General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

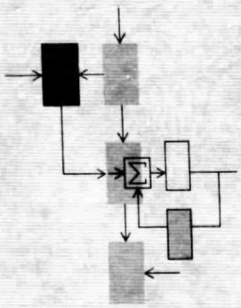
- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

Produced by the NASA Center for Aerospace Information (CASI)

May, 1969

REPORT ESL-R-381
M.I.T. DSR PROJECT 76152
NASA Grant NGL-22-009-019



CIRCAL-2: A GENERAL-PURPOSE ON-LINE CIRCUIT-DESIGN PROGRAM

USER'S MANUAL

James R. Stinger
Michael L. Dertouzos

N70-23656
(ACCESSION NUMBER) 75 (PAGES)
(THRU) 1
(CODE) 08
(CATEGORY)
CR#109363
(NASA CR OR TMX OR AD NUMBER)
FACILITY FORM 602

RECEIVED
APR 1970
NASA STI FACILITY
INPUT BRANCH

Electronic Systems Laboratory

MASSACHUSETTS INSTITUTE OF TECHNOLOGY, CAMBRIDGE 39, MASSACHUSETTS

Department of Electrical Engineering

May, 1969 Report ESL-R-381 Copy _____

CIRCAL-2
A GENERAL-PURPOSE ON-LINE CIRCUIT-DESIGN PROGRAM

USER'S MANUAL

by

James R. Stinger
Michael L. Dertouzos

Electronic Systems Laboratory
Department of Electrical Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

PRECEDING PAGE BLANK NOT FILMED.

ACKNOWLEDGMENT

The preparation and publication of this report, including the research on which it is based, was sponsored under a grant to the Electronic Systems Laboratory, Massachusetts Institute of Technology, DSR Project No. 76152. This grant is being administered as part of the National Aeronautics and Space Administration Research Grant No. NGL-22-009-019. This work was also supported in part by Project MAC, an M. I. T. research project sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Nonr-4102(01). This report is published for information purposes only and does not represent recommendations or conclusions of the sponsoring agencies. Reproduction in whole or in part is permitted for any purpose of the United States Government.

PRECEDING PAGE BLANK NOT FILMED.

CONTENTS

1. INTRODUCTION page 1

1.1 General Description of CIRCAL-2 1
1.2 Access to CIRCAL-2 5
1.3 Unexplainable Errors and/or Behavior 6
1.4 Organization of the Manual 6

2. HOW TO INPUT A NETWORK AND RELATED INFORMATION; THE FILE SYSTEM 7

2.1 Description 7
2.2 The File Types Recognized by CIRCAL-2 7

2.2.1 The Circuit File 7
2.2.2 The Nest File 12
2.2.3 The Defined Command File 13
2.2.4 The Function File 14
2.2.5 The Functional File 15

2.3 File System Commands 16

2.3.1 The DESIGN Command 16
2.3.2 The LISTF Command 17
2.3.3 The PRINTF Command 20
2.3.4 The ERASE Command 20

3. HOW TO ANALYZE A NETWORK; THE OUTPUT OF RESULTS 21

3.1 Description 21
3.2 The OUTPUT and XOUTPT Commands 21
3.3 The Mode File 22

3.3.1 Description of the Mode File 22
3.3.2 Creation of the Mode File 22
3.3.3 Updating the Mode File 22

3.4 Commands for Re-Analysis 25

3.4.1 The REPEAT Command 26
3.4.2 The ALTER Command 27

3.5 Other Output Level Commands 29

3.5.1 The FILE Command 29
3.5.2 The PRINTF Command 29
3.5.3 Commands which Return the System to the Input Level 29

3.6 Analysis Routines Available in CIRCAL-2 29

3.6.1 FREQ 29
3.6.2 PRNTDS 30

3.7 Functions of the Output Variables 31

3.7.1 Description of the Functions 31

3.7.2 Use of the Functions in Specifying the Output page 32

4. HOW TO OPTIMIZE A NETWORK AUTOMATICALLY 34

4.1 Description of the Defined Command Feature 34
4.2 The DEFCOM Primitives 34

4.2.1 The REPEAT Command 34
4.2.2 The IF THEN Statement 35
4.2.3 The GOTO Statement 37
4.2.4 The PRINT Statement 37
4.2.5 The Assignment Statement 37

4.3 Writing a Defined Command 38
4.4 Using a Defined Command 39
4.5 Suggestions to the User of the Defined Command Feature 40

5. UTILITY AND DIAGNOSTIC FEATURES 41

5.1 Utility Commands 41

5.1.1 The MANUAL Command 41
5.1.2 The TIMER Command 41
5.1.3 The QUIT Command and the Interrupt Feature 41

5.2 The CIRCAL-2 Diagnostics 42

5.2.1 The Handling of User Errors in CIRCAL-2 42
5.2.2 Explanation of the User Errors 43

5.3 Possible Difficulties and How to Avoid Them 51

5.3.1 System Errors 51
5.3.2 Other Difficulties 51
5.3.3 Reserved Words and Letters 52

6. EXAMPLE OF A SESSION WITH CIRCAL-2 56

7. CIRCAL-2 SUMMARY 61

LIST OF FIGURES

1.	The Operational Structure of CIRCAL-2	page 3
2.	The Two Command Levels of CIRCAL-2 and Their Associated Commands	4
3.	A Circuit and Its Description	12
4.	A Nest and Its Description	13
5.	A Typical User-Defined Function	14
6.	A Hysteresis Loop	15
7.	Example of the Use of the DESIGN Command	19
8.	Updating the Mode File	26
9.	Example of the Use of the ALTER Command	28
10.	Examples of Output Specifications	33
11.	Examples of IF THEN Statements	36
12.	Examples of Legal Assignment Statements	38
13.	Example of a Defined Command	39

LIST OF TABLES

1.	Generic Names of the Standard CIRCAL-2 Elements	8
2.	The Standard CIRCAL-2 Elements	10
3.	The Edit Subcommands	18
4.	The Mode File Parameters	24
5.	Arguments of the ALTER Command	27
6.	The Standard Functions of CIRCAL-2	32
7.	Explanations of the User Errors	43
8.	Reserved Letters and Reserved Words Used in CIRCAL-2	53

1. INTRODUCTION

1.1 General Description of CIRCAL-2

CIRCAL-2 is a general-purpose on-line circuit analysis program with the following main features:

a) The user may choose any one of several existing analysis routines (such as frequency, nonlinear transient, statistical, etc.) in order to analyze a network that has already been inputted.

b) Networks are inputted in a free format, and network elements may either be chosen from a repertoire of available standard elements, or may be defined in one of two ways: by specifying nonlinear functions (for example, a diode may be defined by specifying the function which yields the voltage across the diode, given the current through it); or by specifying external terminals of an already-inputted network and using this nest, as it is called, as a new element (for example, a transistor nest may be formed by interconnecting diodes and resistors into an Ebers-Moll model). In both of the above cases any desired parameters may be left unspecified until analysis time, and instances of a single defined nest with different parameter values may be used throughout the network to the extent that computer memory permits.

c) Network parameters and network topology may be changed by the user through use of simple incremental commands (which specify only the information that must be changed) prior to re-analysis of the network. Thus, for example, if a user wishes to see the effect of setting the beta of transistor T19 to 100 he would type "repeat beta19 =100", and the program would perform all the previously specified tasks such as choice of the same analysis routine as before, choice of the same analysis increments, the same variables to be saved, and the same variables to be displayed as well as the manner of their display--the only difference would be beta19 which would be set at 100.

d) Computed values are stored in a standard tabular form indexed by the independent variable (which may be time or frequency, depending on the analysis routine to be used). The user may request the output of this information, or of functions of the computed values, such as the logarithm of a voltage versus frequency, and all such output data may be plotted or printed in several different ways, e.g., one variable versus another, one or more variables versus the independent variable, and so forth.

e) The user may specify in terms of several primitive commands an optimization procedure, a DEFCOM (defined command) as it is called, which is a section of the program that behaves as a pseudo-user; that is, it inspects the computed values, changes the network parameters, re-analyzes the network, and keeps repeating this process until conditions specified in the DEFCOM are met. In other words, the DEFCOM feature replaces the user and permits the automatic optimization of a network if the user knows and can therefore specify exactly how this optimization should be conducted.

f) All information inputted into CIRCAL-2, such as networks, nests, functions, and DEFCOMs, is processed by a common text editor which permits the user to make changes and to correct errors; this information is then entered in a file directory for future use. Thus, the CIRCAL-2 program consists not only of the initial program, but of the user-generated information as well, such as new elements, transistor models, optimization commands, designed networks, commonly occurring waveforms (functions) for sources, etc. This feature makes possible the growth and adaptation of the program to the needs of specific classes of users.

g) A variety of utility features are provided to enable the user to obtain information about the program so that he need not remember operational details. Moreover, nearly 100 diagnostic messages exist for protection of the program from user errors. Means are also provided for accelerating the man-machine interaction process for the experienced user and for slowing it down for the beginner; thus, the inexperienced user may operate in a sufficiently slow mode where he is asked explicitly for all the information needed by the program,

while the experienced user may bypass these questions, abbreviate commands, and stack the information that he wishes to input in a nearly interaction-free manner.

h) Additional analysis routines may be written by users with programming experience (see the CIRCAL-2 Programmer's Manual), and may be easily added to the available routines. This feature enables the realization of savings in the writing of analysis programs, since all the "overhead" features concerning input, output, modifications, new elements, etc., are common among all the analysis routines in CIRCAL-2.

The gross operational structure of CIRCAL-2 as reflected by the above features is shown in oversimplified terms in Fig. 1.

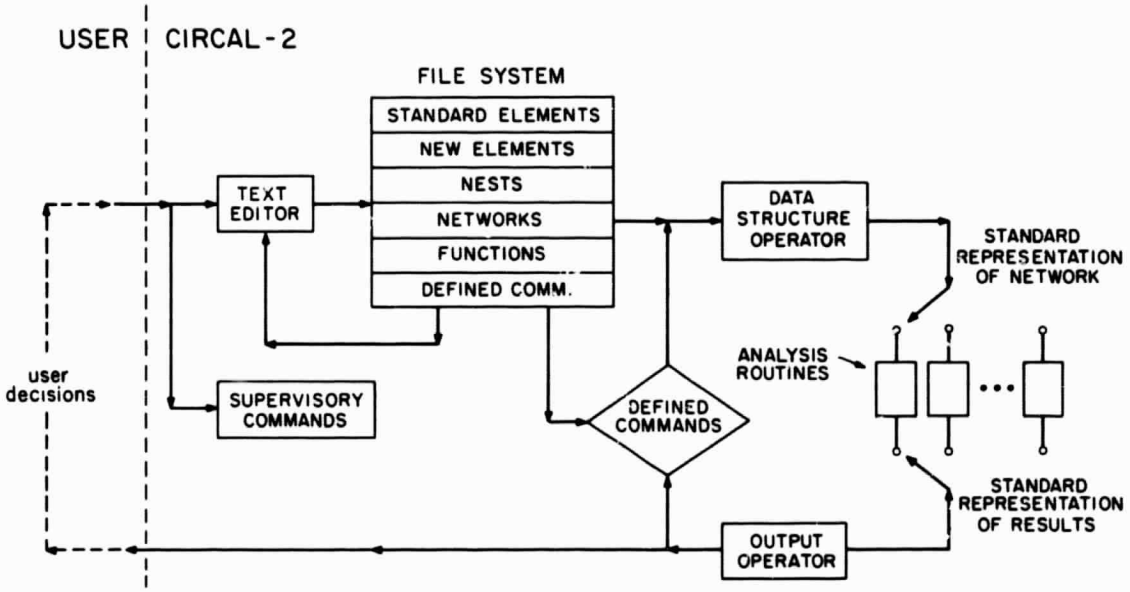


Fig. 1 The Operational Structure of CIRCAL-2

Observe that the user may introduce information into the file system through the text editor, or he may exercise control over the system through the available commands. When the user wishes to analyze a network, the information describing that network is placed in a standard form by the data-structure operator and the desired analysis routine is applied to this data structure. Analysis results are placed

in a standard output data structure, from which they are processed by the output operator for either display to the user or the pseudo-user (the defined command operator).

The gross command structure of CIRCAL-2 is shown in Fig. 2. Observe that there are two levels of commands in CIRCAL-2, the

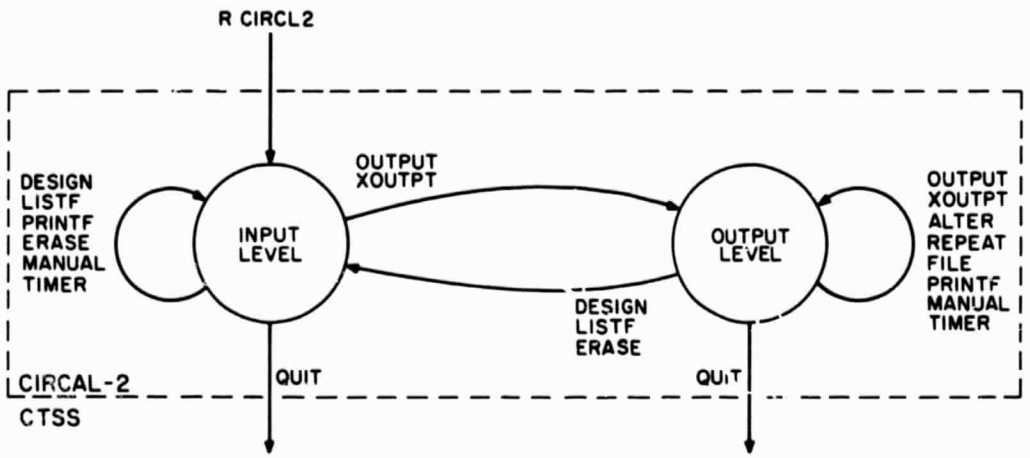


Fig. 2 The Two Command Levels of CIRCAL-2 and Their Associated Commands

input level and the output level. The input level is the level at which new circuits, nests, and functions are described to the system, and existing circuits, nests, etc., are modified through the DESIGN* command. It is also the level where inputted information may be listed by name (LISTF), printed exactly as inputted (PRINTF), or deleted from the system (ERASE), and the level where the user may obtain information about the various commands (MANUAL) or the time spent in computing (TIMER). The output level is the level at which analysis of a network is performed (OUTPUT or XOUTPT), changes in the network are made, re-analysis is conducted (REPEAT, ALTER), and analyzed networks are stored (FILE). In the output level it is also possible to give input-level commands, such as PRINTF, MANUAL, TIMER, DESIGN, LISTF, and ERASE, except

---

* All CIRCAL-2 commands and other special words or letters are given in capital letters throughout this manual to distinguish them but should be typed in lower case when using CIRCAL-2.

that the last three of the above return the system to the input level. Likewise, the analysis commands OUTPUT and XOUTPT when given at the input level take the system to the output level.

