
Masters Theses

Student Theses and Dissertations

1971

A software system for interactive man-machine design and analysis of electronic circuits

Leo William Midden

Follow this and additional works at: https://scholarsmine.mst.edu/masters_theses



Part of the [Electrical and Computer Engineering Commons](#)

Department:

Recommended Citation

Midden, Leo William, "A software system for interactive man-machine design and analysis of electronic circuits" (1971). *Masters Theses*. 3563.

https://scholarsmine.mst.edu/masters_theses/3563

This thesis is brought to you by Scholars' Mine, a service of the Missouri S&T Library and Learning Resources. This work is protected by U. S. Copyright Law. Unauthorized use including reproduction for redistribution requires the permission of the copyright holder. For more information, please contact scholarsmine@mst.edu.

A SOFTWARE SYSTEM FOR INTERACTIVE MAN-MACHINE
DESIGN AND ANALYSIS OF ELECTRONIC CIRCUITS

BY

LEO WILLIAM MIDDEN, 1948-

A THESIS

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI-ROLLA

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING.

1971

Approved by

Robert C. Peirce (Advisor)

James H. Tracy

Allen R. Koch

ABSTRACT

The development of an interactive computer-aided circuit analysis program is presented. A graphics terminal is used as an I/O device while an SCC 650 mini-computer is used as peripheral processor. The central processor is an IBM 360/50.

Formulation, capabilities and use of the system are discussed in detail. Close attention is given to user control, making the system both useful and versatile. Both hard and soft copies of all plots and tabular data are available. Provisions are made for error corrections and changes at each step of the input. Parameter values can be altered before and after each analysis.

Additionally, several existing interactive man-machine circuit analysis systems are described.

ACKNOWLEDGEMENTS

The author would like to thank Dr. Robert Peirson for his guidance and patience during the development of this project. Wayne Omohundro is thanked for his assembler programming assistance and his criticisms.

The author would also like to express his sincere appreciation to the UMR Computer Center Staff and the Computer Operators without whose help and advice this project could not have been completed.

TABLE OF CONTENTS

	Page
ABSTRACT	ii
ACKNOWLEDGEMENTS	iii
LIST OF ILLUSTRATIONS.	vi
LIST OF TABLES	vii
I. INTRODUCTION.	1
II. ON-LINE COMPUTER-AIDED DESIGN	3
A. MIT's On-Line Circuit Analysis (OLCA)	4
B. Circuit Oriented Interactive Fortran (COIF)	8
C. PIXIE	10
D. Other On-line CAD Systems	14
III. UMR'S SYSTEMS ORGANIZATION.	17
A. Hardware.	17
B. Software.	19
IV. SYSTEMS DEVELOPMENT	22
A. Language.	22
B. Core Storage Requirements	23
C. Man-Machine Communications.	23
D. Component Number System	27
E. Standard Branch	29
1. Circuits	29
2. Notation	32
3. Equivalent Circuits.	33
F. Circuits.	33
G. CTD	36

	Page
H. LCAP.	37
V. A USER'S MANUAL	39
A. Electrical Connections.	39
B. Program Loading and Operation	39
C. Analysis Control.	41
D. DC Analysis	42
E. AC and TRansient Analysis	42
F. Frequency or Time Step.	44
G. Modifications	45
H. Parameters.	45
1. Identification	45
2. Values	48
3. Changes.	48
I. Analysis.	50
1. Auto-Save Facility	50
2. Plot	51
J. Messages and Codes.	51
BIBLIOGRAPHY	59
VITA	61
APPENDIX A. A User's Flow Chart for GICNA	62
APPENDIX B. A User's Guide for Rhine's Circuits	72

LIST OF ILLUSTRATIONS

Figures	Page
1. The Operating Environment of OLCA	5
2. Information Flow for OLCA	6
3. COIF Circuit Specification.	12
4. Form of COIF Network Branch	13
5. PIXIE System Data Paths	15
6. Organization of System's Hardware	18
7. Information Flow.	20
8. Communication Languages Between Man and Machine	25
9. CIRCUITS Network.	28
10. Standard Branch for GICNA	30
11. Card Format for CIRCUITS' Components.	31
12. Equivalent Circuits for the Transistors	34
13. Equivalent Circuits for the Diodes.	35
14. Circuit Parameters.	46
15. Labeling Differences Between CIRCUITS and GICNA	47
16. Card Information.	49
17. An Example Circuit for CIRCUITS	75

LIST OF TABLES

Table	Page
I. COIF Circuit Primitives	9
II. COIF Output Primitives.	11
III. Error Messages.	52
IV. Condition Codes	54
V. Change Codes.	58
VI. Standard Elements for CIRCUITS.	73
VII. List of Function for CIRCUITS	78
VIII. Error Messages for CIRCUITS	79

I. INTRODUCTION

The need for an on-line interactive man-machine circuit analysis program has long existed at the University of Missouri-Rolla. All existing circuit analysis programs run in a batch mode environment. The time between formulation of the problem and observation of the results is often several days. This long delay causes one to lose sight of his original objective and thus hinders the learning and/or design process. An effective solution is the use of an on-line analysis program which would allow one to observe the results seconds after entering the problem.

As a means towards this end, a project was initiated to develop a system which would allow the analysis of a circuit from a remote, on-line terminal given the circuit schematic and element values. The system is composed of four parts. The first part, completed under another project, consists of a program called CIRCUITS which allows the drawing of a schematic diagram of an electronic circuit on the screen of the graphics terminal. CIRCUITS stores the network topology in the memory of a mini-computer. CTD, the second part, retrieves the network topology from memory and transmits it over a telephone line to the third part of the system, GICNA. CTD then proceeds to a wait state where it handles all the I/O communications for the graphics terminal. GICNA formats the topology information, requests the element values from the user through the graphics terminal and then calls for the analysis program, LCAP. LCAP, a modified version of ECAP,²⁴ performs the analysis of the circuit and returns a plot to the graphics terminal. GICNA provides the tabular output, selected node voltage and branch currents, upon request. If a change is desired in the circuit, GICNA makes the

change and recalls LCAP.

This system permits a fast and efficient analysis of all types of DC, AC, and TRansient circuits. A plot output, soft or hard copy, can be obtained with an AC or TR analysis. Tabular data output can be obtained with all three types.

The system converts batch processing to a display-type conversational operation allowing fast turn-around time and thus enhances analysis and design of electronic circuits.

II. ON-LINE COMPUTER AIDED DESIGN

When designing and analyzing an electronic circuit it is often difficult to implement a circuit with hardware components for an experimental evaluation of a design.¹ The components are usually expensive and require a considerable amount of time to assemble. Improper connection can damage components and delay further experimentation. Even when a working hardware model is complete, the response may not be what was expected or desired. The result is considerable expense and delay.

One approach to the problem is to formulate a numerical model for experimentation. This eliminates the costly components yet requires considerable time to perform all of the model's numerical computation.

Computer-Aided Design (CAD) proved to be an efficient approach. CAD enabled a design to be evaluated numerically, thus reducing total design/evaluation time. This evaluation ran in a batch mode environment and provided computational speed, rigor in numerical analysis, the ability to perform repetitive operations and a methodical way of presenting the results.² However the complimentary characteristics were still processed by the designer. Invention, selection of potential schemes, adaptation to new problems and experience are the characteristics of the designer. CAD is a utilization of computer characteristics to implement those of the designer. The computer should be readily available to provide a rapid analysis of the designer concepts; however, the batch mode does not provide this type of response.

To provide this response an interactive designer-computer system was designed. An on-line circuit design and analysis system which utilizes a graphics terminal for output and graphic's tool and keyboard for input were developed.^{3,4} In the next sections representative on-line

graphic systems are discussed. The first, OCLA, uses a graphic's tool for input of the circuit. The second, COIF, uses a word description for the circuit input. PIXIE, the third, uses an up-down compiler technique.

Interactive CAD has, of course, proved to be an asset to education.^{5,6} It provides the student with immediate answers to design and analysis problems, thus enhancing and promoting the learning process. For more information on CAD and interactive CAD in design and education consult: Proceedings of the Purdue 1971 Symposium on Applications of Computers to Electrical Engineering Education.

A. MIT's On-Line Circuit Analysis (OLCA)

The operating system for On-Line Circuit Analysis (OLCA)⁷ is summarized in Figure 1. The GRAPHIC 1 console system contains a cathode ray tube, light pen, teletypewriter and a PDP 5 mini-computer with a local core memory. The mini-computer permits the GRAPHIC 1 system to operate on a stand alone basis or in conjunction with a central facility via a data link. The central computing facility is an IBM 7094 machine with various I/O equipment.

OLCA is basically a group of programs dividable into three parts: display-controlling, main handling, and analyzing. Part A, display controlling, is written in the GRIN 1 language and is executed on the PDP 5 satellite processor. Part B, main handling, and Part C, analyzing, are written in GRIN 1 and Fortran respectively. They are used on the central processor. Figure 2 displays the flow of information for the OLCA system.

An analysis session begins by loading Part A. Across the top of

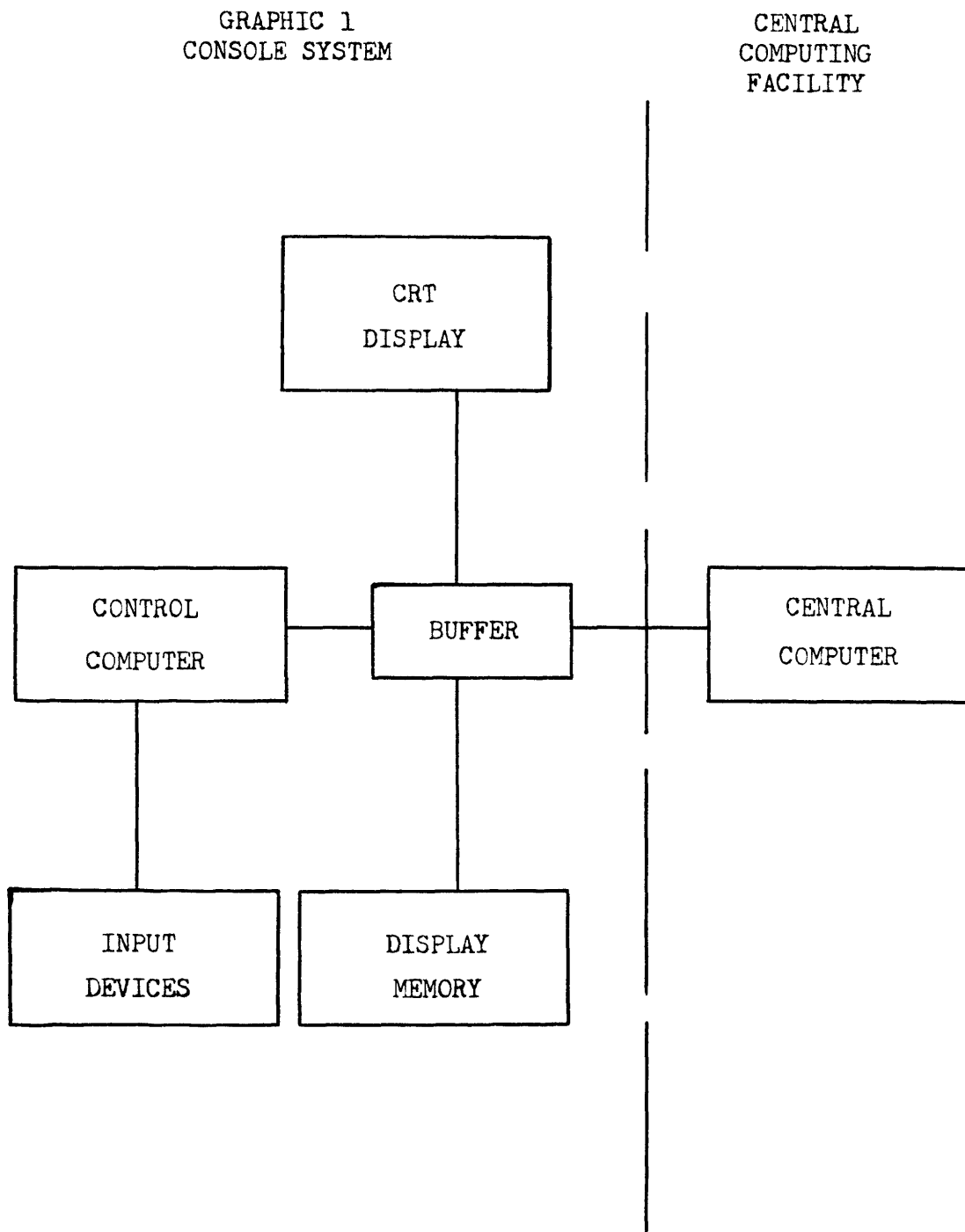


FIGURE 1

The Operating Environment of OLCA³

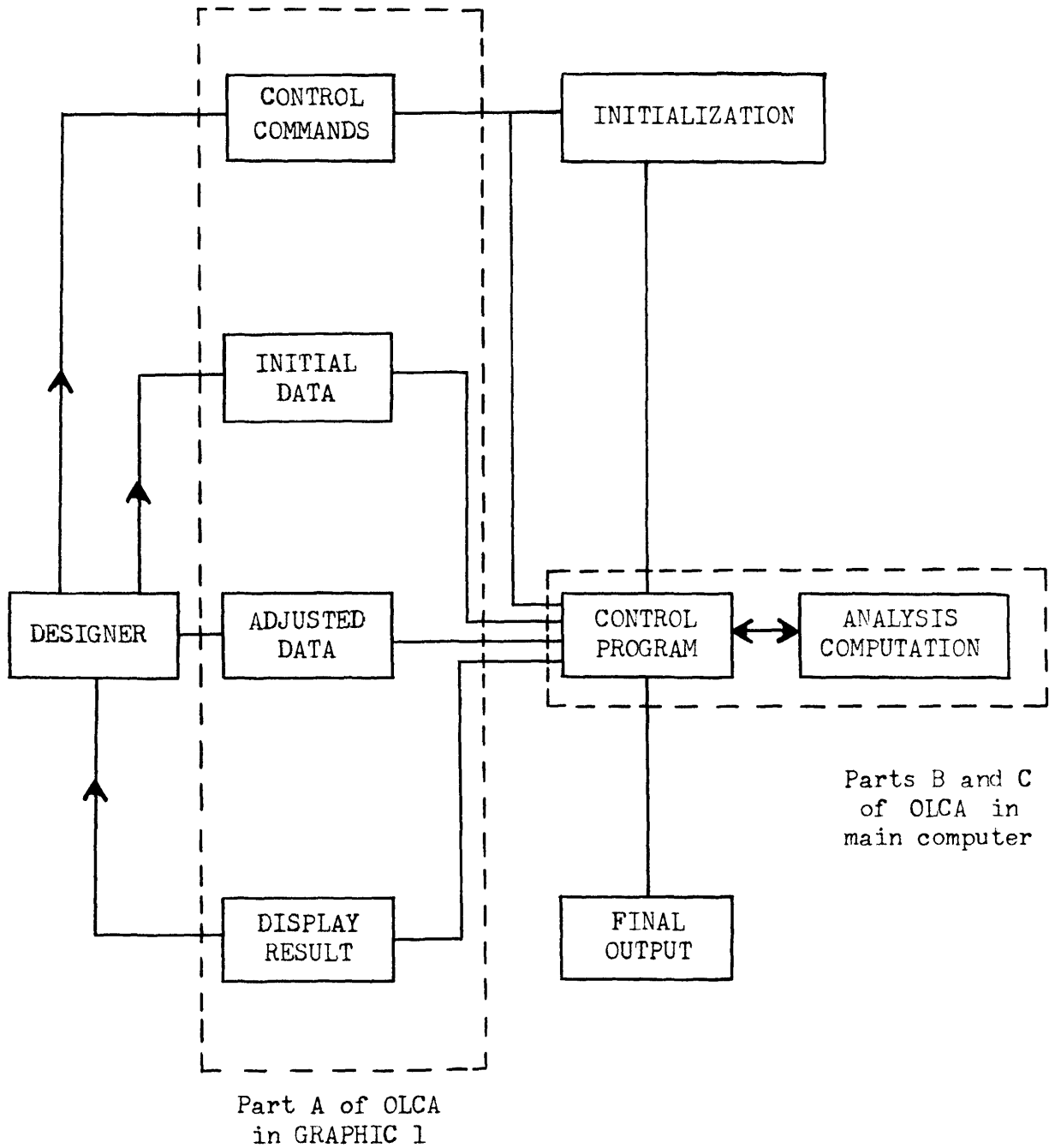


FIGURE 2

Information Flow for OLCA³

the CRT control words such as SERIES, DRAW, DONE, DELETE, VALUE, FREQUENCY, etc., then appear. Displayed across the screen bottom is a library of circuit components symbols. The middle portion of the screen contains the circuit being developed. The operator may point his light pen at a symbol or word and thus specify what action he wishes the computer to perform. When a symbol is indicated, the display is then changed to various questions which quiz the operator concerning the proper connection. If the symbol for a resistor is selected, the operator is quizzed as to which node it is connected to and the orientation desired. The resistor is then inserted and the complete circuit is displayed. When a word is indicated, the display shifts to service and questions the request. When the correct control word, VALUE, is specified, the keyboard can be used to input parameter values. All operations are checked continuously and the operator is informed when he makes an error.

When the circuit specification is complete, the word DONE is selected. The operator is then confronted with nine control commands including: analyze circuit and store results, analyze circuit and print results, analyze circuit and plot results, produce hardcopy, return to main display, etc. At this point Part A conveys the circuit data and the indicated function to the main control program. Part B receives and formats the data for the circuit analysis program. Part B then proceeds to execute the indicated control word.

Analysis is accomplished by the use of the frequency analysis program HYBRID. HYBRID uses matrix techniques to perform the analysis. After analysis, Part B transmits the result to the Graphic 1 terminal or produces the hard copy.

The above sequence of events is repeated until the desired results are obtained. By directing the operator to perform all actions, the system relieves him of having to memorize lengthy and rigid data formats. The operator can proceed through the analysis and design without interruptions.

