 2S7	Shift left 39-p on read-in.
3	40 24L	
	46 253L	
4	L1 F	
	10 20F	
5	L0 1S7	
	42 24L	
6	22 20L	
	L5 1S7	
7	40 24L	Set to read or punch 10 characters
	46 253L	
8	L1 F	See if more are to be read or punched
	32 20L	
9	L5 F	Reset S7
	46 S7	
10	L5 S7	Test
	L0 1S7	
11	32 16L	
	L5 S7	
12	L0 2S7	Set to read or punch p/4-10 characters.
	40 28L	
13	46 255L	Shift left 78-p on read-in.
	L1 S7	
14	L4 1S7	
	10 20F	
15	F0 1S7	
	42 28L	

LOCATION	ORDER		NOTES
16	26 18L		
	F5 1S7		Set to read or punch 10 characters
17	40 28L		
	46 255L		
18	F5 18L		
	42 27L		Plant skip
19	F5 19L		
	42 253L		
20	22 22L		
	L5 26L		Plant skip
21	42 27L		
	F5 17L		
22	42 253L		
	F5 26L		Link
23	42 30L		
	50 3S7		Clear Q
24	81 F		
	10 1F		Read
25	36 26L		Set sign to zero
	L4 4S7		
26	40 F		Store
	S5 30L		Compute excess
27	40 1F		
	27 F		Skip
28	81 F		
	00 F		Read
29	36 30L		Set sign to zero
	L4 5S7		
30	L4 1F		Add excess
	22 ( )F		
31	22 296L	*	
	40 13F		

LOCATION	ORDER		NOTES
32	86 11F		Store initial state
	00 1S4	1+a	
33	L5 F		
	40 12F		
34	86 11F		
	00 S4	a	
35	L5 55L	Function	Set link
	42 F	Routine	
36	26 S6		to function routine
	L5 14F		
37	86 11F		
	00 1706S4	$a+\frac{m}{6}$	
38	41 17L		Clear 17
	L5 15F		Store excited nodes
39	86 11F		
	00 1707S4	$1+a+\frac{m}{6}$	
40	L5 92L	Function	Reset normal exit to function routine
	42 F	Routine	
41	L5 95L		Set to print sequential
	46 285L		
42	F1 1S7		Set sequential
	42 6S7		Test
43	L5 4S7		Set D-Switch
	40 7S7		
44	F5 8S7		Set a-side to 1 state
	F4 4S7		
45	40 10S7		
	81 4F		Test for sexadecimal
46	L0 12S7	10	
	32 67L		
47	L4 12S7	10	
	50 3S7	10	Read in and store in 0

LOCATION	ORDER		NOTES PAGE 4 Q 5
48	74 12S7	10	
	00 4F		
49	91 4F		
	36 48L		
50	S5 62L		
	40 F		
51	81 4F		Test for sexadecimal
	L0 12S7	10	
52	36 68L		
	L4 12S7	10	
53	50 3S7		Read in and store in 1
	74 12S7	10	
54	00 4F		
	91 4F		
55	32 53L		
	S5 36L		
56	40 1F		
	L5 F		Form C(0) - 39
57	L0 13S7		
	36 60L		Test
58	L5 27L		Plant skip
	L4 50L		
59	46 62L		
	L5 F		
60	F4 3S7	0	Form shift address
	42 61L		
61	50 3S7		Clear a
	19 ( )F		Form marker skip
62	22 62L		
	36 63L	w	
63	10 39F		Shift 39
	00 2F		Shift left 2

LOCATION	ORDER		NOTES	PAGE 5 Q 5
64	40 14S7		Store marker	
	S5 71L			
65	40 15S7			
	L5 50L		Reset skip	
66	46-62L			
	41 F		Clear	
67	22 68L		Unwanted end counts	
	41 F			
68	41 1F			
	L1 F			
69	40 16S7		Store end counts	
	L1 1F			
70	40 17S7			
	L5 64L		Link to read in	
71	26 23L			
	40 19S7		Store print marker	
72	L5 F			
	40 18S7			
73	41 18F		Clear term count	
	41 19F		Clear a-side count	
74	24 75L		Stop	
	26 285L		End	
75	L5 9S7		Set b-side	
	L4 4S7		to 0 states	
76	40 11S7			
	41 16F		Clear state count	
77	85 11F			
	00 S4	a		
78	40 8F		Read in i-state	
	36 79L	w		
79	85 11F			
	00 1S4	1+a		