1.2 Access to CIRCAL-2

CIRCAL-2 is available to CTSS users at M.I.T. and is accessed by creating the link

LINK CIRCL2 BCD T378 5166

and giving the CTSS command

RUNCOM CIRCL2

Use of CIRCAL-2 is now initiated by typing

R CIRCL2

The program will respond with

COM-

and wait for the user to type a CIRCAL-2 command.

The following should be read before using CIRCAL-2:

a) The console tabs should be set to one every ten spaces, with the left margin starting at the tenth (or less) space.

b) During its execution, CIRCAL-2 creates several files on disk for the purpose of storing circuit descriptions, as well as analysis data. Therefore it is necessary that the user have <u>at least five records</u> available on disk before using CIRCAL-2 for the first time, and <u>at least one record available</u> for all successive times. These numbers depend on the amount of analysis data and the number of circuit descriptions to be stored, however, so that if many large circuits are to be described to the system and/or a large amount of analysis data is expected, more space should be made available in the user's disk allocation.

c) All of the CIRCAL-2 commands may be abbreviated by typing the first letter of the command only, followed by the arguments of the command, if any. One or more spaces are used as delimiters between the command and its arguments, as well as in describing a

circuit or specifying parameters for the mode file (Section 3.3). Commas also act as delimiters, and may be preferred when specifying such things as argument lists.

1.3 Unexplainable Errors and/or Behavior

Since CIRCAL-2 is still in a developmental stage and has not been used extensively, it is expected that errors are still present in the system, in spite of attempts to eliminate them. Therefore, it is requested that any unexplainable errors or unexplainable behavior be brought to the attention of any of the persons listed below:

James R. Stinger or G. Peter Jessel
M.I.T., Room 37-431
Ext. 2348

M.L. Dertouzos
M.I.T., Room 37-411
Ext. 2145

The authors would also appreciate hearing about any errors in this manual, sections in which the manual seems unclear, or where the manual and the program do not seem to agree. Assistance in using CIRCAL-2 may also be obtained from the same people.

1.4 Organization of the Manual

The first section of this manual deals with introductory material. Section 2 is concerned with the input of information into the program; Section 3 concerns network analysis and the manner in which results are outputted. Section 4 deals with the pseudo-user or defined command feature. Section 5 describes the various utility and diagnostic features, Section 6 gives an example of the use of CIRCAL-2, and Section 7 is a summary of the CIRCAL-2 structure for quick reference by the user.

The reader of this manual is asked to observe that a small number of features described in this manual are currently under implementation. These features or commands are denoted as non-operational in this text. The user should, therefore, consult the MANUAL command of CIRCAL-2 which has a current listing of the latest available commands, before attempting to use any of these commands.

2. HOW TO INPUT A NETWORK AND RELATED INFORMATION; THE FILE SYSTEM

2.1 Description

All information inputted by the user into CIRCAL-2 is placed in the CIRCAL-2 file system in the form of files. For the most part, this file system is similar to that of the Compatible Time-Sharing System (CTSS). It consists of several commands for the creation, modification, and manipulation of files. Each file is given by the user a first name and a second name when the file is created. The first name may be any combination of one to six letters, but the first names of all files created must be different. The second name indicates the type of the file and is one of CKT, NEST, DEFCOM, FNCTN, or FNCTNL, indicating that the file consists of a circuit description, a nest description, a defined command, a function, or a functional respectively.

2.2 The File Types Recognized by CIRCAL-2

2.2.1 The Circuit File

The circuit file consists of the element-by-element description of a circuit and is distinguished from other files by having the word CKT for its second name. Each element in the file is specified by first giving the nodes to which the element is connected, then the name of the element, followed by the value or other identification of the element.

Before describing a circuit to the system, the nodes should be numbered; consecutive numbers need not be used, and the nodes may be numbered in any order as long as the numbers used are distinct non-negative integers. The lowest numbered node is assumed by CIRCAL-2 to be the ground node.

The name of each element appearing in a CKT file is formed from a generic name and the instance number of that element. The names of all elements must be six characters long or less, including the instance number. Table 1 lists the standard elements of CIRCAL-2 and their generic names. To the generic name an instance

number is appended, forming a unique name for each element in the circuit. For nest instances the generic name is the same as the first name of the nest file which contains the nest description.

Table 1

Generic Names of the Standard CIRCAL-2 Elements

Standard Element	Generic Name	Standard Element	Generic Name
Linear Resistor	R	Transformer	X
Linear Inductor	L	Linear Voltage Source	E
Linear Capacitor	C	Linear Current Source	J
Nonlinear Resistor	NR	Nonlinear Voltage Source	NE
Nonlinear Inductor	NL	Nonlinear Current Source	NJ
Nonlinear Capacitor	NC	Switch	S

The value of the element is specified either as a decimal number with decimal point,* a variable whose value is to be specified at an-analysis time, a function whose description exists in a function file, or as a functional whose description exists in a functional file. The arguments of the function or functional may be decimal numbers with decimal point,* parameters whose values are set prior to analysis or the computing dimension (T or F). In addition its arguments may include specifications of network variables, which consist of the type of variable (V, I, Q or LAM) followed, in parentheses, by the name of the element where the variable applies (nests and transformers excepted). Note functions and functionals may not be arguments. By giving an element a variable value, changes in value for that element are easily made by merely specifying a new value for the variable before analysis. If two or more elements are to have the same value, and it is desired to make this value variable, the same variable name can be given as the value of all of these elements. Values for all variables in the circuit

* E format may be used, e.g., 1.0E-3 for 0.001.

are requested prior to the first analysis; only changes in value need be specified thereafter.

A nest instance may have any number of values, including zero, all of which must be specified in a single set of parentheses following the instance name. The values may either be fixed or variable and must be in the order specified in the nest description.

Table 2 gives the format for each of the standard elements of CIRCAL-2. For two-terminal elements, the voltage reference direction is assumed positive with respect to the first node given in the description; the current reference direction is into the element at this node. The first two nodes of the transformer are the nodes to which the primary is connected, the second two are the nodes to which the secondary is connected. The voltage and current conventions given above apply to each pair of nodes separately. For nests, the nodes are specified in the same order as they were in the specification of external nodes in the nest description; all currents are referenced into the nest, and all nest voltages are with respect to the ground node of the circuit.

The transformer is an ideal transformer whose value is the turns ratio (secondary to primary) and therefore must be either a decimal number or a variable. Linear dependent sources and switches are described by giving the type of dependence (V or I) followed by the name of the element, in parentheses, where the dependency occurs. The multiplicative factor, its value, can be either a decimal number or a parameter which is fixed prior to analysis, or a function whose domain includes the computing dimension (T or F) and parameters, but no network variables. For nonlinear dependent sources the domain of the function may also include one or more network variables (except those of the source itself). Dependent sources and switches can depend only on the voltage or current of a two-terminal element, i.e., not a nest or transformer. The value of the independent switch is either 0. (open) or 1. (closed) where the decimal point is necessary, or can be a variable whose value at analysis time is either 0. or 1. The value of the dependent switch is the threshold at which the switch changes state. If the voltage or current of the element where the dependency occurs according to the above convention is below the threshold, the switch is open, while if the voltage or current is above the threshold, the switch is closed. Reverse operation for a particular switch is obtained by reversing both the sign of the threshold and the order of the nodes of the element where the dependency occurs.

Table 2

The Standard CIRCAL-2 Elements

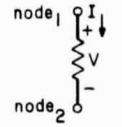
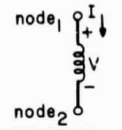
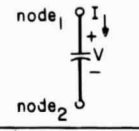
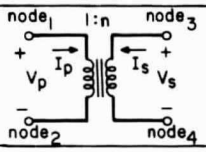
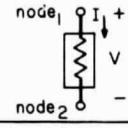
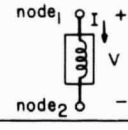
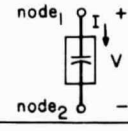
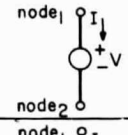
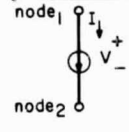
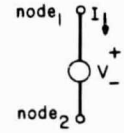
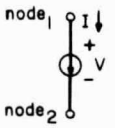
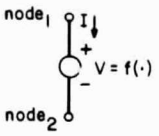
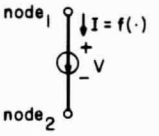
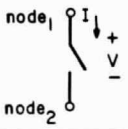
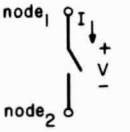
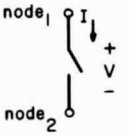
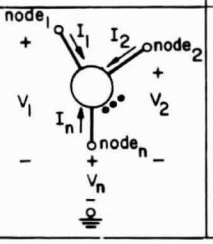
ELEMENT	SCHEMATIC	INPUT FORMAT	VALUE
Linear Resistor		$node_1$ $node_2$ Ri value	a. Decimal number b. Variable name c. fname (args)≠
Linear Inductor		$node_1$ $node_2$ Li value	a. Decimal number b. Variable name c. fname (args)≠
Linear Capacitor		$node_1$ $node_2$ Ci value	a. Decimal number b. Variable name c. fname (args)≠
Ideal Transformer		$node_1$ $node_2$ $node_3$ $node_4$ Xi value	a. Decimal number b. Variable name c. fname (args)*
Nonlinear Resistor		$node_1$ $node_2$ NRi value	a. fname (args)≠ b. fnlname (args)≠
Nonlinear Inductor		$node_1$ $node_2$ NLi value	a. fname (args)≠ b. fnlname (args)≠
Nonlinear Capacitor		$node_1$ $node_2$ NCi value	a. fname (args)≠ b. fnlname (args)≠
Independent Voltage Source		$node_1$ $node_2$ Ei value	a. Decimal number b. Variable name c. fname (args)≠
Independent Current Source		$node_1$ $node_2$ Ji value	a. Decimal number b. Variable name c. fname (args)≠
Linear Dependent Voltage Source		$node_1$ $node_2$ Ei V(elname) value $node_1$ $node_2$ Ei I(elname) value	a. Decimal number b. Variable name c. fname (args)≠

Table 2 (Cont.)

ELEMENT	SCHEMATIC	INPUT FORMAT	VALUE
Linear Dependent Current Source		$node_1$ $node_2$ Ji V(elname) value $node_1$ $node_2$ Ji I(elname) value	a. Decimal number b. Variable name c. fname (args)/
Nonlinear Dependent Voltage Source		$node_1$ $node_2$ NEi value	a. fname (args)≠ b. fnlname (args)≠
Nonlinear Dependent Current Source		$node_1$ $node_2$ NJi value	a. fname (args)≠ b. fnlname (args)≠
Independent Switch		$node_1$ $node_2$ Si value	a. Decimal number (0. or 1.) b. Variable name c. fname (args)*
Voltage-Dependent Switch		$node_1$ $node_2$ Si V(elname) value	a. Decimal number b. Variable name c. fname (args)/
Current-Dependent Switch		$node_1$ $node_2$ Si I(elname) value	a. Decimal number b. Variable name c. fname (args)/
Nest		$node_1$ $node_2$ ... $node_n$ nestnamei (args)	a. Decimal number b. Variable name

NOTES:

i is the instance number of the element.

* cannot be a function of any network variables, or any variable which changes during analysis.

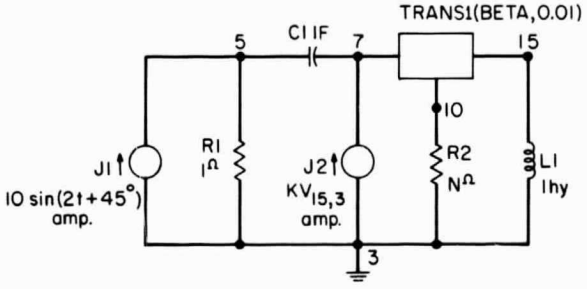
/ cannot be a function of any network variable.

≠ arguments must include one, but not both, of the element variables.

≠ arguments cannot include any of the element variables.

The value of all other elements can be a decimal number, a variable, or a function (for the linear elements R, L, C, E, and J the function cannot depend on any network variables), and, in addition, can be a functional for the nonlinear elements (NR, NL, NC).

Figure 3b gives a proper description for the circuit of Fig. 3a (the order in which the elements are described is not important). Note that the files SINF FNCTN and TRANS NEST must exist before the network can be analyzed.



(a)

```
3    5    J1    SINF(10.,2.,45.)
5    3    R1    1.
5    7    C1    1.
7    15   10    TRANS1(BETA,0.01)
3    7    J2    V(L1) K
15   3    L1    1.
10   3    R2    N
```

(b)

2.2.2 The Nest File

A nest is a circuit with external nodes specified. The nest file consists of a description of the circuit along with a statement specifying the external nodes and a statement listing the arguments of the nest, if any. It is distinguished from other files by having the second name NEST.

The first statement in any nest file must be the XNODES statement which specifies the external nodes of the nest. The form of the XNODES statement is

$$\text{XNODES } n \text{ node}_1 \ldots \text{node}_n$$

where "n" is the number of external nodes which the nest has, and "$\text{node}_1 \ldots \text{node}_n$" is a list of the numbers of the external nodes. The order in which the external nodes are listed determines the correspondence between the external nodes of the nest and the nodes to which an instance of the nest is connected in a circuit.

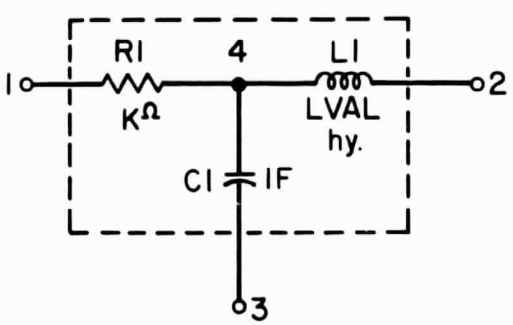
If any of the elements comprising the nest has a variable value, then that variable becomes a nest argument and must be specified

in the ARGS statement. The ARGS statement is the second statement in the nest file (only if the nest has arguments) and has the form

$$\text{ARGS k arg}_1 \ldots \text{arg}_k$$

where "k" is the number of arguments and "$\text{arg}_1 \ldots \text{arg}_k$" is a list of the names of the arguments. As with the external nodes, the order in which the arguments are listed is important since it determines the order in which values for these arguments must later be specified for any instance of the nest in a network.

The remainder of the nest file consists of descriptions of the elements comprising the nest, given exactly as in the circuit file (Section 2.2.1 and Table 2). Figure 4 shows a nest and its corresponding description (note again, that the order in which the elements are described is not important).



(a)

```
XNODES   3   1    2   3
ARGS         2    K    LVAL
1   4        RI   K
4   3        CI   1.
4   2        LI   LVAL
```

(b)

Fig. 4 A Nest and Its Description

The description of a nest may include all of the elements listed in Table 2, including instances of other nests, but may not include any instance of the nest itself.

