DECUS

PROGRAM LIBRARY

DECUS NO.	8-167
TITLE	CIRCUITS
AUTHOR	D. Whiteley
COMPANY	International Computers Limited Staffordshire, England
DATE	October 4, 1968
SOURCE LANGUAGE	

ATTENTION

This is a USER program. Other than requiring that it conform to submittal and review standards, no quality control has been imposed upon this program by DECUS.

The DECUS Program Library is a clearing house only; it does not generate or test programs. No warranty, express or implied, is made by the contributor, Digital Equipment Computer Users Society or Digital Equipment Corporation as to the accuracy or functioning of the program or related material, and no responsibility is assumed by these parties in connection therewith.

CIRCUITS

DECUS Program Library Write-up DECUS No. 8-167

CONTENTS

	PAGE
Introduction	1
Operation of Circuits	2
Circuits Program	4
The System	7
Discussion	15
Conclusion	15
Figure 1	39
List 1	41
List 2	44
List 3	45

INTRODUCTION

This report describes in full the operation and working of the PDP-8 program CIRCUITS, which enables electronic circuits to be drawn quickly and simply using the 338 Display System.

The report also explains how it will be possible to use the 338 Display as a workbench by employing an on-line SYSTEM 4-50 to analyse the circuit and feed back the results for display.

A-C Analysis Program is a KDF-9 Egtran program which determines the AC steady state response of an electrical network consisting of linear elements. Transistor can be included by using the Hybrid π equivalent circuit.

The data for A-C Analysis Program (ACCASS) is punched on cards and requires the number, value and nodal position of each component in the circuit. With this data prepared ACCASS will calculate the voltage vector across any pair of nodes over a range of frequencies defined by the user.

The limitations of ACCASS are not of size or accuracy but of accessability with a reasonable data format. The use of the 338 Display enables circuits to be fed in directly and modified, and reams of data can be displayed in a simple graphical form.

The program CIRCUITS is used soley to draw electronic circuits on the 338 Display and takes no part in any analysis other than to provide a machine orientated data file for the ACCASS program.

Any circuit complete or otherwise can be stored in full by punching out the display file on paper tape. This can be readily fed into the program by its own routine or by the Binary Loader program. Thus the display file is stored in a standard PDP-8 format and can be read by any standard PDP-8 program.

At the moment CIRCUITS is unable to communicate with ACCASS because:

1. SYSTEM 4-50 is not on-line and
2. A translater program has not been written.

The system will consist of a 338 Display conected to an 8K PDP-8 which is on-line to a SYSTEM 4-50.

Having drawn the circuit it will be possible to transfer the relevant information to the main machine. This will translate the data and prepare it for the nodal array in ACCASS. After analysis the results will be returned to the PDP-8 and graphically or otherwise displayed. Requests for any node voltages over a range of frequencies will be called by the operator on the Teletype which is at present unused by CIRCUITS.

It should be possible to calculate the input and output impedance by running ACCASS with modified components i.e. open circuit current scources and a forced potential at the required point.

OPERATIONS OF CIRCUITS

Load the program on the high-speed reader with the binary loader. A Ø accumulator indicates the program has been loaded properly.

The starting address of the program is Ø2ØØ.

The program has no error stops and has no general reset. Therefore, if an error occurs it will most likely be in one of the display files. If starting the program has no effect then a forced punch is suggested with the hope of saving the circuit drawn. To execute a forced punch set the switches to 1215, LOAD ADDRESS press START when punch has finished press STOP.

To reload the program set the switches to the starting address of Ø2ØØ, LOAD ADDRESS and START. Then load the display file tape.

Any component can be called from the Menu on the right by pointing the light pen at the one required. The Menu only shows one orientation of each component, others can be drawn using the rotate and mirror push buttons.

On pointing the light pen at a point in the circuit field a copy of the last component read will be displayed in the square located by the point.

If the component drawn is not in the required orientation it may generally be rotated through 90° by push button 6. This facility is not available to all the Menu i.e. cross wires, straight wires and the diagonal cross over.

Nos. 1, 2, 3, 4, 5, 6.

Having read a component it may be copied into any number of squares and rotated any number of times. Rotation not only acts on a particular square but also on the component stored by the light pen. Thus, once rotated all further fixing will correspond to the last orientation.

As both transistors have a total of eight orientations it was thought necessary to include a mirror facility. This only acts on the transistor components and pressing push button 7 causes the base to swing to the opposite side of the square leaving the emitter and collector as before. Rotate also acts on transistors and both functions may be repeated any number of times.

Pressing push button 11 causes the last component to be erased from the file. The light pen will now erase all squares it is pointed at, until push button 11 is cleared either manually by repressing it or automatically, when a component is read.

It is anticipated that mistakes will be made when drawing the circuit. To reduce the time in modifying the circuit it is possible to delay fixing of the values until the circuit layout is satisfactory.

If a component has just been read into a square then a value can be built up by pointing the light pen at the data across the top of the screen. Each character read will be shifted into the first register in the top right hand corner.

Pointing the light pen at the appropriate register label i.e. V(Value), No, S (Self inductance), M(Mutual inductance) A(Angle) will cause the contents to be erased and the next character string to be built up in that register.

Upon reading each character a copy of the three registers is sent to the Data File and displayed half scale behind the component. The Data File is activated when push button 3 is set.

Calling a square with push button 10 set causes the component to be read from the square into store and the register labels are set accordingly. The required value can then be read in without effecting the component. This enables all values to be read in, after the circuit layout has been completed.

Setting push button 9 causes a copy of the present component and the three registers to be copied into a square when a point is hit with the light pen. This is useful when a circuit has several similar components. If a component number is required it will need changing each time, pointing at (No.) and setting a new one.

Setting push button 5 causes a paper tape copy of the display file to be dumped on the high speed punch.

The tape has four sections:

Leader, code 2ØØ
Component File, 256 characters
Data File, variable length
Trailer, code 2ØØ

and has a format compatible with the binary loader. When the punching is complete the program returns to normal and further drawing is possible.

Place the previously punched Circuit File in the high speed reader with the leader code over the read photocells.

Set push button 4 and the tape will be read in with a modified version of the binary loader.

If Read does not work use the binary loader proceeding with the Circuit File Tape. It is safer to reload the program for the new circuit file will overwrite all information already displayed.

Push Button Functions

PB	0	Display Data
PB	1	Display Drawing Field
PB	2	Display Component Menu
PB	3	Display Data Field
PB	4	Read File from High Speed Reader
PB	5	Punch File from High Speed Punch

PB 6 Rotate Components
PB 7 Mirror Transistor

PB 9 Repeat Components and Value
PB 10 Select Location for Data
PB 11 Erase Location

CIRCUITS PROGRAM

These notes should be read in conjunction with the PAL.III program listing and the program flowchart immediately following.

Fig. 1 shows the general layout of the CIRCUIT display. There are four main fields each called by a push button and controlled by the light pen.

Data Display called by PBØ

1. A set of numerics with signs and a decimal point

   i.e. + 0 1 2 3 4

   - 5 6 7 8 9

2. A set of standard multiples

   i.e. M K

   m μ n p

3. A set of electrical units

   i.e. Ω F M

   A V / .

Each of these symbols can be read by the light pen and shifted into the three display registers. All characters are taken from the CHARACTER GENERATOR.

When a light pen hit occurs on one of the characters the Display Address Counter is read which points to the next instruction address in the display file. Subtracting 1 from this address gives the location in which the character was called. Reading from this address gives the two character codes previously called. As these are two six bit numbers: space (30), character (code), bits 6-11 are selected and stored in TEMP.

The character is then shifted into the appropriate register causing an overflow on the left.

4. Three labelled shift register C,D,E.

Each of these are six words long and can accommodate a total of 12 characters which are cleared when the label is read. All the labels are generated by the program and correspond to the data requirements of each component.

Register C (top) has four states and is reserved for component values and transistor JEDEC codes.

Wires have no value and E is inaccessible having no visible label.

Transistors and Diodes generate their JEDEC codes 2N and 1N in the right hand word which is shifted left when the JEDEC number is read in. No visible register label is generated.

Mutual Inductors generate the label 'S' for the self inductance value of the component.

All other components generate the label 'V' allowing the value to be read into the register.

Register D has two states and is unaccessible for all but mutual inductors and current sources.

Mutual Inductors generate the label 'M' allowing the mutual inductance value to be read.

Current Sources generate the label 'A' allowing the phase angle between the current and its source voltage to be read.

Register E has the label 'NO', for all the menu components and using '/' as a deliminator enables the component number and node numbers to be read.

The characters in the shift register are invisible to the light pen. When a register label is hit with the light pen the LEVEL is set to the appropriate value and CLEARD is entered to clear that particular register.

LABEL	REGISTER	LEVEL
V,S	C	0
A,M	D	1
N°	E	2

For all components except wires LEVEL is initially set to 0. For wires it is set to 2.

The drawing field called by PB1 consists of a 16 by 16 array of dots individually positioned by a POINT EDS instruction (1101). Each dot can be read by the light pen and selected as the origin for the component and blinking square which is used to indicate the present position. The instructions generating each dot allow the component to be changed by overwriting the subroutine jump location. All component subroutines in this field are invisible to the light pen.

When a light pen hit occurs on a dot the Display Address Counter points to any of the three locations after the X co-ordinate. To locate the position exactly requires a backwards search for the only fixed value, POINT EDS. Having located its position the address is stored in PEA and the following address yield Y,X and COMPA.

The Menu column called by PB2 lies to the right of the display and uses the same format as the Drawing Field except that the components are drawn with the light pen on and the dots are invisible. Seventeen components are displayed with access to a total of 54 using the rotate and mirror push buttons. When a light pen hit occurs on one of the components the second word in the push down pointer is read and points to the adress of the next POINT EDS in the display file. Subtracting 1 from this address gives the location of the subroutine jump which can be read to give the subroutine start address. This value is stored in COMP.

The Data field called by PB3 is completely invisible to the light pen and is a chronological file of the data read into the display register. The locating POINT EDS of each data block is a copy of the blinking square instructions with modified X and Y values to cause the characters to lie behind the square. Information to be entered or erased from the file requires a search through.the file for the same X, and Y co-ordinates as the present blinking square.

Routines Generating the Data Field

To clear a data block from the Data Field.

ERASE, Set the data field (PDP-8).to 1.

FPBC, Store the push button state. Clear push button 3.

FERASE,

FSEEK, Search through the Data Field for X and Y co-ordinates corresponding to the blinking square position. If the end of file marker is reached or the X and Y co-ordinates correspond, escape from FSEEK with search address in FT1

, If address found equals FLIMIT (file limit) escape from FERASE.

FSHIFT, Enter when search address ≠ FLIMIT. Shift field down 31 locations over the data block to be erased. Escape when end of field marker found and reduce FLIMIT by 31.

FPBS, Set push buttons to original state.

To read a data block into the Data Field.

PLANT, Set the data field (PDP-8) to 1.

FPBC, Field Push Button Clear as above.

FPLANT, Field Push Button Clear as above.

FSEEK, Field Push Button Clear as above.

, If search address $\neq$ FLIMIT go to FHERE

FMOVE, Transfer display registers onto the end of Data Field, add end of file marker (3000) and increment FLIMIT by 31. Jump out of FPLANT.

FHERE, Data block already present in Data Field. Overwrite with new data using FMOVE.

FTAKE and FLOOP are used in FMOVE to transfer the display registers to the Data Field.

FPBS Field Push Botton Set

THE SYSTEM

This section will form a specification for the programs still necessary to complete the project. At present we have a versatile drawing program and analysis program which require a translator to make them compatible.

This translator consists of six routines and two data files.

Routines

1. A user code routine to read the binary tape from CIRCUITS into a fixed array.
2. A routine to access the PDP-8 on-line and read the relevant information into a fixed array.
3. A translation routine used between 1, 2 and the circuit arrays.
4. An iterative routine to calculate the nodes in the circuits.
5. A translation routine to extract relevant information from the arrays and insert it in ACCASS
6. A routine to interpret instructions from the PDP-8 Teletype.

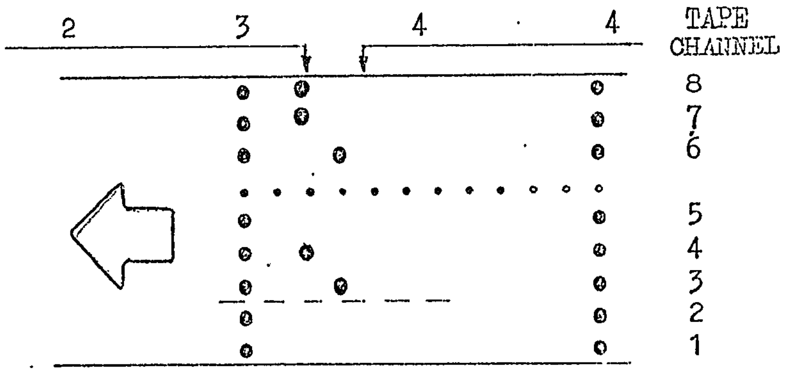
Data Files

1. A store of Hybrid π equivalent circuits for transistors and diodes
2. A store of codes for the components giving type and the connected nodes.

Each Routine has specific specifications and they are as follows:

Routine 1, Binary Tape Format - PDP-8 paper tape is 8-channel and each character corresponds to bits 4-11 of the accumulator at the time of punching. It is, therefore, impossible to reproduce a 12 bit word with one character and the output is in the form of two consecutive six bit characters: i.e.,

Location Bits:	0	1	2	3	4	5	6	7	8	9	10	11
Binary Number:	0	1	0	0	1	1	1	0	0	1	0	0
Octal Number :	2			3			4			4		



This leaves two bits per character unused which are coded in the following way for the first character read.

Tape Channel		
1	2	
0	0	This character and the following one form a single data word
0	1	This character and the following one form a single address word
1	1	This character, channels 3-5 from the new data field
1	0	A leader/trailer code indicating the limits of the tape.

Circuits Tape Format - The tape generated by CIRCUITS is compatible with BINARY TAPE enabling it to be read freely with PDP-8 routines.

The data punched on the tape is listed as follows:

1. Leader Code (2ØØ)
2. Address of subroutine jump in square 1,1
3. Address of subroutine in square 1,1.
   i.e., contents of 2.
4. Address of subroutine jump in square 1,2

5. Address of subroutine in square 1,2

. . . . . (256 squares in a 16 x 16 matrix)

. . . . .

. . . . .

. . . . .

514.Address of subroutine jump in square 16,16

515.Address of subroutine in square 16,16
i.e.,contents of 514

(This completes the circuit field in locations 2125 to 4524)

516.Change Instruction Field to 1 (31Ø)

517.Start Address of Data Field

518.Contents of first location in Data Field

519.Contents of second location in Data Field

. etc.

.

.

. End of file code (3ØØØ)

. Address of FLIMIT

. Value of FLIMIT (last address in Data Field)

. Trailer code (2ØØ)

Read 1 - Having defined the input format, routine 1 is responsible for accepting the binary tape and presenting the integer values to the translator routine 3.

Routine 2 - This must read information from the PDP-8 via the DXC and present it to the first translator. Having direct access the whole 12 bit word may be called and it should be possible to use the DXC File Control Package.

Routine 3 - Having received data from either of the two routines 1 and 2 it now requires translating into the Circuit and Data Arrays. From the information given we have to select the source address in the array and calculate the values.

The format of the input allows two methods for the gating of the information:

1. The address can be decoded to give the array address in which the data should be sent.
i.e., C(x,y) where $y = \frac{\text{address} - 2120 \text{ (Octal)}}{5 \times 20}$

ans $x = \frac{\text{address} - 2120}{5}$ (Octal) $-20 \times y$

Note, the base address is 2125; a subroutine address occurs every 5 locations; the destination array is 16 x 16. 2120 octal 1104 decimal, 20 Octal = 16 decimal. Thus, y and x form the dividend and remainder of a division by 16 having normalised the base address to 1.

2. Accept that 256 sets of data will be transmitted and use a double do loop to read them into the matrix.

```
i.e., do 1 I = 1, 16
          do 1 J = 1, 16

read address (ignore address code)
read data    (forward to :

- 1 continue                      C(I,J,1)
```

The data that follows each address is the starting address in the PDP-8 of a component subroutine. These lie between octal 5100 and 6640 as shown in LIST 1. Each starting address is 20 octal locations from its nearest neighbor apart from INDH and INDV which are 40 because of the increased number of instructions. Due to the matrix format it is possible to normalise the subroutine starting address. These values are given in LIST 1 and LIST 3.

Applying the following equation gives the normalised label.

$$\text{Label} = \frac{\text{Subroutine starting address} - 5060}{20} \text{ (Octal)}$$

This label will be used to access the data in the components code file. INDH and INDV will not effect the file which will just contain two empty and unused locations at 8 and 10.

Having read in the circuit section we now have to translate the variable length data file. Ignoring words 516 and 517 the first word read will contain a POINT EDS code (1101) or a POP code if the end of the file has been reached. Words 2 and 3 will contain the y and x co-ordinates of the data square. From these the destination address in the data array must be calculated. Both x and y are in a fixed format. Bits 0-1 are instructions and should be deleted if present. Bits 2-11 contain the co-ordinate value.

To cause the data characters to lie behind the component the co-ordinates have been modified from the base co-ordinates in the bottom left hand corner.

$$\text{i.e., } y_2 = y_1 + 60 \text{ (Octal)}$$

$$x_2 = x_1 - 60 \text{ with no overflow into bits 0 or 1.}$$

Therefore, to reduce the co-ordinates into base form requires:

$$y_1 = y_2 - 60$$

$$x_1 = x_2 + 60 \text{ (Octal)}$$

and to convert these into array labels requires:

$$I = \frac{y_2-60+70}{70}$$

$$J = \frac{x_2+60+70}{70} \quad \text{(Octal)}$$

Note: The base co-ordinates for square 1,1 are 0 0. Each square is 70 by 70 display units (Octal).