LOCATION	ORDER		NOTES
80	40 9F		
	36 81L	w	
81	85 11F		
	00 1706S4	$a+\frac{m}{6}$	Read in excited nodes
82	40 10F		
	40 6F		
83	85 11F		
	00 1707S4	$1+a+\frac{m}{6}$	
84	40 11F		
	40 7F		
85	L3 10F		
	32 86L		Test for equilibrium
86	22 87L		
	L3 11F		
87	36 248L		
	49 2F		
88	41 3F		Set markers for excited nodes
	F1 2F		
89	40 4F		
	F1 3F		
90	40 5F		
	L5 6F		
91	50 7F		Shift excited nodes
	00 1F		
92	40 6F		
	S5 114L		
93	40 7F		
	L5 6F		
94	36 156L		
	50 8F		
95	S5 8S9		
	J0 2F		

LOCATION	ORDER		NOTES PAGE 7 Q 5
96	SO 2F		
	SO 2F		Form new state
97	L4 2F		
	40 12F		
98	50 9F		
	S5 F		
99	JO 3F		
	SO F		
100	SO F		
	L4 3F		
101	40 13F		
	L5 107L		To end test
102	26 112L		
	00 285L	*	
103	85 11F		
	00 5118S4	b	
104	LO 12F		
	40 F		
105	L3 F		
	36 107L		
106	26 110L		
	00 F	*	Test in b list for duplicate
107	85 11F		
	00 5119S4	1+b	
108	LO 13F		
	40 F		
109	L3 F		
	36 162L		
110	F5 107L		
	40 103L		
111	F5 103L		
	40 107L		

LOCATION	ORDER		NOTES PAGE 8 Q 5
112	L0 11S7		
	36 103L		
113	F5 16F		Count new state
	40 16F		
114	26 S6		to fcn. routine
	L5 14F		
115	86 11F		
	00 6824S4	b+$\frac{m}{6}$	Store excited nodes for new state
116	L5 15F		
	36 117L	w	
117	86 11F		
	00 6825S4	1+b+$\frac{m}{6}$	
118	L5 12F		
	36 119L		
119	86 11F		
	00 5118S4	b	
120	L5 13F		Store new state
	36 121L	w	
121	86 11F		
	00 5119S4	b+1	
122	L5 2F		
	36 123L	w	
123	86 11F		Store causation nodes
	00 8530S4	b+$\frac{m}{3}$	
124	L5 3F		
	36 125L	w	
125	86 11F		
	00 8531S4	1+b+$\frac{m}{3}$	
126	F5 117L		
	40 115L		
127	F5 115L		
	40 117L		Increase drum loop addresses

LOCATION	ORDER	NOTES PAGE 9 Q 5
128	F5 125L	
	40 123L	
129	F5 123L	
	40 125L	
130	F5 121L	
	40 119L	
131	F5 119L	
	40 121L	
132	L0 9S7	
	L0 20S7	Overflow test
133	L0 22S7	
	36 274L	
134	L3 17S7	Test for node change count
	32 142L	
135	50 14S7	
	J0 2F	
136	S1 F	
	32 137L	Test to see if specified node changes
137	22 139L	
	50 15S7	
138	J0 3F	
	S1 F	
139	32 142L	
	F5 17S7	
140	40 17S7	Count node
	32 141L	Test for desired count
141	22 142L	
	L5 102L	
142	42 237L	Plant skip
	50 18S7	
143	J0 2F	
	S1 F	Print Test

LOCATION	ORDER		NOTES
144	36 145L		
	26 147L		
145	50 19S7		
	J0 3F		
146	S1 F		
	36 148L		
147	26 260L		To print
	93 131F		
148	L5 9S7		
	40 103L		Reset
149	F4 3S7		b-search
	40 107L		
150	50 10F		
	J0 4F		
151	S5 10S9		
	J0 14F		Speed-independent or D- test
152	K0 12S9		
	36 256L		
153	50 11F		
	J0 5F		
154	S5 282L		
	J0 15F		
155	K0 147L		
	36 256L		
156	L5 2F		
	50 3F		Shift node markers
157	10 1F		
	40 2F		
158	S5 F		
	40 3F		
159	L3 6F		
	32 160L		