B. Circuit Oriented Interactive Fortran (COIF)

The Politecnice di Milano, Milano, Italy has developed an on-line CAD system for design and analysis.² The system's hardware consists of a UNIVAC 1108 central processor. The satellite processor is a PDP-8 which is connected to a DEC 338 graphics display terminal.

The system software consists of a monitor, a monitor language and a circuit oriented graphic language. The monitor, INGRAM (INteractive GRaphics Monitor) contains all the program interpreters. IMOL (INteractive MOnitor Language) performs the overall control functions of the software systems. COIF (Circuit Oriented Interactive Fortran) describes, generates, and manipulates the circuit parameters.

The COIF language is an extension of Fortran. Circuit oriented functions were added to the list of Fortran primitives to provide for circuit graphic variables (CGV's). A partial listing of these new primitives is given in Table 1. The primitives appear on the right hand side of an equal sign, while the CGV's appear on the left hand side (e.g. N1=NODE(30,40)). The CGV, N1, is assigned a string of commands which activates the structure-handling and display routines. The CGV's must be declared at the beginning of the program by the specification primitive, CIRCUIT. Elements can be changed or deleted by use of the appropriate specifications primitive. The element values are specified

TABLE I

COIF CIRCUIT PRIMITIVES

<u>PRIMITIVE</u>	<u>PURPOSE</u>
NODE(X,Y)	Generates commands for creating a node with x,y coordinates.
INODE(X,Y)	Generates commands for creating an internal node with x,y coordinates.
GROUND(N)	Generates commands for creating a ground symbol at node N.
RESIS(N1,N2,B)	Generates commands for creating a resistor between nodes N1 and N2 on branch B.
CAPAC(N1,N2,B)	Same as RESIS for a capacitor.
INDUC(N1,N2,B)	Same as RESIS for an inductor.
INGEN(T,NS,NA,B)	Generates commands for creating an independent current (if T=A) or voltage (if T=V) generator between nodes NS and NA (oriented from NS to NA) on branch B.
TRASIS(T,NE,NB,BI,NC,BO)	Generates orders for creating a common-emitter transistor of type T. Node NE is connected to the emitter, node NB to the base and node NC to the collector. The base-emitter branch is on branch BI. the collector-emitter branch on branch BO.
CIRCUIT(a,.....z)	Declares the variables a,.....z as CGV's.
CLEAR	Erases the data structure and the display schematic.
DELETE(a,i)	Deletes the instance i of CGV a.
ERASE(a)	Deletes all the instances of the CGV a.

by the use of the COIF command

INPVAL(a) list

where "a" is the statement number of a Fortran statement and list is the names of the CGV's. Output is specified by the output primitives given in Table II.

A set of COIF commands and resulting circuit are in Figure 3. The output will be the current through R1. N4 is specified as an internal node. It is the node between the source and element in the network branch. The network branch is given in Figure 4.

COIF proved to be advantageous because: a designer familiar with Fortran did not need to learn a new language; the designer did not need to input a long list of Fortran call statements; Fortran computation statements could be mixed in with the COIF graphic statements.

C. PIXIE

PIXIE is the name given to the Cambridge Multiple Access Graphics System. PIXIE utilizes a TITAN machine as its central processor. The satellite processor is a PDP 7/340. The satellite processor is used to input and format the circuit. It contacts the main processor only for analysis, storage or hardcopy. The satellite processor can also enter a remote batch analysis job.

PIXIE uses a lightpen to indicate command and control lightbuttons. A lightbutton is a word or character on the screen which activates a set of operations when indicated by the light pen. Command lightbuttons change the mode of the system. The modes of the system are drawing, erasing, compiling, analyzing, etc. Control lightbuttons are used to draw the schematic. They indicate element types and orientation.

TABLE II

COIF OUTPUT PRIMITIVES

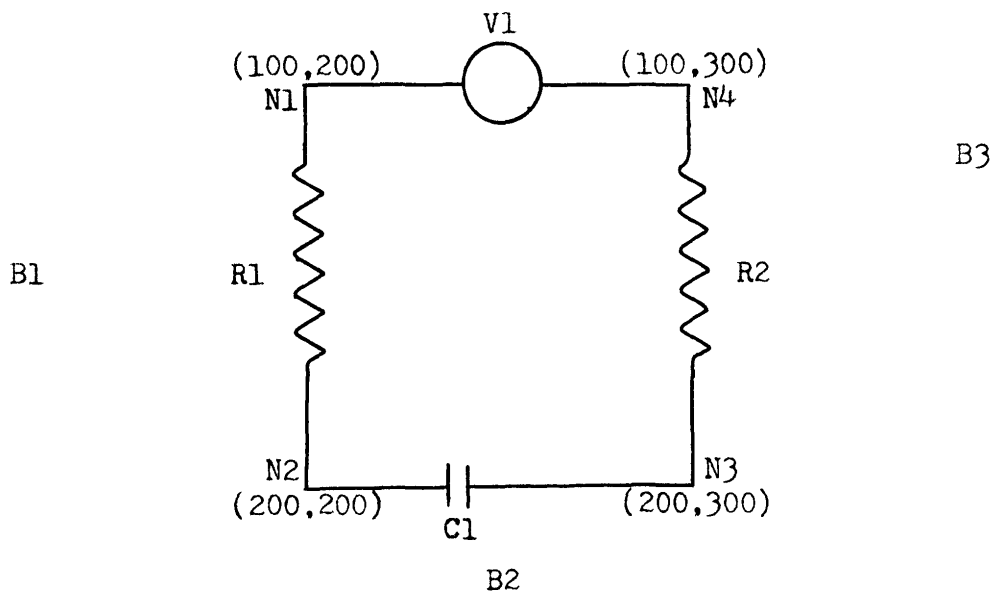
<u>PRIMITIVE</u>	<u>PURPOSE</u>
VOLTG(N1,N2)	Generates commands to build the portion of the output map which selects the voltage between nodes N1 and N2.
CURNT(B)	Generates commands to build the portion of the output map which selects the current in branch B.
OUTPUT(a,b,....,z)	Declares the variables a,b,....,z as output variables.

```

CIRCUIT N1,N2,N3,N4,R1,C1,R2,V1,OUT1
N1=NODE(100,200)
N2=NODE(200,200)
N3=NODE(200,300)
N4=INODE(100,300)
R1=RESIS(N1,N2,1)
C1-CAPAC(N2,N3,2)
R2=RESIS(N1,N3,3)
V1=INGEN(V,N4,N1,3)
INPVAL(100) R1,C1,R2,V1
OUT1=CURNT(1)
OUTPUT OUT1
100 FORMAT(E12.6)

```

(a)



(b)

FIGURE 3

COIF Circuit Specification
(a) COIF Commands (b) COIF Circuit

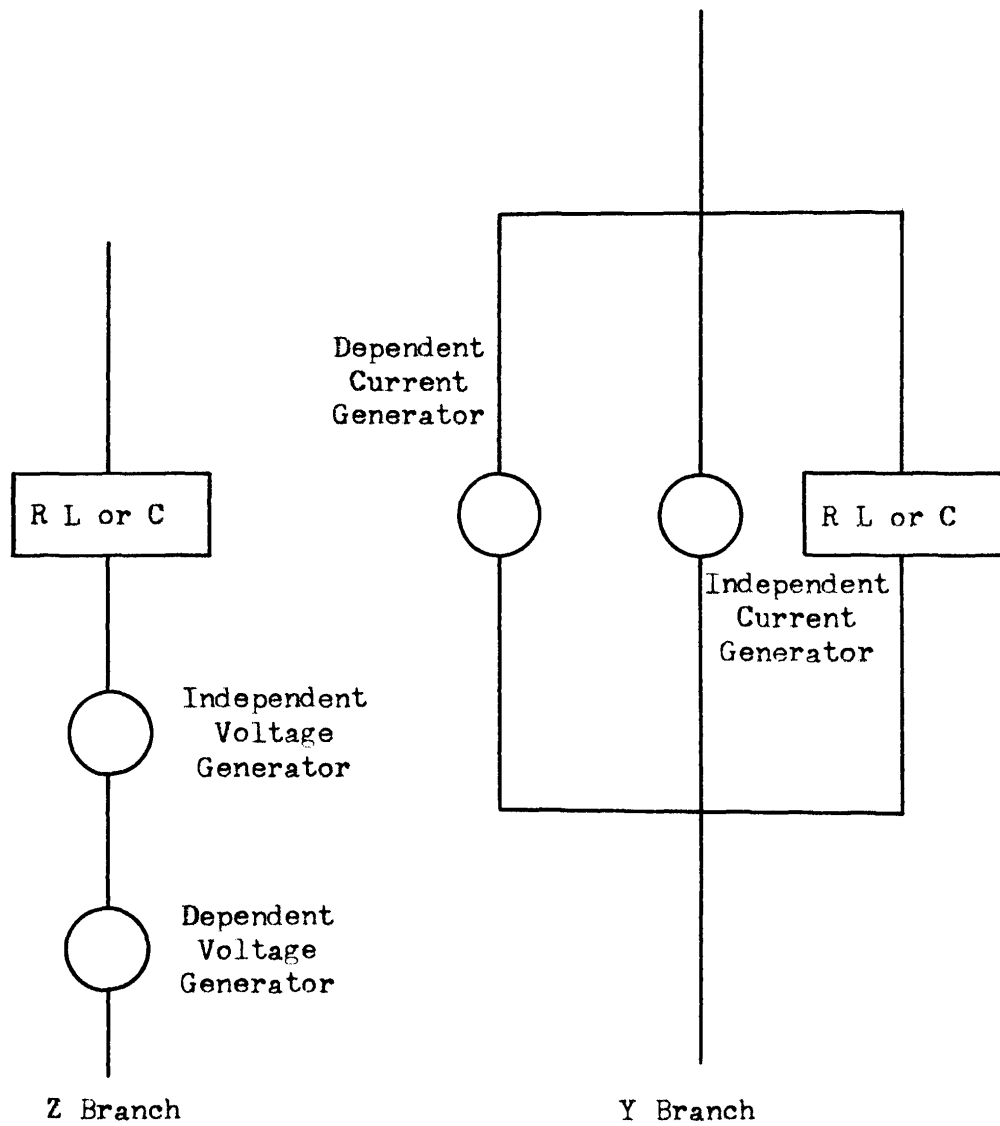


FIGURE 4

Form of COIF Network Branch

A feature of the PIXIE system is its up-down compilers. (Figure 5). The user draws a circuit by entering the drawing mode, indicating the starting point and tracing out the circuit. As the user traces out the circuit, the graphical routines construct a temporary display file. The file contains all the graphical and circuit information. When the user points to the control lightbutton F, the up-compiler is entered. The temporary display file is translated into structured form by the up-compiler. The structured form contains the nodes, branches, lengths of line segments and other graphical information. The node and branch information is stored in a form acceptable to the analysis program. Upon completion of this structure, the down-compiler is entered. This generates the permanent display file. The temporary display is cleared and control is returned to the user.

The information in the temporary file can easily be changed by re-tracing the incorrect part of the circuit. The permanent file can only be changed by entering the erase mode. The user can enter part of his network, put it in the permanent file and continue with his circuit. All of the up-down compiling is accomplished in the satellite processor. Only the needed information for analysis is transmitted to the central processor.

D. Other On-line CAD Systems

There are a number of other interactive systems. SCAMP3 is similar to OLCA but has provisions for a tolerance analysis.⁹ The analysis performed takes into account the random variation of the circuit parameters' values within a given tolerance. The tolerance can be specified by the designer.

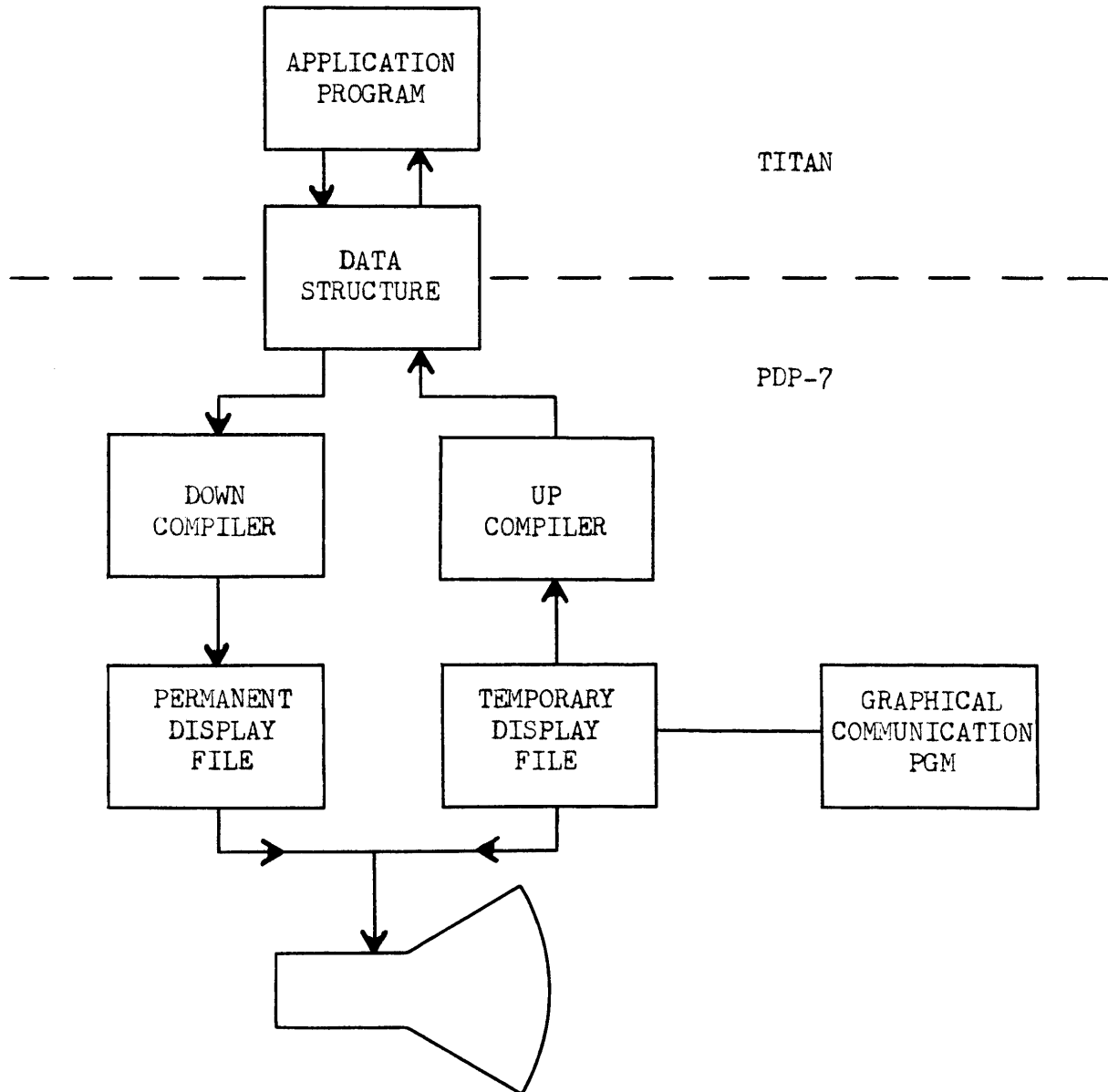


FIGURE 5

PIXIE System Data Paths

M.I.T.'s Project MAC is one of the few on-line utilities that does not use a peripheral processor.^{3,10} All processing and servicing of the graphics terminal is done by the central processor. This requires a high speed link between the graphics terminal and the central processor. This type of system was found to be unsatisfactory.

The OCTOPUS system developed at LRL (Lawrence Radiation Laboratory) is one of the largest on-line systems.¹¹ It employs two PDP-6 computers as a time-sharing supervisor. These supervisors switch programs from the terminal to a large array of bigger computers, including 2-IBM 7094, 2-CDC 6600, and a CDC 3600. A design supervisor selects the analysis program and machine best for the analysis desired.

III. UMR'S SYSTEMS ORGANIZATION

The UMR system can be organized into hardware and software components. The hardware utilized in the system was already existing or was purchased just prior to the start of the project. The existing software routines were modified so that they would be compatible with the programs being written.

A. Hardware

The organization of the system's hardware is shown in Figure 6. The Advance Remote Display Station (ARDS), model 100A,¹² has a keyboard, a set of function keys, and a graphic input device (mouse). The keyboard is standard teletype keyboard except that it has provisions for upper and lower case characters. The function keys are a group of push buttons numbered from one to eleven and are used to set one of the bits in the accumulator of the mini-computer. The mouse, a device similar to the joy stick or light pencil,¹³ is used to manually position the beam (cursor) at a specific point on the screen.

The satellite processor is a Scientific Control Corporation 650 (SCC 650). This mini-computer is used locally to draw the circuit diagram on the ARDS and as a buffer since the data rate between the ARDS and the SCC 650 is 1200 baud while the data rate between the SCC 650 and the IBM 360/50 is 110 baud.

The teletype connected to the SCC 650 is a Teletype model ASR 35. The teletype can be used as an I/O device for all information except graphic. The teletype serves as a backup for the ARDS keyboard.