Character 4 in the data file contains CHAR EDS code 1151 which should be ignored.

The following six words contain 12 character codes from the character generator LIST 2 corresponding to the contents of DISPLAY REGISTER C, i.e., the component value. This has to be converted into a floating point number and stored in the data array D (I,J,1).

The routine must, therefore, be able to convert for example:

1 3. 5 mA/uV into $.135X10^5$

It should also check that the units correspond with the component. The above number would appear as follows in character generator code:

3636 3601 0312 0516 2527 1726

Note that 36 is a space character, 00-11 are their octal equivalents and powers of ten MEGA - PICO are sequential such that exponent = 3 X (code - 15).

If the component is a transistor or diode register c will contain the JEDEC code, i.e., 2N1309A. This will need interpreting in such a way that it is unique and also checked with the list contained in the EQUIVALENT CIRCUIT STORE.

The following word in the display file will contain a CR/LF code and then there will follow a further six word from register D.

Register D is only used for certain components. (i.e., current source as the angle in which case register C must have contained a "/" indicating a reference voltage.)

Any values found should be converted to floating point and stored in D(I,J,2).

After the second CR/LF code six words of register E follow containing the component number and nodal information. The suggested coding for this is as follows with '/' acting as a deliminator.

1. If the component is a wire the first number found will be taken as the node code at that point. This number will not necessarily correspond with

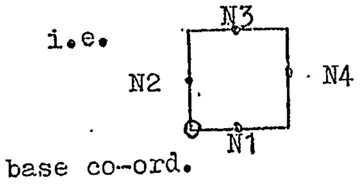
the nodes calculated by routine 4 and will only act as a code.

2. If the component is a dependant current source

```
register E =component / node / node / node / node
            number       n1     n2     C      D
```

Where nodes c and d correspond to those in ACCASS and nodés N1 and N2 are fixed by the square:

i.e.



3. If the component is a mutual inductor

register E = component number / node n1 / node n2

Mutual inductors will be given the same number and the ACCASS translator will calculate nodes c and d.

After reading three display registers there is an ESCAPE FROM CHARACTER MODE instruction before the next block of data is received.

If the first word of any block equals POP(3000) there is no more Data to read.

The last word on the tape is the value of FLIMIT which gives the number of data blocks using the following equation.

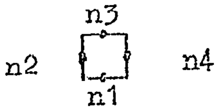
$$\text{Number of blocks} = \frac{\text{FLIMIT} - 600}{31} \quad \text{(Octal)}$$

Note that 600 is the starting address of the Data Field. There are 31 Octal instructions per Data Block.

<u>Routine 4</u> - Each square locates a minimum of two nodes, i.e., n1, n2. The nodes n3, and n4 can be located from adjacent squares

i.e., x,y,n3 =x,y+1,n1

x,y,n4 =x+1,y,n2



ACCASS equires a maximum of 4 nodes as data, a, and b will be calculated c and d must be set in register E when a dependant current source is called.

This results in four coded nodes n1, n2, a, b, and two calculated nodes c, d.

Circuit Array (Integer)

Dimension C(x =1-16, y =1-16, z 1-8)

z = 1 : Component code (1-55) calculated from subroutine starting address

z = 2 : Component number from register E
z = 3 : n1 code number
z = 4 : n2 code number
z = 5 : c code number
z = 6 : d code number
z = 7 : a (n1) calculated node number
z = 8 : b (n2) calculated node number

Data Array

Dimension D(x=1-16,y=1-16,z=1-2)

z = 1 : Component value
z = 2 : Mutual inductance value or current angle

Component Codes Array (Binary)

Dimension E (x=1-55)
6 54 3 21
x = 1 : E = 1 0 1 1 1 1
x = 2 : E = 0 1 0 1 0 1
x = 3 : E = 0 1 1 0 1 0

x = 55: E = 0 0 0 0 0 0

The subscript x and binary value E are given in the component list

The Array E can be converted into a secondary array B which gives Boolean expressions for component states.

Boolean Array

Dimension B(x=1-6) corresponding to bits 1-6 in E where 1=True and 0=False

x = 1 : The component has a connection with n4, true or false
x = 2 : The component has a connection with n3, true or false
x = 3 : The component has a connection with n2, true or false
x = 4 : The component has a connection with n1, true or false
x = 5 : The component is not a wire, true or false
x = 6 : The component is a wire, true or false

Using two arrays where one generates the other, reduces the amount of storage required, Arrays E and B could be combined into one 330 word array but as only one component is called at a time it is unecessary and wasteful.

Before proceeding with routine 4 the calculated nodes require definition.

0-99 A legitimate node number for ACCASS
1000 A node not yet defined
1001 A node impossible because of an open circuited connection, i.e.,not limited by a tag symbol.

From this information it is possible to calculate the nodes a and b through the circuit.

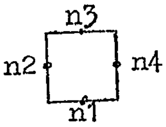
For run 1 initially set all the nodes a,b to undefined (1000).

To initiate run 2 use the component code number as a subscript select a word from Array E and generate from this word Array B which shows the possible node connections in that square. Set all false nodes to the impossible code (1001).

i.e., Component square x y contains a horizontal wire

C(x,y,1) = 2

E(2) = 0 1 0 1 0 1



B(1) = True (n4)
B(2) = False (n3)
B(3) = True (n2)
B(4) = False (n1)

This indicates that n1 and n3 are impossible and should, therefore, be set to (1001).

C(x,y,8) = 1001 (n1)
C(x,y+1,8)=1001 (n3)

Repeat this for all 256 squares.

On the third run ignore all squares that are blank.

i.e., C(x,y,1) = 5

Set CHANGE to ZERO and repeat run 3 until CHANGE is found to be zero at the end of the run.

Consider a square containing a wire. The connected nodes should finally have the same value. There are two initial possibilities.

1. All the nodes are undefined in which case increment the NODE counter and set each possible node to the new value.

2. One or more nodes have values within the range 0-99 in which case select the smallest and force the others to this value.

Consider a square containing a component. If the node is undefined increment the NODE counter and take the new value.

Consider a CROSSOVER wire (3 separate subroutines). Both vertical and diagonal CROSSOVER wires are treated in the same way except that the diagonal CROSSOVERS must account for their opposite half being present in the vertically adjacent square. Take each separate wire separately and treat it exactly the same as the single wire. For these special cases Array B only indicates the nodes connected and not the node pairs, which must be given in the subroutine. If when setting any legitimate node an impossible code is overwritten due to an adjacent component, indicate the error but do not inhibit the change.

If the node value is changed when a set occurs increment the CHANGE counter. If after any pass of the third run CHANGE is Zero, routine 2 should be terminated.

Routines 5 and 6 will not be outlined as both require further study of ACCASS and the system.

DISCUSSION

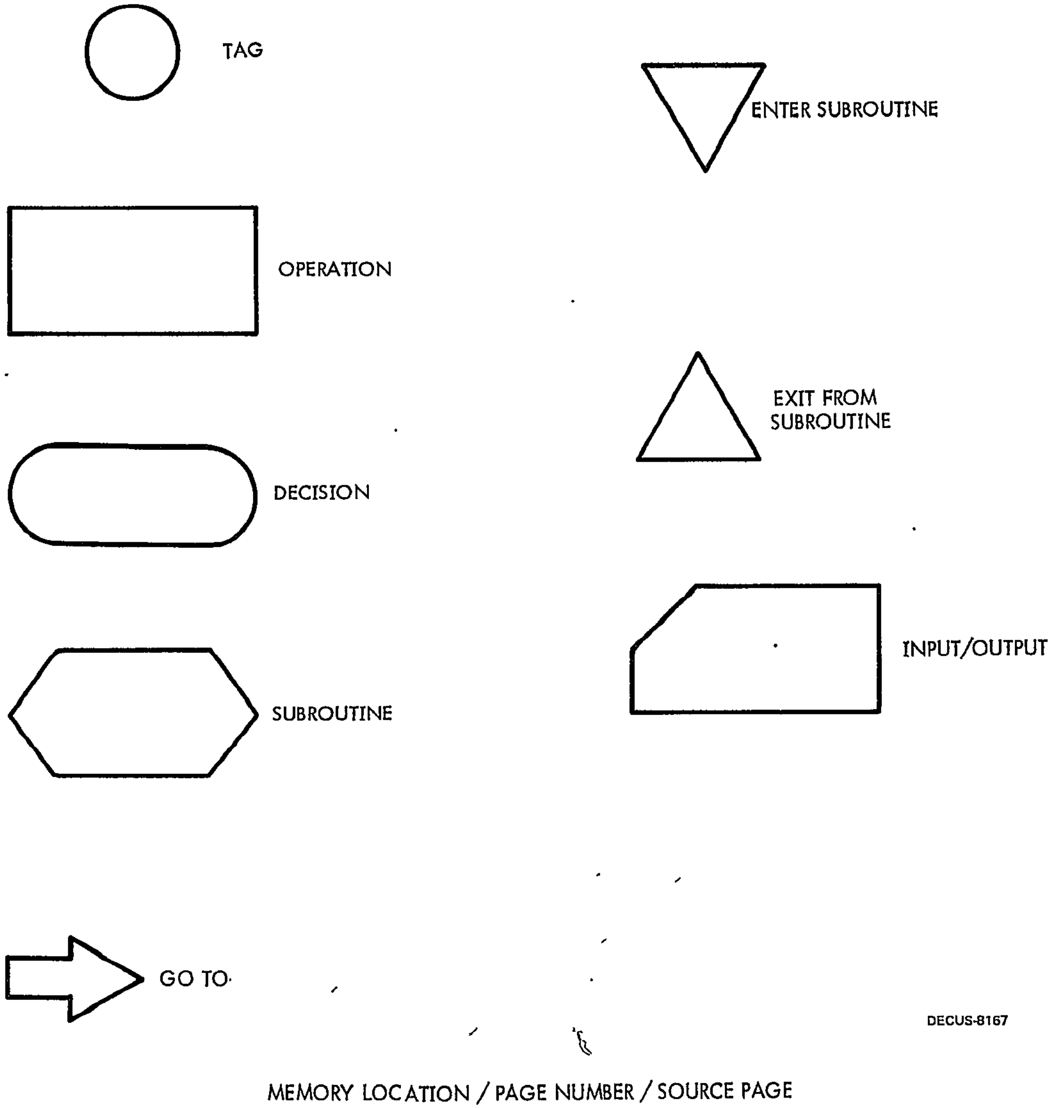
It is realised that the circuit system described is very clumsy with regards to the on-line capabilities of SYSTEM 4-50. With this facility the PDP-8 is reduced to a simple display peripheral whose limited display file is selected from a much larger one in the 4-50. This means CIRCUITS can be written on a much larger scale with a giant drawing area and a greater variety of components. With the 4-50 in control the translation is simplified, as only the display requires coding while all the data is stored in FORTRAN format.

CONCLUSION

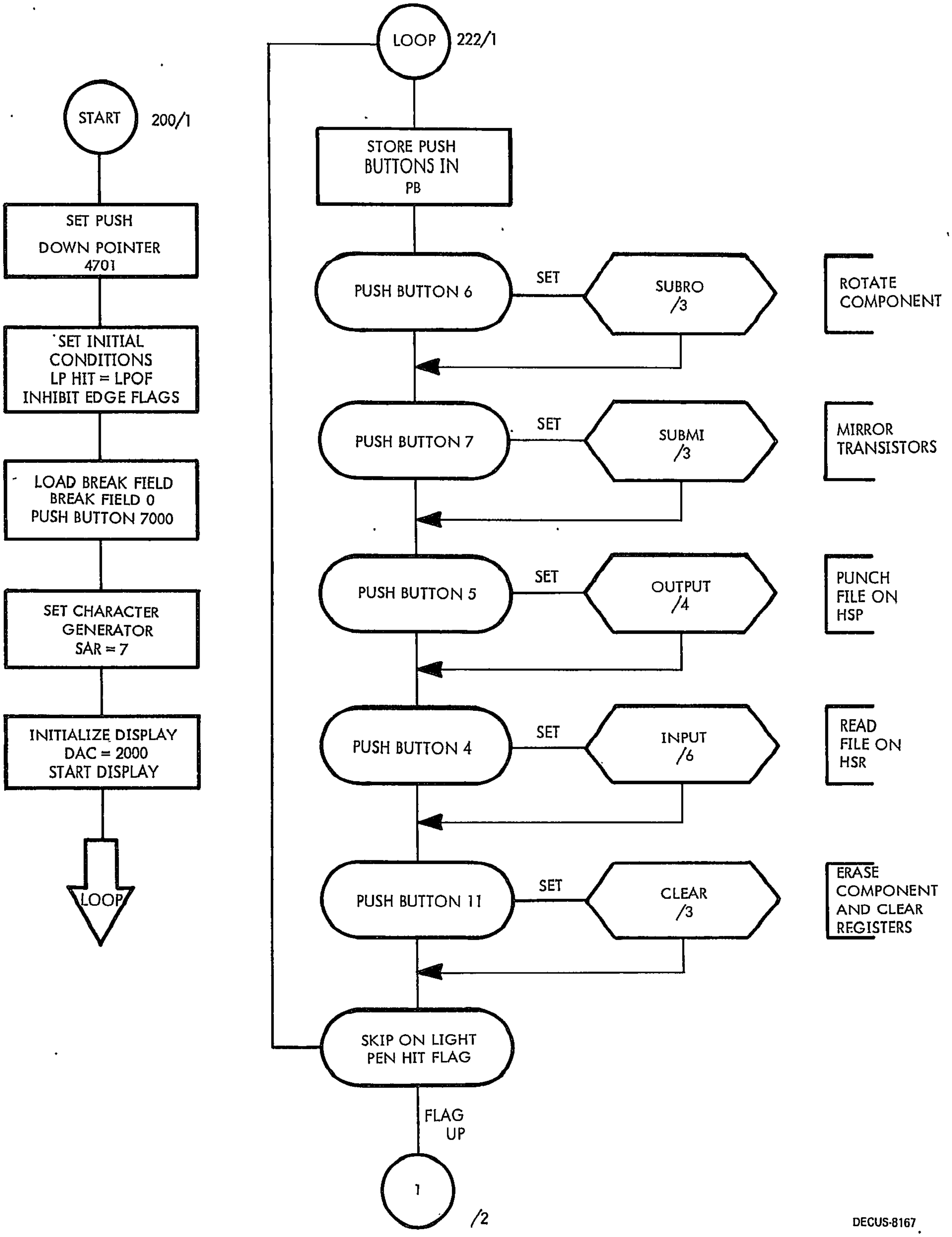
CIRCUITS as a program is most efficient in translating circuit diagrams into computer orientated data. The push buttons and light pen are used most effectively and the comprehensive component file allows the most awkward layouts to be drawn with the greatest of ease.

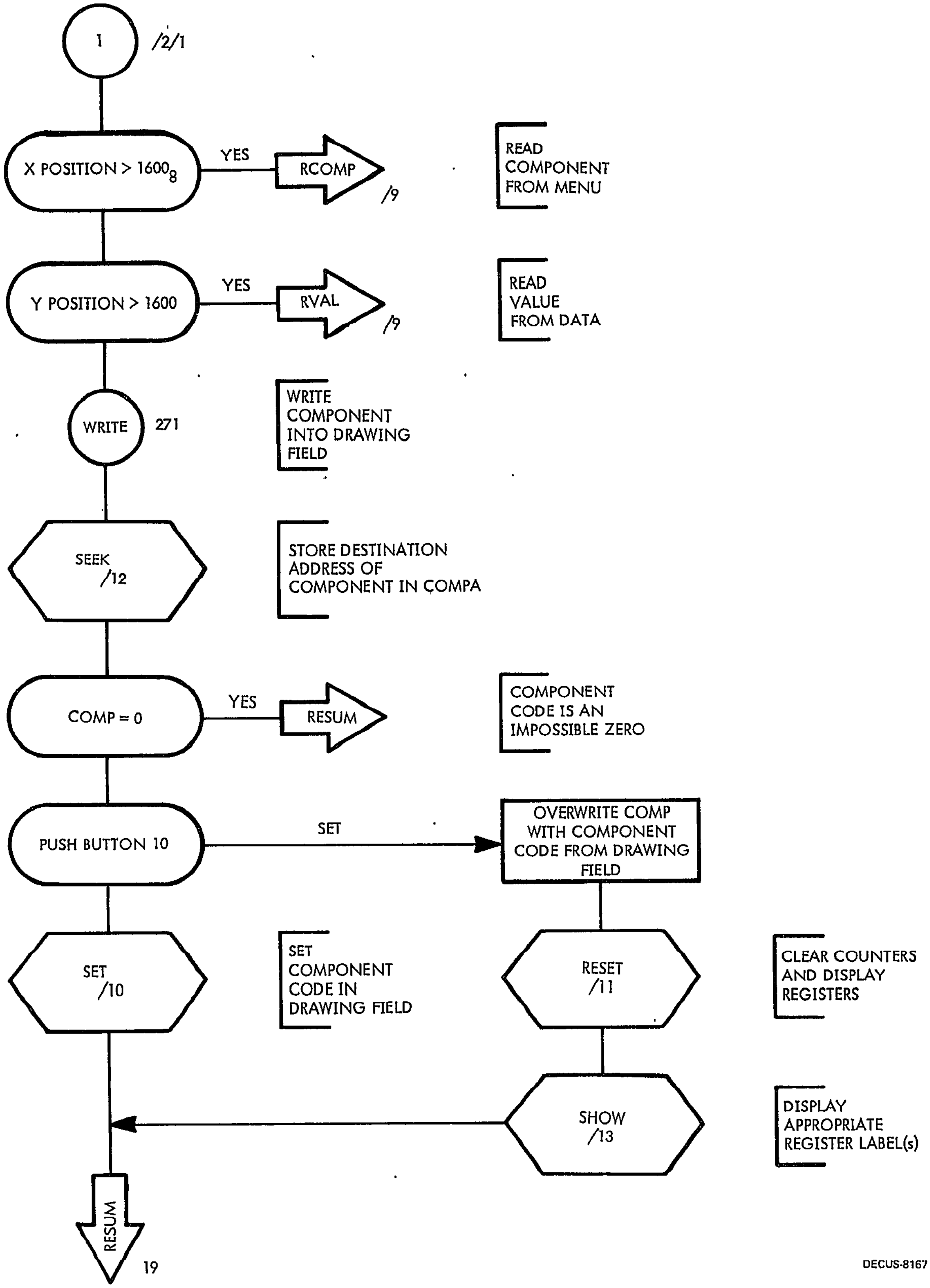
With regards to the Circuit System there still remains a lot of work to do. Rethinking on the role of the on-line display would allow a much more versatile program to be written.