LOCATION	ORDER		NOTES
160	22 88L		Test for end
	L3 7F		
161	36 184L		
	22 88L		
162	L1 7S7		Test for D-run
	36 181L		
163	L5 103L		
	L4 20S7		
164	40 176L		
	L4 20S7		Reset addresses if speed-independent run.
165	40 170L		
	L4 22S7		
166	40 172L		
	F5 170L		
167	40 173L		
	F5 172L		
168	40 175L		
	F5 176L		
169	40 178L		
	36 170L	w	
170	85 11F		
	00 8530S4	$b+\frac{m}{3}$	
171	L4 2F		
	26 172L	w	Adjust causation nodes
172	86 11F		
	00 8530S4	$b+\frac{m}{3}$	
173	85 11F		
	00 8531S4	$1+b+\frac{m}{3}$	
174	L4 3F		
	50 F	w	
175	86 11F		
	00 8531S4	$1+b+\frac{m}{3}$	

LOCATION	ORDER		NOTES PAGE 12 Q 5
176	85 11F		
	00 6824S4	$b+\frac{m}{6}$	Bring out excited (causation nodes)
177	40 14F		
	50 F	w	
178	85 11F		
	00 6825S4	$1+b+\frac{m}{6}$	
179	40 15F		
	L1 7S7		Test for D-Run
180	36 148L		To reset b-search
	26 134L		to node count test
181	L5 103L		Set to bring out causation nodes
	L4 21S7		
182	40 176L		
	F5 176L		
183	40 178L		
	26 176L		To store causation nodes
184	L1 7S7		Test for D-Run
	36 187L		
185	LJ 16F		
	80 1F		Set b-end test constant
186	L4 103L		
	40 11S7		
187	F5 83L		
	40 81L		
188	F5 81L		
	40 83L		Increase addresses on a side
189	F5 79L		
	40 77L		
190	F5 77L		
	40 79L		
191	L0 10S7		Test for end
	36 77L		

LOCATION	ORDER	NOTES
192	L1 7S7	Jump if completed D-run
	40 7S7	
193	36 231L	
	50 10S7	
194	L5 11S7	Interchange a and b end test constants
	40 10S7	
195	S5 264L	
	40 11S7	
196	50 8S7	
	L5 9S7	
197	40 8S7	Interchange a and b markers.
	S5 F	
198	40 9S7	
	L0 8S7	
199	36 223L	Jump if b-side
	L3 19F	check for
200	36 208L	a-side count=0
	L5 19F	
201	80 1F	See if number of times in a-list is
	36 201L	a power of 2.
202	40 F	
	L7 F	
203	36 205L	
	L5 215L	Set to reset after checking
204	42 218L	
	26 208L	
205	L5 79L	
	L0 18F	See if numbers of terms in a-list
206	40 F	are equal
	L3 F	
207	36 208L	
	26 221L	Jump to end of cycle

LOCATION	ORDER		NOTES PAGE 14 Q 5
208	41 F		Clear O, Q, and set initial read-in
	50 3S7		
209	L5 9S7		
	40 210L		
210	85 11F		Read in state
	00 S4		
211	L4 F		Add previous sum shift right 1
	10 1F		
212	S4 F		Form circular sum
	40 F		Store
213	F5 210L		Increment drum address
	40 210L		
214	L0 11S7		Test for end
	36 210L		To read in state
215	L3 19F		Test for 0 times
	36 219L		Do not compare
216	L5 F		
	L0 23S7		
217	40 1F		Compare state check sums
	L3 1F		
218	36 276L		
	26 221L		skip
219	L5 F		Replace state
	40 23S7		Check sum
220	L5 79L		Set new number of terms
	40 18F		
221	F5 19F		Increase no. of times in a-list
	40 19F		
222	L5 207L		
	42 218L		Reset skip
223	L5 9S7		
	40 103L		