The ARDS INTERFACE allows communication between the SCC 650 and the ARDS graphic terminal. The INTERFACE is described in Rhine's thesis.¹⁴

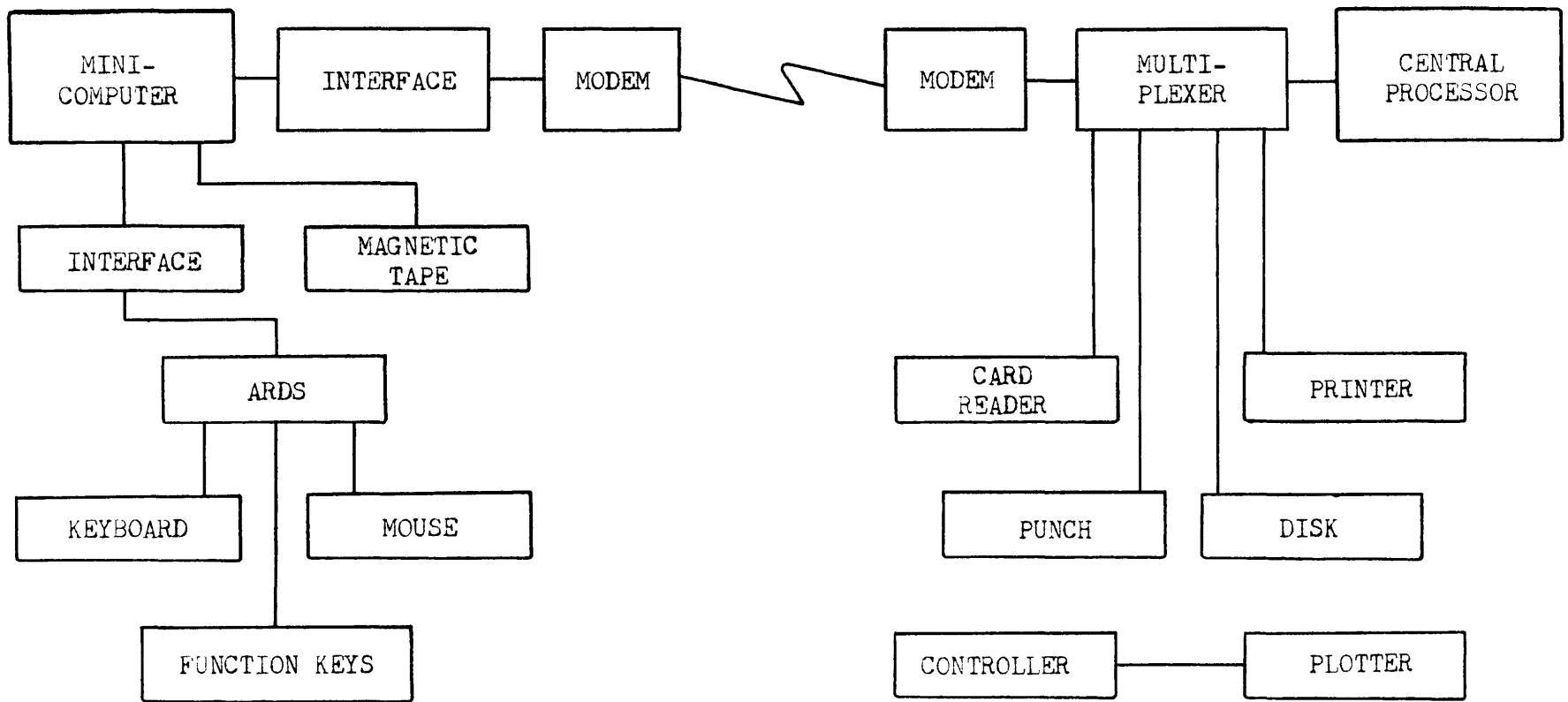


FIGURE 6

Organization of System's Hardware

The digital INTERFACE described in Omohundro's thesis¹⁵ allows communication between the SCC 650 and the modem. The modems are Bell System 103A data sets. They convert the logic information into an audio signal suitable for transmission over telephone lines. Information is transmitted and received at the rate of 110 baud.

The central processor is an International Business Machine System 360 model 50. Its peripheral equipment includes an on-line printer, an off-line plotter, a card reader, a card punch and disk storage. The plotter is a Calcomp 566 Digital Incremental Plotter. The printer is an IBM model 1403 and is capable of printing 1100 lines per minute. The card reader is an IBM model 2501. The disk storage drives are IBM model 2314. The card punch is an IBM model 2540 and is capable of punching 300 cards per minute.

B. Software

The flow of information in the system is pictured in Figure 7.

CIRCUITS is a program written in 650 assembler language for the SCC 650.^{14,16,17} It draws a picture of the desired circuit on the screen of the ARDS graphic terminal. CIRCUITS does all the bookkeeping necessary to alter or add to the circuit. CIRCUITS stores a description of the network topology needed to analyze the circuit in the memory of the SCC 650.

The program CTD (Circuit Topology Data) is also written in SCC 650 assembler language for the mini-computer. CTD retrieves the network topology produced by CIRCUITS and transmits it to the control program, GICNA. Upon completion of the transmission, CTD jumps to a state where its function is to send and receive information between the ARDS and

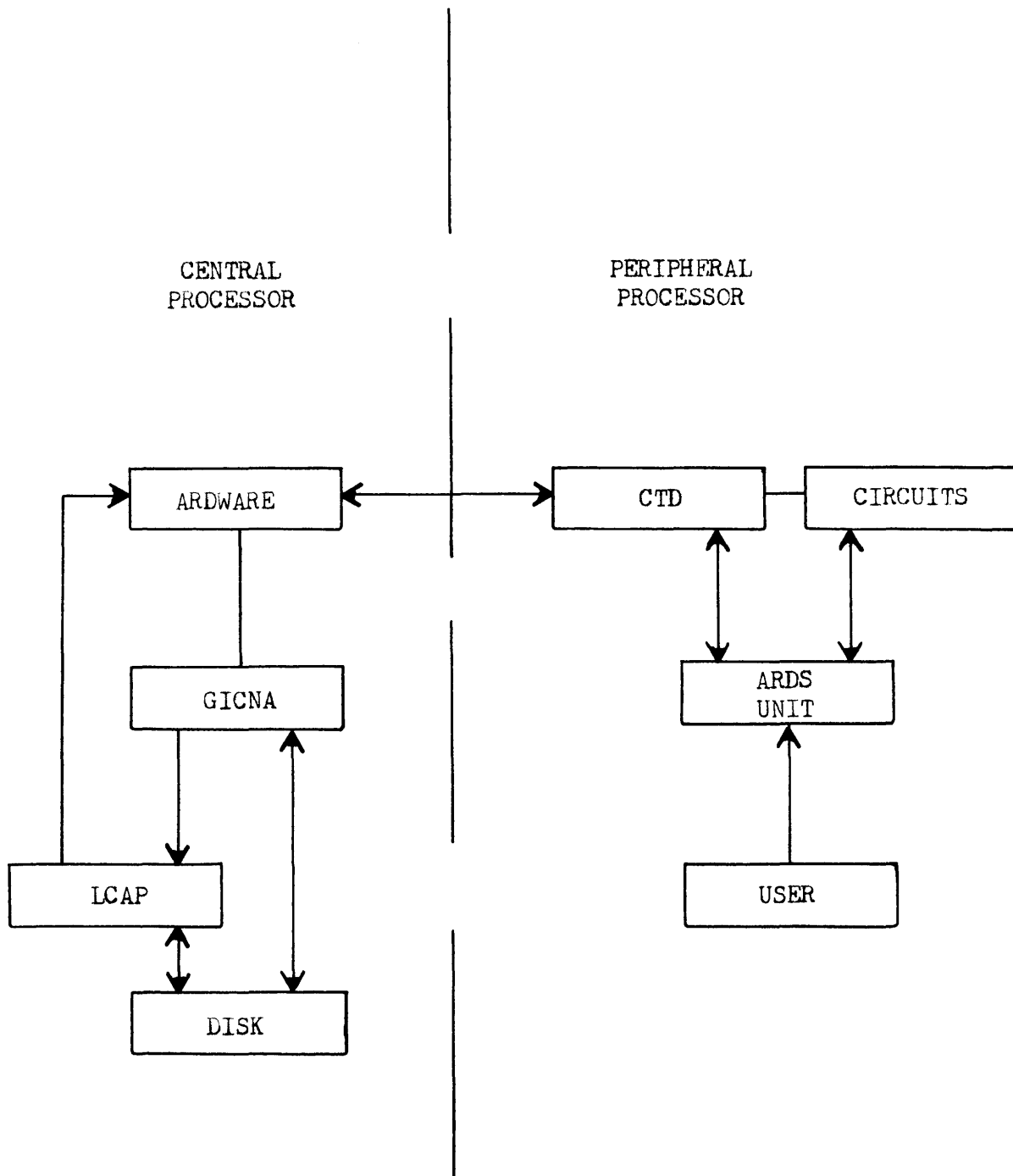


FIGURE 7
Information Flow

central processor.

ARDWARE is the graphic language.^{18,19} The ARDWARE subroutines, written in Fortran and Assembler language, handle all code conversion, mode setting (graphic or alphabetic), and input-output between the ARDS terminal and the main control program. The ARDS subroutines are part of the UMR Library of subroutines.

GICNA (Graphical terminal Input for Computer-aided Network Analysis) is a group of subroutines which convert the electrical circuit topology data into the card format acceptable to the analysis program. The user communicates with the program on-line to define all circuit parameters and element values. The development of GICNA is discussed in the following pages.

The analysis program, ICAP, analyzes the data and produces a plot²⁰ on the ARDS screen. The tabular data is stored on disk and may be retrieved by the user when desired.

IV. SYSTEMS DEVELOPMENT

A. Language

There was a possibility of three different languages to be used to write the GICNA control system: 360 Basic Assembler Language; Programming Language One (PL/1); and Fortran. The 360 Basic Assembler Language was immediately abandoned because of the difficulty involved in readability and transferability.

PL/1 has more advantages than Fortran because of its character handling abilities. PL/1 can compare, add, subtract and alter a string of characters with ease. Fortran can do little more than read and write a character. Most of the manipulation that would have to be done in the control program would involve characters. The analysis program accepts only character input. The graphic terminal also deals only with character strings. Therefore, PL/1 appeared to be the language to use.

However, a major problem developed that forced the control systems to be written in Fortran.²¹ The graphic language, ARDWARE, was written in Fortran and Assembler. Considerable problems were encountered when an attempt was made to call subroutines and pass character arguments between Fortran and PL/1. The non-compatibility of the basic handling structure of the two languages made it impossible to perform the necessary transfer.

Another reason for choosing Fortran is that the analysis programs are also written in Fortran. It is desirable to restrict the overall system to as few languages as possible so that persons who work with the programming would not have to know three different languages.

For these reasons Fortran was chosen as the computer language for the control program.

B. Core Storage Requirements

The large size of the already existing analysis program forced all possible steps to be taken to minimize core requirements. One step taken was in the storage of the card information. Storage must be available for 100 cards each 72 characters long. 100 cards was estimated to be the number that would be required for the largest circuit that could be specified. Normally in Fortran, an integer*4 variable is used to store character information. It would require 29K bytes to store these cards this way. Experimentation found that a character could also be stored in a logical*1 variable. This would result in a 75% saving in core.

Fortran, however, does not allow the comparing of logical*1 variables. It will compare integer*4 characters and determine if they are equal. The logical*1 variable is used in Fortran mainly to store logical information, .TRUE. or FALSE.. This problem of comparison was not solved until an assembler subroutine in ARDWARE was found which would compare logical*1 characters; therefore, all characters in GICNA are stored as logical*1, to conserve core.

C. Man-Machine Communications

There are four approaches that could be taken to provide a communication language between man and machine. These are: direct question and answer; modified question and answer; alphabetic parameter identifier; and numeric parameter identifier.²²

Direct question and answer (DQA) involves asking the user a question which requires an answer of "yes" or "no" to be typed into the keyboard. If the answer is "no" the program proceeds to the next question. If an

answer of "yes" is received, the program then prints a variable name and waits for the user to type in the information. DQA requires a question to be asked for every piece of input information. In DQA the questions can be grouped, where one question asks if any of a list of variables is to be changed. A "no" reply causes progression to the next group. A "yes" reply causes a question to be asked for each variable in the list. DQA in either form is time consuming on the part of both the computer and the user. It also consumes a large amount of core because the program must contain a Hollerith field for every question that is to be asked. However, DQA has the advantage of being straight forward. It allows the use of the program without the necessity of reading a lengthy paper on the use of the program. Figure 8a lists the general form for the DQA method.

Modified question and answer (MQA) is a variation of DQA. Instead of answering the question with "yes" or "no," the user answers with either "no" or the new value of the variable. MQA requires less user response, thus reducing input and user error. MQA has the same advantages and disadvantages as DQA. Figure 8b shows some responses for this method.

The alphabetic parameter identifier (API) gives a listing of all possible variables. Each variable has an alphabetic identifier associated with it. The user inputs the alphabetic identifier and a value for the variables that are to be changed. The identifier is then checked against a list to see which variable is to be changed. This method requires less on the part of the user because he only has to input the variables that he desires to change. Considerable time is saved over the DQA or MQA method. The identifier names chosen are those which

DO YOU WISH TO CHANGE THE VALUE OF
ANY OF THE RESISTORS? YES

DO YOU WISH TO CHANGE RESISTOR ONE? NO

DO YOU WISH TO CHANGE RESISTOR TWO?
RESISTOR TWO=? YES
3K

(a)

DO YOU WISH TO CHANGE THE VALUE OF
ANY OF THE RESISTORS? YES

DO YOU WISH TO CHANGE RESISTOR ONE? NO

DO YOU WISH TO CHANGE RESISTOR TWO? 3K

(b)

THE VARIABLES ARE: RESISTOR ONE R01

RESISTOR TWO R02

RESISTOR THREE R03

THE VARIABLES TO BE CHANGED ARE?

R02=3K

R03=5K

(c)

THE VARIABLES IDENTIFYING NUMBERS ARE:

RESISTOR ONE 1

RESISTOR TWO 2

RESISTOR THREE 3

THE VARIABLES TO BE CHANGED ARE?

2=3K

3=5K

(d)

FIGURE 8

Communication Languages Between Man and Machine

(a) DQA (b) MQA (c) API (d) NPI

usually used to represent the variable, R for resistance, V for voltage, etc. Figure 8c gives an illustration.

The numeric parameter identifier (NPI) associates a number with each variable. NPI requires more effort on the part of the user and less on the part of the programmer, than API. The user must learn what number is associated with which variable. The programmer need no longer check a long list of variables because the identifier number specifies the position of the variable in the list. A considerable amount of computer time is saved using this method. Figure 8d shows some typical changes.

In the design of GICNA, it was desired that the user not have to learn a totally new language in order to use the system. It was also desired that the user not have to memorize a long list of alphabetic or numeric parameters. To meet these requirements, it would appear that the DQA or MQA approach should be used.

However, the analysis program already required 160K bytes of core storage. The program GICNA would be required to store approximately 100 cards each containing 72 characters. Storage for the program instructions and other variables would require considerable core. Including the storage required for the Hollerith field of the DQA or MQA method would bring the total core requirement to between 350 and 400 Kbytes. This large amount of core storage would greatly hinder the scheduling of the job and the length of time for which the job could be run.

The time required to transmit a Hollerith field from the central process to the ARDS terminal was also considered. The telephone line available can only operate at 110 baud. The DQA and MQA would also result in considerable time being consumed answering the questions.

In choosing which method to use for the man-machine communication, a compromise was reached. It involved utilizing both DQA and NPI methods. The DQA method is used in the first pass. It is possible to complete an analysis and never have to specify a numeric identifier. The NPI method is used to make all parameter changes, I/O specifications, and system alterations.

In parts of the program a combination of the two methods is used. Using the DQA a question requiring a "yes" or "no" answer is asked. If "no" is the reply, the program proceeds. If "yes" is the reply, a shift is made to the NPI method and the program requests a numeric identifier. When the question ARE THERE CHANGES? is answered with a "yes," the words ENTER CHANGE CODE appear on the screen. The program is requesting a numeric identifier or code to indicate what changes are to be made.

The above described method proved to be a benefit to both user and programmer. Considerable storage and computer time are saved, yet it is possible for the system to be used without learning a totally new input language.

D. Component Number System

When the topology data from CIRCUITS was converted to card information, the circuit identity was lost. The circuit in Figure 9 is as it would appear from the CIRCUITS program.

The standard card input for this circuit is:

B1	N(1,2),R=	,S=
B2	N(0,1),R=	
B3	N(2,3),R=	
B4	N(0,3),C=	
B5	N(1,2),C=	

A comparison of the circuit and the cards shows that it is difficult to identify the circuit elements. The node numbers have been changed and can not be used as a reference. Resistor, R01, is contained in card B2. Capacitor, C01, is contained in B5. The numbers associated with the components could not be placed on the cards because the analysis program was not designed to accept them.

Three solutions to this problem were studied. The first involved rewriting the program CIRCUITS so that the nodes that were generated would not have to be changed. This was determined to involve more work than it was worth.

The second solution involved limiting the user to only those basic elements that would not result in node changes. The wire, NPN and PNP

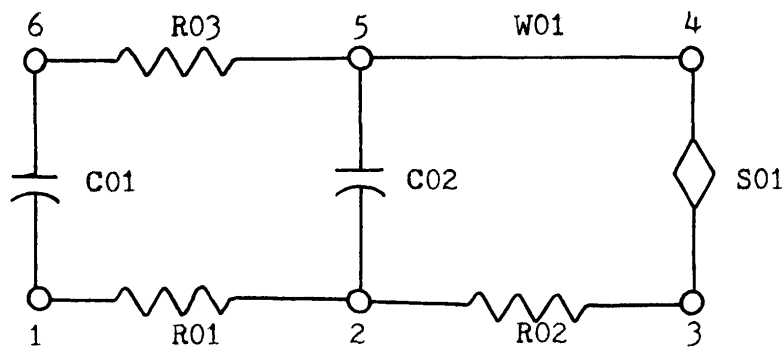


FIGURE 9

CIRCUITS Network

transistor and the forward and reverse diode could not be used. The user would have to input the equivalent circuit for the transistor and

diode. This was found to be too large of a limitation on the user and greatly hindered the drawing of the circuit.

The final solution and the one chosen, involved altering the analysis program to accept the component numbers. This was found to involve considerable tracing through the analysis subroutines to determine where and how the positions were before the equal sign was checked. However, this solution was found to be the most satisfactory. The card information now appears as follows:

```

B1      N(1,2),R02=      ,S01
B2      N(0,1),R01=
B3      N(2,3),R03=
B4      N(0,3),C01=
B5      N(1,2),C02=

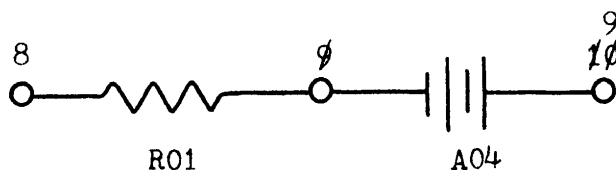
```

E. Standard Branch

1. Circuits. The standard circuit branch refers to the way in which the circuit analysis program will interpret the input data. The standard branch for this system is shown in Figure 10. The current direction is assumed to be from the last node to the current node. While each branch must contain a passive element, the inclusion of an active element is optional.