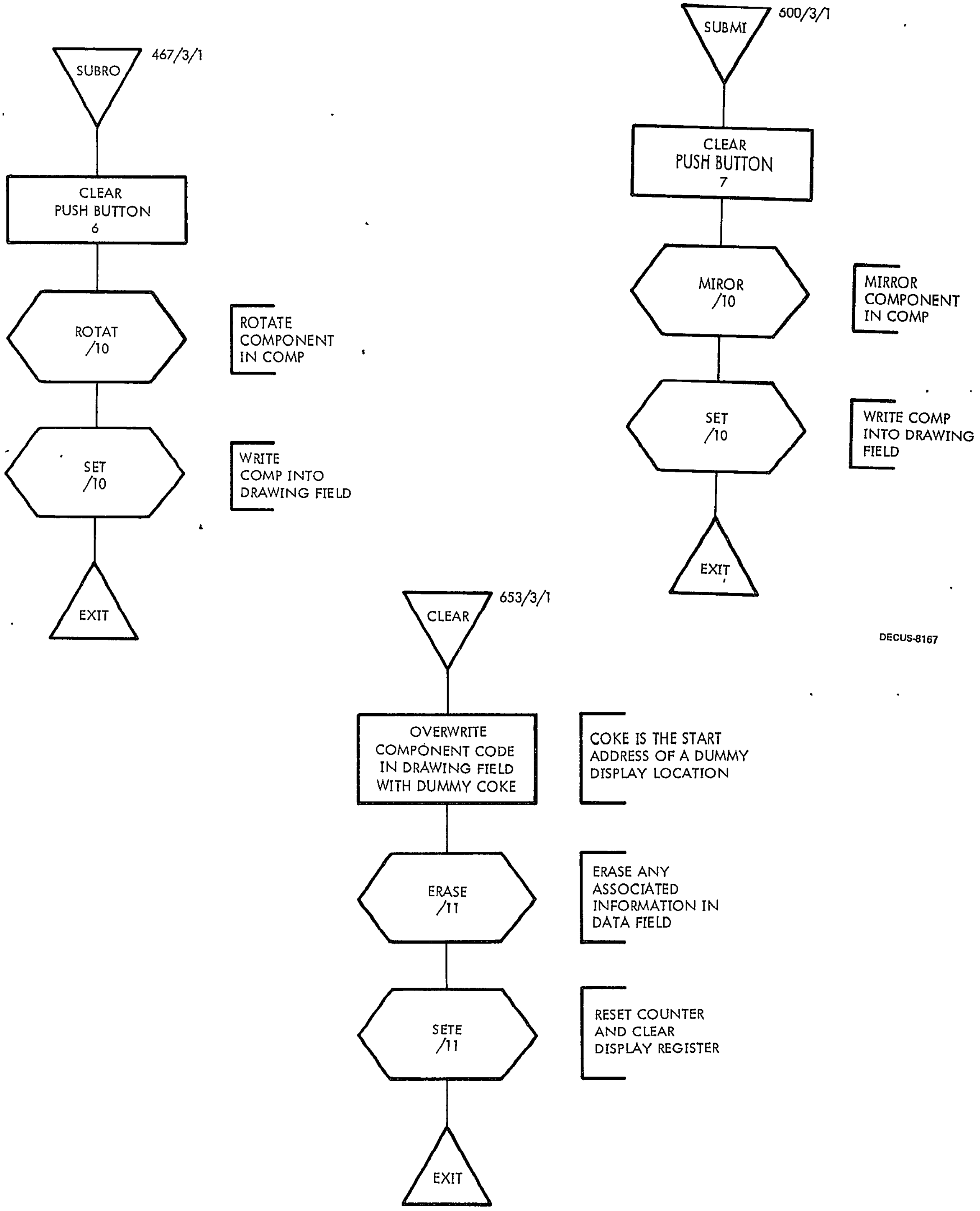
CIRCUITS PROGRAM
FLOWCHART CODING



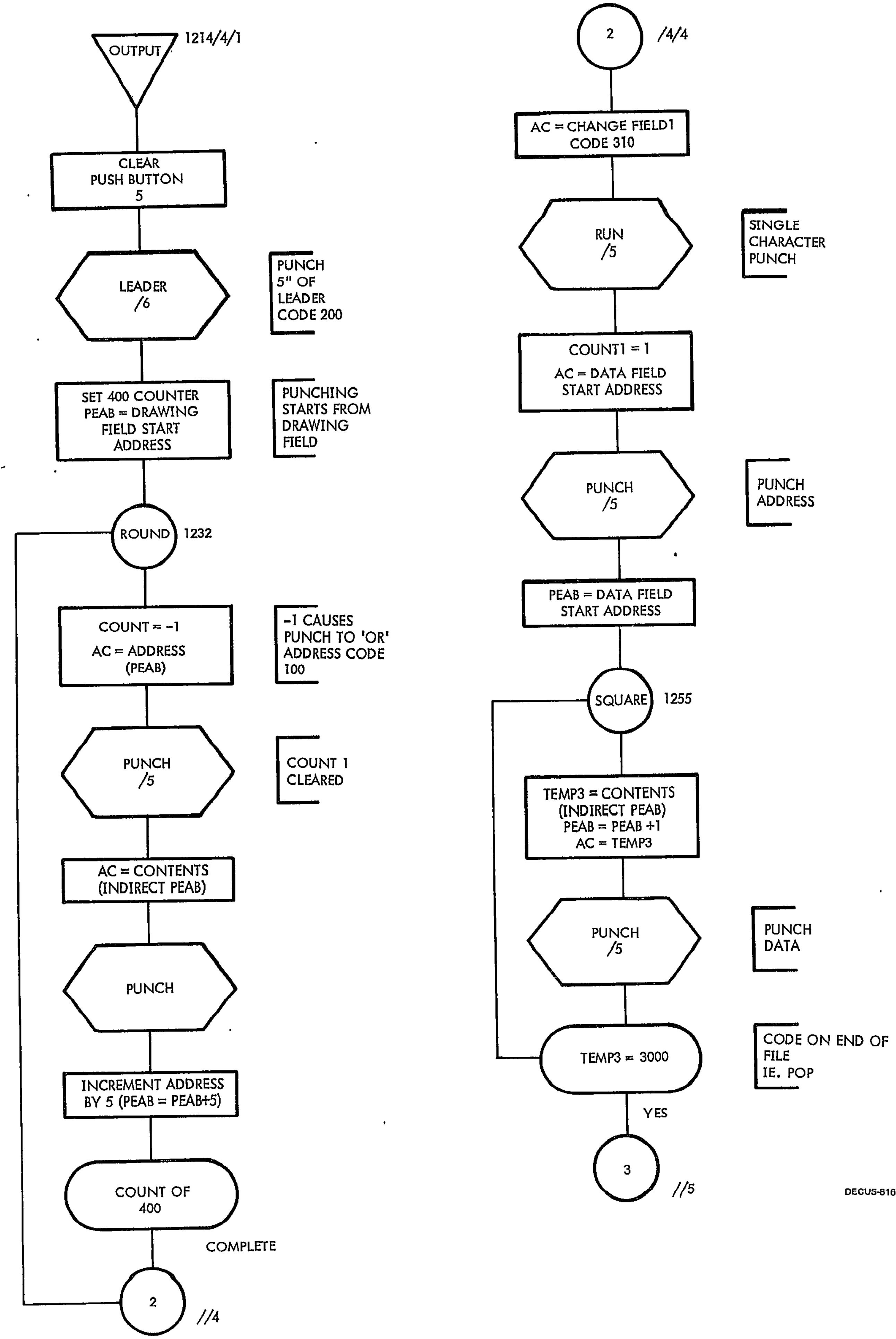
DECUS-8167

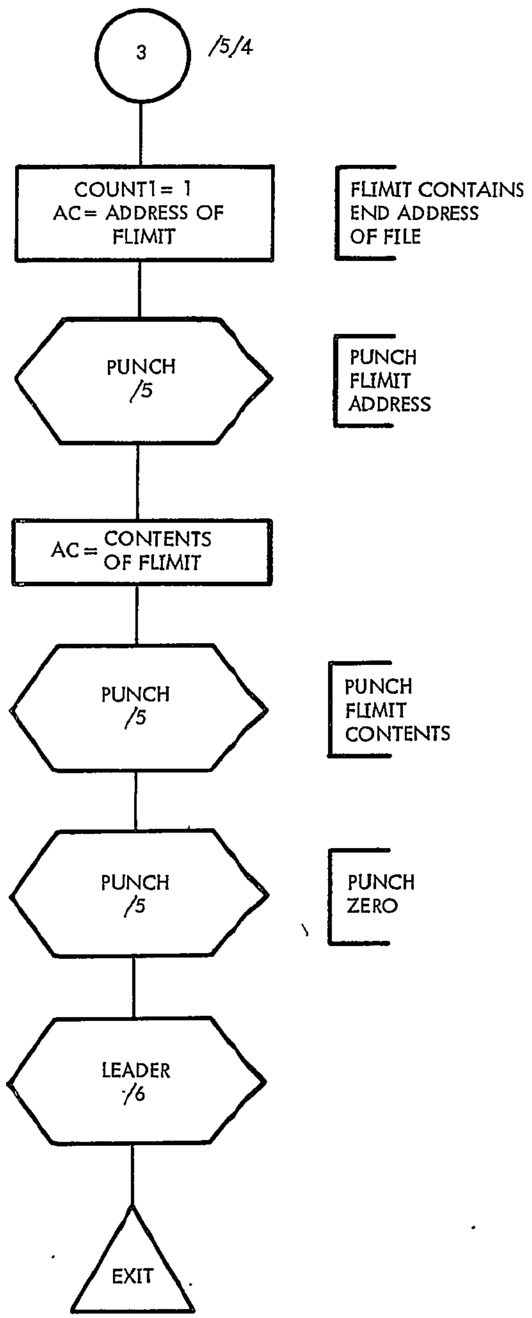

START
200/1
SET PUSH DOWN POINTER 4701
SET INITIAL CONDITIONS LP HIT = LPOF INHIBIT EDGE FLAGS
LOAD BREAK FIELD BREAK FIELD 0 PUSH BUTTON 7000
SET CHARACTER GENERATOR SAR = 7
INITIALIZE DISPLAY DAC = 2000 START DISPLAY
LOOP
LOOP
222/1
STORE PUSH BUTTONS IN PB
PUSH BUTTON 6
SET
SUBRO /3
ROTATE COMPONENT
PUSH BUTTON 7
SET
SUBMI /3
MIRROR TRANSISTORS
PUSH BUTTON 5
SET
OUTPUT /4
PUNCH FILE ON HSP
PUSH BUTTON 4
SET
INPUT /6
READ FILE ON HSR
PUSH BUTTON 11
SET
CLEAR /3
ERASE COMPONENT AND CLEAR REGISTERS
SKIP ON LIGHT PEN HIT FLAG
FLAG UP
1
/2

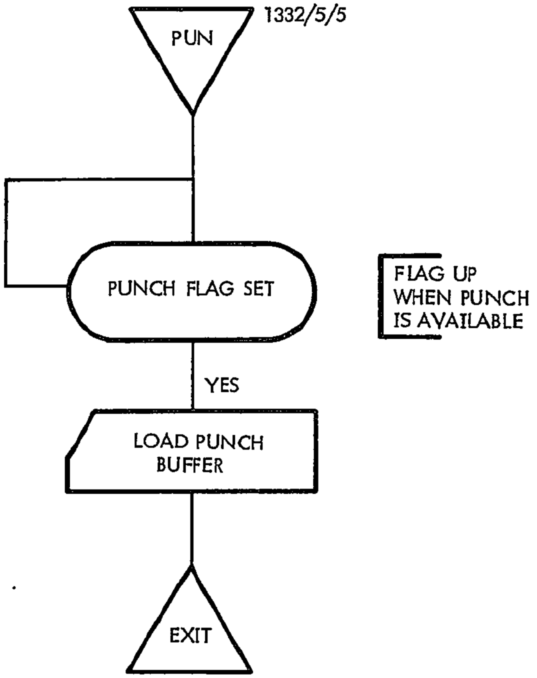

1
/2/1
X POSITION > 1600_8
YES
RCOMP
/9
READ COMPONENT FROM MENU
Y POSITION > 1600
YES
RVAL
/9
READ VALUE FROM DATA
WRITE
271
WRITE COMPONENT INTO DRAWING FIELD
SEEK /12
STORE DESTINATION ADDRESS OF COMPONENT IN COMPA
COMP = 0
YES
RESUM
COMPONENT CODE IS AN IMPOSSIBLE ZERO
PUSH BUTTON 10
SET
OVERWRITE COMP WITH COMPONENT CODE FROM DRAWING FIELD
SET /10
SET COMPONENT CODE IN DRAWING FIELD
RESET /11
CLEAR COUNTERS AND DISPLAY REGISTERS
SHOW /13
DISPLAY APPROPRIATE REGISTER LABEL(s)
RESUM
19

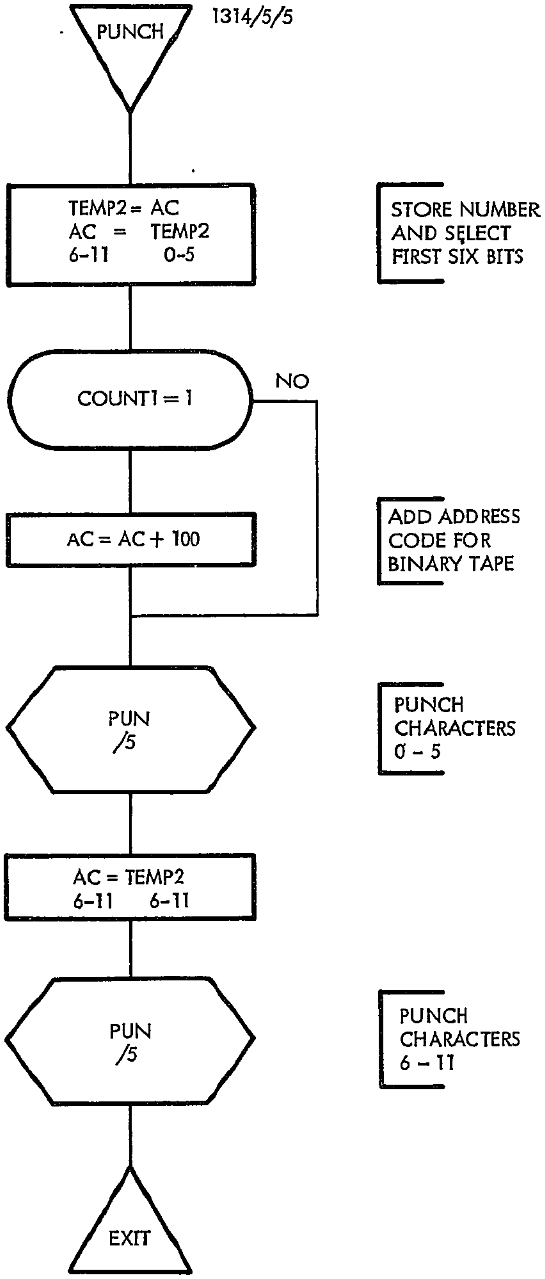

SUBRO
467/3/1
CLEAR PUSH BUTTON 6
ROTAT /10
ROTATE COMPONENT IN COMP
SET /10
WRITE COMP INTO DRAWING FIELD
EXIT
SUBMI
600/3/1
CLEAR PUSH BUTTON 7
MIROR /10
MIRROR COMPONENT IN COMP
SET /10
WRITE COMP INTO DRAWING FIELD
EXIT
CLEAR
653/3/1
OVERWRITE COMPONENT CODE IN DRAWING FIELD WITH DUMMY COKE
COKE IS THE START ADDRESS OF A DUMMY DISPLAY LOCATION
ERASE /11
ERASE ANY ASSOCIATED INFORMATION IN DATA FIELD
SETE /11
RESET COUNTER AND CLEAR DISPLAY REGISTER
EXIT