LOCATION	ORDER	NOTES
224	L4 22S7	
	40 119L	
225	L4 20S7	
	40 115L	Plant b addresses
226	L4 20S7	
	40 123L	
227	F5 103L	
	40 107L	
228	F5 119L	
	40 121L	
229	F5 115L	
	40 117L	
230	F5 123L	
	40 125L	
231	L5 8S7	
	40 77L	
232	L4 20S7	Plant a addresses
	40 81L	
233	F5 77L	
	40 79L	
234	F5 81L	
	40 83L	
235	L1 7S7	D-Test
	36 242L	
236	L5 16F	
	L4 6S7	Jump if S-test fails
237	36 245L	
	26 238L	Skip from node change
238	L3 16S7	Is state count needed
	36 74L	
239	L5 16F	Add states
	L4 16S7	

LOCATION	ORDER		NOTES
240	40 16S7		
	36 285L		Test
241	26 74L		To start
	00 F	*	
242	L5 8S7		Set to pick up causation nodes
	L4 21S7		
243	40 81L		
	F5 81L		
244	40 83L		
	26 77L		To D-run
245	43 6S7		Block S-test
	F1 7S7		Set D-switch
246	40 7S7		
	L5 151L		Set to punch distributive
247	46 285L		
	22 237L		To node change skip
248	J0 S9		Punch equilibrium
	50 248L		
249	26 SK		
	L5 247L		Link
250	46 256L		
	L5 252L		Set pickup address
251	46 254L		
	L5 8F		Get First word.
252	50 9F		
	00 1F		
253	82 ( )F		State print routine
	26 ( )F		
254	L5 ( )F		
	00 2F		
255	82 F		
	92 131F		

LOCATION	ORDER		NOTES	PAGE 17	Q 5
256	26 ( )F				
	L1 7S7		Test for D-run		
257	36 272L				
	40 17F		Set failure switch		
258	J0 2S9		Punch failure		
	50 258L				
259	26 SK				
	36 260L	w			
260	L5 195L		Link		
	46 256L				
261	93 967F		Spaces		
	L5 263L		Set pickup address		
262	46 254L				
	L5 8F		Get first word		
263	50 9F				
	22 252L		To state print		
264	10 1F	w			
	L5 262L				
265	L4 96L		Step up addresses		
	42 262L				
266	L5 263L				
	L4 96L				
267	46 263L				
	L0 24S7		Test for print loop continuation		
268	36 261L				
	L5 251L				
269	42 262L		Reset loop		
	L5 252L				
270	46 263L				
	L5 17F		Normal jump		
271	32 147L				
	26 287L		To end reset		

LOCATION	ORDER		NOTES PAGE 18 Q 5
272	43 7S7		Block D-test
	L5 152L		Set to punch semimodular
273	46 285L		
	26 223L		to reset
274	J0 4S9		Punch overflow
	50 274L		
275	26 SK		
	26 285L		To end
276	J0 14S9		Punch cycling
	50 276L		
277	26 SK		
	26 285L		To end
278	J0 6S9		
	50 278L		Punch disallowed
279	26 SK		
	L5 24S7		
280	L0 171L		
	46 24S7		Print information from disallowed
281	L5 154L		state
	46 271L		
282	26 260L		
	L5 24S7		
283	L4 171L		
	46 24S7		
284	L5 155L		
	46 271L		
285	J0 F		
	50 285L		Punch Type
286	26 SK		
	92 135F		
287	F5 287L		Reset node count skip
	42 237L		

LOCATION	ORDER	NOTES
288	L5 9S7	Test if a-side
	L0 8S7	
289	32 294L	
	L5 9S7	
290	50 8S7	
	40 8S7	Interchange a and b markers
291	S5 F	and end tests
	40 9S7	
292	L5 11S7	
	50 10S7	
293	40 10S7	
	S5 F	
294	40 11S7	
	50 235L	Set end skip in reset
295	L5 31L	
	40 235L	
296	26 223L	To reset
	S5 F	reset end of reset
297	40 235L	
	41 5F	Clear 5
298	93 135F	To compiler
	20 S5	
<u>S</u> <u>6</u>		
0	F5 6L	Set initial storage location for mark
	42 5L	
1	41 14F	Clear 14, 15
	41 15F	
2	L5 12F	
	40 F	Load new state
3	L5 13F	
	40 1F	
4	L5 F	
	50 1F	Form and store marker
5	00 1F	
	40 F	