2.2.3 The Defined Command File

For the more experienced user, CIRCAL-2 provides the facility for the definition of new commands to control the repeated analysis of a circuit until an optimal design has been reached. Such commands are described in the defined command file, which is

distinguished from other files by the second name DEFCOM. The defined command feature is described in detail in Section 4.

2.2.4 The Function File

A function file contains a user-defined function and is distinguished from other files by the second name FNCTN. User-defined functions, along with the standard CIRCAL-2 functions, may be used in:

a.) the definition of a new function;

b.) the definition of a new functional;

c.) the description of an element in a circuit or nest file;

d.) the output of analysis results; and

e.) the definition of a new command (DEFCOM).

Function files are created by the user through use of the DESIGN command. Once defined through the DESIGN command, these functions may be used at any time in any of the five items listed above. Figure 5 gives an example of a function that could be defined by the user. Note that this function involves the use of the standard square root function.

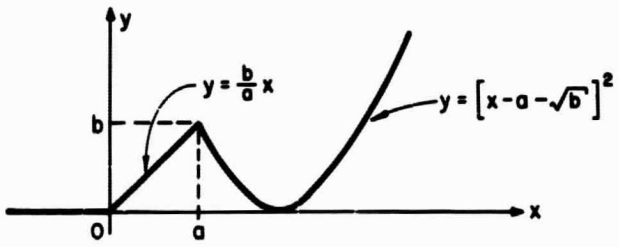


Fig. 5 A Typical User-Defined Function

At present, functions cannot be defined through the DESIGN command since the mechanism for processing user-defined functions has not been fully implemented. Functions can, however, be added to CIRCAL-2 by writing them in the AED language, compiling them, and updating the CIRCAL-2 function library with the resulting BSS code. This requires the entire system to be reloaded, however, and this procedure for adding functions to CIRCAL-2 is therefore not recommended.

The standard CIRCAL-2 functions are available for use, but only in specifying the output of analysis results (see Section 3.7). These standard functions are: ATAN, COS, COT, EXP, LOG, LOG10, MAG, PHASE, SIN, SQRT, and TAN. They are listed in Table 6 on page 32 along with restrictions on their arguments.

2.2.5 The Functional File

Elements with memory (e.g., a magnetic core) are obtained through the use of functionals in the element description. User-defined functionals are contained in functional files which are distinguished from other files by the second name FNCTNL.

The only difference between a function and a functional is that the functional must retain one or more previous values of its arguments and/or of the functional itself, in order to determine the present value of the functional, given the present values of its arguments. For example, a hysteresis loop may be considered as being composed of two functions, f_1 and f_2, as shown in Fig. 6. If x_1 is the previous value of x and x_2 is the present value, then the

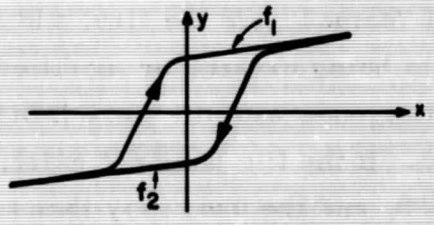


Fig. 6 A Hysteresis Loop

present value, y_2, of this hysteresis functional is

$$y_2 = \begin{cases} f_1(x_2) & \text{if} \quad x_2 - x_1 > 0 \\ f_2(x_2) & \text{if} \quad x_2 - x_1 < 0 \\ y_1 & \text{if} \quad x_2 - x_1 = 0 \end{cases}$$

where y_1 is the previous value of the functional. Thus, both the previous value of x and the previous value of y are necessary, in

addition to the present value of x, to determine the present value of y. For some functionals, more than one previous value may be required to determine the present value of the functional.

User-defined functionals, along with the standard CIRCAL-2 functionals, may be used in:

a.) the definition of a new functional;

b.) the description of an element in a circuit or nest file; and

c.) the output of analysis results.

At present functionals are not allowed in CIRCAL-2, but provision for the definition of functionals at the input/edit level will be included in future versions of CIRCAL-2, along with a few standard functionals.

2.3 File System Commands

2.3.1 The DESIGN Command

The DESIGN command is used to create or modify files. Its format is

DESIGN name1 name2

where "name1" and "name2" are the first and second names respectively of the file being created or modified. If the file already exists the system responds with the word EDIT, and the user may then modify the file. If the file does not exist, the system responds with the word INPUT, and the user may then type in the file.

In the input mode the user may type anything since no error checking is done at this time. In general, however, the user will type a circuit description, a nest description, a defined command description, a function, or a functional. Any number of lines may be typed in this mode; after the last line has been typed (or at any time when it is desired to modify what has so far been typed) an extra carriage return causes the system to enter the edit mode and respond with the word EDIT.

In the edit mode, the user may modify any line in the file by using the edit subcommands. To modify a line it must first be located, then the modifications made by replacing part of the located line. The system keeps track of the line being referred to by

maintaining a "pointer" to that line. This pointer is moved by the edit subcommands (except for the print and change subcommands). In addition to modifying a line, any number of lines may be deleted and any number of lines may be printed, beginning with the located line.

The edit subcommands and their meaning are given in Table 3. All of the edit subcommands, except for the top subcommand, produce a response from the system. Any number of the edit subcommands may be used in editing a file.

If the user is in the edit mode and desires to insert one or more lines at any point in the file, the procedure is as follows: first, move the pointer to the line just before the point where the lines are to be inserted by using either the locate or the next subcommand (or the top subcommand if the lines are to be inserted before the first line in the file), then give an extra carriage return to enter the input mode. Now the lines to be inserted may be typed. When all lines have been typed, an extra carriage return returns the user to the edit mode.

After all inputting and editing has been completed, the design process is terminated by giving the file subcommand while in the edit mode. If it is desired to store the file under a new name, the new first and second names of the file are given as arguments to the file subcommand and the file will be stored under this new name. If no arguments to the file subcommand are given, the file is stored under the old name.

Figure 7 gives an example of the use of the DESIGN command.

2.3.2 The LISTF Command

The LISTF command is used to obtain a list of the names of all (or a subset) of the existing files. If a list of all files in the directory is desired, the format of the command is

```
LISTF
```

If a listing of only the file "name1 name2" is desired, the format of the command is

```
LISTF    name1    name2
```

Table 3

The Edit Subcommands

Subcommand	Format	Response	Meaning
top	T	None	Move pointer to top of file
next	N k	Line at which pointer stops or INPUT is printed	Advance pointer k lines from current line; if k is greater than the number of lines in the file after the current line, the pointer is advanced to the bottom of the file and the input mode is entered; k is assumed to be 1 if unspecified.
print	P k	k or fewer lines are printed	Print k lines of the file beginning with the current line; if k is greater than the number of lines in the file after the current line, only the remaining lines are printed; k is assumed to be 1 if unspecified; the pointer is not moved.
delete	D k	O.K. or END OF FILE REACHED	Delete k lines of the file beginning with the current line; if k is greater than the number of lines in the file after the current line, only the remaining lines are deleted; k is assumed to be 1 if unspecified; the pointer is moved to the line preceding the first line deleted.
change	C qstring1qstring2q	Line after changes have been made is printed	Change first occurrence (left to right) of string1 to string2 in current line; string1 is any character string in the current line, string2 is any character string, q is any character not in string1 or string2; current line is unchanged if string1 cannot be found in the current line; the pointer is not moved.
locate	L string	Line with first occurrence of string is printed	Beginning with the line following the current line, search the file for an occurrence of string; if the end of the file is reached before locating string, the search is resumed at the top of the file and continued to the current line; string is any string of characters in the file; the pointer is moved to the first line encountered containing string or is not moved if string is not found.
file	F name1 name2 or FILE name1 name2	*	Store the file being modified in the master file under the name name1 name2; if name1 and name2 are not specified, the file is stored under the name specified when the DESIGN command was given, deleting the old version of the file, if any; the system returns to command status.

```
COM- design simple ckt
INPUT
1  0  j1  1.
1  0  r1  1.
1  0  c1  1.

EDIT
1  r1
1  0   R1  1.

c  /1./k/
1  0  R1  K

file
*

COM-
```

(a)

```
COM- design simple ckt
EDIT
n  2
1  0  C1  1.
d
END OF FILE REACHED

INPUT
1  2  11  1.

EDIT
1  r
1  0  R1  K

c  /1/2/
2  0  R1  K

t
p  10

1  0  J1  1.
2  0  R1  K
1  2  L1  1.
f  nother ckt
*

COM-
```

(b)

Fig. 7 Example of the Use of the DESIGN Command to (a) Create a File and (b) Modify an Existing File

For a listing of all files with first name "name1" the format of the command is

LISTF name1 *

while for a listing of all files with second name "name2" the format of the command is

LISTF * name2

Information such as the length of the file in words is given for each file listed.

2.3.3 The PRINTF Command

The PRINTF command is used when it is desired to print the contents of a file on the console. The format of the PRINTF command is

PRINTF name1 name2

where "name1" and "name2" are the first and second names respectively of the file whose contents are to be printed.

2.3.4 The ERASE Command

The ERASE command is used when it is desired to delete an entire file. The format of the ERASE command is

ERASE name1 name2

where "name1" and "name2" are the first and second names respectively of the file to be deleted. The file directory is updated accordingly when a file is deleted.

3. HOW TO ANALYZE A NETWORK; THE OUTPUT OF RESULTS

3.1 Description

To analyze a network the user must first transfer the system to the output level by giving either the OUTPUT or XOUTPT command. The mode file for the circuit being analyzed is then created and the user is allowed to make modifications and additions to the information in the mode file. Now the circuit is analyzed and the results displayed, and the system returns to command status at the output level. At this point any CIRCAL-2 command may be given, but in general commands for altering and re-analyzing the network will be used until a satisfactory design has been achieved. To analyze a different network when the system is already at the output level, the OUTPUT or XOUTPT command must be given again for the new network.

3.2 The OUTPUT and XOUTPT Commands

The OUTPUT or the XOUTPT command is used when it is desired to analyze a circuit.* The format of the OUTPUT command is

OUTPUT namel $\text{analysis}_1 \ldots \text{analysis}_n$

where "namel" is the first name of the file containing the description of the circuit to be analyzed and "$\text{analysis}_1 \ldots \text{analysis}_n$" is an optional list of the analysis routines to be used in analyzing the circuit (see Section 3.6). If this list of analysis routines is given, the circuit to be analyzed is handled more efficiently; however, for small networks the difference is negligible. The OUTPUT command is used for updating the mode file in the slow mode (Section 3.3.3); to update the mode file in the fast mode, the XOUTPT command is used. The format is

---

* If a series of consecutive analyses are to be performed on a circuit, the OUTPUT or XOUTPT command is used only for the first analysis. Successive analyses are performed using the REPEAT command (see Section 3.4.1).

XOUTPT namel $analysis_1 \ldots analysis_n$

where "namel" and "$analysis_1 \ldots analysis_n$" are as described above.

3.3 The Mode File

3.3.1 Description of the Mode File

The mode file is created by the system prior to the first analysis of a circuit. It contains values for the analysis and output start points, the analysis and output end points, and the analysis and output increments. It also contains the names of the variables to be saved for output, the names of the variables to be plotted or printed, the name of the analysis routine to be used, and the independent variable for analysis.

The purpose of the mode file is to act as a storage for these parameter values, accessible by the analysis and output routines, so that values for the parameters in the mode file need only be specified when a change in value is desired. This eliminates the need to specify values for all of the parameters before each analysis.

3.3.2 Creation of the Mode File

A mode file is created for each different circuit analyzed during a session with CIRCAL-2. It is created prior to the first analysis of the circuit by making a copy of the preset mode file which contains preset values for all of the parameters except for the name of the analysis routine, the names of the variables to be saved for output, and the names of the variables to be plotted or printed. Values for the parameters with no preset value must be specified prior to the first analysis; values for the other parameters need only be specified if a value other than the preset value is desired.

Once the mode file has been created, it is saved for use in future analyses of the circuit to which it corresponds (during the current session with CIRCAL-2 only).

3.3.3 Updating the Mode File

There are two modes in which the mode file parameters may be updated: the slow mode and the express mode. The slow mode of mode file update is designed for the beginning user of CIRCAL-2 who

is not familiar with the contents of the mode file and the manner in which parameter values are specified. In the slow mode, the value of each parameter is requested by the system until all values have been specified by the user. Updating of the mode file takes longer in this mode because of the increased interaction between the user and the program. The express mode of mode file update is designed for the more experienced user who is familiar with the mode file parameters and with the manner of specifying their values. In the express mode, the user types values for all parameters he wishes to change, without waiting for a system response. This results in less interaction between the program and the user and hence, faster updating of the mode file.

If the OUTPUT command was given in transferring to the output level, the system will ask for the value of each of the mode file parameters with statements such as "TYPE ANALYSIS INCREMENT, AI." Table 4 lists the mode file parameters along with their preset values and formats for specifying a change in value. As the value of each parameter is requested, the user must either type in a value according to the proper format or, if no change from the preset value is desired, give a carriage return. A value must be given, however, for those parameters listed in Table 4 as having no preset value.

If the XOUTPT command was given in transferring to the output level, the system will respond with "TYPE PARAMETERS." Values for all mode file parameters in which a change from the preset value is desired can now be typed, following the formats given in Table 4. Spaces or commas may be used as delimiters and if more than one line is required to type the values, each line except the last is ended with a "$" to indicate continuation. If none of the mode file parameters are to be changed, just a carriage return is given.

Values for PI, PS, and PE need only be specified if they are to be different than the values for AI, AS, and AE since whenever AI, AS, and AE are specified, PI, PS, and PE are set to the same value. Therefore, values for the output parameters, when specified, must be given after values for the analysis parameters have been specified. Also, the variables to be saved for output must be specified before

Table 4

The Mode File Parameters

Parameter	Preset Value	Format for Specifying a Change in Value	Comments
Analysis routine	None	analysisname	analysisname is the name of the available analysis routines (see Section 3.6)
Analysis increment	0.1	AI = newval	newval is a fixed decimal number (with decimal point); E format may be used (e.g., 1.0E-3 for 0.001)
Analysis start point	0.0	AS = newval	newval is the same as for AI
Analysis end point	10.0	AE = newval	newval is the same as for AI
Independent variable of analysis	T	IV = indvar	indvar is either the letter T (time) or the letter F (frequency)
Variables to be saved for output	None	SAVE n var_1 ... var_n	n is the number of variables to be saved for output; var_i is the name of a variable to be saved for output (see Section 3.6)
Output increment	0.1	PI = newval	newval is the same as for AI
Output start point	0.0	PS = newval	newval is the same as for AI
Output end point	10.0	PE = newval	newval is the same as for AI
Variables to be plotted or printed	None	outype n $f_1(a_{11}, \ldots, a_{1m}), \ldots, f_n(a_{n1}, \ldots, a_{np})$ or PRINT n $f_1(a_{11}, \ldots, a_{1m}), \ldots, f_n(a_{n1}, \ldots, a_{np})$VS $f_{n+1}(a_{n+1,1}, \ldots, a_{n+1,q})$	outype is either the word PLOT or the word PRINT; n is the number of things to be plotted or printed and must be 1 for plotting, ≤ 5 for printing; $f_i(a_{i1}, \ldots, a_{ij})$ is the specification for a function (see Section 3.7) which may be the identity function
Network Parameter	None	parname = newval	parname is the name of the network parameter; newval is the same as for AI

the variables to be plotted or printed (see Section 3.7 for a more detailed explanation of the specification of output variables).