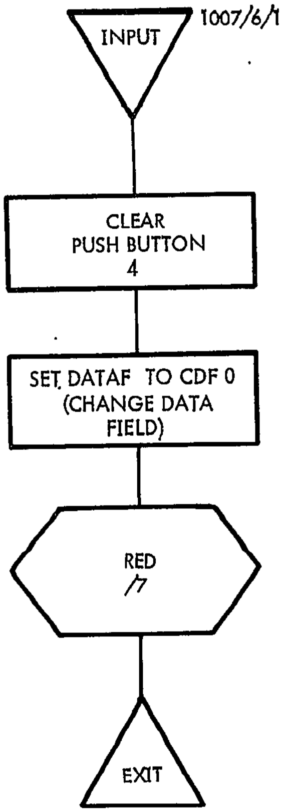
DECUS-8167

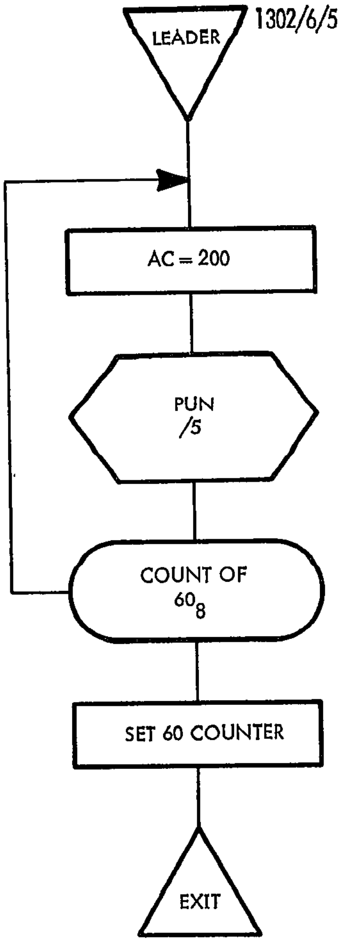

OUTPUT
1214/4/1
CLEAR
PUSH BUTTON
5
LEADER
/6
PUNCH
5" OF
LEADER
CODE 200
SET 400 COUNTER
PEAB = DRAWING
FIELD START
ADDRESS
PUNCHING
STARTS FROM
DRAWING
FIELD
ROUND
1232
COUNT = -1
AC = ADDRESS
(PEAB)
-1 CAUSES
PUNCH TO 'OR'
ADDRESS CODE
100
PUNCH
/5
COUNT 1
CLEARED
AC = CONTENTS
(INDIRECT PEAB)
PUNCH
INCREMENT ADDRESS
BY 5 (PEAB = PEAB+5)
COUNT OF
400
COMPLETE
2
//4
2
/4/4
AC = CHANGE FIELD1
CODE 310
RUN
/5
SINGLE
CHARACTER
PUNCH
COUNT1 = 1
AC = DATA FIELD
START ADDRESS
PUNCH
/5
PUNCH
ADDRESS
PEAB = DATA FIELD
START ADDRESS
SQUARE
1255
TEMP3 = CONTENTS
(INDIRECT PEAB)
PEAB = PEAB +1
AC = TEMP3
PUNCH
/5
PUNCH
DATA
TEMP3 = 3000
CODE ON END OF
FILE
IE. POP
YES
3
//5

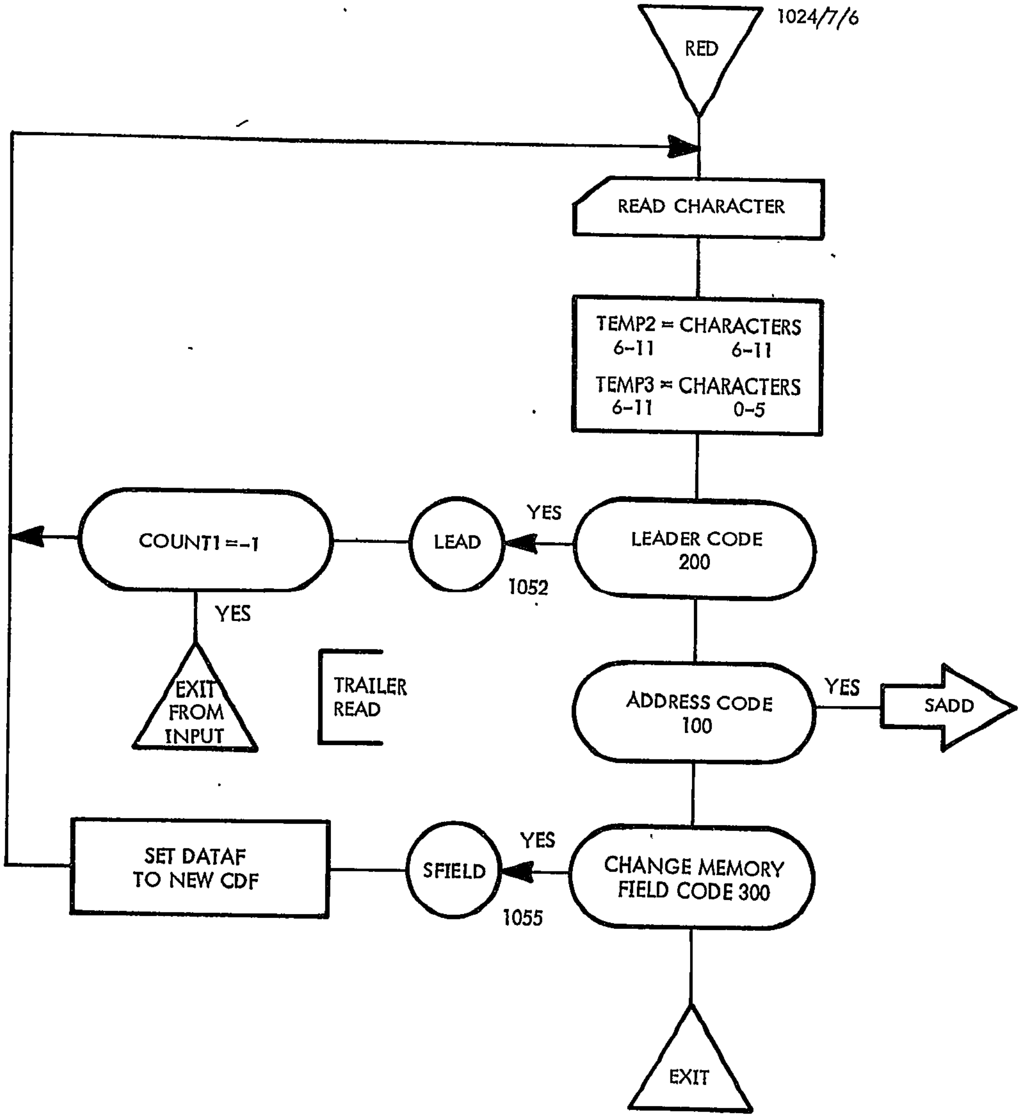

3
/5/4
COUNT1 = 1
AC = ADDRESS OF FLIMIT
FLIMIT CONTAINS END ADDRESS OF FILE
PUNCH /5
PUNCH FLIMIT ADDRESS
AC = CONTENTS OF FLIMIT
PUNCH /5
PUNCH FLIMIT CONTENTS
PUNCH /5
PUNCH ZERO
LEADER /6
EXIT


PUN
1332/5/5
PUNCH FLAG SET
FLAG UP WHEN PUNCH IS AVAILABLE
YES
LOAD PUNCH BUFFER
EXIT


PUNCH
1314/5/5
TEMP2 = AC
AC 6-11 = TEMP2 0-5
STORE NUMBER AND SELECT FIRST SIX BITS
COUNT1 = 1
NO
AC = AC + 100
ADD ADDRESS CODE FOR BINARY TAPE
PUN /5
PUNCH CHARACTERS 0 - 5
AC 6-11 = TEMP2 6-11
PUN /5
PUNCH CHARACTERS 6 - 11
EXIT

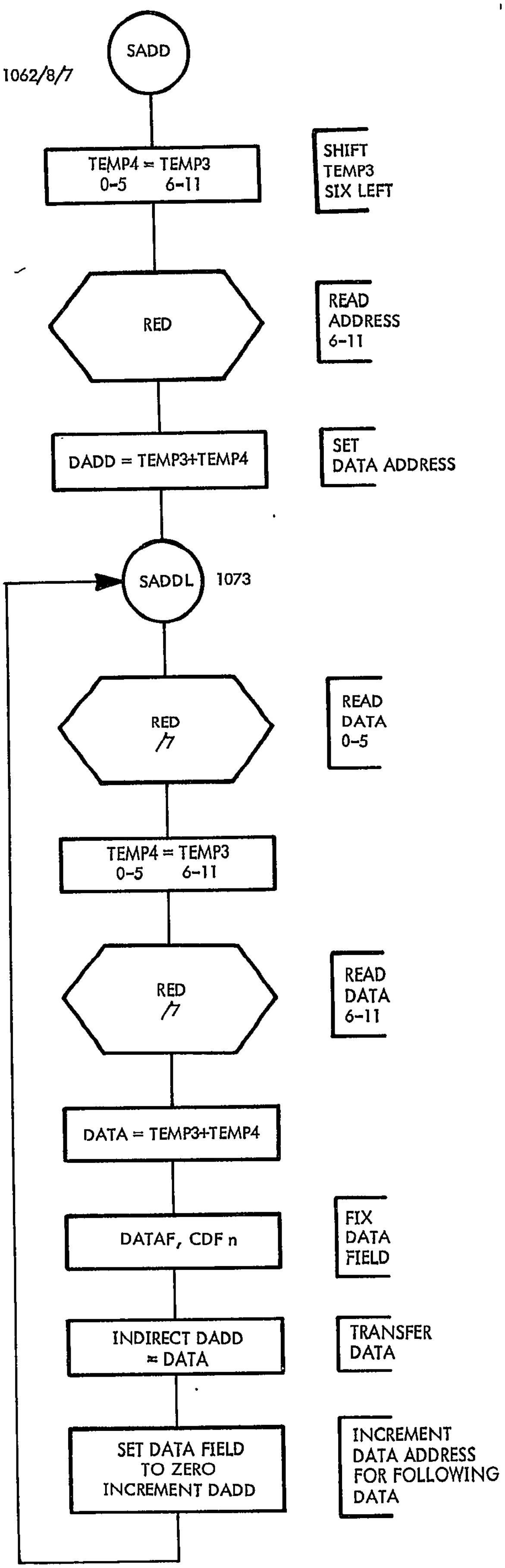

INPUT
1007/6/1
CLEAR
PUSH BUTTON
4
SET DATAF TO CDF 0
(CHANGE DATA
FIELD)
RED
/7
EXIT

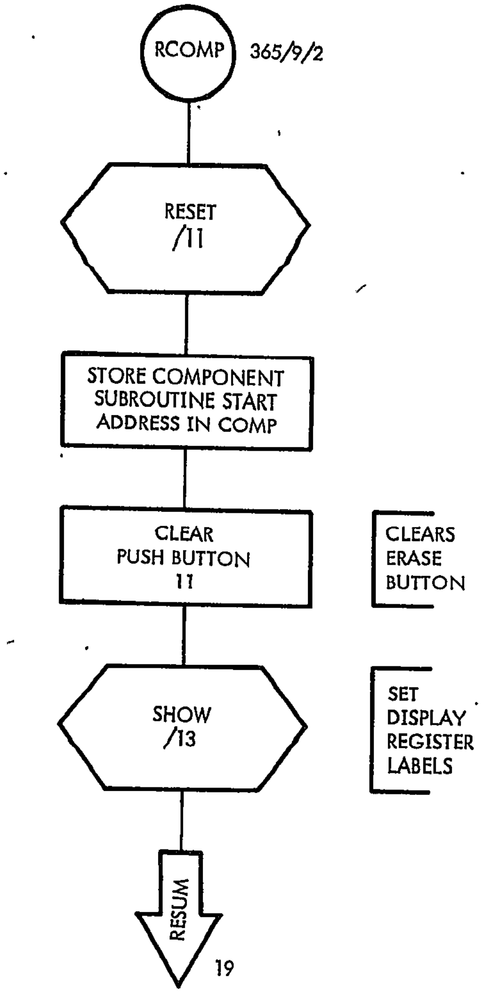

LEADER
1302/6/5
AC = 200
PUN
/5
COUNT OF
60_8
SET 60 COUNTER
EXIT

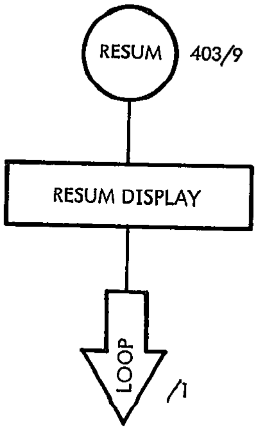


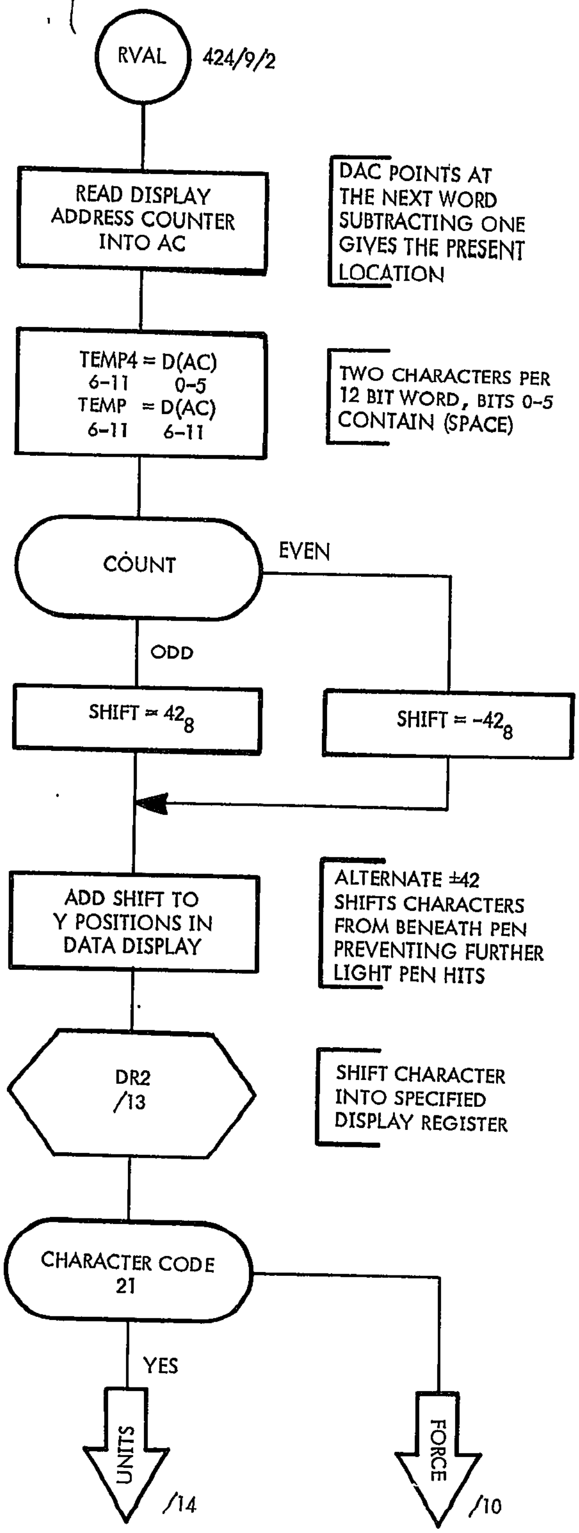
The instructions leading to Gate perform the function of the column of decisions above.