LOCATION	ORDER	NOTES
6	40 F	
	S5 9L	
7	40 1F	
	F5 5L	Reset store address
8	42 5L	
	F0 18S8	Test for end
9	36 4L	
	26 F	Exit
<u>S</u> <u>8</u>		
0	80 F	Test no. for 5th hole delay
	00 1F	
1	00 F	
	00 6F	6×2^{-39}
2	F1 F	Prime
	36 F	Termination
3	L5 F	Unprimed
	36 F	
4	35 F	
	26 F	+
5	50 F	Auxiliary
	40 F	Function
6	19 F	Marker
	00 1F	generator
7	99 40F	End test for marker count
	00 1F	
8	L4 14F	
	40 14F	Planting instructions
9	L4 15F	
	40 15F	
10	50 12F	
	S5 F	

LOCATION	ORDER		NOTES PAGE 21 Q 5
11	JO 14F		
	SO F		
12	L4 14F		
	SO F		Last
13	50 F	w	8
	40 14F		words
14	50 13F		
	S5 F		
15	JO 15F		
	SO F		
16	L4 15F		
	SO F		
17	40 15F		
	22 F		
18	80 1F		Stor. Loc. for Aux. markers
	40 F		
19	50 F		
	26 278S3		Failure term.
S 7			
0	81 F		Temp. storage for read-in, print
	10 1F		
1	81 40F		Read in 10 char.
	10 1F		
2	00 F		To obtain 81 00 order pair
	10 F		
3	00 F		0
	00 F		
4	80 F		-1
	00 F		
5	40 F		1/2
	00 F		

LOCATION	ORDER		NOTES
6	LL 4095		
	LL 4094		Sequential test
7	80 F		
	00 F		D switch
8	85 11F		
	00 S4		a reset
9	85 11F		
	00 5118S4		b reset
10	05 11F		
	00 S4		a end
11	05 11F		
	00 5118S4		b end
12	00 F		
	00 10F		10 x 2^{-39}
13	00 F		
	00 39F		39 x 2^{-39}
14	00 F		Storage for node change count constant
	00 F		
15	00 F		
	00 F		
16	00 F		
	00 F		New state count
17	00 F		
	00 F		Node change count
18	00 F		Print control constant
	00 F		
19	00 F		
	00 F		
20	00 F		
	00 1706F		m/6
21	00 F		
	00 3412F		m/3

LOCATION	ORDER		NOTES
22	01 F		
	00 F		Interchange constant
23	00 F		
	00 F		Cycling Test state
24	80 15F		
	J2 245S3		End test for print
S 5			
0	OF F		
	91 4F		Test for fifth-hole delay
1	40 F		
	L7 F		
2	30 22S3		
	41 1F		Clear aux. count
3	41 10S6		Inhibit 26-plant
	47 6S8		clear marker generator
4	81 4F		Read integer
	10 S8	80F 001F	
5	36 L		
	L4 S8	80F 001F	
6	50 3S7	0	
	74 12S7	10	
7	00 4F		
	91 4F		
8	32 6L		Test for)
	LO 1S8	6	
9	40 4F		
	43 12L		
10	L7 4F		
	36 12L		
11	F5 12L		Change address for storing integer
	42 12L		

LOCATION	ORDER		NOTES PAGE 24 Q 5
12	S1 10S8		Store integer
	40 F		
13	F5 12L		Should we read another integer
	00 39F		
14	36 4L		
	L5 1S8	6	
15	10 1F		
	L0 F		
16	10 2F		Form and Store read-punch constant
	50 3S7		
17	00 22F		
	46 S7		
18	L5 6S6		
	42 3F		Store location of auxiliary markers
19	L0 F		
	42 3F		
20	L0 1F		Store start of functions
	42 18S8		
21	F4 3S7	1	
	42 9S6		Set location to start fcn. routine
22	42 33L		
	42 35L		
23	81 4F		
	L0 12S7	10	Test for sexadecimal
24	36 39L		
	L4 12S7	10	
25	50 3S7	0	
	74 12S7	10	Read in integer
26	00 4F		
	91 4F		
27	32 25L		
	F0 1S8	6	