Note, no active element may be in a branch without a passive element. Consult the section on equivalent circuits to see how the transistor and diode are handled. The battery, sources and element cards are shown in Figure 11.

GICNA checks the topology data and combines the passive and active elements that are in series into a branch. The topology data for



would result in the cards information

B6 N(8,9),RER01= ,EA04=

Node nine in the diagram would be an inaccessible node and could not be specified in any print or plot statement. The equivalent node for 10 would be 9. If the node 10 were specified in a print or plot statement, the number 9 would be inserted. If a current source was specified by the user, the passive element in series with it is not combined into one branch. A new resistor is "generated" by the current source.

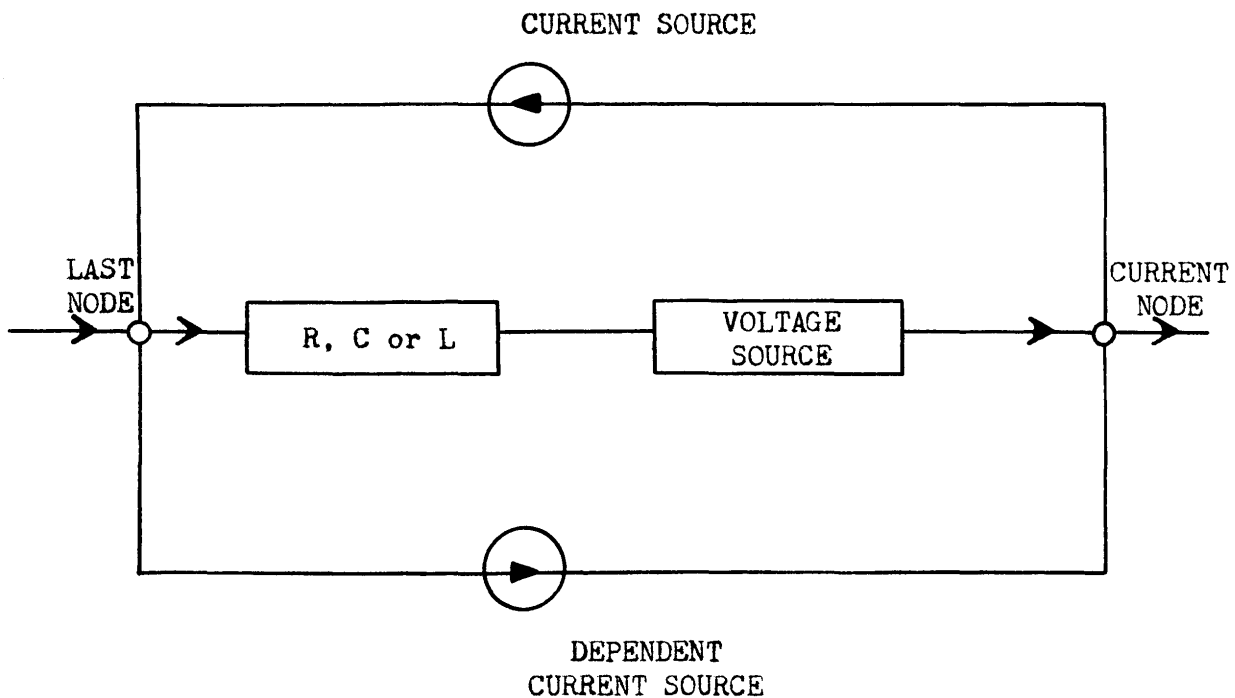


FIGURE 10

Standard Branch for GICNA

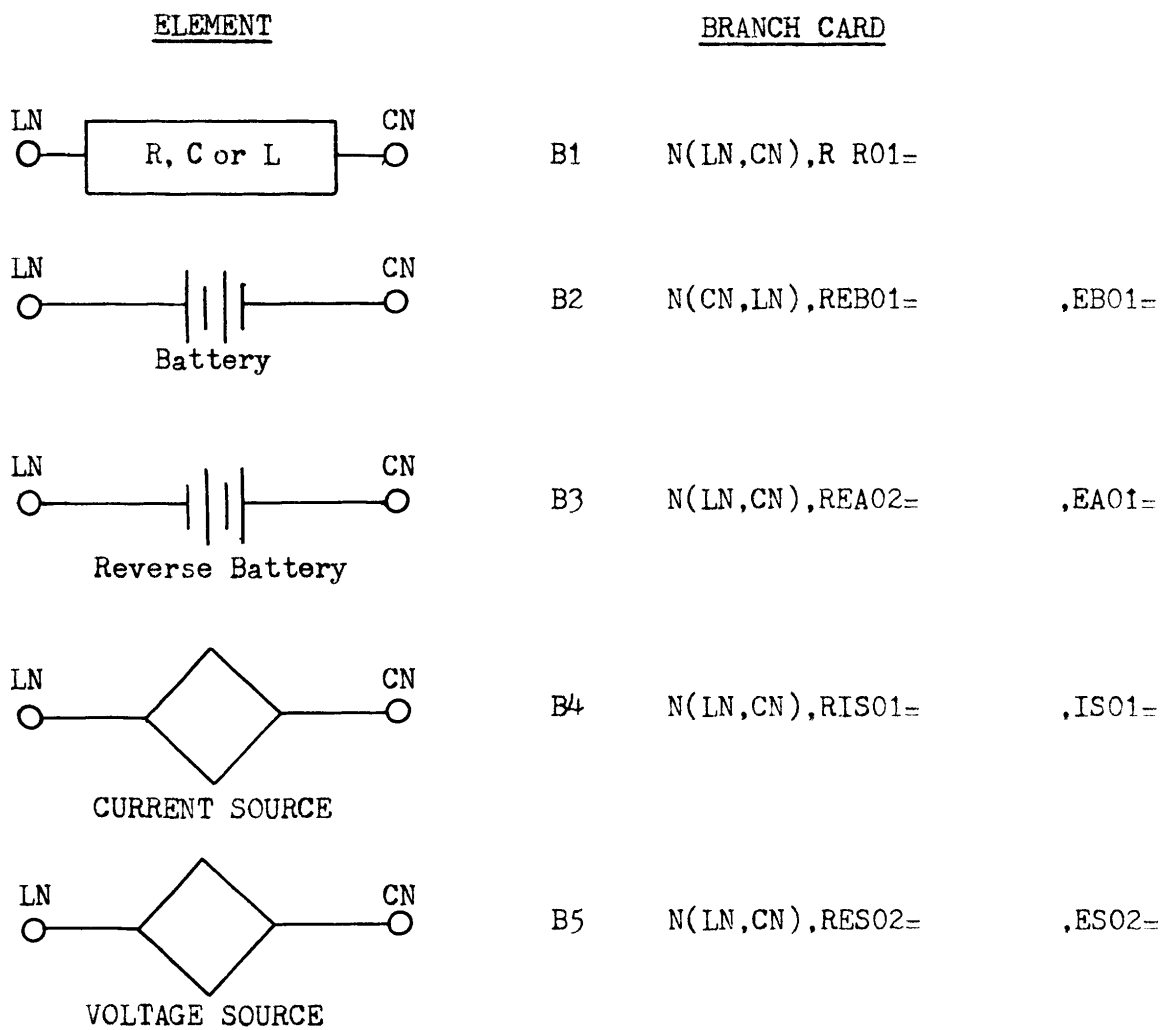
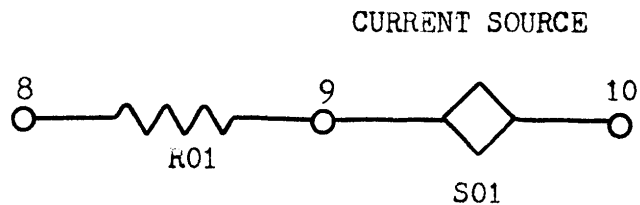


FIGURE 11

Card Format for CIRCUITS' Components

The circuit



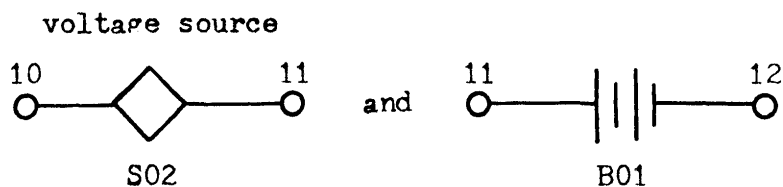
produces the cards

B7 N(8,9),R R01=

B8 N(9,10),RIS01= ,IS01=

The standard branch defines the resistor as being in parallel with the current source.

Any batteries or sources that are not in series with a passive element also "generate" a resistor. The circuit



produce the cards

B9 N(10,11),RES02 ,ES02=

B10 N(11,12),REB01= ,EB01=

2. Notation. In the card

B13 N(12,13),RERO4= ,EA04=

the group of five characters, RERO4, serve as an identifier for the user. The first character, in this case an R, indicates that the type of passive element in this branch is a resistor. This character is interpreted by the analysis program. The remaining four are ignored. The second character, E, indicates what type of active element is in the branch. A blank in this position means that no active element is contained in the branch. This position always contains an E, I, or a

blank. The third, fourth, and fifth characters, R04, are the component number system. They indicate what generated the passive element. In this case the passive element, a resistor, was generated by element R04 of the topology data.

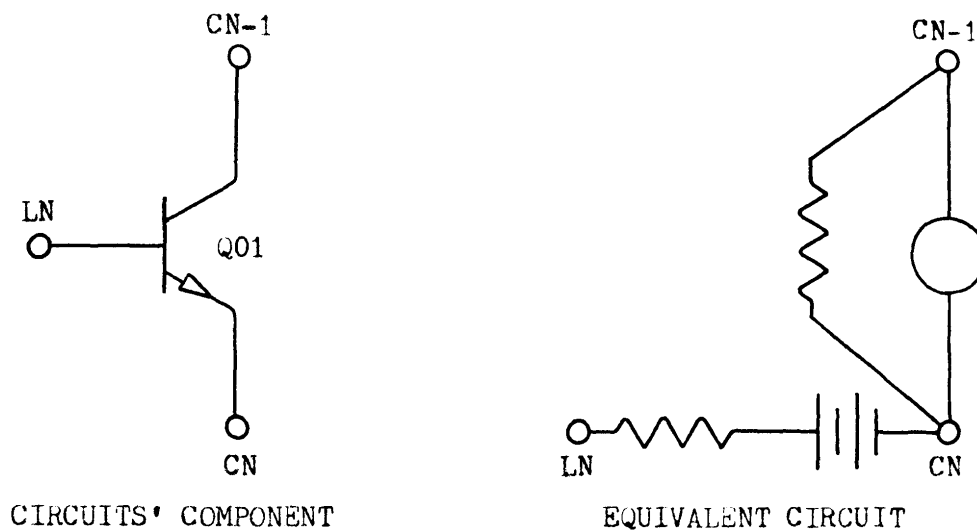
Referring to the last card, the second group of characters, EA04, is the active element identifiers. Only the first character is interpreted by the analysis program. This character is always an E or an I. In this case the E indicates that the branch contains a voltage source. The next three characters are the component numbering system and indicate what generated the active element in this branch.

3. Equivalent Circuits. There are several existing circuit analysis programs which accept transistors and diodes as circuit components; however, LCAP is not one of these. An equivalent circuit consisting of components acceptable to LCAP must be substituted. The equivalent circuits which the GICNA will substitute for the NPN and PNP transistor are given in Figure 12. The card information is also given in the figure.

The equivalent circuit and card information for a forward and reverse diode are shown in Figure 13. In both cases the source and resistor are combined in one branch according to the standard branch format. These equivalent circuits were chosen for their simplicity and good approximation to the actual component.

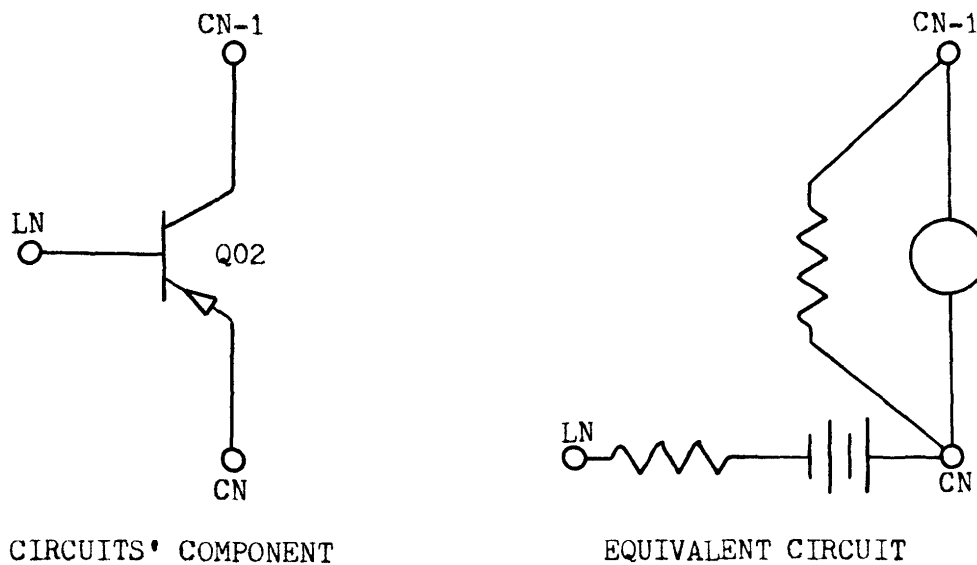
F. CIRCUITS

The program CIRCUITS was written by George Rhine¹⁴ and requires octal locations '0 through '5277 of the memory of the SCC 650 for the program instructions and variable storage. The remaining octal



B1 N(CN, LN), REQ01= ,EQ01=
 B2 N(CN-1, CN), R Q01=
 T1 B(1,2), BETA Q01=

(a) NPN Transistor

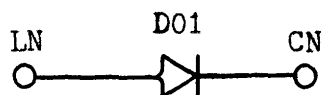


B3 N(LN, CN), REPO2= ,EPO2=
 B4 N(CN, CN-1), R P02=
 T2 B(3,4), BETA P02=

(b) PNP Transistor

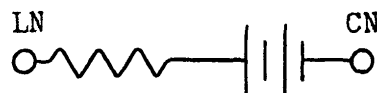
FIGURE 12

Equivalent Circuits for the Transistor



CIRCUITS' Component

B5

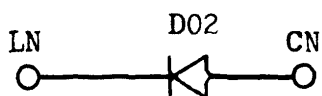


Equivalent Circuit

N(CN, LN), RED01=

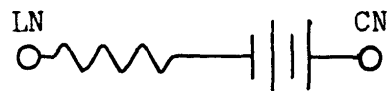
, ED01=

(a) Diode



CIRCUITS' Component

B6



Equivalent Circuit

N(LN, CN), REE02=

, EE02=

(b) Reverse Diode

FIGURE 13

Equivalent Circuits for the Diodes

locations from '6000 to '7777 are used for the storage of an element table and node table. These two tables contain all of the topology information for the network.

Two changes had to be made to CIRCUITS in order to adapt it for use with the system. The first change involved providing for an entry to the program which would perform the data transmission. This was accomplished by adding the function F3F6 to the list of existing functions. When the F3F6 is entered a direct jump to CTD is executed.

For each element in the network four locations in memory are used for the element table. The first location contains the element name and number. The second location was intended to contain a pointer to the location of the value of the element. However, the program required such a large majority of the storage that this idea was abandoned. The second location is, therefore, not used. The third and fourth locations are used to store the NODE1 and NODE2 for each element. NODE1 is the number that was the last node when the element was specified. NODE2 contains the current node.

The case of the three node device, a transistor, is where the second change had to be made. Originally both NODE1 and NODE2 contained the base node. This gave no indication of what the emitter or collector nodes were. NODE2 was changed to contain the emitter node. The collector node can be found by subtracting one from the emitter node.

G. CTD

CTD sends the element table to GICNA. Upon decoding a control C, CTD transmits the highest node number followed by a control S. Seven characters for each element are then sent. The characters are sent in

the following order: element name; element number; NODE1 number; NODE2 number. For each of the numbers two characters are always sent, with all numbers between one and nine preceded by a zero. After every 14 elements, 98 characters, a control S is transmitted. This clears the buffer area and prepares for more data. At the completion of the transmission a control N followed by a control S is transmitted by CTD. The control N is decoded as the end of data by GICNA. CTD then jumps to a halt state, after which, on an interrupt basis it sends and receives data.

CTD starts in octal location '5300 and ends at approximately '5600.

H. LCAP

LCAP is a version of ECAP (Electrical Circuit Analysis Program), a circuit analysis program written for the IBM 360/50 computer.^{23,24,25} LCAP and ECAP have the same basic subroutines except that several modifications have been made to LCAP's subroutines.

ECAP permits input only from the card reader and allows output only to the printer or plotter. LCAP permits I/O from any of the systems I/O devices. LCAP can input data from a file and output data to the ARDS terminal.

When the system 360 detects the end of the ECAP data, the program is terminated with a system error. LCAP prevents the operating system from generating an error, generates an error of its own, and returns control to the calling main line.

ECAP terminates execution in four different subroutines. Termination occurs when an error in the format of the data is found or when the analysis is complete. Upon detection of an error or completion of the

computation LCAP prevents termination and returns control to the calling main line for correction or hard copy output.

LCAP has also been modified to accept standard branches that have elements and source with component numbering system.

B1 N(0,1),R01=10,E01=5

would be accepted by LCAP while data subgroup error would be generated by ECAP. The same branch in a format acceptable to ECAP would read

B1 N(0,1),R=10,E=5

It should be noted that LCAP will consider a zero a part of a component number system while the alphabetic character "O" is considered as an initial condition.

B2 N(1,2),C01=5E-6,E01=.5

is interpreted as a voltage source and a capacitor, while

B2 N(1,2),C01=5E-6,E001=.5

is taken to be a capacitor with an initial condition of five tenths of a volt.