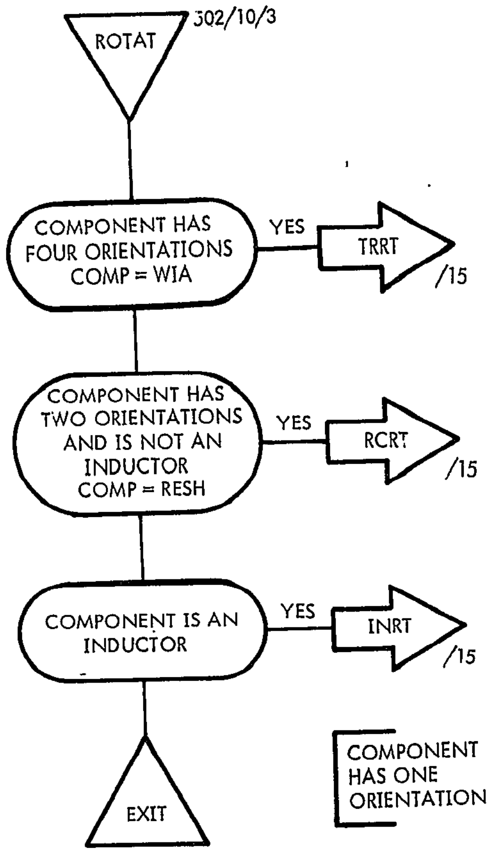
DECUS-8167

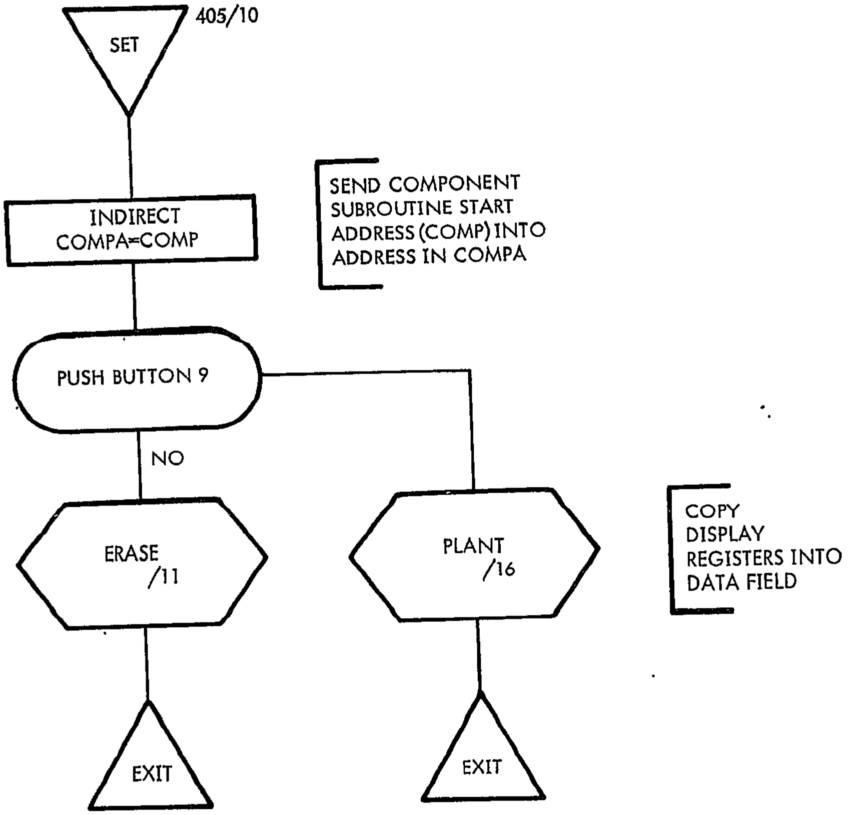

SADD
1062/8/7
TEMP4 = TEMP3
0-5 6-11
SHIFT
TEMP3
SIX LEFT
RED
READ
ADDRESS
6-11
DADD = TEMP3+TEMP4
SET
DATA ADDRESS
SADDL
1073
RED
/7
READ
DATA
0-5
TEMP4 = TEMP3
0-5 6-11
RED
/7
READ
DATA
6-11
DATA = TEMP3+TEMP4
DATAF, CDF n
FIX
DATA
FIELD
INDIRECT DADD
= DATA
TRANSFER
DATA
SET DATA FIELD
TO ZERO
INCREMENT DADD
INCREMENT
DATA ADDRESS
FOR FOLLOWING
DATA

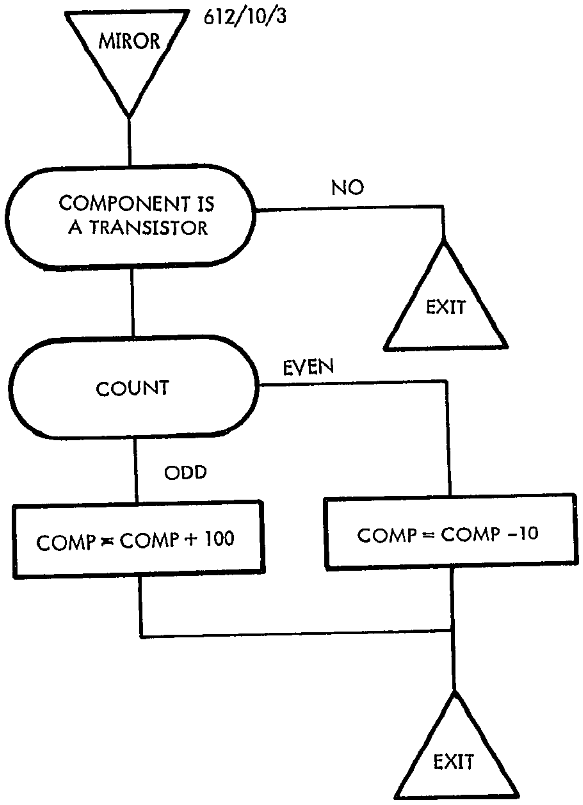

RCOMP
365/9/2
RESET
/11
STORE COMPONENT
SUBROUTINE START
ADDRESS IN COMP
CLEAR
PUSH BUTTON
11
CLEARS
ERASE
BUTTON
SHOW
/13
SET
DISPLAY
REGISTER
LABELS
RESUM
19

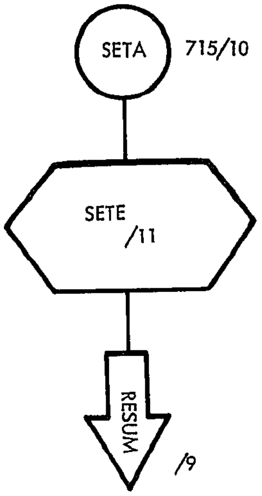

RESUM
403/9
RESUM DISPLAY
LOOP
/1

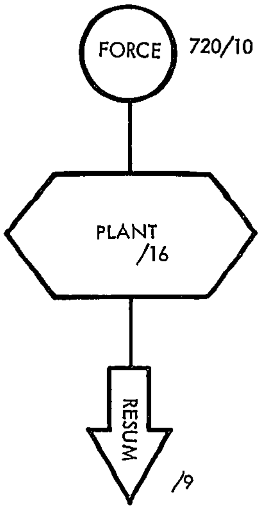

RVAL
424/9/2
READ DISPLAY
ADDRESS COUNTER
INTO AC
DAC POINTS AT
THE NEXT WORD
SUBTRACTING ONE
GIVES THE PRESENT
LOCATION
TEMP4 = D(AC)
6-11 0-5
TEMP = D(AC)
6-11 6-11
TWO CHARACTERS PER
12 BIT WORD, BITS 0-5
CONTAIN (SPACE)
COUNT
EVEN
ODD
SHIFT = 42₈
SHIFT = -42₈
ADD SHIFT TO
Y POSITIONS IN
DATA DISPLAY
ALTERNATE ±42
SHIFTS CHARACTERS
FROM BENEATH PEN
PREVENTING FURTHER
LIGHT PEN HITS
DR2
/13
SHIFT CHARACTER
INTO SPECIFIED
DISPLAY REGISTER
CHARACTER CODE
21
YES
UNITS
/14
FORCE
/10

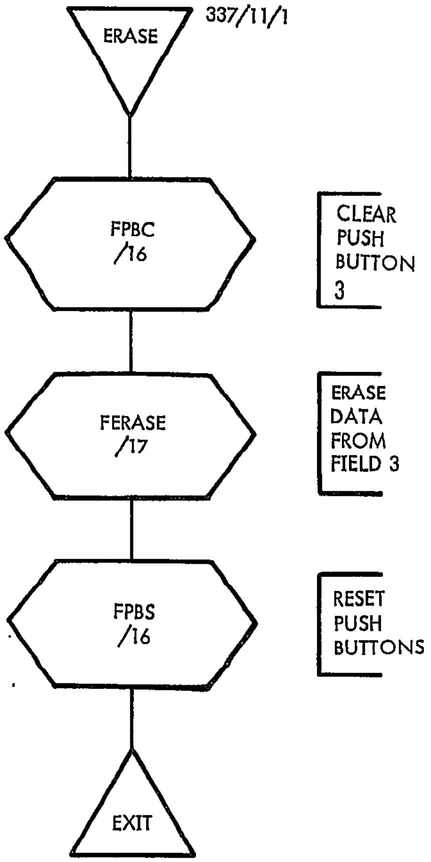

ROTAT
302/10/3
COMPONENT HAS FOUR ORIENTATIONS COMP = WIA
YES
TRRT
/15
COMPONENT HAS TWO ORIENTATIONS AND IS NOT AN INDUCTOR COMP = RESH
YES
RCRT
/15
COMPONENT IS AN INDUCTOR
YES
INRT
/15
EXIT
COMPONENT HAS ONE ORIENTATION

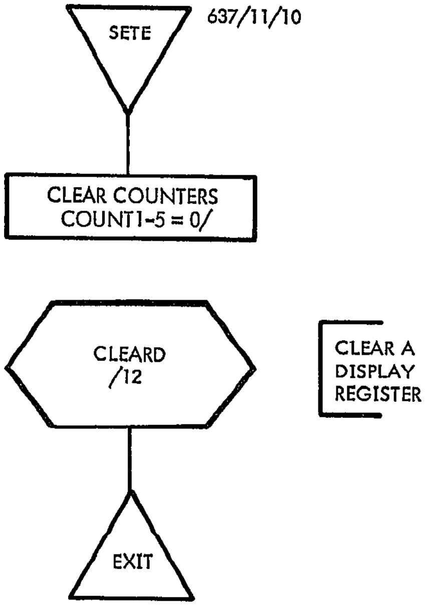

SET
405/10
INDIRECT COMPA=COMP
SEND COMPONENT SUBROUTINE START ADDRESS (COMP) INTO ADDRESS IN COMPA
PUSH BUTTON 9
NO
ERASE
/11
PLANT
/16
COPY DISPLAY REGISTERS INTO DATA FIELD
EXIT
EXIT

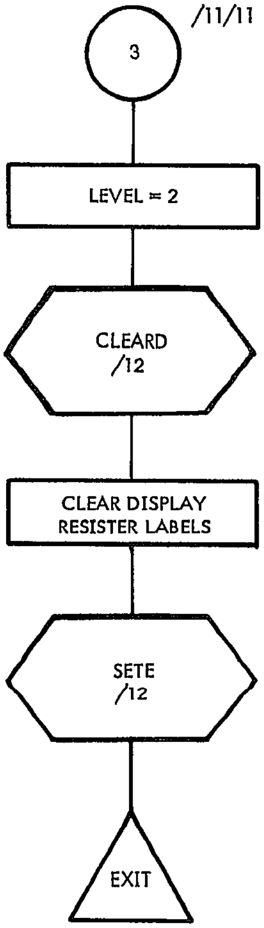

MIROR
612/10/3
COMPONENT IS A TRANSISTOR
NO
EXIT
COUNT
EVEN
ODD
COMP = COMP + 100
COMP = COMP -10
EXIT

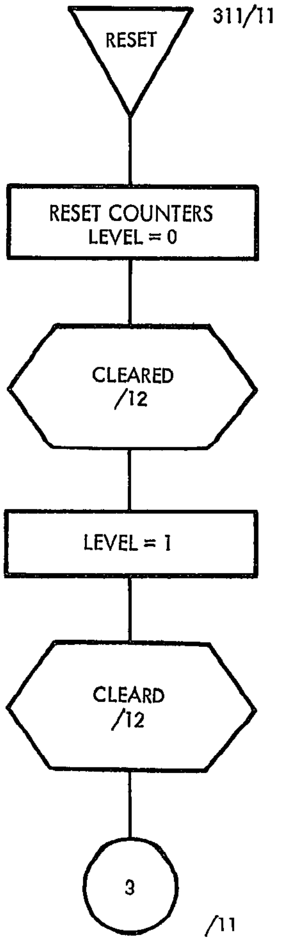

SETA
715/10
SETE
/11
RESUM
/9

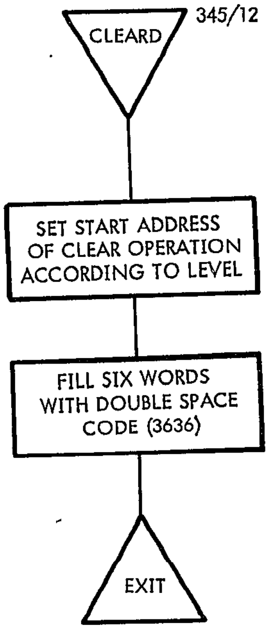

FORCE
720/10
PLANT
/16
RESUM
/9


ERASE
337/11/1
FPBC
/16
CLEAR
PUSH
BUTTON
3
FERASE
/17
ERASE
DATA
FROM
FIELD 3
FPBS
/16
RESET
PUSH
BUTTONS
EXIT

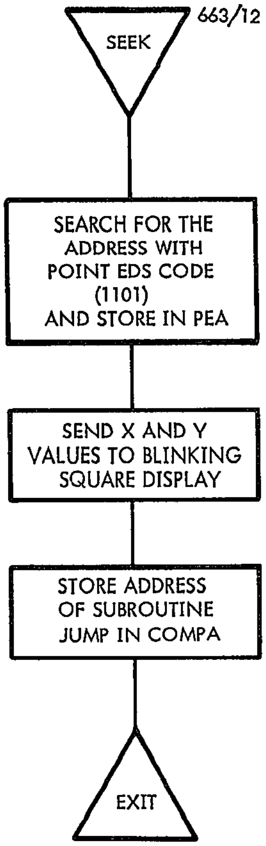

SETE
637/11/10
CLEAR COUNTERS
COUNT1-5 = 0/
CLEARD
/12
CLEAR A
DISPLAY
REGISTER
EXIT


3
/11/11
LEVEL = 2
CLEARD
/12
CLEAR DISPLAY
RESISTER LABELS
SETE
/12
EXIT


RESET
311/11
RESET COUNTERS
LEVEL = 0
CLEARED
/12
LEVEL = 1
CLEARD
/12
3
/11

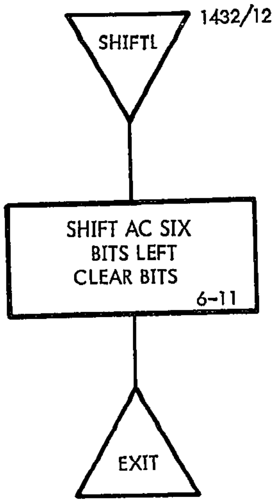

CLEARD
345/12
SET START ADDRESS OF CLEAR OPERATION ACCORDING TO LEVEL
FILL SIX WORDS WITH DOUBLE SPACE CODE (3636)
EXIT

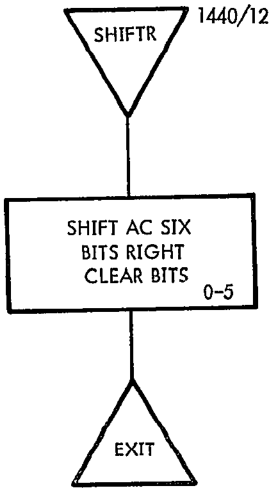
LEVEL	DISPLAY REGISTER
0	C1
1	D1
2	E1


SEEK
663/12
SEARCH FOR THE ADDRESS WITH POINT EDS CODE (1101) AND STORE IN PEA
SEND X AND Y VALUES TO BLINKING SQUARE DISPLAY
STORE ADDRESS OF SUBROUTINE JUMP IN COMPA
EXIT

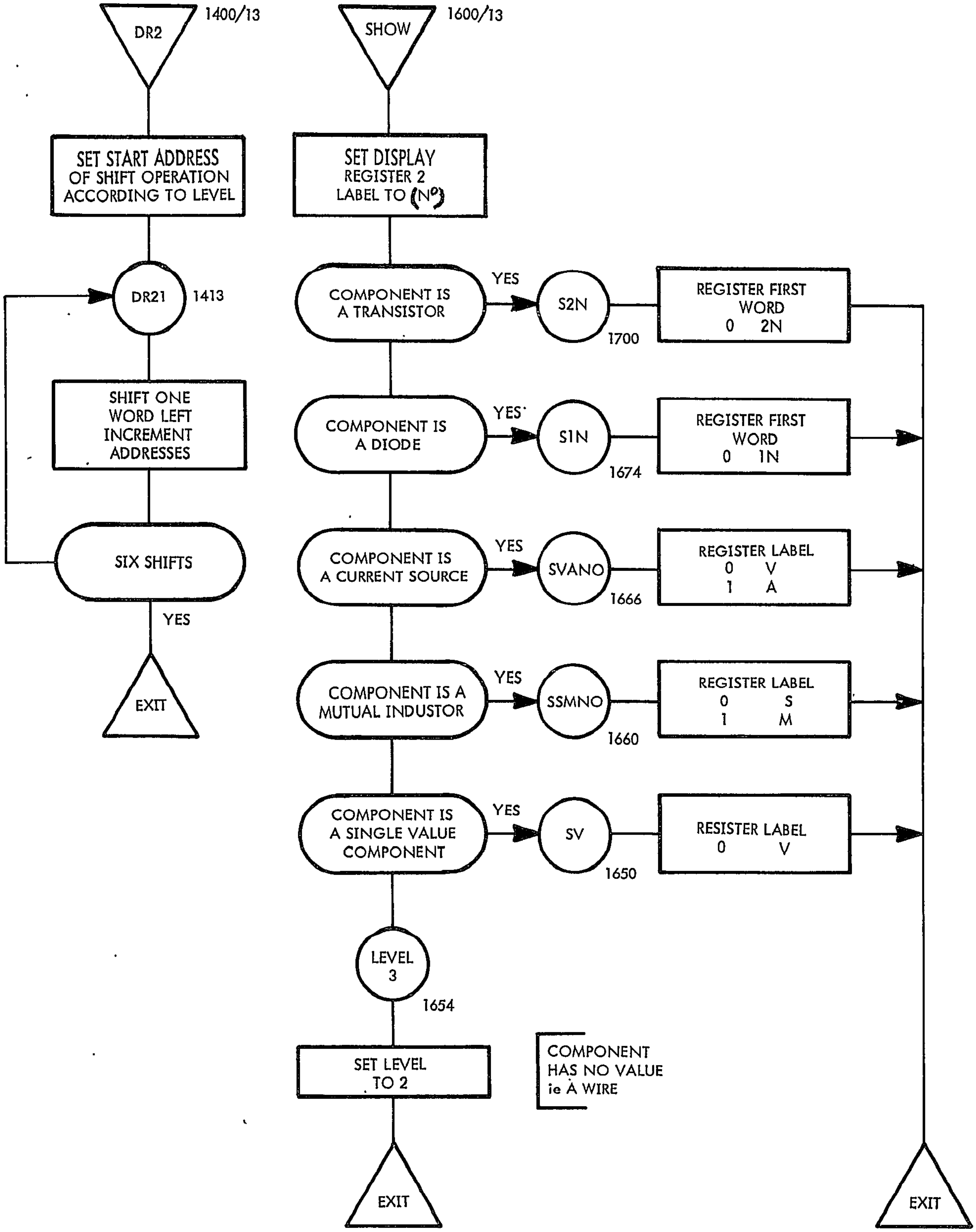
DISPLAY STOPS AFTER POINT EDS, IN A RANDOM POSITION A BACKWARDS SEARCH IS REQUIRED