If an error or an interrupt occurs during the updating of the mode file in either mode, the system returns to command status at the output level. Updating of the mode file may then be continued from the point where the error or interrupt occurred by using the REPEAT command (Section 3.4.1), or the OUTPUT or XOUTPT command may be given again.

After all updating of the mode file has been completed, a check is performed to make sure values have been specified for all parameters whose value cannot be preset. If any of these parameters has not been given a value, the system issues a statement similar to that used in the slow mode of update and the user must then specify a value for the parameter.

Figure 8 gives an example of the updating of the mode file in both the slow mode and the fast mode.

Following the updating of the mode file, the circuit is analyzed using the information in the mode file and the results of analysis are either plotted or printed.

3.4 Commands for Re-Analysis

Normally the user will re-analyze the network through use of the REPEAT command, followed by the mode file changes that he wishes to make. In order for this to be possible, however, the user must have had the foresight to declare as variable the element values that he wishes to change (e.g., the value of resistor R19 could be set to K, and the user could then give a new value for K prior to re-analysis).

Occasionally, however, the user may wish to make changes which were not initially planned. This is possible through the ALTER command, as explained below, which operates directly on the network data structure. Note that changes performed via the ALTER command are not reflected in the network file (CKT).

For any significant changes the user should use the DESIGN command to edit the appropriate network. Note that where this alternative is exercised, some delay is experienced since the network data structure must be recreated. The mode file is saved, however.

```
COM- output simple
TYPE ANALYSIS ROUTINE
freq
TYPE ANALYSIS INCREMENT, AI
ai=0.25
TYPE ANALYSIS START POINT, AS
as=1.
TYPE ANALYSIS END POINT, AE
ae=6.
TYPE INDEPENDENT VARIABLE, IV
iv=f
TYPE VARIABLES TO BE SAVED FOR OUTPUT
save 2 vr 1 vi 1
TYPE OUTPUT INCREMENT, PI
pi=0.5
TYPE OUTPUT START POINT, PS

TYPE OUTPUT END POINT, PE
pe=5.
TYPE OUTPUT
plot 1 mag(vr 1, vi 1)
TYPE VALUES FOR THE NETWORK PARAMETERS, IF ANY
k=2.

GRAPH OF
```

(a)

```
COM- xoutpt simple

TYPE PARAMETERS
iv=f freq save 2 vr 1 vi 1 as=1. ai=.25 ae=6. pi=.5 $
pe=5. k=2. plot 1 mag(vr 1 vi 1)

GRAPH OF
```

(b)

Fig. 8 Updating the Mode File in (a) The Slow Mode and (b) The Fast Mode. The circuit "simple" is given in Fig. 7a.

In summary, the user should try to make changes in the network through the REPEAT command. If this is not possible it is recommended that he use the DESIGN command.

3.4.1 The REPEAT Command

The REPEAT command is used to change any of the mode file parameters and to re-analyze the circuit. The format of the REPEAT

command is

REPEAT $par_1 \ldots par_n$

where "$par_1 \ldots par_n$" are mode file parameter specifications, i.e., any of the formats of Table 4, or any variables declared by the user upon design of a CKT. The "$" may be used to indicate continuation on the next line if necessary. This command causes the mode file to be updated in the fast mode, the circuit to be re-analyzed, and the analysis results to be plotted or printed. If an error is detected in any of the "par_i", the REPEAT command may be used again to continue on from the point where the error occurred.

3.4.2 The ALTER Command (not operational in some versions of CIRCAL-2)

The ALTER command is used to make changes in the network being analyzed without having to leave the output level (as is the case if the DESIGN command is used to make the changes). The format of the ALTER command is

ALTER $filename_1$ $argset_1 \ldots filename_n$ $argset_n$

where "$filename_i$" is the first name of the file containing the description of the circuit or nest where the changes indicated by "$argset_i$" are to be made, and "$argset_i$" consists of any number of pairs of the form "op_{ij} $opargs_{ij}$" where "op_{ij}" and "$opargs_{ij}$" are explained in Table 5.

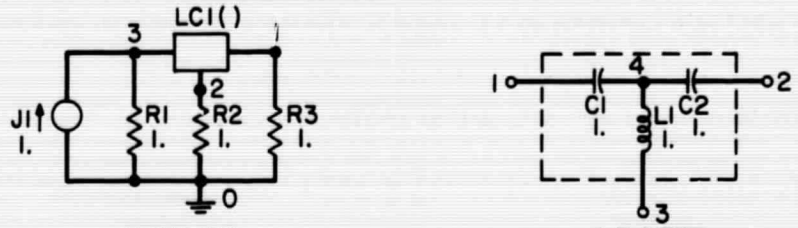
Table 5

Arguments of the ALTER Command

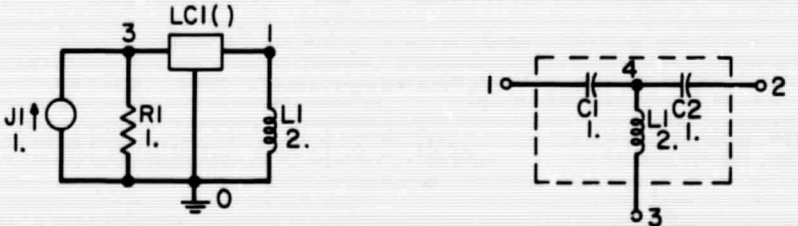
op_{ij}	$opargs_{ij}$	Meaning
C	elname newvalue	Change the value of element "elname" to "newvalue".
D	elname	Delete element "elname" from the network.
A	$node_1 \ldots node_k$ elname value	Add element "elname" between nodes "$node_1 \ldots node_k$" with value "value".
R	$elname_1$ $elname_2$ value	Replace element "$elname_1$" by element "$elname_2$" with value "value".

It is necessary to specify the name of the circuit or nest when the ALTER command is used since changes are allowed both in the main circuit and in any nests used. These changes are made only in the network data structure, however, and are not made in the description of the circuit or nest. Therefore the only record of the changes made is the sequence of ALTER commands used to make the changes.

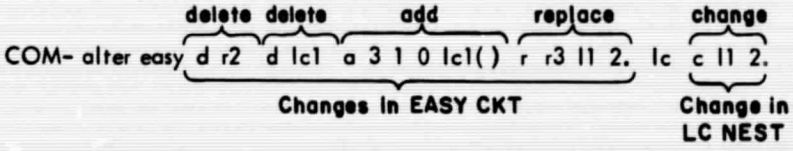
Should more than one line be required for the ALTER command, a "$" must be given at the end of each line except the last, to indicate continuation. Figure 9c illustrates the use of the ALTER command in changing the circuit of Fig. 9a to that of Fig. 9b.



(a)



(b)



(c)

Fig. 9 Example of the Use of the ALTER Command (c) To Change the Circuit of (a) To the Circuit (b)

3.5 Other Output Level Commands

3.5.1 The FILE Command (not operational)

The FILE command allows data structures to be saved so that they need not be created each time analysis is performed. It also allows changes made by the ALTER command on the data structure to be reflected by corresponding changes in the circuit or nest description; the latter is then saved as a new circuit or nest.

3.5.2 The PRINTF Command

The PRINTF command has already been described in connection with the CIRCAL-2 file system (Section 2.3.3), and is described here as well since at the output level this command can also be used to print the contents of the mode file. The format of the PRINTF command is

PRINTF name1 name2

where "name1" and "name2" are the first and second names respectively of the file whose contents are to be printed. If "name1" is MODE and "name2" is FILE then the contents of the mode file for the circuit being analyzed will be printed.

3.5.3 Commands which Return the System to the Input Level

While all CIRCAL-2 commands may be given at the output level, several of them return the system to the input level when given at the output level. The commands which do this are the DESIGN command (Section 2.3.1), the LISTF command (Section 2.3.2), and the ERASE command (Section 2.3.4). The OUTPUT and XOUTPT commands (Section 3.2), when given at the output level, take the system to the input level where execution of the command returns the system to the output level.

3.6 Analysis Routines Available in CIRCAL-2

3.6.1 FREQ

FREQ is a linear frequency analysis routine which performs nodal analysis by matrix inversion. The elements allowed in any circuit to be analyzed by FREQ are: linear, time-invariant resistors, inductors, and capacitors; independent current sources; voltage

dependent current sources (linearly dependent only); and nests. Values for all elements must be either fixed decimal numbers with decimal point* or variables whose values are to be specified at analysis time. No functions or functionals are allowed. No restriction is placed on the number of levels of nesting allowed.

The output of FREQ is in the form of the real and imaginary parts of the voltage at any node with respect to the ground node (taken to be the node with the lowest number) versus frequency. The real part of the voltage at node "n" with respect to ground is specified by "VR n" while "VI n" specifies the imaginary part of the voltage at that node. To obtain the magnitude and phase of the voltage at a node, the MAG and PHASE functions must be used (see Section 3.7).

No analysis is performed by FREQ for negative frequencies so AI, AS, and AE must all have positive values and AE must be greater than AS. If AS is negative or zero it will automatically be incremented by AI until it is positive, at which point analysis begins. Analysis continues until a frequency which is greater than (AE-AI) is reached, which is the final analysis point.

To use FREQ in analyzing a circuit, the word "FREQ" is given as the name of the analysis routine to be used, and the letter "F" is given as the value of the independent variable, IV, when updating the mode file (Section 3.3.3). AI, AS, AE, PI, PS, and PE must be specified in terms of frequency; the variables to be saved for output and the variables to be plotted or printed must be specified in terms of the real and imaginary parts of the voltage at a node with respect to ground, as indicated above.

3.6.2 PRNTDS

PRNTDS is not an analysis routine but is used in the same manner. It is a routine which prints on-line the data structure created by the system for the circuit being analyzed, and is included for use when a new analysis routine is being incorporated into CIRCAL-2.

---

* E format may be used, e.g., 1.0E-3 for 0.001.

The output of PRNTDS is a printout of the data structure, in octal, with headings and values of pointers added to aid in interpreting the printout. Since the printout of the data structure for even a small network (five elements) takes several pages, the PRNTDS routine should be used only when necessary, although the interrupt feature (Section 5.1.3) can be used to terminate the printout.

The mode file is not used by the PRNTDS routine, but since PRNTDS is treated as if it were an analysis routine, values for the mode file parameters which cannot be preset must be specified when using PRNTDS. If another analysis routine is to be used on the circuit in addition to PRNTDS, values for the mode file parameters which apply to that analysis routine can be given, otherwise the dummy statements "SAVE 1 X 1" and "PRINT 1 X 1" will satisfy the system (see Section 3.3.3). The word "PRNTDS" is given as the name of the analysis routine to be used when a printout of the data structure is desired.

3.7 Functions of the Output Variables

3.7.1 Description of the Functions

The CIRCAL-2 user is allowed not only to plot or print the results of analysis directly but to plot or print functions of the results as well. These functions are specified when updating the mode file and may be used alone or in combination with output variables which are not arguments of any function. Each function specification must be of the form

$$f(a_1, \ldots, a_n)$$

where f is the name of the function and $a_1, \ldots, a_n$ are the arguments of the function. Each a_i must be the name of a variable saved for output or the independent variable of analysis (the commas are optional).

Table 6 lists the standard functions which are available for use in specifying the output. New functions may be defined by the user, as explained in Section 2.2.4. There are only two standard functions having more than one argument, and of these two the order of the arguments is important only for the PHASE function. Here the first

argument must be the real part of the variable and the second argument must be the imaginary part.

Table 6

The Standard Functions of CIRCAL-2

Function Specification	Description	Restrictions
ATAN(x)	Arctangent of x	None
COS(x)	Cosine of x	None
COT(x)	Cotangent of x	$n\pi < x < (n+1)\pi$ $(n=0, \pm1, \pm2, \ldots)$
EXP(x)	e^x	None
LOG(x)	$\log_e x$	$x > 0$
LOG10(x)	$\log_{10} x$	$x > 0$
MAG(x_1,x_2)	$\sqrt{x_1^2 + x_2^2}$	None
PHASE(x_1,x_2)	$\tan^{-1} x_2/x_1$	None
SIN(x)	Sine of x	None
SQRT(x)	$\sqrt{x}$	$x \geq 0$
TAN(x)	Tangent of x	$\frac{n\pi}{2} < x < (\frac{n}{2}+1)\pi$ $(n = \pm1, \pm3, \ldots)$

3.7.2 Use of the Functions in Specifying the Output

There are two ways of specifying the output of analysis results, one for plotting or printing the results versus the independent variable of analysis, the other for printing the results versus a function of the independent variable or versus a function of the saved variables.

To specify the output versus the independent variable only, the format is

$$\text{outype } n\ f_1(a_{11}, \ldots, a_{1m}), \ldots, f_n(a_{n1}, \ldots, a_{np})$$

where "outype" is either the word PRINT or the word PLOT, "n" is the number of functions to be printed or plotted versus the independent variable, and the $f_i(a_{i1}, \ldots, a_{ij})$ are specifications of the functions as described in Section 3.7.1. Any or all of the functions may be identity functions of one argument, i.e., an a_{ik} by itself. Up to five functions may be printed versus the independent variable, only one function may be plotted at a time.

To specify the printing of functions of the output variable versus a function of the independent variable or versus a function of the saved variables the format is

$$\text{PRINT } n\ f_1(a_{11}, \ldots, a_{1m}), \ldots, f_n(a_{n1}, \ldots, a_{np})\ \text{VS}\ f_{n+1}(a_{n+1,1}, \ldots, a_{n+1,q})$$

where "n" is the number of functions to be printed versus the (n+1)st function, and the $f_i(a_{i1}, \ldots, a_{ij})$ are the function specifications. A maximum of five functions may be printed versus a function and no plotting versus a function is allowed. Again, any of the functions may be identity functions.

Figure 10 gives some examples of correct statements for specifying the output using functions, provided the network variables VR 1, VI 1, VR 2, VI 2, and VI 4 have previously been specified as being saved for output.

(a) PRINT 3 MAG(VR 1, VI 1), VR 2, EXP(VI 4)
(b) PLOT 1 PHASE(VR 2, VI 2)
(c) PRINT 2 VI 1, SIN(VI 2) VS LOG(F)
(d) PRINT 1 VI 4 VS VI 2

Fig. 10 Examples of Output Specifications (commas are optional)

4. HOW TO OPTIMIZE A NETWORK AUTOMATICALLY

4.1 Description of the Defined Command Feature

The defined command feature allows the user to define his own commands to automatically control the repeated analysis of a circuit until specified conditions are met by the results of analysis. A defined command, therefore, provides the facility for incorporating network optimization procedures into CIRCAL-2. Once defined, such a command is used in the same manner as is a standard CIRCAL-2 command.