LCAP is a system in itself. It can run in batch mode on a stand alone basis or can be linked with another program and run in an on-line mode.

V. A USER'S MANUAL

A. Electrical Connections

The initial procedure in the electrical connection of the GICNA system is to apply power to SCC 650 mini-computer and the ARDS terminal. The cable link from the SCC 650 is then connected to the CPU 1 socket on the back of the interface cabinet. Also on the back of the interface cabinet, the plugs from the ARDS, the Data Set, and the Magnetic Tape Deck are connected to the CPU 1 channels. The lines going to the ARDS unit and the modem should be checked for proper connection. The function keyboard and mouse should be hooked up to the back of the ARDS. The ARDS terminal should be in the on-line mode. The modem should be up full and in the originate mode. The telephone should be present and plugged into its socket. The toggle switches for the Tape Deck, ARDS Interface and Digital Link should be turned on. All necessary equipment is now connected and ready to operate.

B. Program Loading and Operation

The program GICNA is a load module stored on disk in the computer center. It can be loaded using the GICNA JCL deck.²⁶ This action should be coordinated with the computer operators.

The programs CIRCUITS and CTD²⁷ can be loaded from magnetic tape or from paper tape. The programs are loaded under the name '1001 on the magnetic tape. If paper tape is used, there are four parts that need to be loaded.

CIRCUITS' starting location is '100. The use of this program is described in Appendix B.

When the circuit to be analyzed is complete, depress function

keys three and six (F3F6). The SCC 650 should leave the halt state and go into the run mode. Dial the TWX line at 364-8081. When the line is answered a high pitched sound can be heard on the phone. Place the phone in the cradle of the modem with the cord at the end marked "cord." Turn-on the modem. The SCC 650 should now be in the halt state. All communications with the SCC 650 are accomplished from the halt state on an interrupt basis.

The program GICNA is basically a question and answer program. A "yes" reply is indicated by a control S (CS) and a "no" reply by a control N (CN) followed by a control S (CS). Thence forth a CN will be used to indicate a CN CS series. An error results if a CN or CS is not received in answer to a question.

All non-question requests must also be terminated by a CN or CS. The CS is used to indicate two things. One, a CS is decoded as the end of input character. Two, a CS indicates that the information that was sent contains no errors. If the information did contain an error then a CN deletes that information and awaits the correct input.

The user is checked at each state of input for possible errors. A list of these errors is given in Table III.

When the screen is full the words END PAGE appear at the bottom of the screen. The program waits until the user is ready to continue. A CS continues the program.

After the telephone connection has been made, the ARDS bell should ring four times. This indicates that the program is ready to start. A CS starts the program. The first thing to appear on the screen is the program name and related information. This is followed by a command instruction to the user to input a control C (CC). If a CC is not

supplied, the program issues an instruction looking for a condition code (see Table IV). When a CC is entered the CTD program takes control and transmits the network data to the IBM 360. A CS should not be inserted after the CC as the program automatically inserts one. When the transmission is complete the SCC 650 should return to the halt state. The user is now questioned to see if the data is correct. If an error occurred during transmission or the phone line was dropped and the data was not sent, the user can restart at this point. A "no" answer to this question causes the program to request a condition code.

C. Analysis Control

If there are voltage sources or transistors in the circuit, the user is next questioned concerning them. A "no" answer to the question, IS THIS A VOLTAGE SOURCE? results in a current source being inserted in the circuit. A "yes" reply inserts a voltage source. A "no" reply to the question DOES THE USER WANT A BETA WITH ____? results in a gamma being inserted. A "yes" reply places a beta in the circuit. The type of analysis is now requested. The two letter abbreviations, AC, DC, TR, should be inserted. All alphabetic information can be in upper or lower case. The program automatically shifts all lower case letters to upper case.

ARE THERE ANY COMMENTS? is the next question. A "no" reply causes progression. A "yes" reply will allow the user to input any comments that are desired. Sixty-four characters are allowed on each line. These comments appear at the beginning of the program. They should include the user's name, the course number, the program number and related information. It is also advisable to provide the numbers of the nodes

which are to be plotted or printed. All node numbers are changed from those that were in the topology data because of the formatting that has to be done. A list of the equivalent node numbers may be obtained by specifying the correct condition code.

These comments are in a group referred to as a ring. In a ring the way in which each line is ended is important. A "yes" ending causes a new line to be made available. A "no" ending deletes the current line and makes a new line available. When all the desired comments have been made, enter a control A (CA). The CA ends the ring and progresses to the next question.

D. DC Analysis

If a DC analysis was specified, the print specification is the next information requested. The form for the print statement is

NV(9), CA(3,8,10) .

Only NV (node voltages) and CA (element currents) with a specified node should be used. The specified nodes give selective print-out which reduces the space needed to store the output. Modifications is the next information required for a DC analysis (see Modification).

E. AC and TRansient Analysis

If the type of analysis specified was an AC or TR, the next question checks to see if a plot is desired. An answer of "no" advances the program to the frequency and print-out state. A "yes" answer causes the program to request the plot label information. This information is used to label the output plot. It must include the user's name. There are thirty-one character positions available.

If only one plot is required, specify "yes" to the next question, IS THERE ONE PLOT? A reply of "no" means that two plots are required. Two plots, however, are the maximum number that can be specified for one analysis.

The starting and ending frequency or time step for the plot are now requested. These numbers must be in long hand notation. The number one thousand must be entered as 100 and not 1E3. Fourteen character positions are available. A decimal point following the number is optional.

A request for the plot type symbols will next be displayed. The two letter symbols and their meanings are as follows:

NV	Node Voltages
CA	Element Currents
BV	Branch Voltages
Z	Branch Voltages/Element Current

All four may be used with an AC plot but only NV and CA may be used with a TR plot.

The plot nodes are now requested. One or two numbers can be specified. When two numbers are given the ratio of the voltages or currents are plotted, the two numbers should be separated by a blank. The form is

9 or 9 10 .

It should be noted that the equivalent node for these nodes is used in the card specification. Therefore, a different number appears in the final specification than is specified here.

F. Frequency or Time Step

The AC Frequency or TR Time Step for the printed output is now requested. The frequency or time step can be in either long hand notation (1000) or exponential notation (1E3). For a plot, the frequency must also be specified over a range by the range notation

$$1000(1.5)3E6$$

The first number indicates the starting frequency. The middle number can be calculated to be a specified number of frequency steps. The equation is

$$M=(L/F)^{(1/K-1)}$$

where M is the middle number, F is the first number, L is the last number, and K is the number of frequency steps. The final number is the ending frequency. When a plot is specified a frequency range corresponding to the plot frequency range must be specified in the modification statement.

For the TR analysis a request is now made for additional specifications. The form and names are

OUTPUT INTERVAL = 1

FINISH TIME = 10E49

INITIAL TIME = 0

The entire word must be specified. The values shown above are the default values when the parameter is not specified. These specifications are in a ring. CS makes a new line available. CN deletes the present line and makes a new line available. CA ends the ring.

The print specification for AC and TR analysis are the same as for the DC analysis.

G. Modifications

For AC or DC analysis the next question asks if there are modifications to be made. An answer of "yes" makes column seven available. An answer of a CA makes column one available. A "no" causes progression. Modifications take the form

```

                Modify
                Frequency=1000(1.5)3E6
B10            N(0,4),R=10000
                Execute

```

The words Modify and Execute must be present and start in column seven. Frequency is optional but must start in column seven. The branch is optional but the B must be in column one and the N in column seven. Modification statements are in ring fashion and require a CS, CN, CA, line ending.

H. Parameters

1. Identification. The circuit parameters will appear on the screen in one of the formats in Figure 14. In form A the first C indicates that this branch contains a capacitor. The blank following the C means that no source is in this branch. The C01 is the label given this element from the program CIRCUITS. Its purpose is to serve as an identifier for the user.

Form B is a branch containing an element and a source. C02 and S01 were placed in the same branch in accordance with the standard branch. The C02 identifies the capacitor C as having existed in the topology data. ES01 indicates that the source was in the topology data and that the user specified it to be a voltage source.

Form C shows a source that generated a resistor. A source cannot

be in a branch by itself. Source S02 is in the topology data but the resistor was added according to the standard branch.

The resistor and voltage source in Form D did not exist in the topology data but were generated as an equivalent circuit for the diode, D01. (See Equivalent Circuits.)

FORM	BRANCH PARAMETERS	
A	C C01=	
B	CEC02=	,ES01=
C	RIS02=	,ES02=
D	RED01=	,ED01=
E	REQ01=	,EQ01=
	R Q01=	,

FIGURE 14

Circuit Parameters

Two branches were generated for the equivalent circuit for the transistor Q01, Form E. Consult the section on equivalent circuits for the circuit layout.

When the program CIRCUITS draws a transistor on the screen, it is labeled with a Q and given a number. NPN and PNP transistors can be distinguished by their different symbols. However, when a transistor is placed in card format the symbol is not present to distinguish between the two. Therefore, a different letter is used to indicate each type. Refer to Figure 15. The Q is used for a NPN while P is used for a PNP. Note that the identification number is not changed, making

NETWORK ON SCREEN	NETWORK ON CARDS	MEANING
Q01	Q01	NPN Transistor
Q02	P02	PNP Transistor
Q03	P03	PNP Transistor
Q04	Q04	NPN Transistor
B01	B01	Battery
B02	A02	Reverse Battery
B03	B03	Battery
D01	E01	Reverse Diode
D02	D02	Diode
D03	E03	Reverse Diode

FIGURE 15

Labeling Differences Between CIRCUITS and GICNA

it possible to identify the location of the transistor. The above also applies to the battery and the diode.

2. Values. The value of each parameter can be specified following the appearance on the screen of the identification parameter. The value can be long hand or exponential notation followed by a CS. A CN deletes the value that was just specified, repeats the identification parameter, and allows a new specification. Eight character positions are allowed for the values.

3. Changes. If the answer to the question, DO YOU WANT THE CARDS DISPLAYED? is "yes," then all the cards will be displayed. If the user is not sure what is on all the cards, this will allow him to see them as they will appear for analysis. If a card contains an error, the next question allows him to make the change. The question asks if there are any changes to be made in the cards. A "yes" reply will ask for the change code. The change code is a two-part number code. The first number indicates in which of the four groups of cards the change is to be made. Group One includes all the cards before the first B type card. Group Two includes all of the B type cards. Group Three includes all of the T type cards. All of the cards following the T type are in Group Four. Figure 16 indicates the groups. The plot specification cards are in Group One while the modify and print cards are in Group Four.

The second number is the number of the card in the group that is to be changed. Referring to Figure 16, the change code

23

results in

B3 N(2,0),RER01=1000 ,ES01=5.0

being displayed and thus made available for changes. The change code

41

allows a change to be made in the frequency card. After the card to be changed has appeared on the screen, the exclamation mark (!) is used to space over to the column which is to be changed. The change is then made. The exclamation mark is the only character which is ignored. All other characters are placed on the card in the position in which they are entered. Any information that needs to be deleted should be blanked over. If the value of a resistor is 10000 and the user wants to change it to $3E6$, then two zeros must be blanked out or $3E600$ will result.

```
CARD      B3      N(2,0),RER01=10000      ,ES01=5.0
CHANGE    !!!!!!!!!!!!!!!!!!!!!!!3E6
NEW CARD  B3      N(2,0),RER01=3E6        ,ES01=5.0
```

	<u>GROUP</u>	<u>CARD</u>
	C	
ONE	C	MIDDEN, LEO
	C	
		AC ANALYSIS
	B1	N(0,1), L L01=5E-4
TWO	B2	N(1,2),C C01=.05
	B3	N(2,0),RER01=1000 ,ES01=5.0
		FREQUENCY=1000
FOUR		PRINT,NV(2)
		EX

FIGURE 16

Card Information

Ending a line with a CS makes the next card available for changes. A CN makes the card preceding the present one available. A CA returns the program to the change code position. Table V lists some other change codes. Change code 71 advances the program.

I. Analysis

DOES THE USER WISH ANALYSIS? is the next question asked. A "yes" reply performs the analysis. A "no" reply returns control to the condition code state. This provision allows the user to abort the analysis if a major problem has developed or if he has accidentally arrived at this state by an incorrect code.

1. Auto-Save Facility. Before the analysis is performed, the auto-save facility comes on. The auto-save facility is a provision which allows the user to temporarily save a plot or other data which appears on the screen. This information is stored by the page and can be recalled by the user. A Header appears on the screen when the facility is on. It consists of four boxes, labeled THIS PAGE, NEXT PAGE, LAST PAGE, and RETURN. The page control mode is entered by ending a page with a control E, pointing the cursor to one of the boxes, pressing set point on the mouse, and entering a control S. In this way the user can move forward (NEXT PAGE), backward (LAST PAGE) or redisplay the present page (THIS PAGE). Once the page control mode has been entered the RETURN box must be indicated to return control to the main program. The auto-save facility is turned on before each analysis. The facility saves the card data and the plot(s). If desired, the user can turn the facility on or off by using the proper condition code.

2. Plot. If a plot was specified, it will next appear on the screen. If the plot is not displayed, an error occurred. A major error causes the error message to be displayed. The condition code for changes is then entered, the corrections are made and the analysis is repeated. If a minor error occurred, the condition code for tabular data should be entered. After the corrections have been made, the analysis can be attempted again.

When the plot is finished, the condition code for changes can be entered. This allows the user to specify alteration in the circuit parameters and request another analysis. Hard copy output can be obtained after each plot by specifying the correct condition code.

The program can be ended by specifying the exit condition code. The word `TERMINATE` will be displayed. A CA must be entered, followed by a CS. The word `TERMINATE 1` appears and another CA CS must be entered. The program is then terminated.

J. Messages and Codes

Table III is a list of the error messages generated by the program GICNA. After each error is a probable cause and the course of action that should be taken.

The condition code is the main control section of the program. It is divided into three parts: input, output, and alterations. These codes are listed in Table IV.

Table V contains the specification codes used to change the state of the system. These codes allow changes to be made in the parameters and the parameter values.

TABLE III

ERROR MESSAGES

ERROR #		CAUSE and ACTION
1	Conversion Error	All numbers are received as characters. They are then converted to a number. When this conversion was attempted, a non-numeric was detected. Try the input again.
2	Option Does Not Exist	A wrong code number was specified. Consult the code table and then specify the correct code.
3	Maximum Length Exceeded	A maximum of eight characters is permitted in the value field of the first identifier parameter on each card. Use exponential notation to express the value.
4	Maximum Length Exceeded	More information was specified than can be placed on one card. If possible use two cards, otherwise limit the information.
5	Row Does Not Exist	The number of the row that was specified in a change code was greater than the total number of rows in that group. Check to be sure that the right row number was specified. Check to be sure that the correct group number was specified.
6	Node Does Not Exist	A node for a plot or print statement is larger than the highest node in the circuit. Or the node specified is the node between an element and a source in a standard branch. Specify a different node.
7	Zero Node	This error occurs only with input condition code three. The last name specified had a zero in the first node. Most likely, a new line was indicated but no input was entered. No action needs to be taken.

TABLE III (Continued)

8	Maximum Number of Cards Exceeded	The maximum number of cards for one of the four groups was exceeded. A maximum of 15 for group one, 50 for group two, 20 for group three, and 15 for group four are allowed. The network is too big for use with this system. Reduce the circuit size or run in batch mode.
9	Label Length Exceeded	The plot label information has exceeded 31 characters. Shorten the information and respecify.
10	Wrong Answer	An answer of yes or no was not specified in reply to a question; or a needed operation could not be performed. Repeat the input giving the correct reply.
11	Wrong File Information	The information contained on the file did not match the variables into which it was read. For example, the topology data was read into the card variables. NOTE, the system is in an indeterminate state. The total system must be initialized before proceeding. Try again, specifying the correct marker code.
12	Error On Unit 12	An error developed when information was read from the tabular data storage disk. Or a punch was attempted before an analysis was performed. Try again to obtain the information. If the error re-occurs, repeat the analysis.
13	Hard-copy Analysis Error	A major error was found in the card data when a hard copy plot was specified. Rerun the analysis and produce a soft copy of the output. Then check all of the tabular data for possible formatting errors.
14	Error On Unit 14	An error developed on a read from the file that contained the desired information. Initialize the total system and try again.

TABLE III (Continued)

15	Error On Unit 15	An error developed when the card punch was specified. Go through the change code again. Specify no the questions ARE THERE CHANGES and DO YOU WISH ANALYSIS. Then retry the punch.
----	---------------------	--

TABLE IV

CONDITION CODES

CODE #		EFFECT
A. INPUT CODES		
1	Input from CTD	The circuit topology data is to come from the program CTD and CIRCUITS. The name, number and two nodes are sent for each element. If CTD does not complete its transmission and the program appears to be in a loop, insert a CN CS. The program should check to see if the data is sent. Specify an answer of no.
2	Card Format	The circuit is to be specified by card format. The plot, print, and modify information is first requested and placed in the proper position on the cards. The group code can then be entered. Group code 1 and 4 make column one of a card available. Group code 2 and 3 make column seven available and automatically place the proper B or T and the row number in the first columns. These cards are in ring fashion. Group code 5 displays all the cards on the screen. Group code 6 returns control to the main line.
3	Topology Data Form	The input is in the form that is generated by the program CIRCUITS. Seven characters for each element must be typed-in. The elements are in ring fashion.

TABLE IV (Continued)

4	Card Format from File	The cards are loaded from a file. The cards were placed on file by a previous run or through the card reader by the program LCI (Lost Card Information). The data is stored under a marker code. An error is generated if the file does not contain card information.
5	Topology Data from File	The topology data is loaded from file. The information was placed here by a previous run. A marker code is required. An error is generated if the file does not contain topology data.
6-10	Error	These codes are not used at the present time. An error message results if they are specified.