LOCATION	ORDER		NOTES PAGE 25 Q 5
28	00 1F		
	10 1F		Test for)
29	36 31L		
	F5 34L		Change address for constant
30	42 34L		pickup
	26 26L		
31	L0 1S8	6	Test for prime
	36 33L		
32	L5 2S8		Select F1-36
	22 33L		
33	L5 3S8		Select L5-36
	40 ( )F		Store instruction
34	S5 F		
	L4 2F		Plant address of marker
35	00 20F		
	46 ( )F		
36	L5 18L		Reset marker address
	42 34L		
37	F5 33L		
	42 33L		Increase plant addresses
38	42 35L		
	26 23L		To next term
39	F0 3S7	-1	Test for end of function
	36 47L		
40	50 33L		
	L5 33L		
41	F0 3S7	-1	
	42 42L		
42	42 45L		
	L5 F		
43	00 7F		
	36 53L		Set 36 transfers

LOCATION	ORDER	NOTES	PAGE 26 Q 5
44	10 7F		
	36 45L	w	
45	K5 8S8		
	42 ( )F		
46	L5 42L		
	26 41L		
47	FO 3S7	-1	
	FO 3S7	-1	Test for F termination
48	40 6F		
	36 50L		
49	L3 1F		
	36 51L		
50	L5 33L		
	22 51L		Set address for next function
51	F5 33L		
	40 5F		
52	50 5F		
	22 40L		
53	F1 5F		Jump if at end of function
	36 55L		
54	L5 4S8		Select 36-26
	22 33L		
55	L5 33L		
	42 63L		
56	42 65L		Set address for end plant
	42 7F		
57	00 20F		
	46 73L		
58	L5 5F		Set address for next function
	42 33L		
59	42 35L		
	L5 6F		Test for f

LOCATION	ORDER		NOTES
60	32 72L		termination
	L3 1F		Test for aux. function
61	32 64L		
	L5 9S6		
62	L4 1F		
	42 5S8		Plant 50 F 40 F
63	L5 5S8		
	40 ( )F		
64	22 73L		To set transfers
	19 18F		
65	L4 6S8		
	40 F		
66	46 6S8		
	L0 7S8		Set address of
67	36 70L		09 1F 10 ( )F
	47 6S8		
68	L5 71L		
	L4 28L		
69	46 71L		
	22 64L		
70	F5 65L		Store
	42 71L		
71	L5 8S8		
	40 ( )F		
72	22 73L		To set transfers
	L5 19S8		Plant failure
73	40 ( )F		end 50 F 26 F
	L5 7F		
74	F0 3S7	-1	
	42 75L		
75	42 77L		
	L5 ( )F		
76	00 6F		
	32 78L		Set 26 transfers

LOCATION	ORDER		NOTES PAGE 28 Q 5
77	L5 7F		
	42 ( )F		
78	22 79L		
	00 1F		
79	32 80L		
	L5 75L		
80	26 74L		
	41 5F		Clear 5
81	L5 6F		Test if failure function
	36 37L		
82	L3 1F		Test if auxiliary function
	36 85L		
83	F5 1F		Increase aux. fcn. count
	40 1F		
84	26 37L		Return
	50 F	*	
85	F5 F		Increase node count
	40 F		
86	F1 F		Test for end
	36 37L		
87	F5 71L		
	42 88L		
88	L5 10S8		
	40 ( )F		
89	00 3F		Store last eight words
	32 92L		
90	L5 88L		
	L4 28L		
91	46 88L		
	42 88L		
92	26 88L		
	L5 88L		

LOCATION	ORDER	NOTES
93	42 35S3	Plant address of last word
	42 40S3	
94	L5 12L	Reset "Last 8" plant
	46 88L	
95	L5 45L	Set to plant L4 14F
	46 71L	40 14F originally
96	93 139F	
	24 S3	To main routine

S - parameters

00 3K	
00 F 00 166F	Main Routine
00 F 00 2560F	First usable location on drum
00 F 00 49F	Compiler
00 F 00 490F	Function routine
00 F 00 465F	Storage for main routine
00 F 00 146F	Storage for compiler
00 F 00 20F	Compacted words to be printed by P7
00 F 00 36F	P-7 Print routine

On Tape

S - parameters

P - 7

Interlude to compact and store words to be printed

Words to be printed

Compiler (overwrites interlude locations)

Storage for compiler

Main Routine

Storage for main routine

Function routine (first ten words)

Sum check from SADOI