Defined commands are constructed from the following (DEFCOM) primitives: the REPEAT command, the IF THEN statement, the GOTO statement, the PRINT statement, and the assignment statement, all of which are explained in Section 4.2. Appropriate combinations of these primitives control the modification of network parameters and the corresponding re-analysis of the circuit, making decisions which would normally be made by the user. Printing or plotting of the analysis results is suppressed for all but the last analysis of the circuit called for by the defined command (i.e., following satisfaction of the necessary conditions). Defined commands may have any number of arguments.

Writing a defined command is essentially the same as writing a computer program and hence, involves the same pitfalls and drawbacks. Therefore, experience in programming as well as experience in using CIRCAL-2 is a prerequisite to using the defined command feature.

4.2 The DEFCOM Primitives

4.2.1 The REPEAT Command

The REPEAT command, as described in Section 3.4.1, updates the mode file and causes re-analysis. When used as a DEFCOM primitive, however, this command can have only network parameters as arguments; no other mode file parameters may be changed. These network parameters can be specified in any of the forms taken

by the assignment statement described in Section 4.2.5. The format for the REPEAT command, when used in defining a new command, is

REPEAT $par_1 \ldots . par_n$

where "$par_1 \ldots par_n$" are assignment statements involving only network parameters and arguments to the defined command (see Section 4.3). For example, "REPEAT K = K + B" is a correct use of the REPEAT command where K is a network parameter and B is an argument of the command.

4.2.2 The IF THEN Statement

The IF THEN statement is used when it is desired to perform some action conditional upon the value of a boolean expression. The format of the IF THEN statement is

IF boolean expression THEN statement

where "boolean expression" is an expression whose value is either true or false and "statement" is any DEFCOM primitive.

Only one form for the boolean expression in the IF THEN statement is allowed. It involves comparing the value of one variable with the value of another variable and has the format

var1 op var2

where "var'" and "var2" are either fixed decimal numbers with decimal point*, network parameters, arguments of the defined command, or specifications of analysis data, and "op" is one of the six boolean operators: GRT(>), GEQ($\geq$), EQL(=), NEQ($\neq$), LEQ($\leq$), and LES(<). Specifications of analysis data are of the form

varname (analysispt)

where "varname" is the name of a network variable saved for output and "analysispt" is the analysis point (i.e., the value of the independent variable) at which the value of the network variable is to be taken. The analysis point at which the value of the network variable is to be taken may be either a fixed decimal number (with decimal

* E format may be used, e.g., 1.0E-3 for 0.001.

point)* or an argument of the defined command, but not a network parameter, and must lie between AS and AE. For example, the boolean expression

VR 1 (2.5) GRT 5.

will be true only if the value of the network variable VR 1 is greater than 5 when the value of the independent variable of analysis is 2.5. If the analysis point and value are to vary from one use of the command to the next, they may be specified as arguments to the command in which case the expression becomes

VR 1 (Y) GRT Z

where Y and Z have been previously specified as arguments of the defined command. As another example, the boolean expression

K LEQ B

where K is a network parameter and B is an argument of the command, will be true only if the value of K is less than or equal to the value of B.

The statement following the word THEN in the IF THEN statement can be any of the DEFCOM primitives, including another IF THEN statement. It is executed only if the value of the boolean expression is true and is skipped over otherwise.

Some examples of correct IF THEN statements are given in Fig. 11.

(a) IF VI 4 (6.) EQL -3. THEN GOTO DONE
(b) IF VAL GEQ VR 10 (3.5) THEN REPEAT K = K + 1
(c) IF PAR LES 0. THEN PAR = 1.
(d) IF 7. LEQ K THEN IF PAR NEQ K THEN GOTO LOOP

Fig. 11 Examples of IF THEN Statements (VAL, K, and PAR are either network parameters or arguments of the defined command, DONE and LOOP are statement labels.)

---

* E format may be used, e.g., 1.0E-3 for 0.001.

4.2.3 The GOTO Statement

When an unconditional transfer to a statement other than the next statement to be executed is desired, the GOTO statement is used. The format of the GOTO statement is

GOTO label

where "label" is a label which has been assigned to the statement to which a transfer is to be made. A label can be any combination of letters and numbers, six characters or less, and beginning with a letter. The manner in which a label is assigned to a statement is explained in Section 4.3.

4.2.4 The PRINT Statement

The PRINT statement allows the on-line printing of values for any network parameters or arguments of the defined command. The format for the PRINT statement is

PRINT n $var_1 \ldots var_n$

where "n" is the number of variables whose values are to be printed and "$var_1 \ldots var_n$" is a list of the variable names. Each variable value is printed on a separate line along with its name. For example, the statement

PRINT 2 PAR VAL

will cause the values of PAR and VAL to be printed.

4.2.5 The Assignment Statement

The assignment statement is used to change the value of either a network parameter or an argument of the defined command. Only simple assignment statements of the form

var1 = var2 op var3

are allowed, where "var1" is either a network parameter or an argument of the command, "var2" and "var3" are either network parameters, arguments of the command, constants, or specifications of analysis data, and "op" is one of the arithmetic operators: +, -, *, or /. One or more spaces must appear on either side of the operator in the assignment statement to be interpreted correctly. Minus signs

in front of variable names are ignored completely. Thus the assignment statement "K = K - B" will be interpreted as "K = K + B". Specifications of analysis data are explained in Section 4.2.2.

No other form of assignment statement is allowed. Some examples of legal assignment statements are given in Fig. 12. Notice that the operator and the third variable can be missing, as in Fig. 12a.

(a) K = 1.
(b) K = K + B
(c) B = 2. / K
(d) B = VR 3 (1.) * VR 3 (1.)

Fig. 12 Examples of Legal Assignment Statements (K is a network parameter, B is a DEFCOM argument.)

4.3 Writing a Defined Command

Writing a defined command consists of combining the DEFCOM primitives with labels and certain special symbols to form a sequence of executable statements which, when executed, perform some useful function. The defined command is described to the system through the DESIGN command (Section 2.3.1) and is kept in a file with second name DEFCOM. The first name of the file is the name of the command.

A defined command may have any number of arguments, the values of which are given each time the command is used. These arguments must be specified in the description of the command by the ARGS statement, which has the form

$$\text{ARGS } k \text{ } arg_1 \ldots arg_k$$

where "k" is the number of arguments and "$arg_1 \ldots arg_k$" is a list of the names of the arguments. If a defined command has any arguments, this statement must be the first statement in the command description.

All statements in the description of a defined command must end with a right parenthesis. This is necessary since the description is free-format. Labels are assigned to a statement by writing the name of the label, followed by an apostrophe, directly preceding the statement to which it is to be assigned. The last statement to be executed

in the description of the command must be the END statement. It consists merely of the word END followed by a right parenthesis.

Since a label may be assigned to any statement in a command description, labels may be assigned to the statement following the word THEN in an IF THEN statement, if desired. A statement preceded by a label is also a statement and may be assigned a label as well.

An example of a defined command description is given in Fig. 13.

```
            ARGS 1 B )
            K = B )
LOOP '      REPEAT  K = K + 1.)
            IF VI 1 (2.) GEQ -0.1 THEN GOTO PRNT)
            IF K GRT 9. THEN GOTO DONE)
            GOTO LOOP)
PRNT '      PRINT 2 B K)
DONE '      END)
```

Fig. 13 Example of a Defined Command (K is a network parameter.)

4.4 Using a Defined Command

Defined commands may only be used at the output level after the mode file has been updated and the network on which the defined command is to be used has been analyzed one or more times.

To use a defined command on a circuit, the OUTPUT or XOUTPT command for that circuit must be given first. The mode file is then updated with the values for the parameters which are to be used during the execution of the defined command since they cannot be changed by the defined command. After the analysis of the circuit and the output of analysis results, the system returns to command status at the output level, at which point the defined command may be used. This is done by giving the name of the command followed by values, in order, for any arguments. These values must be fixed decimal numbers (with decimal point)*. The name of the

---

* E format may be used, e.g., 1.0E-3 for 0.001.

defined command is the first name of the file with second name DEFCOM which contains the description of the defined command.

As an example of the use of a defined command, suppose the defined command of Fig. 13 is stored in a file named OPTMIZ DEFCOM. Then typing "OPTMIZ 1." at the output level will initiate execution of the defined command with the variable B assigned the value 1.

4.5 Suggestions to the User of the Defined Command Feature

Since the output of analysis results is suppressed for all but the last analysis during the execution of a defined command, it is difficult to determine whether or not the command is executing properly. Therefore it is suggested that the command description contain several PRINT statements to print out values as the execution of the command is proceeding, especially during the execution of a loop. This will enable such things as infinite loops to be easily detected when the command is executed. Once the command has been debugged, the PRINT statements can be removed.

It is also suggested that an upper limit be placed on the number of times the command causes re-analysis of the circuit, to avoid an unnecessary use of computer time should the optimization procedure fail to converge.

If at any time it is suspected that a defined command has entered an infinite loop or otherwise gone wrong, the interrupt feature (Section 5.1.3) should be used to halt the execution of the command. The TIMER command (Section 5.1.2) may then be used.

It is worth stating again that only those users who have had experience both with programming and the use of CIRCAL-2 should attempt to make use of the defined command feature.

5. UTILITY AND DIAGNOSTIC FEATURES

5.1 Utility Commands

There are three utility commands which can be given at either the input level or the output level without a resulting change in level. They are described below in Sections 5.1.1, 5.1.2, and 5.1.3.

5.1.1 The MANUAL Command

CIRCAL-2 has a "built-in" user's manual, parts of which may be printed on-line by using the MANUAL command. The format of the MANUAL command is

MANUAL arg

where "arg" is either the word COM, in which case a list of the standard CIRCAL-2 commands is given, or the name of any standard CIRCAL-2 command, in which case a brief description of that command is given.

5.1.2 The TIMER Command

By giving the command TIMER (no arguments) the user can find out the execution time in milliseconds that has elapsed since use of CIRCAL-2 was initiated. Therefore, to find out the amount of time used in the execution of a command, TIMER must be given both before and after execution of the command, the difference in the two values being the desired time.

5.1.3 The QUIT Command and the Interrupt Feature

By giving the command QUIT (no arguments) the user can leave CIRCAL-2 and return to command level in CTSS.

The user may interrupt the execution of any of the CIRCAL-2 commands, the analysis of a circuit, or the output of analysis results by pushing the interrupt button on the console once. Upon receiving the resulting interrupt signal, the system stops whatever it is doing and, except when the interrupt is given during the execution of an edit subcommand, returns to command status at either the input level or the output level depending on where the interrupt

occurred. If the interrupt is given during the execution of an edit subcommand, the program returns to the edit level and the user may then continue to edit his file.

This interrupt feature will be found useful when the program is typing information that the user is no longer interested in (e.g., the plot of a variable that has reached a constant value long before the end of the plot), or when the execution of a command or the analysis of a circuit seems to be taking an unusually long time. There are instances, however, when use of the interrupt should be avoided because its use might jeopardize the file system. Such instances are listed in Section 5.3.2.

5.2 The CIRCAL-2 Diagnostics

5.2.1 The Handling of User Errors in CIRCAL-2

Extensive error checking is performed by CIRCAL-2 on all user statements. This is done to detect errors before they become serious and the source of the error becomes difficult to find. It is especially important to have extensive error checking in an on-line system such as CIRCAL-2 where the user learns by using the system.

Two types of user errors can occur in CIRCAL-2: fatal errors and non-fatal errors. If a fatal error occurs, the system can no longer continue whatever it was doing when the error was detected. Therefore, an error message is printed on the console and the system returns to command status at either the input level or the output level depending on where the error occurred. The next step to be taken by the user is to correct the error and repeat the command which caused the error (i.e., the last command given before the error occurred).

If a non-fatal error occurs, the system is still able to continue whatever it was doing when the error occurred, generally by ignoring altogether the condition which caused the error. To let the user know that an error has occurred, however, an error measage is printed before the system continues on. Most likely the user will want to correct the situation which caused the non-fatal error, but need not in all cases.

If at any time the user types an illegal character, the error message "ERROR IN COLUMN xx OF LINE..." will be printed,

followed by the line in which the error occurred and the character causing the error. The system then returns to command status where the entire line must be retyped with the error corrected.

In addition to error messages, the system prints out informational messages when certain conditions occur. For example, when modifying a file the message "END OF FILE REACHED" will be printed when the user tries to delete more lines than there are in the file. These conditions are not errors and in general do not require any corrections to be made on the part of the user.

5.2.2 Explanation of the User Errors

All error messages are printed on-line when an error occurs and begin with the letter "E" followed by a number. Error messages are distinguished in this manner from informational messages.

Table 7 gives explanations of the user errors by error number. The explanations include information as to whether the error is fatal or not and what steps should be taken to correct the error.

Table 7

Explanations Of The User Errors

E1 RECORD QUOTA OVERFLOW - fatal

There is no more space on disk to allow new files to be created or existing files to be modified. Either delete or change to temporary mode, one or more files now in permanent mode. Do not delete or change the mode of the files CIRCAL F.D., CFILE 000000, CFILE 000001, OUTPUT ARRAY, and PLTPRT ARRAY.

E2 NOT A VALID COMMAND - non-fatal

The letter or word typed is not a legal edit subcommand. See Table 3 for a list of the allowed edit subcommands.

E3 CORRECT FORMAT IS 'L STRING' - non-fatal

The locate subcommand was given improperly. See Table 3.

E4 THE ARGUMENT OF THIS COMMAND MUST BE AN INTEGER - non-fatal

The next, print, and delete subcommands take an integer argument. See Table 3.

E5 CORRECT FORMAT IS 'C *STRING1*STRING2*' - non-fatal

The change subcommand was given improperly. See Table 3.

E6 STRING NOT IN FILE - non-fatal

The string given as the argument to the locate subcommand was not found anywhere in the file.

E7 STRING1 NOT IN CURRENT LINE - non-fatal

The first string given in the argument to the change subcommand was not found in the current line.

E8 ERROR IN OPEN - fatal

E9 ERROR IN WRWAIT - fatal

E10 ERROR IN RDWAIT - fatal

E11 ERROR IN BUFFER - fatal

These four errors are in the routines which read from and write into the files on disk. If any of these errors occur, notify the persons listed in Section 1.3.

E12 FILE NOT IN DIRECTORY - fatal

The first and second names given as the name of a file were not found in the file directory (i.e., the file does not exist). Use LISTF to check.

E13 NOT A COMMAND - fatal

The word or letter typed is not a valid CIRCAL-2 command. See Section 7.

E14 COMMAND TYPED IS AN OUTPUT COMMAND - fatal

The word or letter typed is a command that can only be used at the output level. See Fig. 2.