B. OUTPUT CODES

11	Tabular Data Displayed on ARDS	The information that was generated by the analysis program is displayed on the ARDS screen. The error message is displayed if an error occurred. If no error occurred, the tabular output is displayed.
12	Tabular Data on Printer	The same information that was placed on the ARDS under code 11 is now printed on the printer.
13	Punch Cards	The card information for the last circuit that was analyzed is punched on IBM cards.
14	Card Information on File	The card information for the circuit presently in core is permanently stored on file. A marker code between one and eight is required. This marker code is also needed to retrieve the information.*

*The same file is used for codes 14 and 15. Therefore, care should be taken that the information stored is not under the same marker code. Also the information that was previously in the file under the specified marker code is destroyed.

TABLE IV (Continued)

15	Topology Data on File	The topology data is stored permanently on file. A marker code between one and eight is required.*
16	Topology Data on Printer	The topology data and a list of the equivalent nodes is printed on the printer.
17	Equivalent Nodes on ARDS	A list of the equivalent nodes that was generated for the circuit is displayed on the ARDS screen.
18	Topology Data on ARDS	A list of the topology data and the branch in which the element was placed is displayed on the ARDS screen.
19	Card Information on ARDS	The card information for the circuit is displayed on the ARDS screen.
20	Hard Copy	A hard copy of the result of the analysis is produced. If a plot was specified, it is plotted on the plotter. The tabular data is printed on the printer.

C. Alteration

21	Initialization	The system is initialized. This code should be entered before a totally new circuit is specified. This code is automatically initiated when the system is first started.
22	Partial Initialization	This partial initialization can be specified after input condition code 5. It allows all of the card information to be produced again. If this code is not specified, it is assumed that the card information already exists in the system.

*The same file is used for codes 14 and 15. Therefore, care should be taken that the information stored is not under the same marker code. Also the information that was previously in the file under the specified marker code is destroyed.

TABLE IV (Continued)

23	Start Processing	After the circuit has been loaded by one of the input codes, this code starts the processing of the information.
24	Change Code	This code places the program in the change code state.
25	Set Plot	The plot indicator is set to one. This is automatically done when a plot is specified in the Analysis Control section. It need only be specified when a plot is added to an existing circuit.
26	Auto Save Off	The auto save facility is turned off.
27	Auto Save On	The auto save facility is turned on.
28	Restart	The program is restarted. The program name and heading information is displayed. The system is initialized.
29	Error	This code is not used.
30	Exit	This code is the program exit command. Ending states are generated and the program ends.

TABLE V

CHANGE CODES

CODE	EXPLANATION
1N	Group 1 and row N of the card information is made available.
2N	Group 2 and row N of the card information is made available.
3N	Group 3 and row N of the card information is made available.
4N	Group 4 and row N of the card information is made available.
51	The card information is displayed on the ARDS screen.
61	Input condition code 2 is called. This allows cards to be added to the circuit.
71	The equivalent nodes for the circuit are displayed on the ARDS screen.
91	The change code state is ended and the program proceeds to the analysis.

BIBLIOGRAPHY

1. Franklin H. Branin, Jr., "Computer Methods of Network Analysis," Proceedings of the IEEE, vol. 55, November 1967, pp. 1787-1801.
2. Giampio Bracchi and Marco Somalvico, "An Interactive Software System for Computer-Aided Design: An Application to Circuit Project," Communications of the Association of Computer Machinery, vol. 13, September 1970, pp. 537-545.
3. Michael L. Dertouzos, "An Introduction to On-Line Circuit Design," Proceedings of the IEEE, vol. 55, November 1967, pp. 1961-1971.
4. Gerald J. Herskowitz, Computer-Aided Integrated Circuit Design, McGraw Hill, Inc., 1968.
5. Michael L. Dertouzos, "Educational Uses of On-Line Circuit Design," IEEE Transactions on Education, vol. E-12, September 1969, pp. 197-221.
6. M. A. Murray, "Inter-University Development Programs," IEEE Transactions on Education, vol. E-12 December 1969, pp. 225-251.
7. Hing C. So, "OLCA: An On-Line Circuit Analysis System," Proceedings of the IEEE, vol. 55, November 1967, pp. 1954-1961.
8. Wiseman, Lemke, and Hiles, "PIXIE - A New Approach to Graphical Man-Machine Communications," International Conference on Computer Aided Design, 15-18 April 1969, pp. 463-471.
9. G. Bracchi and M. Somaltrico, "Man-Machine Communication in Computer-Aided Tolerance Analysis of Electronic Circuits," Advance in Electronics - Proceedings of the 16th Electronics Congress, 24-27 March 1969, pp. 271-280.
10. Lee and Thornton, "An Experiment in Computer-Aided Education," Proceedings of the IEEE, vol. 55, November 1967, pp. 2001-2005.
11. Dawson, Kuo, and Magnuson, "Computer-Aided Design of Electronic Circuits: A User's View Point," Proceedings of the IEEE, vol. 55, November 1967, pp. 1946-1954.
12. Advanced Remote Display Station (ARDS 100A) Reference Manual, Computer Displays, Inc.
13. Samuel Davis, Computer Data Display, Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1969.
14. George Rhine, A Hardware and Software Interface Between a Graphics Terminal and the SCC 650 Computer, Master of Science Thesis, University of Missouri-Rolla, 1971.

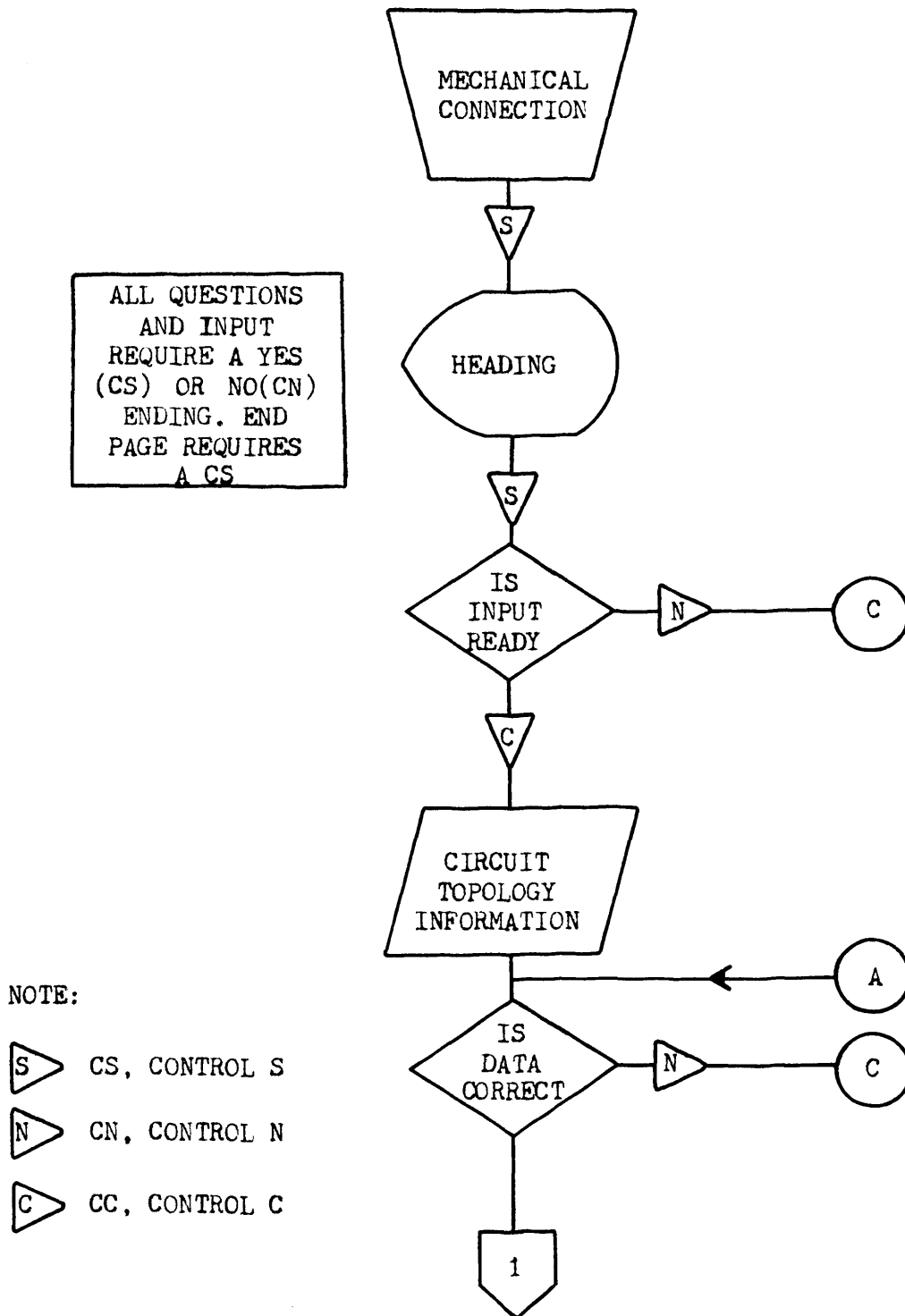
15. Wayne Everett Omohundro, The Design of a Data Set Interface, Master of Science Thesis, University of Missouri-Rolla, 1971.
16. R. F. Crall and J. H. Tracey, "Operation of the SCC 650 Computer," Department of Electrical Engineering, University of Missouri-Rolla, Technical Bulletin CRL 68.1, August 1968.
17. G. I. Rhine and R. F. Crall, "Notes on the Operation of SCC 650 Assembler," Department of Electrical Engineering, University of Missouri-Rolla, Technical Report CRL 69.3, December 1969.
18. ARDWARD the Basic Software System for ARDS: Programing Manual, Computer Display, Inc., Document Number 013-02, June 1970.
19. ARDWARE the Basic Software System for ARDS: Installation Manual, Computer Display, Inc., Document Number 013-11, July 1970.
20. Alan Cowgill, "ECAP Remote Graphics Input," Seminar Paper EE195 University of Missouri-Rolla, Unpublished.
21. Fortran IV (G and H) Programmer's Guide, IBM 360 Operating System, June 1970.
22. McKinlay, "Conversational Techniques Developed for Remote Access Computer-Aided Design," IEEE International Conference on Computer-Aided Design, 15-18 April 1969, pp. 162-170.
23. Electronic Circuit Analysis Program (ECAP/360-E): Notes, University of Missouri-Rolla, June 1970.
24. Randal W. Jensen and Mark D. Lieberman, IBM Electronic Analysis Program, Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1968.
25. 1620 Electronic Circuit Analysis Program: User's Manual, IBM Applications Program, 1965.
26. Leo William Midden, "Documentation on the GICNA Software System," Department of Electrical Engineering, University of Missouri-Rolla, Technical Report CRL 73.1, April 1973.
27. Leo William Midden, "CTD and Related SCC 650 Programs," Department of Electrical Engineering, University of Missouri-Rolla, Technical Report CRL 73.2, April 1973.

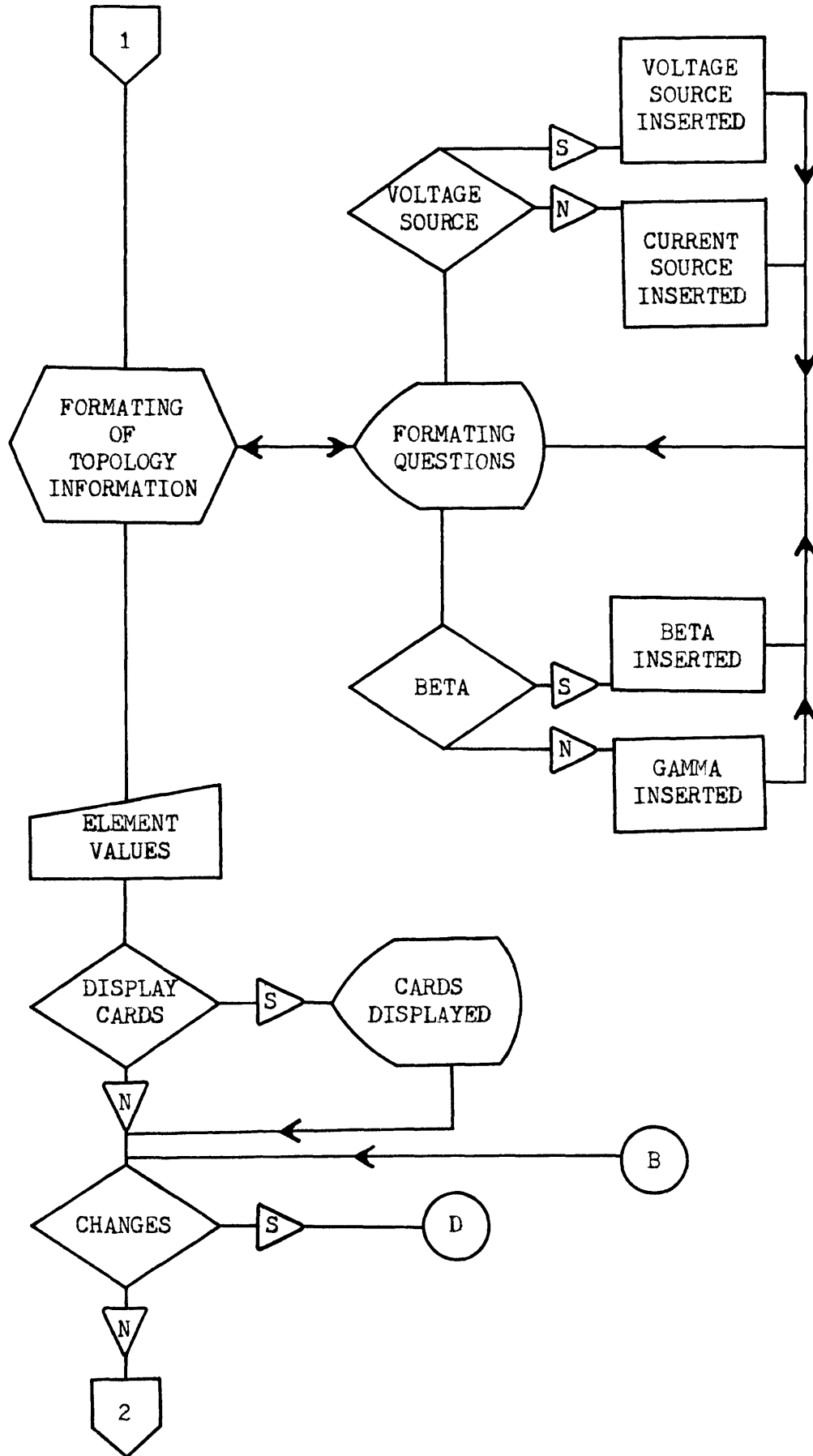
VITA

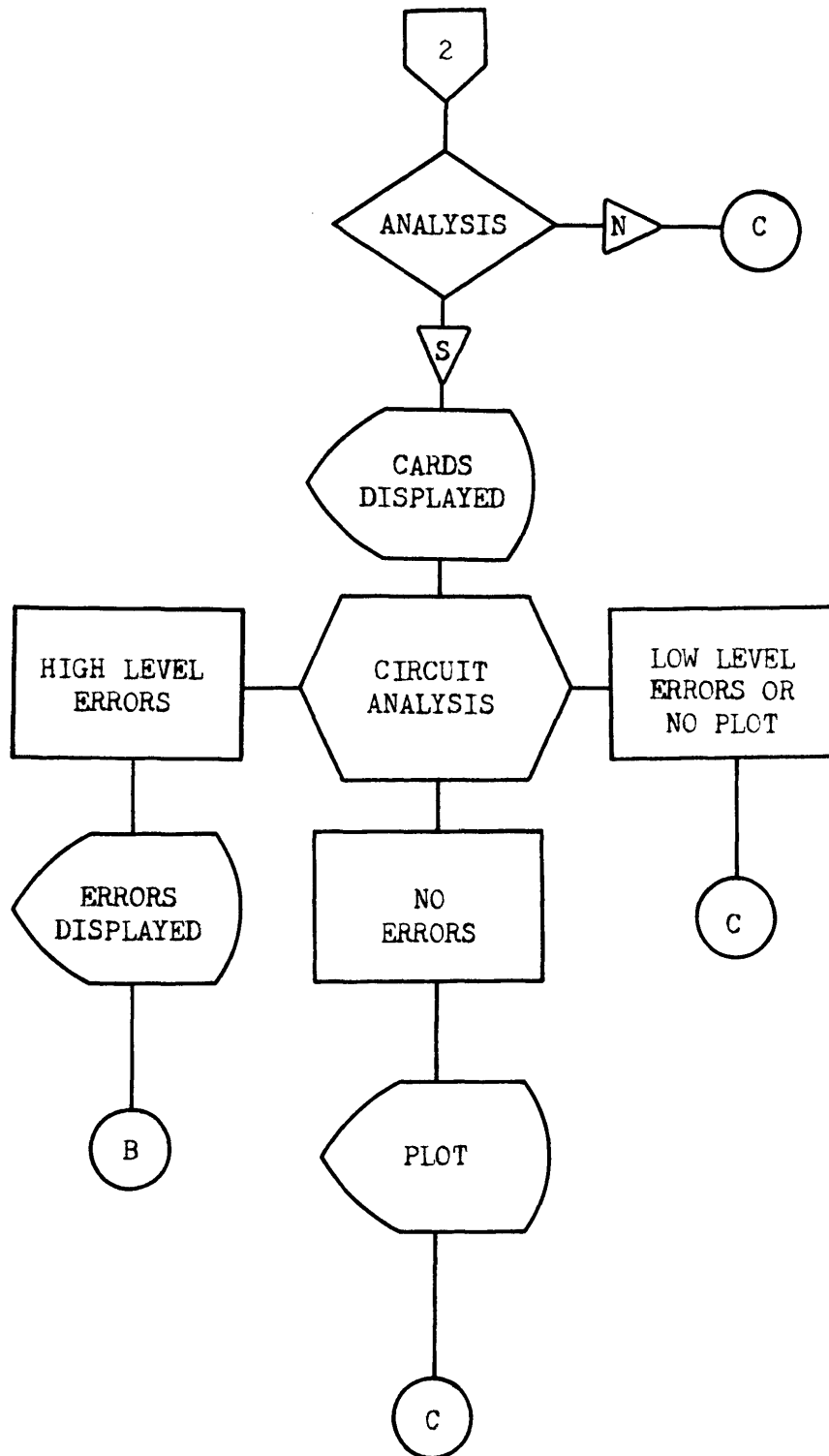
Leo William Midden was born on July 3, 1948, in Springfield, Illinois. He received his primary and secondary education in the city of his birth. He attended Springfield College in Illinois where he received an Associate of Art degree. In September 1968, he enrolled in the University of Missouri-Rolla and received a Bachelor of Science degree in Electrical Engineering in May 1970.