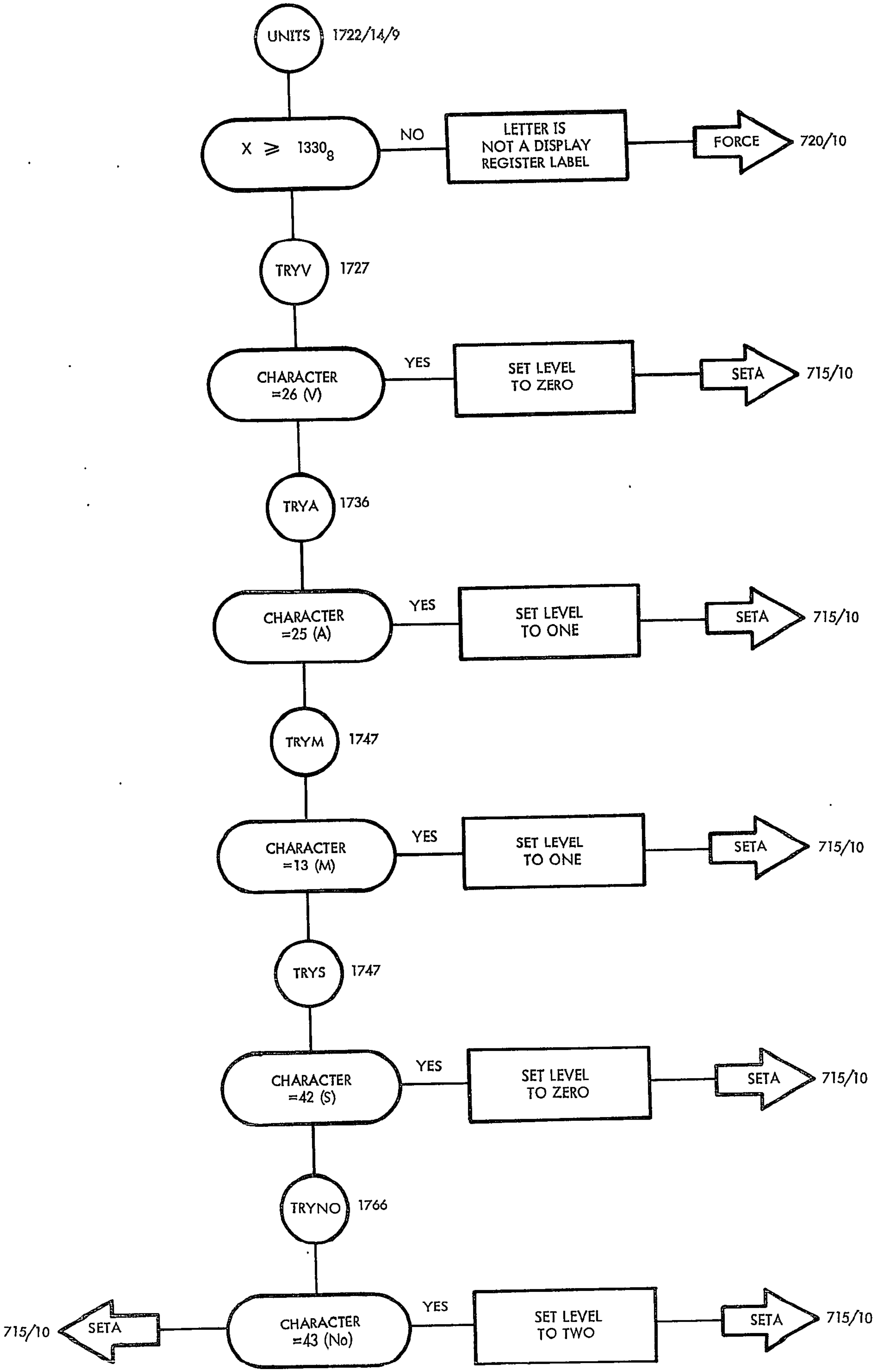
PRODUCES BLINKING SQUARE OVER PRESENT POSITION

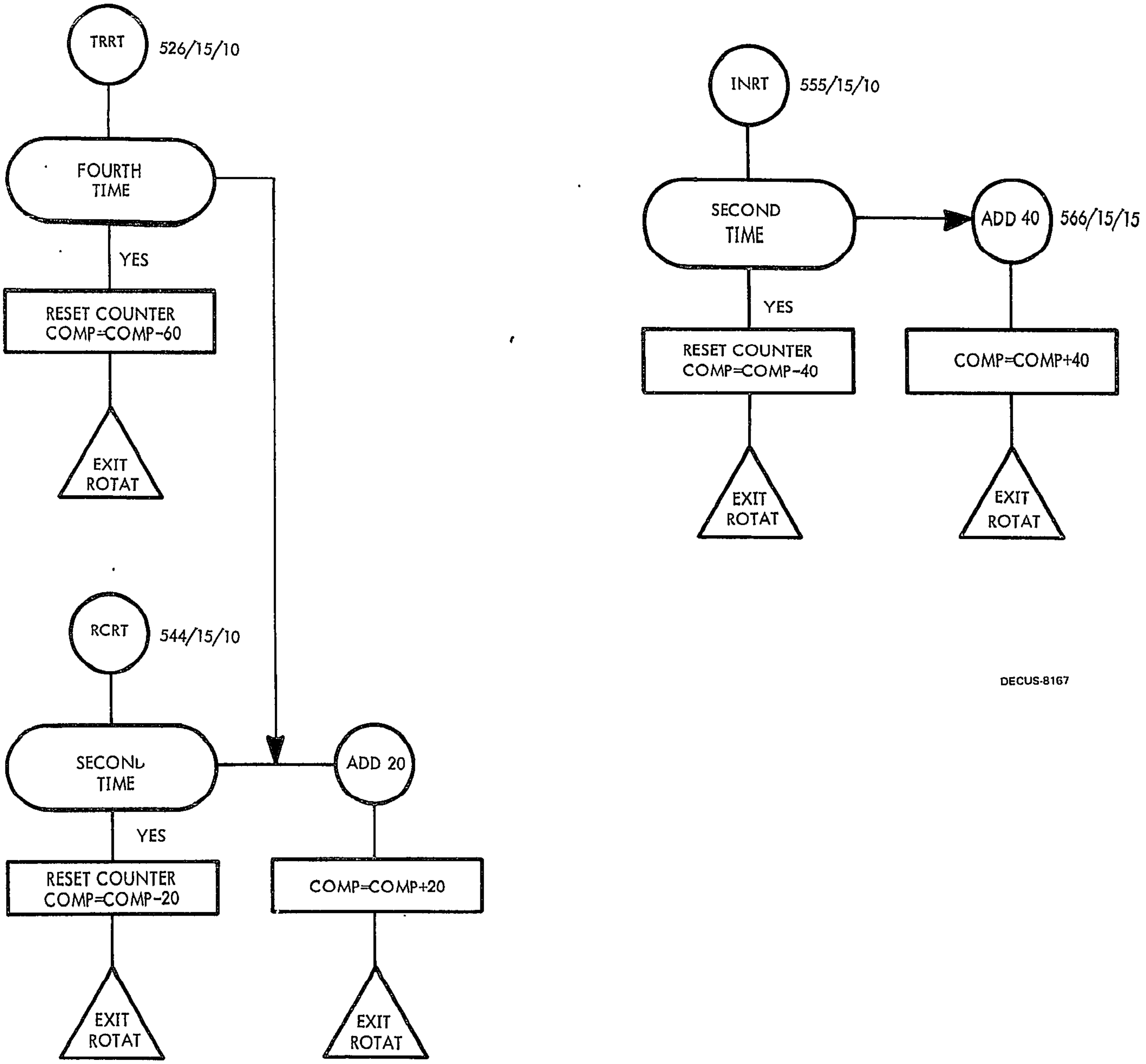


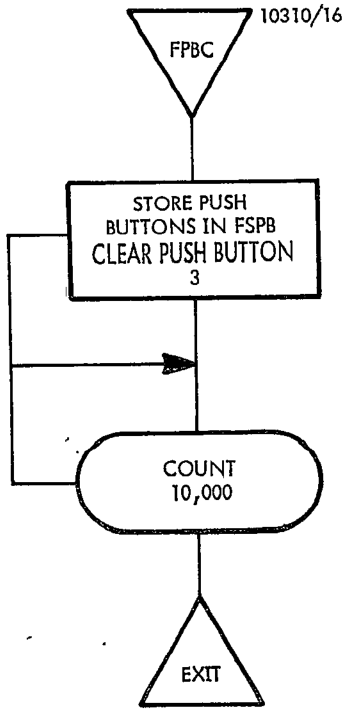


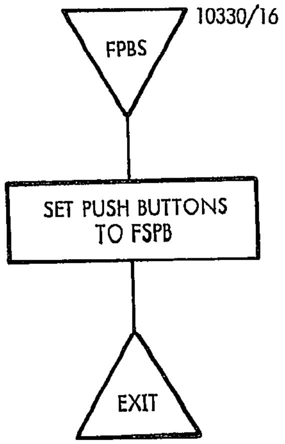
DECUS-8167

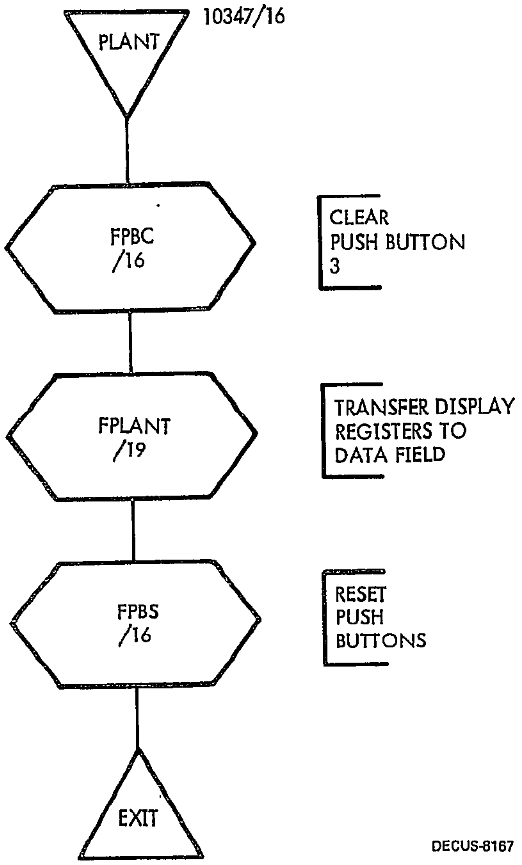

DR2
1400/13
SET START ADDRESS OF SHIFT OPERATION ACCORDING TO LEVEL
DR21
1413
SHIFT ONE WORD LEFT INCREMENT ADDRESSES
SIX SHIFTS
YES
EXIT
SHOW
1600/13
SET DISPLAY REGISTER 2 LABEL TO (N°)
COMPONENT IS A TRANSISTOR
YES
S2N
1700
REGISTER FIRST WORD
0 2N
COMPONENT IS A DIODE
YES
S1N
1674
REGISTER FIRST WORD
0 1N
COMPONENT IS A CURRENT SOURCE
YES
SVANO
1666
REGISTER LABEL
0 V
1 A
COMPONENT IS A MUTUAL INDUSTOR
YES
SSMNO
1660
REGISTER LABEL
0 S
1 M
COMPONENT IS A SINGLE VALUE COMPONENT
YES
SV
1650
RESISTER LABEL
0 V
LEVEL 3
1654
SET LEVEL TO 2
COMPONENT HAS NO VALUE ie A WIRE
EXIT
EXIT


UNITS
1722/14/9
X ⩾ 1330₈
NO
LETTER IS NOT A DISPLAY REGISTER LABEL
FORCE
720/10
TRYV
1727
CHARACTER =26 (V)
YES
SET LEVEL TO ZERO
SETA
715/10
TRYA
1736
CHARACTER =25 (A)
YES
SET LEVEL TO ONE
SETA
715/10
TRYM
1747
CHARACTER =13 (M)
YES
SET LEVEL TO ONE
SETA
715/10
TRYS
1747
CHARACTER =42 (S)
YES
SET LEVEL TO ZERO
SETA
715/10
TRYNO
1766
CHARACTER =43 (No)
YES
SET LEVEL TO TWO
SETA
715/10
715/10
SETA

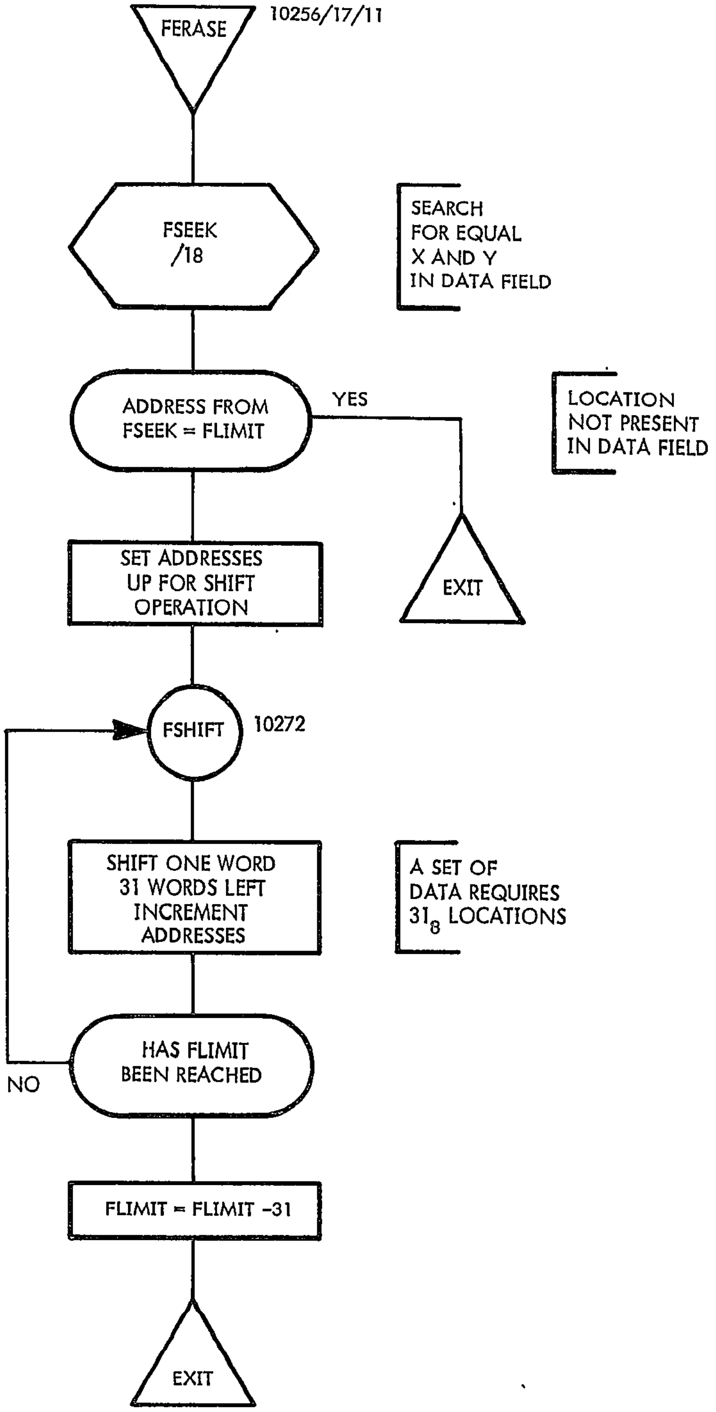

TRRT
526/15/10
FOURTH TIME
YES
RESET COUNTER COMP=COMP-60
EXIT ROTAT
RCRT
544/15/10
SECONL TIME
YES
RESET COUNTER COMP=COMP-20
EXIT ROTAT
ADD 20
COMP=COMP+20
EXIT ROTAT
INRT
555/15/10
SECOND TIME
YES
RESET COUNTER COMP=COMP-40
EXIT ROTAT
ADD 40
566/15/15
COMP=COMP+40
EXIT ROTAT
DECUS-8167