E15 INCORRECT ELEMENT IN NETWORK - fatal

The network being analyzed has an element which is not allowed by the analysis routine being used (see Section 3.6). Use the DESIGN command to remove the element.

E16 CALL TO NEST SEARCH WITHOUT PROCEDURE - fatal

This error occurs in the analysis routine and is not a user error. If it occurs, notify the persons listed in Section 1.3.

E17 INCORRECT COMPUTING DIMENSION - fatal

The value given for the independent variable of analysis, IV, is incorrect (see Table 4). Use the REPEAT command to re-specify the value.

E18 ELEMENT SPECIFIED BY A FUNCTION - fatal

The network being analyzed contains an element described by a function. This is not allowed by the analysis routine being used (see Section 3.6). Use the DESIGN command to remove the element.

E19 ELEMENT CONNECTED BETWEEN IDENTICAL NODES IS IGNORED - non-fatal

An element in the network being analyzed is connected to the same node at both terminals. Analysis proceeds as if the element were not in the network. Use the DESIGN command to fix the element.

E20 INCORRECT NO. OF ARGS IN NEST INSTANCE - fatal

A nest instance in the network being analyzed has either too few or too many arguments. The number of arguments given for the nest instance must be equal to the number specified in the nest description. Use PRINTF command to check.

E21 CIRCUIT NOT IN FILE DIRECTORY - fatal

The circuit file whose first name was given in the OUTPUT or XOUTPT command was not in the file directory (i.e., the file does not exist). Use LISTF to check.

E22 ANALYSIS ROUTINE NOT FOUND - fatal

The analysis routine name specified in the OUTPUT or XOUTPT command or in updating the mode file is incorrect (see Section 3.6). Retype the OUTPUT or XOUTPT command or use the REPEAT command with the correct analysis routine name.

E23 NEST NOT IN FILE DIRECTORY - fatal

The network being analyzed contains a nest instance whose description was not found in the file directory (i.e., does not exist). Use the DESIGN command to either remove the nest instance or input its description.

E24 FUNCTION OR FUNCTIONAL NOT IN FILE DIRECTORY - fatal

The network being analyzed contains an element described by a function or functional whose description was not found in the file directory (i.e., does not exist). Use the DESIGN command to either remove the element or input the function or functional description.

E25 PARAMETER NOT FOUND - fatal

The mode file parameter given was specified incorrectly (see Table 4). Use the REPEAT command to specify the parameter correctly.

E26 NETWORK PARAMETER NOT PROPERLY SPECIFIED - fatal

The value for a network parameter has been incorrectly specified (see Table 4). Use the REPEAT command to specify the network parameter correctly.

E27 ANALYSIS ROUTINE NOT SPECIFIED IN OUTPUT COMMAND - fatal

The analysis routine name given in updating the mode file is not one of the names given in the OUTPUT or XOUTPT command. Use the REPEAT command with another analysis routine name or give the OUTPUT or XOUTPT command again.

E28 OUTPUT NOT PROPERLY SPECIFIED - fatal

The variables to be plotted or printed have been specified incorrectly (see Table 4 and Section 3.7.2). Use the REPEAT command to re-specify the output.

E29 ANALYSIS AND PLOT PARAMETERS INCOMPATIBLE - fatal

The range of the independent variable for the output extends beyond the range of the independent variable for the analysis. Use the REPEAT command to re-specify AS, AE, PS, or PE.

E30 PI, PS, PE INCOMPATIBLE - fatal

PI, PS, and PE must satisfy the equation $(PE-PS)/PI \geq 0$; i.e., the increment PI must take the output from PS to PE. Use the REPEAT command to re-specify PI, PS, or PE.

E31 AI, AS, AE INCOMPATIBLE - fatal

AI, AS, and AE must satisfy the equation $(AE-AS)/AI \geq 0$; i.e., the increment AI must take analysis from AS to AE. Use the REPEAT command to re-specify AI, AS, or AE.

E32 PI and AI INCOMPATIBLE - fatal

PI and AI must satisfy the equation $|PI| \geq |AI| > 0$. Use the REPEAT command to re-specify AI or PI.

E33 AI NOT PROPERLY SPECIFIED - fatal

E34 AS NOT PROPERLY SPECIFIED - fatal

E35 AE NOT PROPERLY SPECIFIED - fatal

E36 IV NOT PROPERLY SPECIFIED - fatal

E37 VARIABLES TO SAVE NOT PROPERLY SPECIFIED - fatal

E38 PI NOT PROPERLY SPECIFIED - fatal

E39 PS NOT PROPERLY SPECIFIED - fatal

E40 PE NOT PROPERLY SPECIFIED - fatal

These eight errors indicate that a mode file parameter has been incorrectly specified (see Table 4). Use the REPEAT command to re-specify the parameter.

E41 MORE THAN 50 ITEMS, ALL ABOVE 50 IGNORED - non-fatal

The buffer in which the items typed by the user are stored only has room for 50 items. If more than 50 items are typed, items 51 and above are ignored. The next error to occur will indicate the last item processed. Repeat the command with the ignored items as arguments.

E42 ERROR IN OUTPUT ARRAY - fatal

Generally this error means there is a record quota overflow (see E1). If it does not, notify the persons listed in Section 1.3.

E43 NODE SPECIFIED IN 'SAVE' COMMAND NOT IN NETWORK - fatal

A node specified in the name of a variable to be saved for output does not exist in the network being analyzed. Use the REPEAT command to re-specify the save variables.

E44 ONLY REAL AND IMAG PART OF VOLTAGE CALCULATED BY ANALYSIS - fatal

A variable to be saved for output is other than the real or imaginary part of a voltage, the only things calculated by the analysis being used (see Section 3.6). Use the REPEAT command to re-specify the save variables.

E45 ONLY A SINGLE NODE SHOULD BE SPECIFIED IN 'SAVE' COMMAND - fatal

The names of the variables to be saved for output should include only a single node for the analysis being used (see Section 3.6). Use the REPEAT command to re-specify the save variables.

E46 VARIABLE FOR OUTPUT NOT SAVED - fatal

A variable to be plotted or printed was not specified as a variable to be saved for output (see Sections 3.3.3 and 3.7.1). Use the REPEAT command to re-specify the save variables.

E47 NEST PARAMETER NOT FOUND IN 'ARGS' LIST - fatal

A variable specified for the value of an element in a nest was not found among the variables specified in the ARGS statement (see Section 2.2.2). Use the DESIGN command to change the value of the element.

E48 ONLY 5 VARIABLES MAY BE PRINTED AT A TIME - fatal

A maximum of five variables may be printed at a time. Use the REPEAT command to re-specify the output.

E49 ONLY 1 VARIABLE MAY BE PLOTTED AT A TIME - fatal

Only one variable may be plotted at a time. Use the REPEAT command to re-specify the output.

E50 ELEMENT HAS IMPROPER NUMBER OF NODES - fatal

An element in the network being analyzed has too few or too many terminals (see Table 2). Use PRINTF to check and DESIGN to change the element.

E51 INSTANCE NUMBER MISSING FROM ELEMENT - fatal

All element names must include an instance number appended to the generic element name (see Table 2). Use PRINTF to check and DESIGN to change the element.

E52 TOO MANY ARGUMENTS -fatal

Too many arguments were given for the command (see Section 7). Give the command again with the proper number of arguments.

E53 TOO FEW ARGUMENTS - fatal

Not enough arguments were given for the command (see Section 7). Give the command again with the proper number of arguments.

E54 CKT OR NEST NOT FOUND - fatal

The circuit or nest specified in the ALTER command was not found (see Section 3.4.2). Give the ALTER command again with the correct circuit or nest name.

E55 FUNCTION OR FUNCTIONAL FILES CANNOT BE 'ALTER'ED - fatal

The ALTER command cannot be used to change function or functional files. Use the DESIGN command.

E56 OP INCORRECT OR NO CKT OR NEST SPECIFIED - fatal

Either an ALTER op was specified incorrectly or no circuit or nest in which the changes were to be made was specified (see Section 3.4.2). Give the ALTER command again correctly.

E57 OP INCORRECT OR MISSING - fatal

Either an ALTER op has been specified incorrectly or not specified at all (see Section 3.4.2). Give the ALTER command again with the correct op.

E58 ELEMENT NOT FOUND - fatal

An element specified in the ALTER command is not in the circuit or nest specified (see Section 3.4.2). Use PRINTF to check and give the ALTER command again.

E59 TOO MANY OPS, USE ALTER TWICE WITH FEWER OPS - fatal

The size of the buffer and the manner in which the ALTER command is processed limits the number of ops that can be specified in a single use of the ALTER command. Use the ALTER command twice, each with fewer ops.

E60 CKT OR NEST HAS TWO ELEMENTS WITH SAME NAME - fatal

All elements in a circuit or nest must have a unique name (see Section 2.2.1). Use PRINTF to check and DESIGN to change the element name.

E61 XNODES STATEMENT NOT ALLOWED IN CKT FILE - fatal

Circuits cannot have external nodes (see Section 2.2.1). Use DESIGN to remove the XNODES statement.

E62 NEST FILE WITH NO XNODES STATEMENT - fatal

All nest files must have an XNODES statement (see Section 2.2.2). Use DESIGN to add the XNODES statement.

E63 ALL NODES IN XNODES STATEMENT MUST BE DIFFERENT - fatal

No two nodes specified in the XNODES statement of a nest can be the same. Use DESIGN to change the XNODES statement.

E64 NEST MUST HAVE AT LEAST TWO EXTERNAL NODES - fatal

At least two nodes must be specified in the XNODES statement of a nest. Use DESIGN to change the XNODES statement.

E65 XNODES STATEMENT INCORRECT - fatal

The XNODES statement of a nest was specified incorrectly (see Section 2.2.2). Use PRINTF to check and DESIGN to change the XNODES statement.

E66 ARGS STATEMENT INCORRECT - fatal

The ARGS statement of either a circuit or nest was specified incorrectly (see Section 2.2.2). Use PRINTF to check and DESIGN to change the ARGS statement.

E67 ALL ARGUMENTS IN ARGS STATEMENT MUST BE DIFFERENT - fatal

Two or more argument names listed in the ARGS statement of a circuit or nest are the same. Use PRINTF to check and DESIGN to change the ARGS statement.

E68 ELEMENT SPECIFIED INCORRECTLY - fatal

An element in the circuit being analyzed is specified incorrectly (see Table 2).

E69 NUMERICAL UNDERFLOW - VARIABLE SET TO ZERO - non-fatal

Some number was too small and was set to zero.

E70 NUMERICAL OVERFLOW - fatal

Some number was too large for the computer to handle. Check element values and values given for the mode file parameters to determine source of trouble.

E71 NUMERICAL ERROR - NOT UNDERFLOW OR OVERFLOW - fatal

Some numerical error (neither an underflow nor an overflow) occurred. If this error occurs, notify the persons listed in Section 1.3.

E72 RESISTOR OR INDUCTOR HAS ZERO VALUE - fatal

The analysis routine being used does not allow resistors and inductors to have zero value. Use DESIGN command (or REPEAT command if network parameter was given zero value) to give element non-zero value.

E73 NO ANALYSIS DONE - AI LESS THAN ZERO - fatal

The analysis routine being used does not allow the analysis increment to be negative. Use the REPEAT command to respecify AI.

E74 ERROR IN PLTPRT ARRAY - fatal

When this error occurs it generally means there is a record quota overflow (see E1). If not, notify the persons listed in Section 1.3.

E75 SAVE VARIABLES MUST BE SPECIFIED BEFORE OUTPUT VARIABLES - fatal

The variables to be saved for output must be specified first since the variables to be plotted or printed are checked against the save variables. Use the REPEAT command to re-specify the parameters in the correct order.

E76 ARGUMENT NOT RECOGNIZED - fatal

An argument of the command given is not one of the allowed items (see Section 7). Retype the command with correct arguments.

E77 DEFCOM ARGUMENT INCORRECT - fatal

Whenever a defined command with arguments is used, values for the arguments must be specified as fixed decimal numbers with decimal point (see Section 4.4). Give the command again with correct argument values.

E78 ILLEGAL ANALYSIS POINT IN STATEMENT - fatal

The analysis point given in an IF THEN statement or in an assignment statement of a defined command is outside the range of analysis, or not a fixed decimal number or a DEFCOM argument (see Sections 4.2.2 and 4.2.5). Use DESIGN to change the statement.

E79 ILLEGAL BOOLEAN EXPRESSION IN IF STATEMENT - fatal

The boolean expression in an IF THEN statement in a defined command description is incorrect (see Section 4.2.2). Use DESIGN to change the IF statement.

E80 ILLEGAL IF STATEMENT - fatal

An IF THEN statement in the defined command is incorrect (see Section 4.2.2). Use DESIGN to correct the statement.

E81 ILLEGAL GOTO STATEMENT - fatal

A GOTO statement in the defined command is incorrect (see Section 4.2.3). Use DESIGN to correct the statement.

E82 LABEL NOT FOUND - fatal

A label specified as the argument of a GOTO statement was not found in the defined command description. Use DESIGN to assign the label to a statement or change the argument of the GOTO statement.

E83 ILLEGAL PRINT STATEMENT - fatal

A PRINT statement in the defined command description is incorrect (see Section 4.2.4). Use DESIGN to correct the statement.

E84 NO END STATEMENT OR MISSING ')' - fatal

All statements in a defined command description must end with a right parenthesis and the last statement to be executed must be the END statement (see Section 4.3). Use DESIGN to correct the error.

E85 ILLEGAL ASSIGNMENT STATEMENT - fatal

Only assignment statements of a specific format are allowed in a defined command (see Section 4.2.5). Use DESIGN to correct the statement.

E86 DIVISION BY ZERO IN A FUNCTION - fatal

Division by zero was attempted in one of the standard CIRCAL-2 functions. This generally means the value of the function went to infinity at some point.

E87 ARGUMENT OF A FUNCTION LESS THAN ZERO - fatal

Negative arguments are not allowed for the standard functions LOG and SQRT. In addition, the argument of the LOG function cannot be zero. (See Table 6).

5.3 Possible Difficulties and How to Avoid Them

5.3.1 System Errors

It is conceivable that errors could occur in CIRCAL-2 which are not caused by the user, but rather by some error in the system which has not been detected as yet. These errors may result in unexplainable behavior or may cause termination of the program and the return of the user to CTSS command status.

There are three errors that cause termination of the program: protection mode violation, program stop, and illegal use of the free storage routines (FREE, FREZ, FREC, and FRET). If any of these errors occur, the persons listed in Section 1.3 should be notified.

Another condition which will cause termination of the program, but which is not an error, is the unavailability of free storage. This generally means that there is not enough room in core to analyze the circuit. Should this condition occur, reduce the size of the network and try analyzing it again.

5.3.2 Other Difficulties

If the system does not recognize what appears to be a legal command or specification of a mode file parameter, the item should be retyped since it is possible that an error has occurred in transmission.