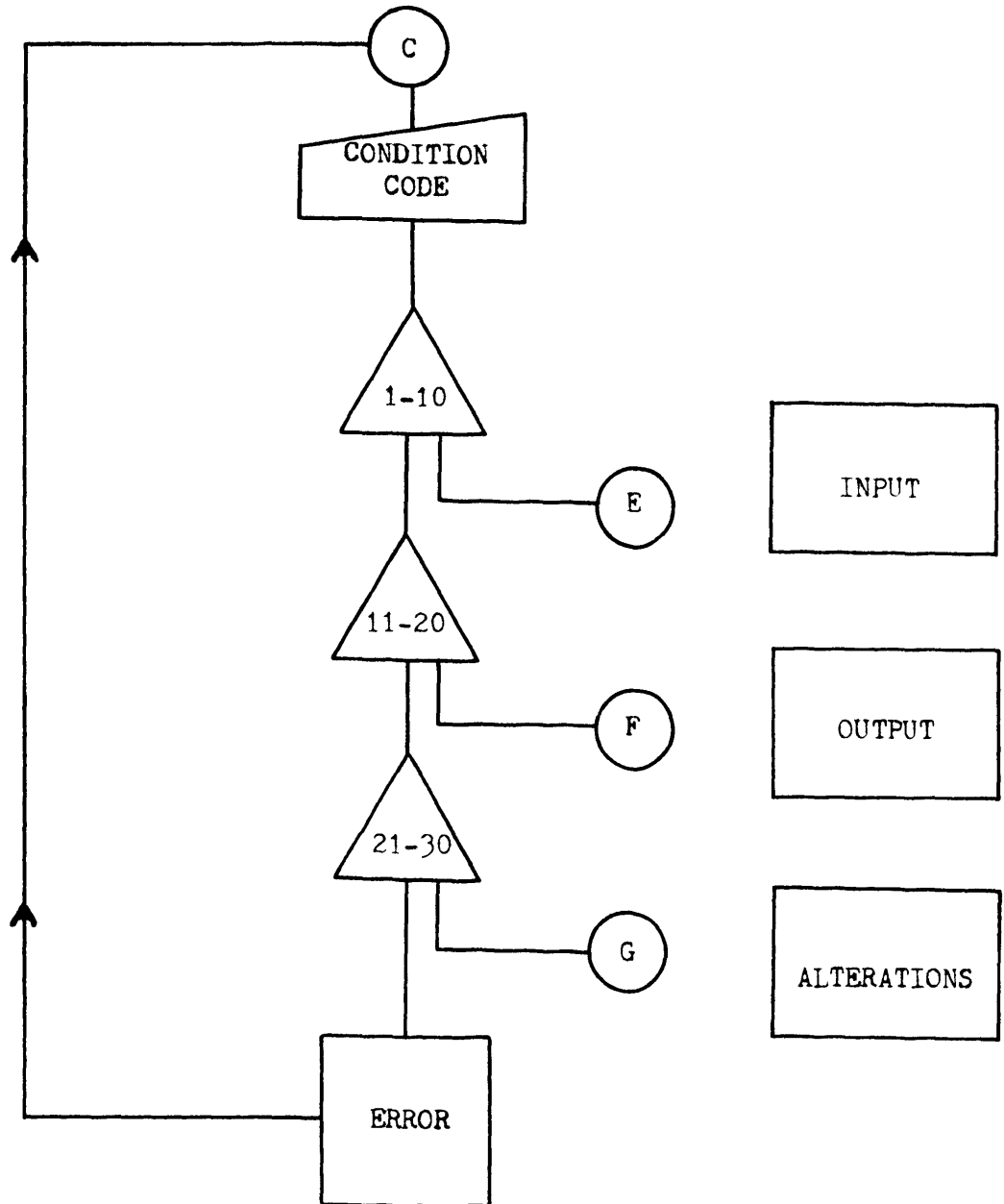
He enlisted in the United States Air Force and will spend the next four years in the service of his country.