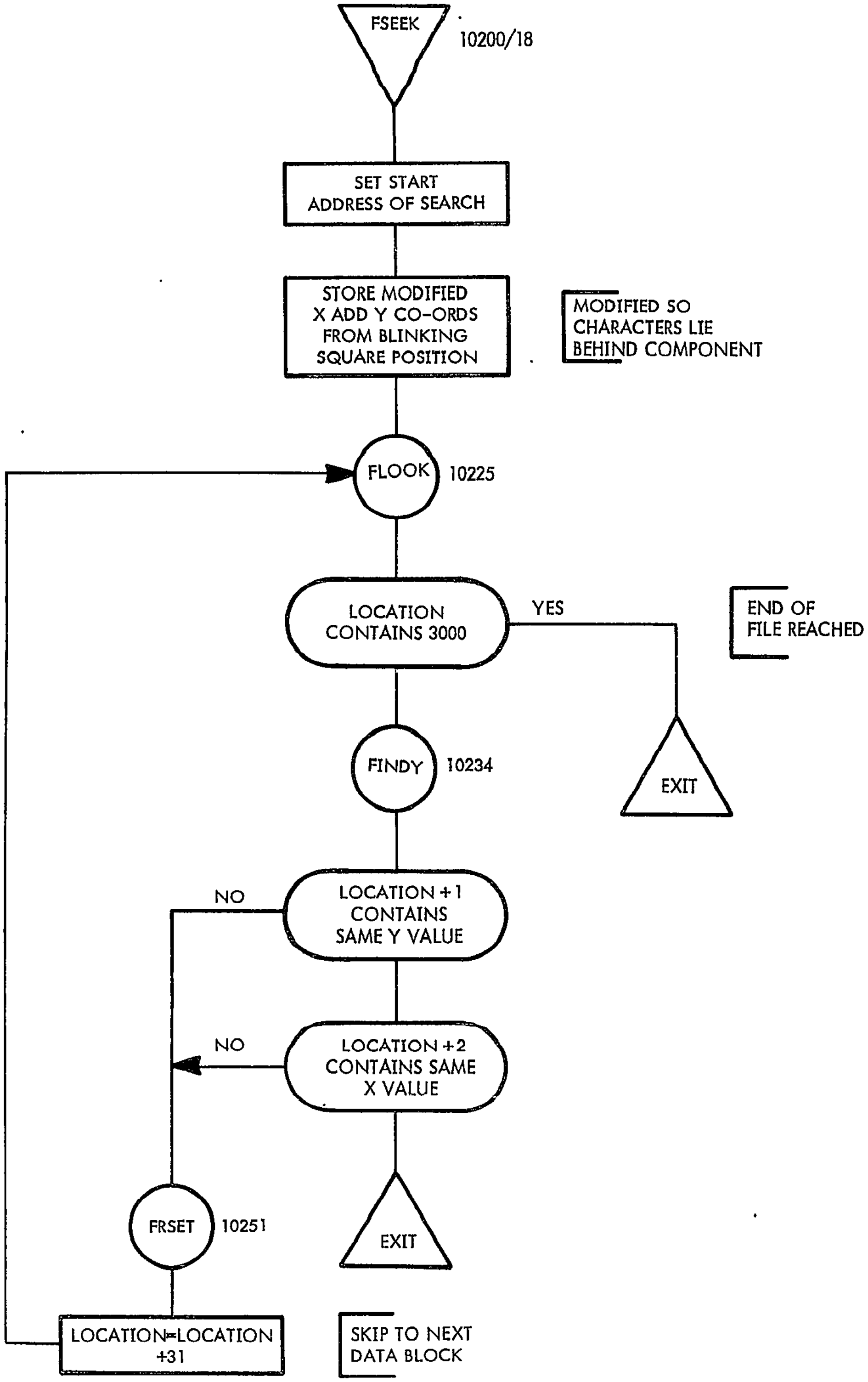

FPBC
10310/16
STORE PUSH BUTTONS IN FSPB
CLEAR PUSH BUTTON 3
COUNT 10,000
EXIT


FPBS
10330/16
SET PUSH BUTTONS TO FSPB
EXIT

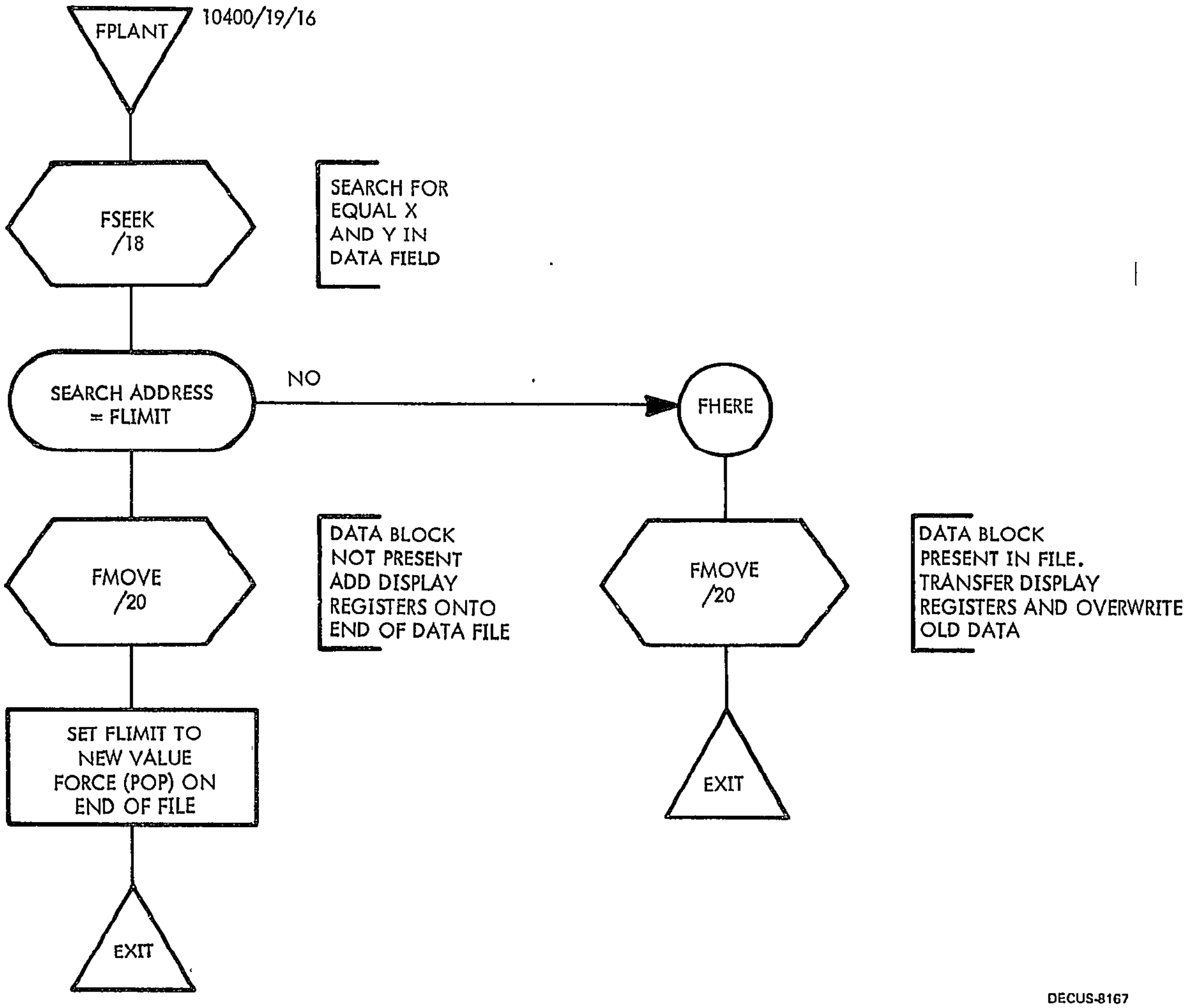

PLANT
10347/16
FPBC /16
CLEAR PUSH BUTTON 3
FPLANT /19
TRANSFER DISPLAY REGISTERS TO DATA FIELD
FPBS /16
RESET PUSH BUTTONS
EXIT

DECUS-8167

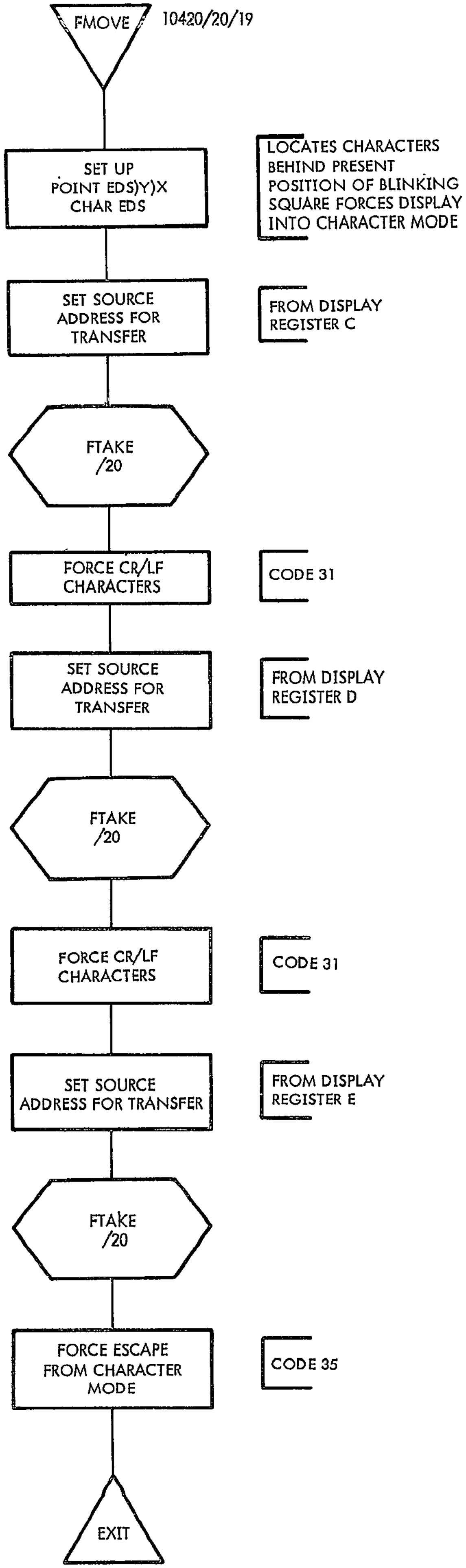

FERASE
10256/17/11
FSEEK
/18
SEARCH FOR EQUAL X AND Y IN DATA FIELD
ADDRESS FROM FSEEK = FLIMIT
YES
LOCATION NOT PRESENT IN DATA FIELD
EXIT
SET ADDRESSES UP FOR SHIFT OPERATION
FSHIFT
10272
SHIFT ONE WORD 31 WORDS LEFT INCREMENT ADDRESSES
A SET OF DATA REQUIRES 31_8 LOCATIONS
HAS FLIMIT BEEN REACHED
NO
FLIMIT = FLIMIT -31
EXIT

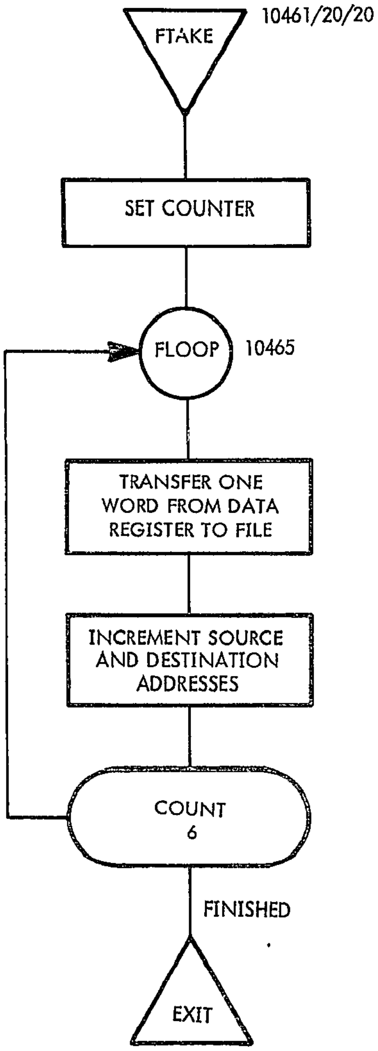

FSEEK
10200/18
SET START ADDRESS OF SEARCH
STORE MODIFIED X ADD Y CO-ORDS FROM BLINKING SQUARE POSITION
MODIFIED SO CHARACTERS LIE BEHIND COMPONENT
FLOOK
10225
LOCATION CONTAINS 3000
YES
END OF FILE REACHED
EXIT
FINDY
10234
NO
LOCATION +1 CONTAINS SAME Y VALUE
NO
LOCATION +2 CONTAINS SAME X VALUE
FRSET
10251
EXIT
LOCATION=LOCATION +31
SKIP TO NEXT DATA BLOCK

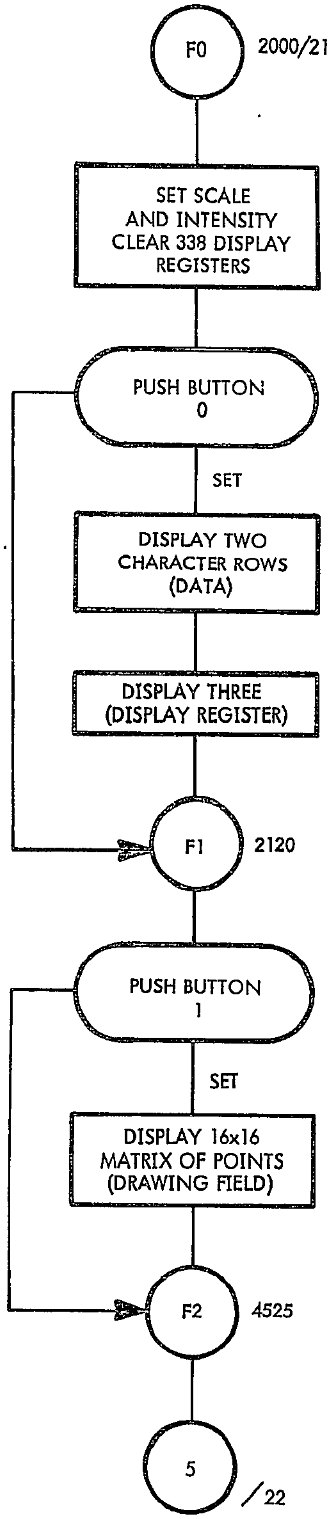
DECUS-8167


FPLANT
10400/19/16
FSEEK
/18
SEARCH FOR
EQUAL X
AND Y IN
DATA FIELD
SEARCH ADDRESS
= FLIMIT
NO
FHERE
FMOVE
/20
DATA BLOCK
NOT PRESENT
ADD DISPLAY
REGISTERS ONTO
END OF DATA FILE
FMOVE
/20
DATA BLOCK
PRESENT IN FILE.
TRANSFER DISPLAY
REGISTERS AND OVERWRITE
OLD DATA
SET FLIMIT TO
NEW VALUE
FORCE (POP) ON
END OF FILE
EXIT
EXIT

DECUS-8167


FMOVE
10420/20/19
SET UP
POINT EDS)Y)X
CHAR EDS
LOCATES CHARACTERS
BEHIND PRESENT
POSITION OF BLINKING
SQUARE FORCES DISPLAY
INTO CHARACTER MODE
SET SOURCE
ADDRESS FOR
TRANSFER
FROM DISPLAY
REGISTER C
FTAKE
/20
FORCE CR/LF
CHARACTERS
CODE 31
SET SOURCE
ADDRESS FOR
TRANSFER
FROM DISPLAY
REGISTER D
FTAKE
/20
FORCE CR/LF
CHARACTERS
CODE 31
SET SOURCE
ADDRESS FOR TRANSFER
FROM DISPLAY
REGISTER E
FTAKE
/20
FORCE ESCAPE
FROM CHARACTER
MODE
CODE 35
EXIT


FTAKE
10461/20/20
SET COUNTER
FLOOP
10465
TRANSFER ONE
WORD FROM DATA
REGISTER TO FILE
INCREMENT SOURCE
AND DESTINATION
ADDRESSES
COUNT
6
FINISHED
EXIT


F0
2000/21
SET SCALE AND INTENSITY CLEAR 338 DISPLAY REGISTERS
PUSH BUTTON 0
SET
DISPLAY TWO CHARACTER ROWS (DATA)
DISPLAY THREE (DISPLAY REGISTER)
F1
2120
PUSH BUTTON 1
SET
DISPLAY 16x16 MATRIX OF POINTS (DRAWING FIELD)
F2
4525
5
/22

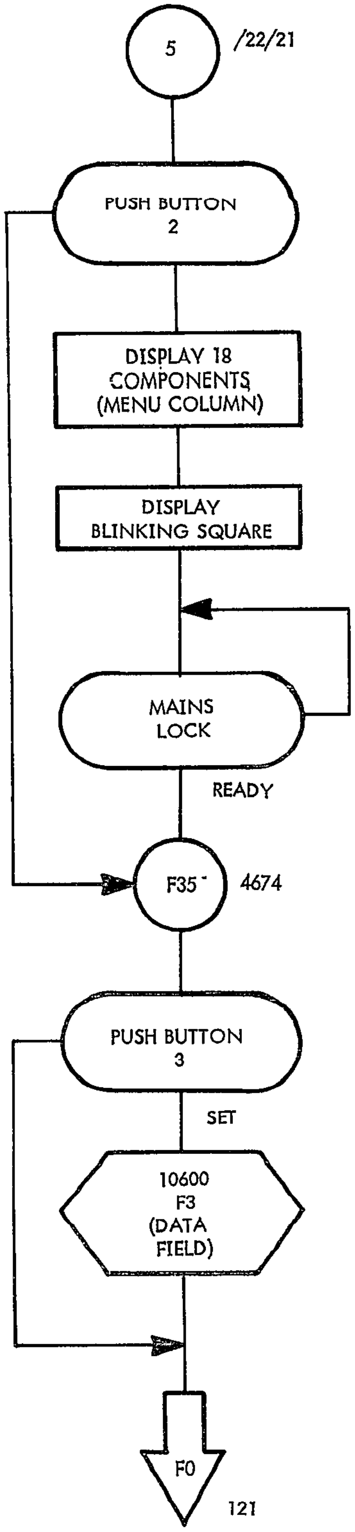
FORMAT

VERTICALLY SHIFTED BY CHANGING Y1 AND Y2.

```
POINT EDS)Y1)X)CHAR EDS) 3033)3000)3001) etc.
                         ␣+  ␣ 0 ␣  1
POINT EDS)Y2)X)CHAR EDS) 3034)3005)3006) etc.
                         ␣ -  ␣ 5 ␣ 6
```

```
POINT EDS)Y)X)C0)C1)C2)C3)C4)C5)C6)4135
                 )LABEL LPOF)DATA      )LPON ESCAPE
POINT EDS)Y)X)D0)D2)D3)D4)D5)D6)4135
                  AS ABOVE
POINT EDS)X)Y)E1)E2)E3 etc.
```

```
POINT EDS)X)Y)PJMP LPOB)COKE
               )ENTER SUBROUTINE)SUBROUTINE
               )W14 LIGHT PEN OFF) START ADDRESS
EXIT FROM A SUBROUTINE REQUIRES (POP)
FOR A DUMMY COMPONENT COKE = POP
FOR A HORIZONTAL RESISTOR COKE = RESH
```



```
POINT EDS)Y)X)PJMP SC2)RESH
                  )ENTER SUBROUTINE)HORIZONTAL
                  )SCALE 2             )RESISTOR
ETC.
```

```
POINT EDS)B1)B2)BICON)LPOF)SC8
             )B1 and B2 LOCATE ORIGIN
             )SWITCH BLINK ON, PEN OFF, SCALE 8
SVEC EDS  ) DATA          ) BKOF
ENTER SHORT VECTOR MODE, DRAW SQUARE
                 SWITCH BLINK OFF
ESCAPE WHEN MAINS PULSE IS PRESENT
```

```
ENTER SUBROUTINE F3,SCALE1, LIGHT PEN OFF
POINT EDS)Y)X)CHAR EDS)DISPLAY REGISTER.C)3131
                              DISPLAY REGISTER.D)3131
                              DISPLAY REGISTER.E)3535
POINT EDS) etc.
3000
ESCAPE FROM SUBROUTINE
```