If the error occurs again, the manual should be consulted to make sure the format of the item is correct. If the format is correct, the situation should be explained to one of the persons listed in Section 1.3.

It has been found that use of the interrupt feature will sometimes cause the system to print out a line of extraneous characters before printing succeeding information. This line should be ignored.

There are several phases of system operation where it is unadvisable to use the interrupt feature. One of these phases is during the execution of the ERASE command. Should the user mistakenly give an interrupt at this point, the LISTF and PRINTF commands should be used to check the file system. Another unadvisable place to use the interrupt is during the execution of the file subcommand of the DESIGN command. Here again, the LISTF and PRINTF commands should be used to check the file system if an interrupt is given. The interrupt feature also should not be used during execution of the ALTER command. While using the interrupt here will have no effect on the file system, it will leave the user unaware of what changes have been made in the data structure. If an interrupt is given at this point, it is advisable to start over with the OUTPUT or XOUTPT command.

Another possible source of difficulty is in the naming of files. All user-created files must have different first names regardless of the second name of the file.

5.3.3 Reserved Words and Letters

Several words and letters have special meaning in CIRCAL-2 and the user is cautioned against using them except when that meaning is intended. Although in general no harm will be done if these words and letters are used with other than their intended meaning, it is suggested that such use be avoided to prevent possible trouble and confusion. One way of avoiding the unintentional use of reserved words and letters is to begin all variable names, nest names, etc., with the same letter (such as x, for example) and make them two characters long or longer.

A complete list of the reserved words and letters, with the meaning of each, is given in Table 8. Where more than one meaning is given for a word or letter, the correct meaning is determined by

context. All reserved words and letters used in this manual are given in upper-case letters to distinguish them from other words and letters, but should be given in lower-case letters when using CIRCAL-2.

Table 8

Reserved Letters (a) and Reserved Words (b)
Used in CIRCAL-2

Reserved Letter	Meaning
A	Abbreviation for ALTER command; ALTER argument
C	Generic element name; ALTER argument; DESIGN subcommand
D	Abbreviation for DESIGN command; ALTER argument; DESIGN subcommand
E	Abbreviation for ERASE command; generic element name; used in E format number specification
F	Abbreviation for FILE command; frequency; DESIGN subcommand
I	Current
J	Generic element name
L	Abbreviation for LISTF command; generic element name; DESIGN subcommand
M	Abbreviation for MANUAL command
N	DESIGN subcommand
O	Abbreviation for OUTPUT command
P	Abbreviation for PRINTF command; DESIGN subcommand
Q	Abbreviation for QUIT command; charge
R	Abbreviation for REPEAT command; ALTER argument; generic element name
S	Generic element name
T	Abbreviation for TIMER command; time; DESIGN subcommand
V	Voltage
X	Abbreviation for XOUTPT command; generic element name

(a)

Reserved Word	Meaning
AE	Analysis end point
AI	Analysis increment
ALTER	Command
ARGS	Used to specify arguments
AS	Analysis start point
ATAN	Arctangent function
CKT	Second name of a circuit file
COM	MANUAL argument
COS	Cosine function
COT	Cotangent function
DEFCOM	Second name of a defined-command file
DESIGN	Command
END	Used in defined-command description
EQL	Boolean operator
ERASE	Command
EXP	Exponential function
FILE	Command; DESIGN subcommand
FNCTN	Second name of a function file
FNCTNL	Second name of a functional file
FREQ	Analysis routine name
GEQ	Boolean operator
GOTO	Used in defined-command description
GRT	Boolean operator
IF	Used in defined-command description
IV	Independent variable of analysis
LAM	Flux
LEQ	Boolean operator
LES	Boolean operator
LISTF	Command
LOG	Natural log function
LOG10	Log_{10} function
MAG	Magnitude function
MANUAL	Command
NC	Generic element name

Reserved Word	Meaning
NE	Generic element name
NEQ	Boolean operator
NEST	Second name of a nest file
NJ	Generic element name
NL	Generic element name
NR	Generic element name
OUTPUT	Command
PE	Output end point
PHASE	Phase function
PI	Output increment
PLOT	Output type
PRINT	Output type; used in defined-command description
PRINTF	Command
PRNTDS	Analysis routine name
PS	Output start point
QUIT	Command
REPEAT	Command
SAVE	Used to specify variables to be saved for output
SIN	Sine function
SQRT	Square root function
TAN	Tangent function
THEN	Used in defined-command description
TIMER	Command
VI	Imaginary part of voltage
VR	Real part of voltage
XNODES	Used to specify external nodes
XOUTPT	Command

(b)

6. EXAMPLE OF A SESSION WITH CIRCAL-2

An example of a session with CIRCAL-2 is given on the following pages. The user's input to the system is in lower-case letters while the system response is in upper case. Comments have been added along the right margin to aid in understanding the example.

```
r circ12
W 1513.1
EXECUTION.

COM- manual com                                  Use the MANUAL command to
                                                 obtain a list of the commands
THE COMMANDS ARE ...                             available in this version. The
                                                 letter in parentheses indicates
DESIGN (D)                                       the command may be abbreviated
ERASE (E)                                        by the first letter only.
LISTF (L)
OUTPUT (O)
XOUTPT (X)
QUIT (Q)
MANUAL (M)
TIMER (T)
PRINTF (P)
REPEAT (R)

THE FOLLOWING COMMANDS ARE NOT OPERATIONAL IN THIS VERSION OF CIRCAL-2 ...

ALTER (A)
FILE (F)

COM- m listf                                     Use the MANUAL command to
                                                 obtain a description of the
THE FORMAT OF THE LISTF COMMAND IS               LISTF command.
      (A) LISTF
      (B) LISTF NAME1 NAME2
      (C) LISTF   *   NAME2
      (D) LISTF NAME1   *
WHERE (A) GIVES A LISTING OF ALL FILES, (B) LISTS ONLY THE FILE
NAME1 NAME2 IF IT EXISTS, (C) LISTS ALL FILES WITH SECOND NAME NAME2,
AND (D) LISTS ALL FILES WITH FIRST NAME NAME1.

COM- listf                                       List all files.

NAME1  NAME2  MODE LENGTH SUPFIL
IEEE   CKT       0     90      1
ALPHA  NEST      0     78      1
FILTER CKT       0     25      1
DEMO   CKT       0     14      1
LARGE  CKT       0     12      1
COLPIT CKT       0     14      1
TEST   DEFCOM    0     33      1
T      NEST      0     12      1
SIMPLE CKT       0      6      1
NOTHER CKT       6      6      1

COM- printf simple ckt                           Print the contents of the
                                                 file SIMPLE CKT.
1 0 J1 1.
1 0 R1 K
1 0 C1 1.
```

```
COM- erase simple ckt
```

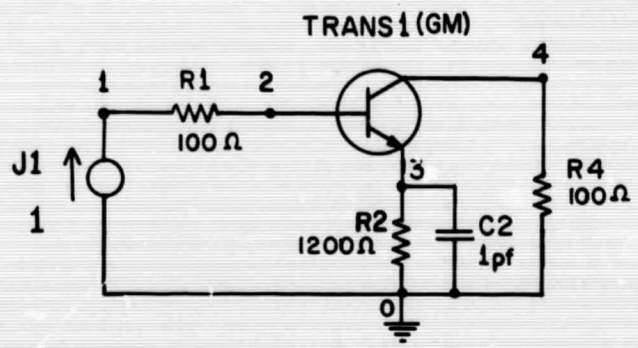
Delete the file SIMPLE CKT.

```
COM- design amp ckt
INPUT
0 1 j1 1.
1 2 r1 100.
3 2 4 trans1(gm)
3 0 r2 1200.
3 0 c2 1.e-12
4 0 r4 100.

EDIT
1 gm
3 2 4 TRANS1(GM)

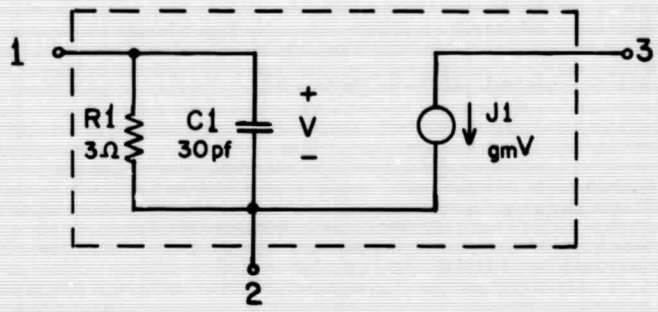
c /3 2/2 3/
2 3 4 TRANS1(GM)

f
*
```




```
COM- d trans nest
INPUT
xnodes 3 1 2 3
args 1 gm
1 2 r1 3.
1 2 c1 3.e-11
3 2 j1 v(c1) gm

EDIT
f
*
```




```
COM- output amp
TYPE ANALYSIS ROUTINE
freq
TYPE ANALYSIS INCREMENT, AI
ai=5.e8
TYPE ANALYSIS START POINT, AS
as=1.e9
TYPE ANALYSIS END POINT, AE
ae=1.e10
TYPE INDEPENDENT VARIABLE, IV
iv=f
TYPE VARIABLES TO BE SAVED FOR OUTPUT
save 2 vr 1 vi 1
TYPE OUTPUT INCREMENT, PI

TYPE OUTPUT START POINT, PS

TYPE OUTPUT END POINT, PE

TYPE OUTPUT
plot 1 mag(vr 1,vi 1)
TYPE VALUES FOR THE NETWORK PARAMETERS, IF ANY
gm=.3
```

Transfer to output level with the slow mode of mode file update. Values for PI, PS, and PE could have been given but were not since they were to be the same as AI, AS, and AE. Note the use of the MAG function in specifying the output.

```
GRAPH OF
  1 MAG(VR 1 VI  1)
VS F
ORDINATE TIMES 10 TO THE 1

          8         13        18        23        28        33
          |----+----|----+----|----+----|----+----|----+----|
 1.000E+09                                          *
                              *
                         *
                    *
                  *
                 *
 3.500E+09       *
                *
                *
                *
                *
 6.000E+09      *
                *
                8#INT. 1
```

Interrupt--not interested in remainder of plot.

```
COM- timer
TOTAL RUNNING TIME=        11833 MS
```

Have used 11.833 sec. in execution so far.

```
COM- repeat gm=.5 pe=6.e9
```

Re-analyze with new value for GM. Note, only the new value for GM need be specified, all else remains the same.

```
GRAPH OF
  1 MAG(VR 1 VI 1)
VS F
ORDINATE TIMES 10 TO THE 1

          8         13        18        23        28        33        38
          |----+----|----+----|----+----|----+----|----+----|----+----|
 1.000E+09                                                   *
                              *
                         *
                    *
                  *
                 *
 3.500E+09       *
                *
                *
                *
                *
 6.000E+09      *


COM- t
TOTAL RUNNING TIME=        14450 MS
```

Re-analysis and plotting took 2.617 sec.

```
COM- d varygm defcom
INPUT
args 4 apt gms gmi gme )
repeat gm = gms )
loop ' if vr 1 (apt) les 0.0 then goto done )
if gm geq gme then goto exit )
repeat gm = gm + gmi )
goto loop )
done ' print 1 gm )
exit ' end )

EDIT
f
*
```

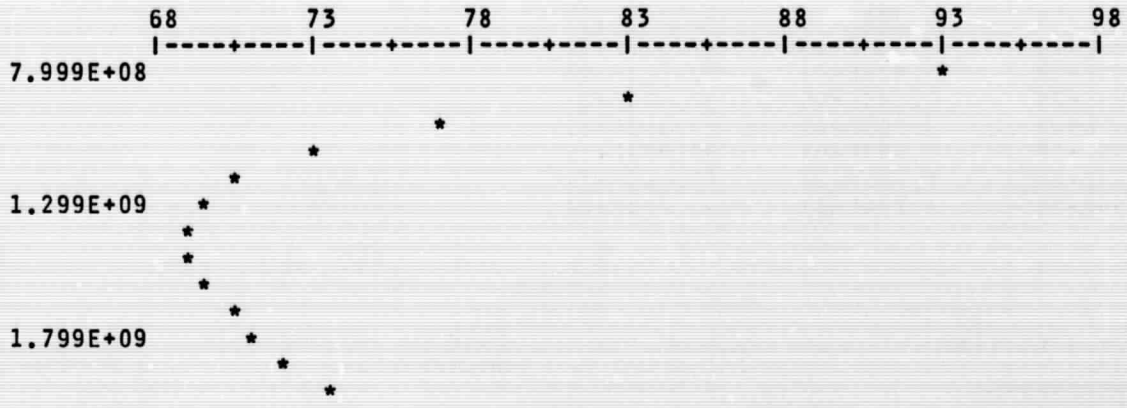
Define a command to vary the value of GM from GMS in increments of GMI until the real part of the voltage at node 1 with respect to ground goes negative at analysis point APT, or until GM $\geq$ GME.

```
COM- x amp

TYPE PARAMETERS
as=8.e8 ai=1.e8 ae=2.e9 plot 1 vr 1
SPECIFY VALUE(S) FOR GM
gm=.3
```

Transfer to output level with fast mode of mode file update. Note that the mode file previously created for this circuit still exists, and only the parameters to be changed from the previous values are specified. Network parameters must be specified in any case.

```
GRAPH OF
  1 VR 1
VS F
ORDINATE TIMES 10 TO THE 0
```

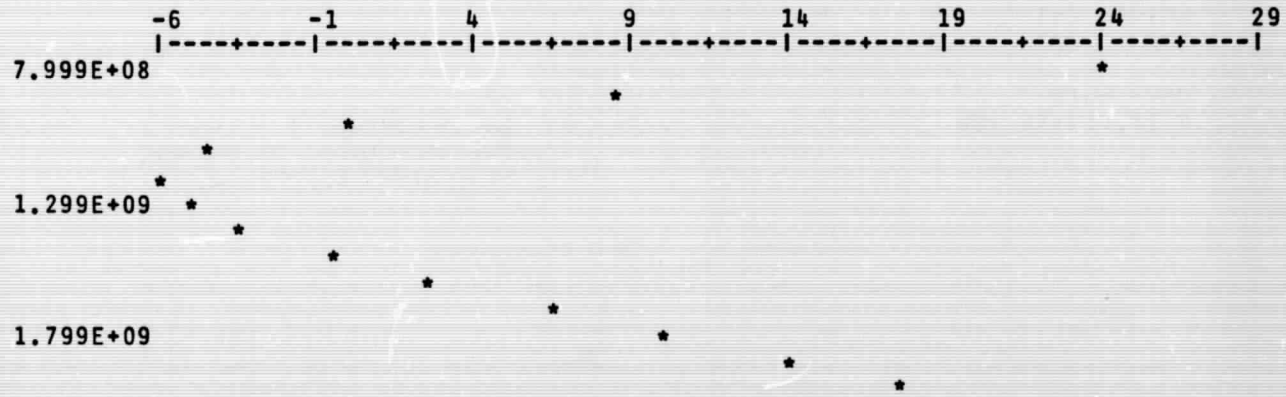



```
COM- varygm 1.4e9 0.5 0.05 1.0
GM= 7.999E-01

GRAPH OF
  1 VR 1
VS F
ORDINATE TIMES 10 TO THE 0
```

Use defined command VARYGM with APT=1.4e9, GMS=0.5, GMI=0.05, and GME=1.0.