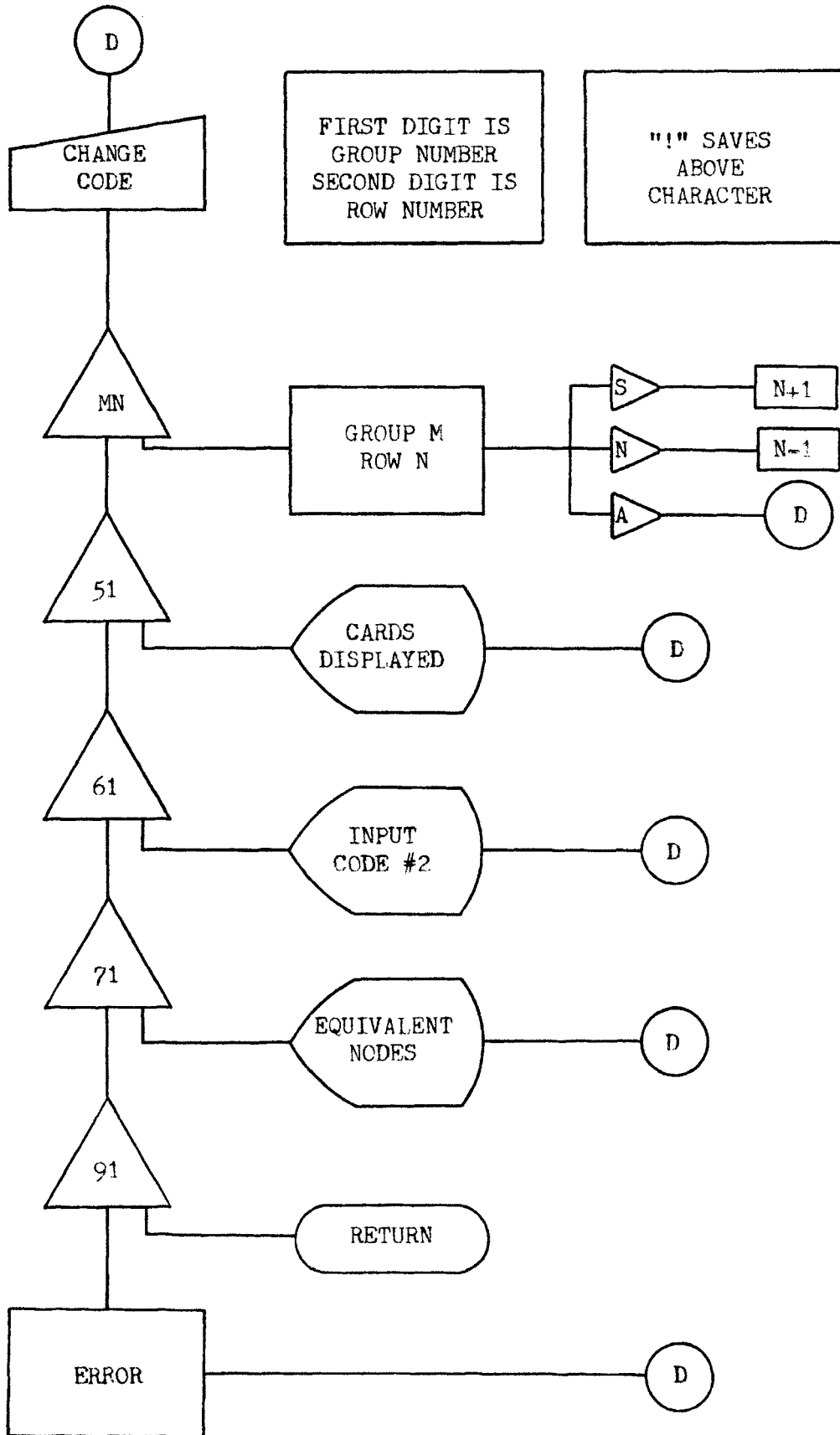
APPENDIX A

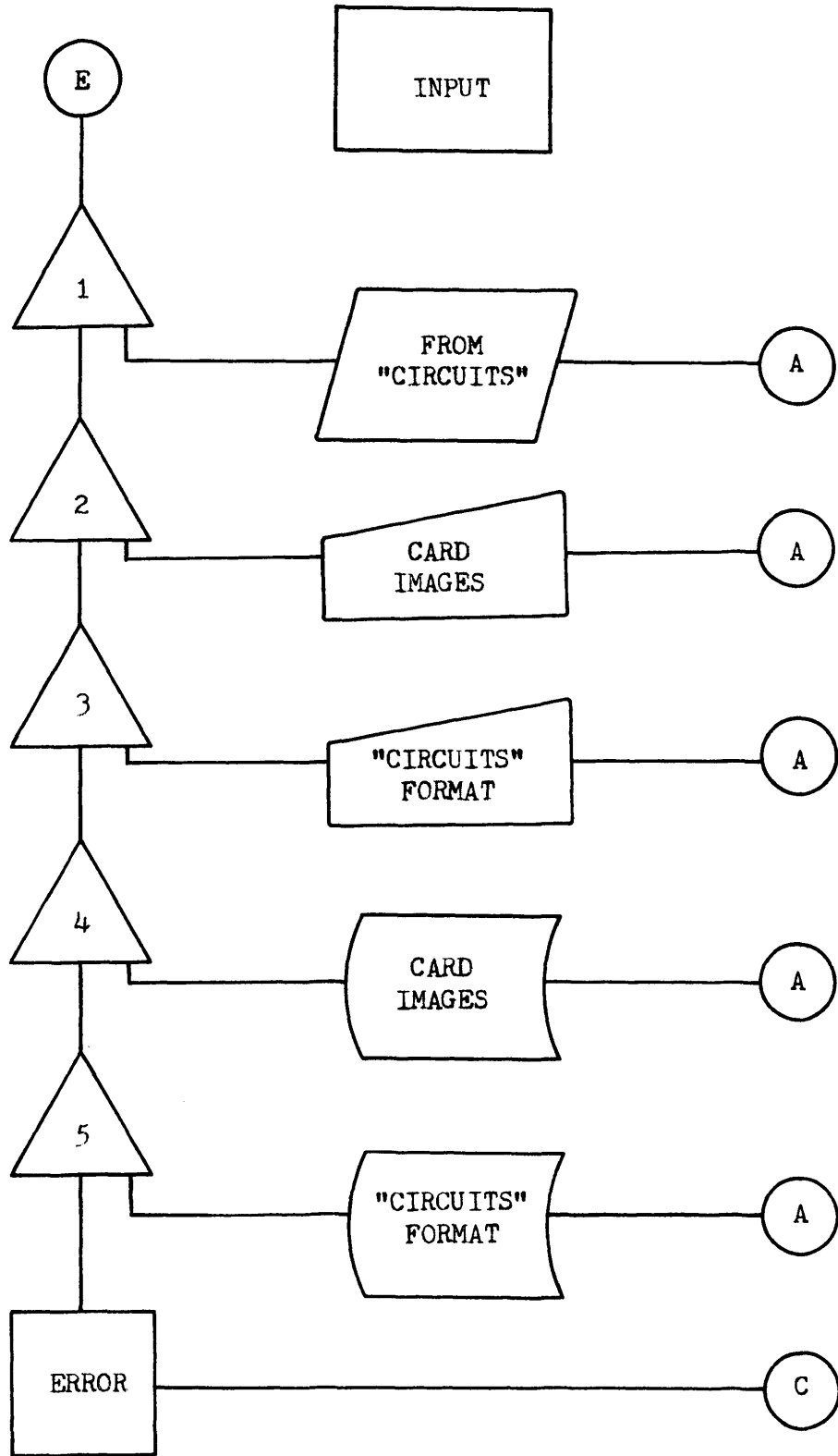
A USER'S FLOW CHART
FOR GICNA

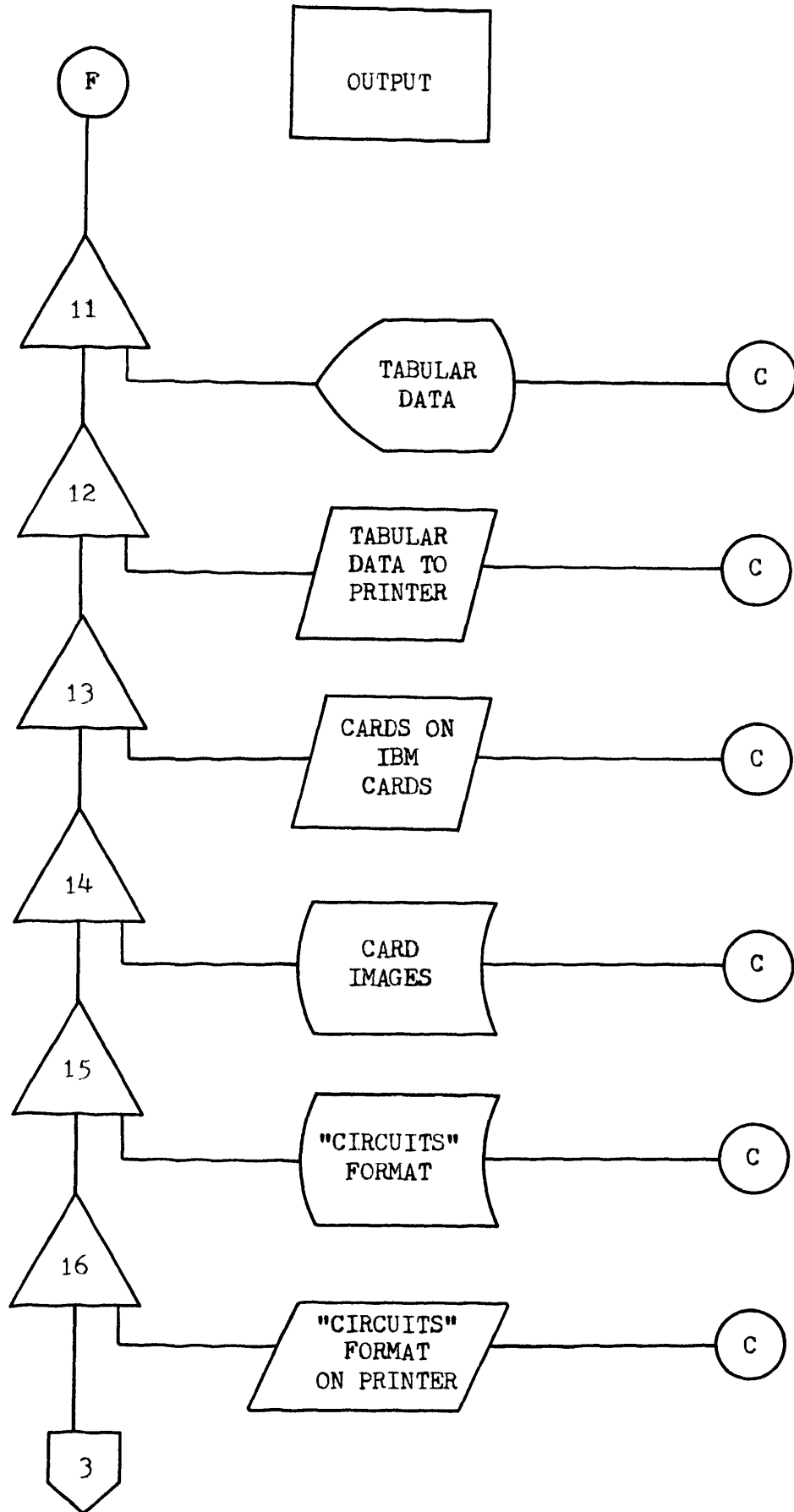


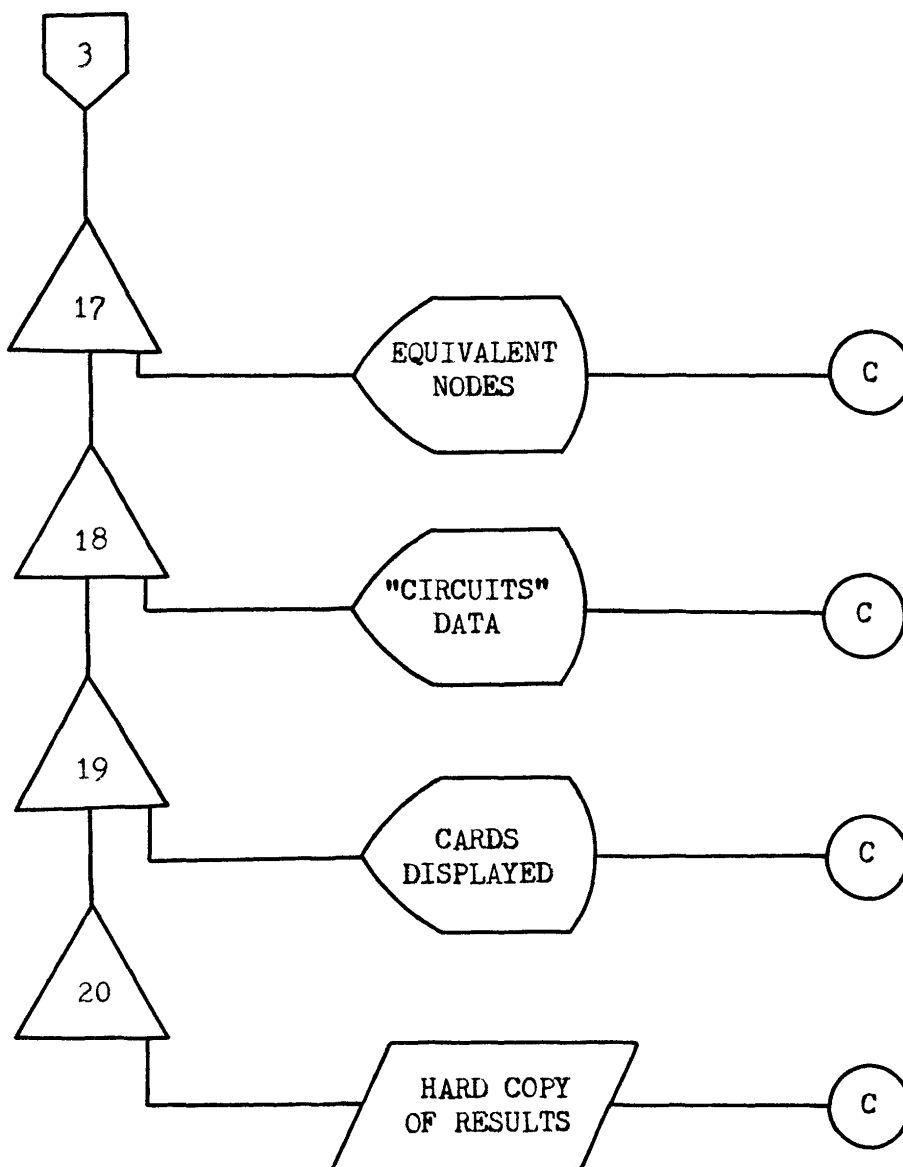


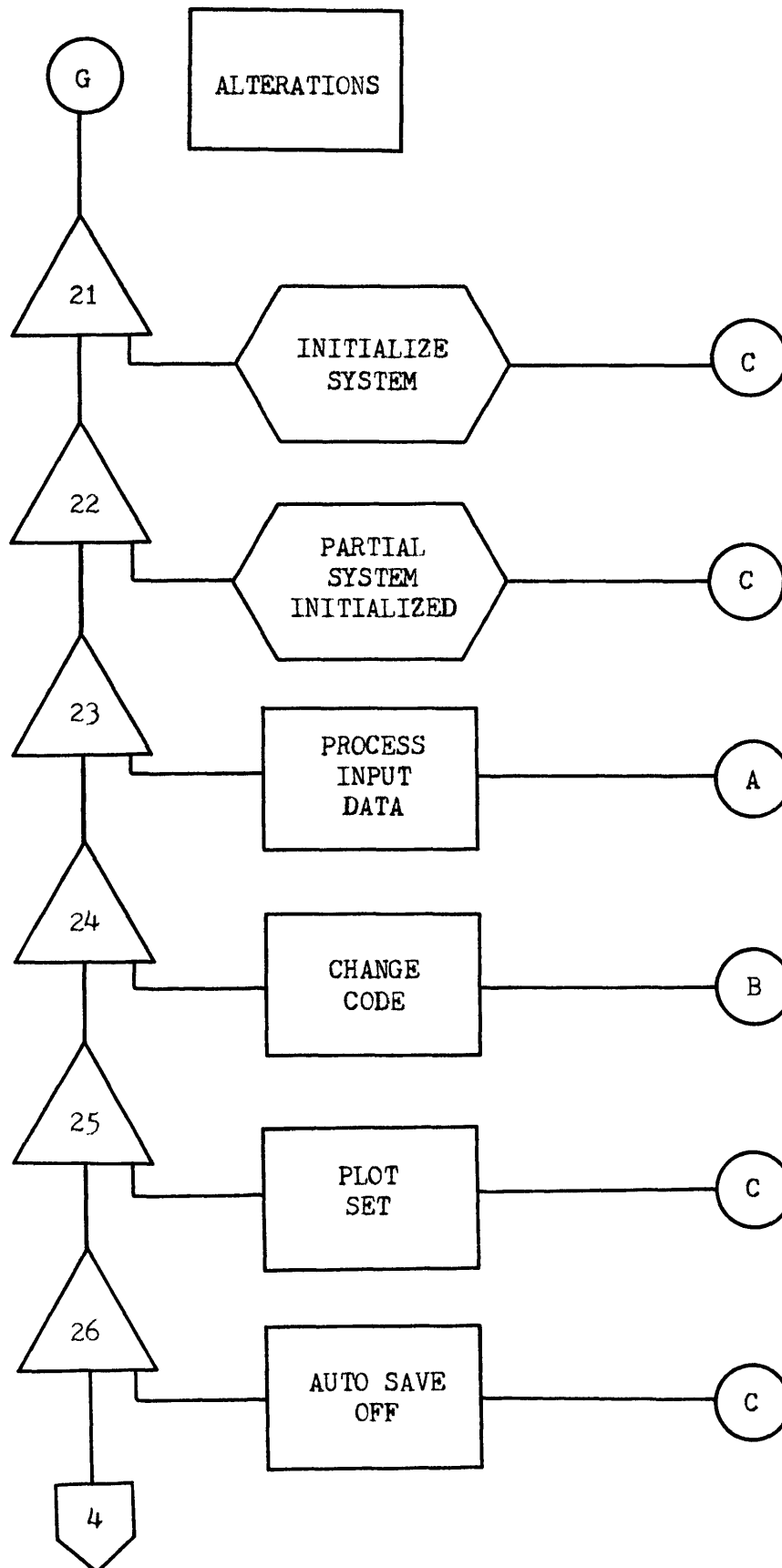


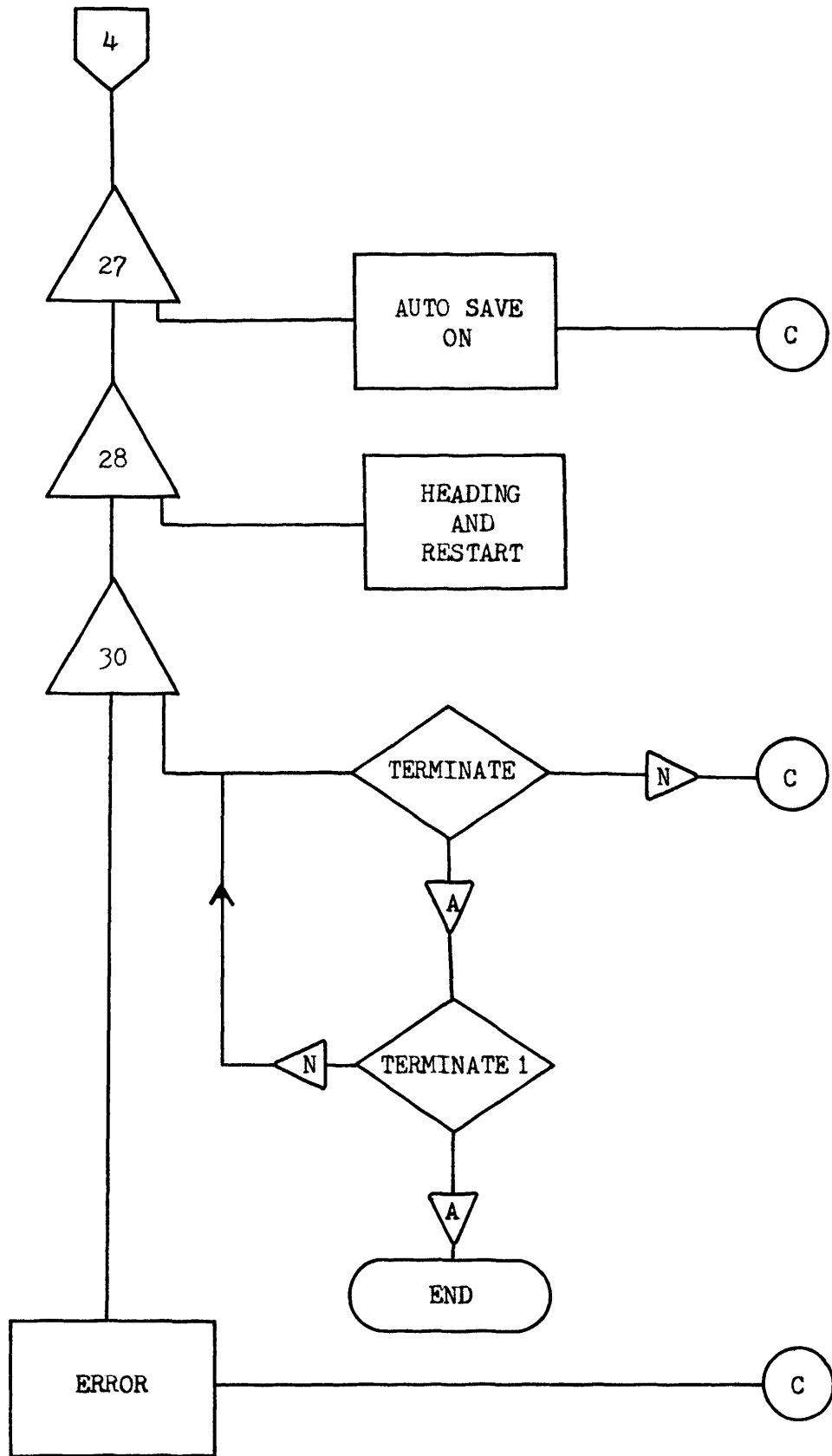












APPENDIX B

A USER'S GUIDE FOR RHINE'S CIRCUITS

A. Circuit Specifications

CIRCUITS is a program which allows the user to interactively draw a circuit schematic on the ARDS graphics terminal. The program also constructs a dictionary of the elements and connections. This dictionary allows the circuit to be redrawn when the user desires. The dictionary also contains all of the circuit topology data necessary to analyze the circuit.


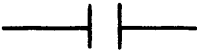


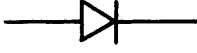


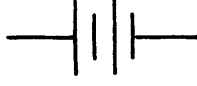
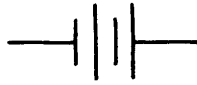
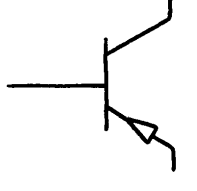
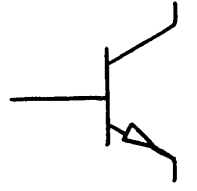
A device referred to as a mouse is used to indicate the location of the nodes on the screen. The middle button on the mouse (set point) sends the point location to the program. The set of function keys is used to indicate what element or action is desired by the user. A list of the functions and their description is given in part C of this section. The element figures which appear as a result of these function keys are given in Table VI.

The current node is the node that was last specified preceding the current node. When node #1 is entered, it is the current node. When node #2 is entered, it becomes the current node and node #1 becomes the last node. All elements are placed between the last node and the current node.

The maximum allowable schematic that CIRCUITS can produce contains 24 nodes. A maximum of four elements can be connected to each node.

The program CIRCUITS can be loaded from paper or magnetic tape. There are three parts in absolute format that need to be loaded from

TABLE VI.
STANDARD ELEMENTS FOR CIRCUITS *

<u>ELEMENT</u>	<u>SYMBOL</u>
NODE	*
RESISTOR	
CAPACITOR	
INDUCTOR	
WIRE	
RIGHT/DOWN DIODE	
LEFT/UP DIODE	
SOURCE	
BATTERY	
MINUS BATTERY	
PNP TRANSISTOR	
NPN TRANSISTOR	

* Table taken from Rhine's thesis.

paper tape. A direct jump to the start of the absolute loader is executed at the end of each tape. Therefore, the run button need only be lifted and depressed to load the next part. On magnetic tape the total program is stored under the name '1001. It can be loaded using the magnetic tape loader program.

The starting location for the program is '100. The program should go into the halt state and place a dotted line across the bottom of the ARDS screen. The circuit should not be drawn below this line, because this area is used for error messages. All specifications are done on an interrupt basis from the halt mode.

B. Example Program

The following is taken entirely from Rhine's thesis.

This next section will present the steps necessary to draw a circuit on the ARDS. Figure 17 will be used as a typical circuit that a user may wish to construct for study.

The first step that is taken is to initialize the program by entering function number 11 (F 11) through the function keyboard. This will erase the screen and draw a dotted line across the lower portion of the screen. The user should refrain from entering into this area since error messages will be displayed here.

Using the mouse and the middle button (set point), the user would now point to the location on the screen where node #1 should appear. Since most analysis programs consider the lowest numbered node to be GROUND, this node should be entered first. Next F10 is entered and the terminal will respond with the symbol for a node and its number on the screen. Next the user points to the location of node #2 and enters this via the mouse and function keys. Now entering F9, the terminal will respond with the symbol for a signal source between nodes one and two. Next node #3 is entered, and then F1 is entered specifying a resistor. A resistor will then be drawn from node two to three. Notice that only one new node was entered; the element is always drawn from the last node to the current node. After drawing the resistor, node #3 will become the last node. This same process continues until R2 and B1 have been drawn.

At this point, node #5 is the last node and we wish to draw a wire to node one. By pointing the mouse to node one and entering F14, node one is redefined as the current node but no output is generated. After entering F5, a wire will be drawn between node one and node 5. Using F14 again, node three is redefined as the current node and then the user enters node six and resistor #3. Redefining node three again, the user enters F13 and a NPN transistor is drawn on the screen and two new nodes are created, node seven at the collector, and node eight at the emitter. Node eight is also the current node after the transistor is drawn. Entering node nine and F1 will draw a resistor from the emitter to node nine. Redefining node four and entering F5 will draw W2.

In order to create node ten at the intersection of two grid lines from node six and node seven, the user should first redefine node six and then redefine node seven. Now, if F15 is entered, a node will be created horizontally from the first redefined node (node six) and vertically from the last redefined node (node seven). If F5 is entered, W3 will be drawn. The rest of the elements and nodes can now be drawn in a similar manner.

As an example of the deletion capabilities, assume that when the diode was drawn F4 had been entered instead of F3. This would have resulted in a diode pointing to the left. To correct this, nodes seven and eleven are both redefined and F16 is entered. This will delete the diode from the dictionary but not the screen. To see the circuit without the element enter F23. Now redefining nodes seven and eleven and entering F3 will cause the proper diode to be drawn.

It is advisable to periodically dump on paper tape, F1F9, the part of the circuit that has already been drawn. A machine error can cause loss of the circuit that is stored in memory. The paper tape back-up can save considerable time if the error occurs. F1F8 reloads and redraws the circuit.

The complete circuit should also be dumped on paper tape. With the paper tape the circuit can be reproduced relatively easy at some future time. If an error occurs in communication with the analysis program, the paper tape makes it simple to reload the circuit.

C. Function Keys

A list of the function keys is given in Table VII. The function keys for the elements are self-explanatory. The following is taken from Rhine's thesis and explains the use of the special function keys.

The first of these is RESTART. This function will erase the screen, initiate the program and dictionary, and allow the user to begin drawing the circuit diagram. REDRAW will redraw the circuit on the screen from the dictionary. DELETE ELEMENT will delete the element between the last node and current node from the dictionary. It will automatically resequence all elements with the same name and a higher number. DELETE NODE will delete the current node from the dictionary if there are no elements connected to it. Also, all nodes higher than the deleted one will be resequenced. REDEFINE will redefine the referenced node as the current node without displaying the node symbol or changing the dictionary entry. SAVE will punch the dictionary description on paper tape so that it may be used at a later time. LOAD will read a paper tape produced by the SAVE function and construct a dictionary for that circuit. The circuit will then be drawn on the screen and the user may add to it, modify it, or call for an analysis. CORNER will place a node on the screen at the intersection of two grid lines extending horizontally from the last node and vertically from the current node. ATTENTION will stop all processing and return control to the user. This is particularly useful when an error has been noticed during a REDRAW sequence.

D. Error Messages for CIRCUITS

Table VIII is a list of error messages produced by the program CIRCUITS. In order to return to normal operation, it is important to follow the action stated.

TABLE VII. *

LIST OF FUNCTION FOR CIRCUITS

<u>FUNCTION</u>	<u>DESCRIPTION</u>
F0	Attention
F1	Resistor
F2	Capacitor
F3	Right/Up Diode
F4	Left/Down Diode
F5	Wire
F6	Inductor
F7	Battery
F8	Minus Battery
F9	Source
F10	Node
F11	Restart
F1F2	Transistor-PNP
F1F3	Transistor-NPN
F1F4	Redefine
F1F5	Corner Point
F1F6	Delete Element
F1F7	Delete Node
F1F8	Load Circuit
F1F9	Save Circuit
F2F3	Redraw
F3F6	CTD

* Table taken from Rhine's thesis.

If the ASR 35 teletype bell sounds, the function indicated was not complete. This can occur in the case of REDEFINE, DELETE ELEMENT, and DELETE NODE. When the ARDS bell sounds, the function is completed. In attempting to redefine a node, the user may find it necessary to try several times before the correct coordinates are sent.

If the program appears to be stuck in a loop or does not function properly, try the ATTENTION function. If at some point the user is not sure of the state of the circuit, specify a REDRAW. This shows the state of the circuit in the dictionary. If the program seems to be functioning improperly, redefine a node other than the current or last node. Then redefine the last node and then the current node. An error could cause the program to lose the coordinates of the current and last node.

TABLE VIII.

Error Messages for CIRCUITS

LETTER CODE	MESSAGE	ACTION
UN	Undefined Function	The function specified does not exist. Check the function table for the correct code and retry.
SI	Point Too Close	The distance between the last node and the current node is not sufficient to draw an element. DELETE ELEMENT must be specified. Then DELETE NODE to delete the current node can be entered. Move the point farther away and try again.

IP	Incorrect Point	The last node and current node are not on a horizontal or vertical grid line. The first step is to delete the element, F1F6. Next, delete the current node, F1F7. Then specify a current node which lies on a grid line with the last node.
IC	Illegal Character	An illegal character was found. Execute F0 and try again.
CF	Changed Function	If a node was specified by the function keys and no point was defined by the mouse, when a second function is specified this warning results. Continue in a normal manner.
RS	Redraw Specification	A non-recoverable error occurred in the node or element table pointer. Restart the program. If necessary reload the program.
IR	Illegal Reference	An attempt was made to reference or delete a node or element that is not properly defined. Define properly and repeat.
DS	Deletion Specification	An attempt was made to delete a node that had an element connect to it. First delete the element and then the node.
CN	Count Exceeded	A total of four elements may be connected to one node. When the fifth is attempted this error results. An F1F6 must be executed before proceeding. This deletes the element that was to be added. Proceed in a normal manner.
DF	Dictionary Full	The storage space for the circuit information is full. No more input is accepted. The size of the circuit must be reduced.