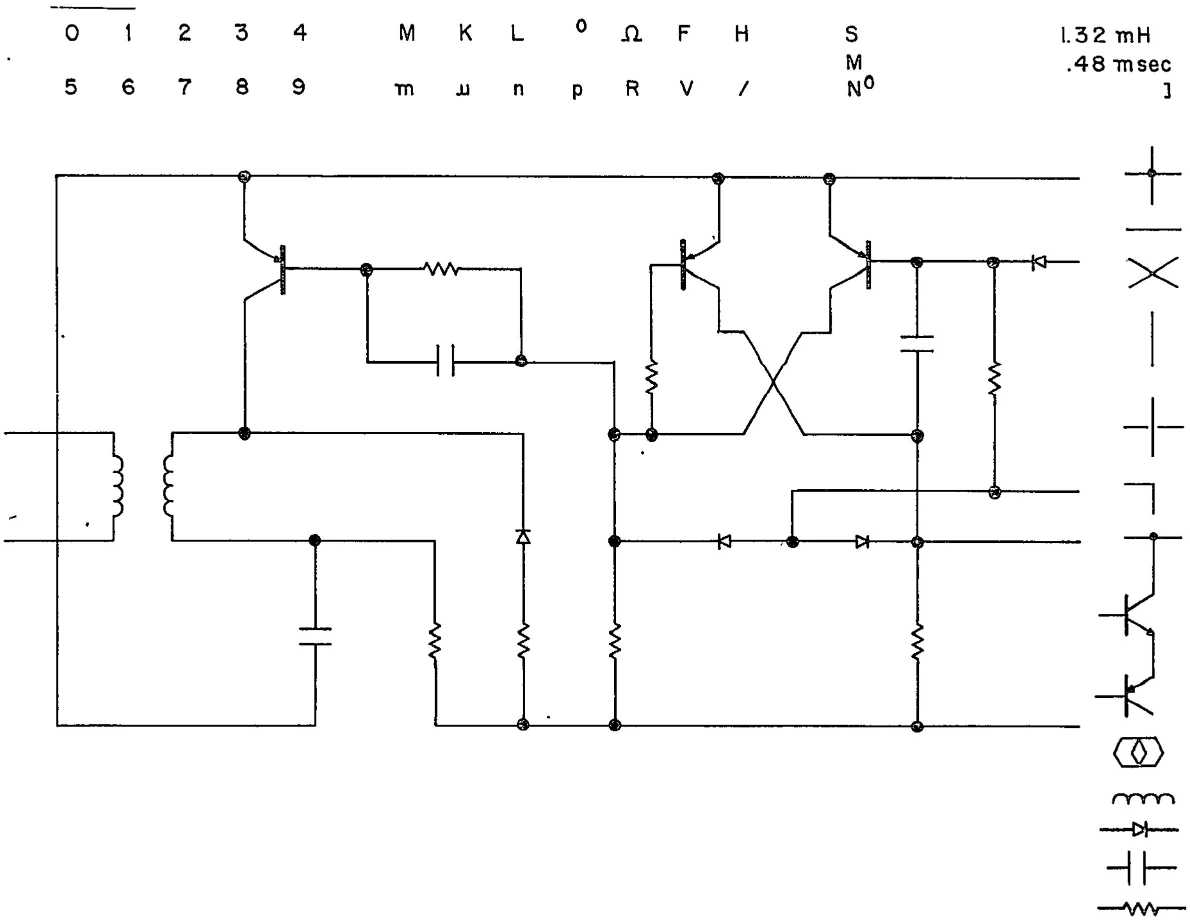
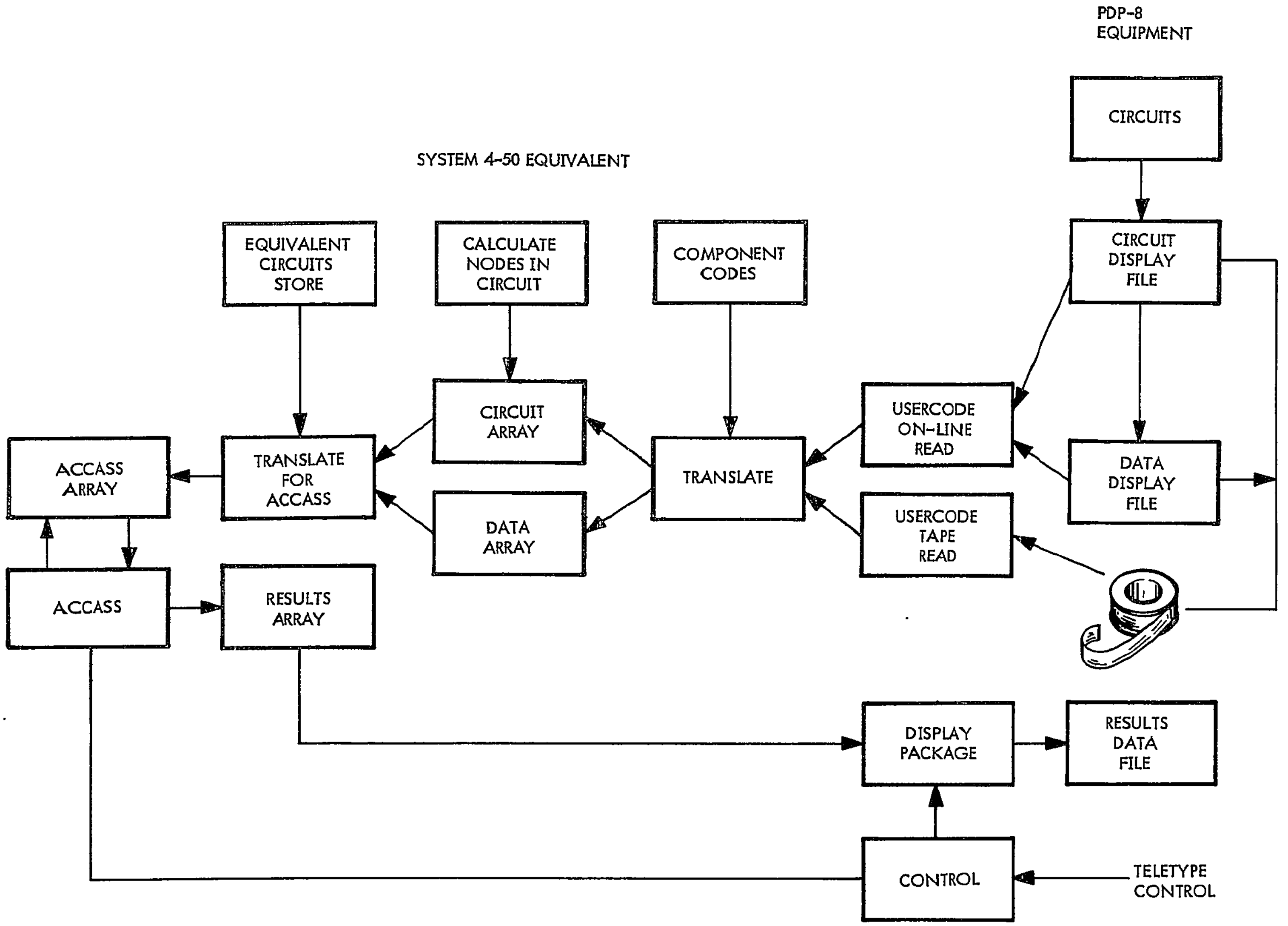


Figure 1.

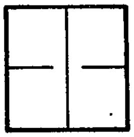
DEC PULSE AMPLIFIER W640
DRAWN WITH CIRCUITS
(Terminations and values can
now be labelled)



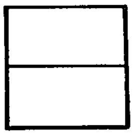
BLOCK DIAGRAM OF THE CIRCUIT SYSTEM

COMPONENT SUBROUTINES LIST 1

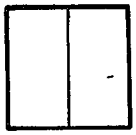
W1/1
5100
101111



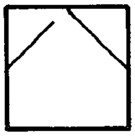
W10/2
5120
010101



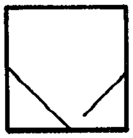
W9/3
5140
011010



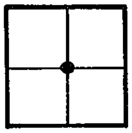
W13/4
5160
100101



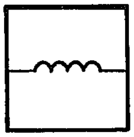
W12/5
5200
100101



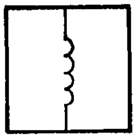
W0/6
5220
011111



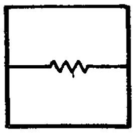
INDH/7
5240
100101



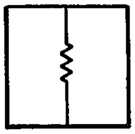
INDV/9
5300
101010



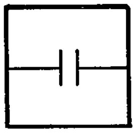
RESH/11
5340
100101



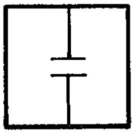
RESV/12
5360
101010



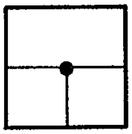
CAPH/13
5400
100101



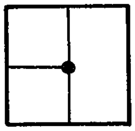
CAPV/14
5420
101010



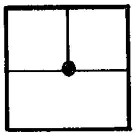
W1/15
5440
011101



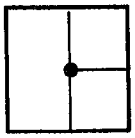
W2/16
5460
011110



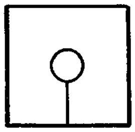
W3/17
5500
010111



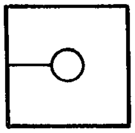
W4/18
5520
011011



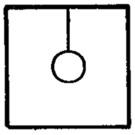
T1/19
5540
011000



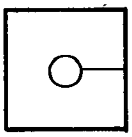
T2/20
5560
010100



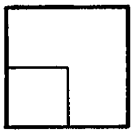
T3/21
5600
010010



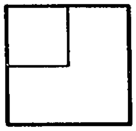
T4/22
5620
010001



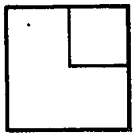
W5/23
5640
011100



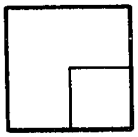
W6/24
5660
010110



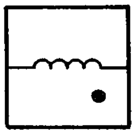
W7/25
5700
010011



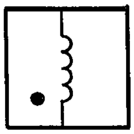
W8/26
5720
011001



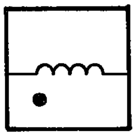
MUHR/27
5740
100101



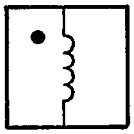
MUVB/28
5760
101010



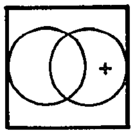
MUHL/29
6000
100101



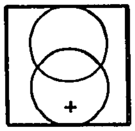
MUVT/30
6020
101010



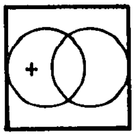
IHR/31
6040
100101



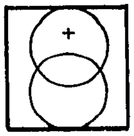
IHB/32
6060
101010



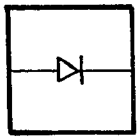
EHL/33
6100
100101



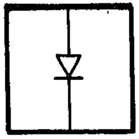
IHT/34
6120
101010



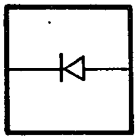
DIR/35
6140
100101



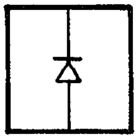
DIB/36
6160
101010



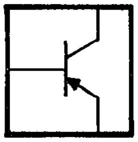
DIL/37
6200
100101



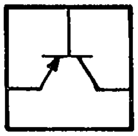
DIT/38
6220
101010



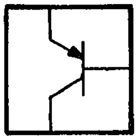
PEDBL/39
6240
101110



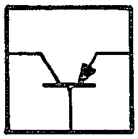
PELBU/40
6260
100111



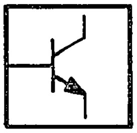
PEUBR/41
6300
101011



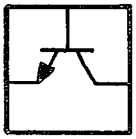
PERBD/42
6320
101101



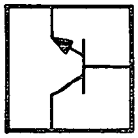
NEDBL/43
6340
101110



NELBU/44
6360
100111



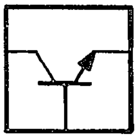
NEUBR/45
6400
101011



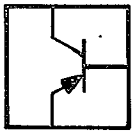
CODING

LABEL/CODE NUMBER
SUBROUTINE START ADDRESS
ARRAY E NUMBER

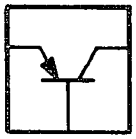
NERBD/46
6420
101101



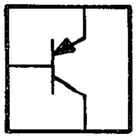
PEDBR/47
6440
101011



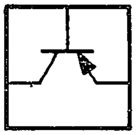
PELBD/48
6460
101101



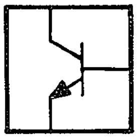
PEUBL/49
6500
101110



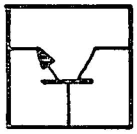
PERBU/50
6520
100111



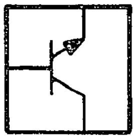
NEDBR/51
6540
101011



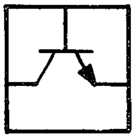
NELBD/52
6560
101101



NEUBL/53
6600
101110



NERBU/54
6620
100111



COKE/55
6640
000000

LISTING OF THE CHARACTER GENERATOR LIST 2

OCTAL CALLING CODE	CHARACTER	SYMBOL
0	ZERO	0
1	ONE	1
2	TWO	2
3	THREE	3
4	FOUR	4
5	FIVE	5
6	SIX	6
7	SEVEN	7
10	EIGHT	8
11	NINE	9
12	DPOINT	.
13	MEGA	M
14	KILO	K
15	UNIT	I
16	MILLI	m
17	MICRO	u
20	NAND	n
21	PICO	p
22	OHMS	
23	FARAD	F
24	HENRY	H
25	AMP	A
26	VOLTS	V
27	SLASH	/
30	SKIPI	
31	SKIP2	
32	SKIP3	
33	PLUS	+
34	MINUS	-
35	6040	escape
36	BLANK	
37	NANCY	N
40	4040	LPOF
41	4060	LPON
42	SELF	S
43	LABEL	N°
44	DEGREE	o

LISTING OF THE CHARACTER GENERATOR LIST 3

```
51ØØ = ØØØ1 = ØØØ1
512Ø = ØØØ2 = ØØØ2
514Ø = ØØØ3 = ØØØ3
516Ø = ØØØ4 = ØØØ4
52ØØ = ØØØ5 = ØØØ5
522Ø = ØØØ6 = ØØØ6
524Ø = ØØØ7 = ØØØ7
526Ø = ØØ1Ø = ØØØ8
53ØØ = ØØ11 = ØØØ9
532Ø = ØØ12 = ØØ1Ø
534Ø = ØØ13 = ØØ11
536Ø = ØØ14 = ØØ12
54ØØ = ØØ15 = ØØ13
542Ø = ØØ16 = ØØ14
544Ø = ØØ17 = ØØ15
546Ø = ØØ2Ø = ØØ16
55ØØ = ØØ21 = ØØ17
552Ø = ØØ22 = ØØ18
554Ø = ØØ23 = ØØ19
556Ø = ØØ24 = ØØ2Ø
56ØØ = ØØ25 = ØØ21
562Ø = ØØ26 = ØØ22
564Ø = ØØ27 = ØØ23
566Ø = ØØ3Ø = ØØ24
57ØØ = ØØ31 = ØØ25
572Ø = ØØ32 = ØØ26
574Ø = ØØ33 = ØØ27
576Ø = ØØ34 = ØØ28
6ØØØ = ØØ35 = ØØ29
6Ø2Ø = ØØ36 = ØØ3Ø
6Ø4Ø = ØØ37 = ØØ31
6Ø6Ø = ØØ4Ø = ØØ32
61ØØ = ØØ41 = ØØ33
612Ø = ØØ42 = ØØ34
614Ø = ØØ43 = ØØ35
616Ø = ØØ44 = ØØ36
62ØØ = ØØ45 = ØØ37
622Ø = ØØ46 = ØØ38
624Ø = ØØ47 = ØØ39
626Ø = ØØ5Ø = ØØ4Ø
63ØØ = ØØ51 = ØØ41
632Ø = ØØ52 = ØØ42
634Ø = ØØ53 = ØØ43
636Ø = ØØ54 = ØØ44
64ØØ = ØØ55 = ØØ45
642Ø = ØØ56 = ØØ46
644Ø = ØØ57 = ØØ47
646Ø = ØØ6Ø = ØØ48
65ØØ = ØØ61 = ØØ49
652Ø = ØØ62 = ØØ5Ø
654Ø = ØØ63 = ØØ51
656Ø = ØØ64 = ØØ52
66ØØ = ØØ65 = ØØ53
662Ø = ØØ66 = ØØ54
664Ø = ØØ67 = ØØ55
```

SUBROUTINE START ADDRESS = OCTAL CODE =
ARRAY E NUMBER COMPONENT CODE