Network was analyzed seven times before conditions were met. Results of the last analysis are plotted.



```
COM- r print 2 vr 1 vi 1
```

Re-analysis is not required with this use of the REPEAT command since the information requested still exists from the last analysis.

```
VALUES FOR
  1 VR 1
  2 VI 1
VS F

    IV              1              2
 7.999E+08      2.403E+01     -6.094E+02
 8.999E+08      8.593E+00     -5.289E+02
 1.000E+09     -9.608E-02     -4.634E+02
 1.099E+09     -4.394E+00     -4.092E+02
 1.199E+09     -5.778E+00     -3.638E+02
 1.299E+09     -5.208E+00     -3.255E+02
 1.399E+09     -3.332E+00     -2.928E+02
 1.500E+09     -5.974E-01     -2.647E+02
 1.599E+09      2.685E+00     -2.404E+02
 1.699E+09      6.295E+00     -2.193E+02
 1.799E+09      1.007E+01     -2.008E+02
 1.899E+09      1.391E+01     -1.846E+02
 2.000E+09      1.774E+01     -1.704E+02
```

```
COM- p mode file
```

Print the contents of the mode file.

```
ANALYSIS=FREQ
AI= 1.000E+08  AS= 7.999E+08  AE= 2.000E+09
IV=F
PI= 1.000E+08  PS= 7.999E+08  PE= 2.000E+09
SAVE 2 VR 1 VI 1
PRINT 2 VR 1 VI 1 VS F
```

```
COM- l * ckt
```

List all files with second name CKT.

```
NAME1  NAME2  MODE  LENGTH  SUPFIL
IEEE   CKT       0      90       1
FILTER CKT       0      25       1
DEMO   CKT       0      14       1
LARGE  CKT       0      12       1
COLPIT CKT       0      14       1
NOTHER CKT       6       6       1
AMP    CKT       0      14       1
```

```
COM- quit
R 28.683+35.700
```

Leave CIRCAL-2
Execution time = 28.683 sec.
Swap time = 35.700 sec.

MISCELLANEOUS NOTES

1. Tabs must be set one every ten spaces for proper plotting of graphs.

2. There must be at least one record available on disk before using CIRCAL-2.

3. Commands may be abbreviated by the first letter only.

4. The first names of all user-created files must be different.

5. The names of all elements must be six characters or less, including the instance number.

6. Node numbers need not be consecutive, but must be non-negative integers.

7. The node with the smallest number is taken to be the ground node.

COMMANDS

Command Format	Operation Performed	Level Change?	Reference (Section)
ALTER $filename_1$ $argset_1$ ... $filename_n$ $argset_n$ (where $argset_i = op_{il}$ $opargs_{il}$ ... op_{ik} $opargs_{ik}$)	Makes changes in circuit or nest data structure	No	3.4.2
DESIGN name1 name2	Allows creation and modification of files	Only when used at the output level	2.3.1
ERASE name1 name2	Deletes an entire file	Only when used at the output level	2.3.4
FILE name1 name2	Saves data structures and modified CKT and NEST files	No	3.5.1
LISTF LISTF name1 name2 LISTF * name2 LISTF name1 *	Lists names of existing files	Only when used at the output level	2.3.2
MANUAL arg	Prints sections of the User's Manual	No	5.1.1
OUTPUT name1 $analysis_1$ ... $analysis_n$	Transfers to output level for analysis of circuit name1 CKT	Only when used at the input level	3.2
PRINTF name1 name2	Prints the contents of a file	No	2.3.3 3.5.2
QUIT	Returns user to CTSS command status	---	5.1.3
REPEAT par_1 ... par_n	Re-analysis of the circuit	No	3.4.1 4.2.1
TIMER	Prints execution time since CIRCAL-2 was resumed	No	5.1.2
XOUTPT name1 $analysis_1$ ... $analysis_n$	Transfers to output level for analysis of circuit name1 CKT	Only when used at the input level	3.2

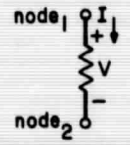
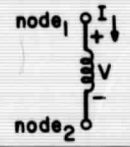
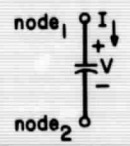
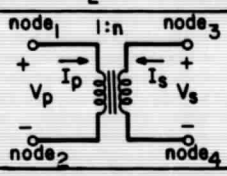
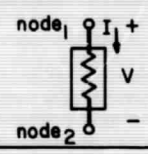
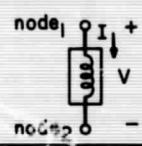
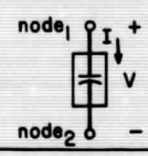
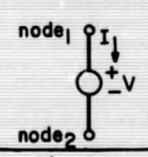
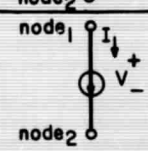
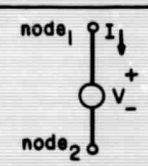
Arguments of the ALTER Command

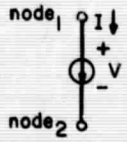
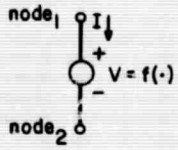
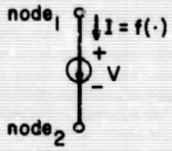
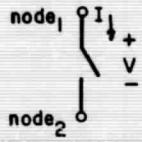
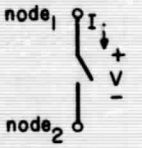
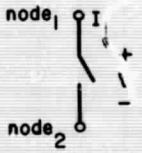
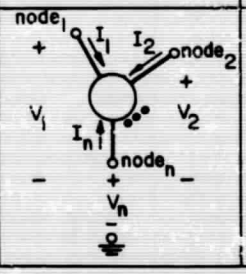
op_{ij}	$opargs_{ij}$	Meaning
C	elname newvalue	Change the value of element "elname" to "newvalue".
D	elname	Delete element "elname" from the network.
A	$node_1$ ... $node_k$ elname value	Add element "elname" between nodes "$node_1$ ... $node_k$" with value "value".
R	$elname_1$ $elname_2$ value	Replace element "$elname_1$" by element "$elname_2$" with value "value".

The Edit Subcommands

Subcommand	Format	Response	Meaning
top	T	None	Move pointer to top of file
next	N k	Line at which pointer stops or INPUT is printed	Advance pointer k lines from current line; if k is greater than the number of lines in the file after the current line, the pointer is advanced to the bottom of the file and the input mode is entered; k is assumed to be 1 if unspecified.
print	P k	k or fewer lines are printed	Print k lines of the file beginning with the current line; if k is greater than the number of lines in the file after the current line, only the remaining lines are printed; k is assumed to be 1 if unspecified; the pointer is not moved.
delete	D k	O.K. or END OF FILE REACHED	Delete k lines of the file beginning with the current line; if k is greater than the number of lines in the file after the current line, only the remaining lines are deleted; k is assumed to be 1 if unspecified; the pointer is moved to the line preceding the first line deleted.
change	C qstring1qstring2q	Line after changes have been made is printed	Change first occurrence (left to right) of string1 to string2 in current line; string1 is any character string in the current line, string2 is any character string, q is any character not in string1 or string2; current line is unchanged if string1 cannot be found in the current line; the pointer is not moved.
locate	L string	Line with first occurrence of string is printed	Beginning with the line following the current line, search the file for an occurrence of string; if the end of the file is reached before locating string, the search is resumed at the top of the file and continued to the current line; string is any string of characters in the file; the pointer is moved to the first line encountered containing string or is not moved if string is not found.
file	F name1 name2 or FILE name1 name2	*	Store the file being modified in the master file under the name name1 name2; if name1 and name2 are not specified, the file is stored under the name specified when the DESIGN command was given, deleting the old version of the file, if any; the system returns to command status.

The Standard CIRCAL-2 Elements

ELEMENT	SCHEMATIC	INPUT FORMAT	VALUE
Linear Resistor		node$_1$ node$_2$ Ri value	a. Decimal number b. Variable name c. fname (args)$^{\neq}$
Linear Inductor		node$_1$ node$_2$ Li value	a. Decimal number b. Variable name c. fname (args)$^{\neq}$
Linear Capacitor		node$_1$ node$_2$ Ci value	a. Decimal number b. Variable name c. fname (args)$^{\neq}$
Ideal Transformer		node$_1$ node$_2$ node$_3$ node$_4$ Xi value	a. Decimal number b. Variable name c. fname (args)*
Nonlinear Resistor		node$_1$ node$_2$ NRi value	a. fname (args)$^{\neq}$ b. fnlname (args)$^{\neq}$
Nonlinear Inductor		node$_1$ node$_2$ NLi value	a. fname (args)$^{\neq}$ b. fnlname (args)$^{\neq}$
Nonlinear Capacitor		node$_1$ node$_2$ NCi value	a. fname (args)$^{\neq}$ b. fnlname (args)$^{\neq}$
Independent Voltage Source		node$_1$ node$_2$ Ei value	a. Decimal number b. Variable name c. fname (args)$^{\neq}$
Independent Current Source		node$_1$ node$_2$ Ji value	a. Decimal number b. Variable name c. fname (args)$^{\neq}$
Linear Dependent Voltage Source		node$_1$ node$_2$ Ei V(elname) value node$_1$ node$_2$ Ei I(elname) value	a. Decimal number b. Variable name c. fname (args)$^{\neq}$

ELEMENT	SCHEMATIC	INPUT FORMAT	VALUE
Linear Dependent Current Source		node$_1$ node$_2$ Ji V(elname) value node$_1$ node$_2$ Ji I(elname) value	a. Decimal number b. Variable name c. fname (args)/
Nonlinear Dependent Voltage Source		node$_1$ node$_2$ NEi value	a. fname (args)/# b. fnlname (args)/#
Nonlinear Dependent Current Source		node$_1$ node$_2$ NJi value	a. fname (args)/# b. fnlname (args)/#
Independent Switch		node$_1$ node$_2$ Si value	a. Decimal number (0. or 1.) b. Variable name c. fname (args)*
Voltage-Dependent Switch		node$_1$ node$_2$ Si V(elname) value	a. Decimal number b. Variable name c. fname (args)/
Current-Dependent Switch		node$_1$ node$_2$ Si I(elname) value	a. Decimal number b. Variable name c. fname (args)/
Nest		node$_1$ node$_2$ ... node$_n$ nestnamei (args)	a. Decimal number b. Variable name

NOTES:

i is the instance number of the element.

* cannot be a function of any network variables, or any variable which changes during analysis.

/ cannot be a function of any network variable.

/ arguments must include one, but not both, of the element variables.

/# arguments cannot include any of the element variables.

The Mode File Parameters

Parameter	Preset Value	Format for Specifying a Change in Value	Comments
Analysis routine	None	analysisname	analysisname is the name of the available analysis routine (see Section 3.6)
Analysis increment	0.1	AI = newval	newval is a fixed decimal number (with decimal point); E format may be used (e.g., 1.0E-3 for 0.001)
Analysis start point	0.0	AS = newval	newval is the same as for AI
Analysis end point	10.0	AE = newval	newval is the same as for AI
Independent variable of analysis	T	IV = indvar	indvar is either the letter T (time) or the letter F (frequency)
Variables to be saved for output	None	SAVE n var_1 ... var_n	n is the number of variables to be saved for output; var_i is the name of a variable to be saved for output (see Section 3.6)
Output increment	0.1	PI = newval	newval is the same as for AI
Output start point	0.0	PS = newval	newval is the same as for AI
Output end point	10.0	PE = newval	newval is the same as for AI
Variables to be plotted or printed	None	outype n $f_1(a_{11},\ldots,a_{1m}),\ldots,f_n(a_{n1},\ldots,a_{np})$ or PRINT n $f_1(a_{11},\ldots,a_{1m}),\ldots,f_n(a_{n1}\ldots,a_{np})$VS $f_{n+1}(a_{n+1,1},\ldots,a_{n+1})$	outype is either the word PLOT or the word PRINT; n is the number of things to be plotted or printed and must be 1 for plotting, ≤ 5 for printing; $f_i(a_{i1},\ldots,a_{ij})$ is the specification for a function (see Section 3.7) which may be the identity function
Network Parameter	None	parname = newval	parname is the name of the network parameter; newval is the same as for AI

The Standard Functions of CIRCAL-2

Function Specification	Description	Restrictions
ATAN(x)	Arctangent of x	None
COS(x)	Cosine of x	None
COT(x)	Cotangent of x	$n\pi < x < (n+1)\pi$ $(n=0, \pm1, \pm2, \ldots)$
EXP(x)	e^x	None
LOG(x)	$\log_e x$	$x > 0$
LOG10(x)	$\log_{10} x$	$x > 0$
MAG(x_1, x_2)	$\sqrt{x_1^2 + x_2^2}$	None
PHASE(x_1, x_2)	$\tan^{-1} x_2/x_1$	None
SIN(x)	Sine of x	None
SQRT(x)	$\sqrt{x}$	$x \geq 0$
TAN(x)	Tangent of x	$\frac{n\pi}{2} < x < (\frac{n}{2}+1)\pi$ $(n = \pm1, \pm3, \ldots)$

FILE TYPES

File Type	Name2	Contents of File	Comments
Circuit	CKT	[Element-by-element description of a circuit]	Order of the elements not important
Nest	NEST	XNODES k $n_1 \ldots n_k$ ARGS n $arg_1 \ldots arg_n$ [Element-by-element description of a circuit]	External nodes must be specified; all network variables must be in ARGS list; order of the elements not important
Function	FNCTN	[Description of a mathematical function which can use standard functions and/or other defined functions]	Defined functions presently not allowed in CIRCAL-2
Functional	FNCTNL	[Description of a mathematical functional which can use standard functionals and/or other defined functionals]	Defined functionals presently not allowed in CIRCAL-2
Defined Command	DEFCOM	ARGS n $arg_1 \ldots arg_n$) [Description of a defined command consisting of a series of DEFCOM primitives] END)	All variables used which are not network variables must be in ARGS list; all statements must end with); all labels must be followed by '

DEFCOM PRIMITIVES

Primitive	Operation Performed	Comments
REPEAT $par_1 \ldots par_n$	Re-analysis of the circuit	"par_i" are assignment statements containing only network parameters and DEFCOM arguments
IF booleanexp THEN statement	"statement" is executed only if "booleanexp" has the value TRUE.	"booleanexp" must be of the form "var1 op var2" where "var_i" are numbers, variables, or specifications of analysis data and "op" is one of GRT, GEQ, EQL, NEQ, LEQ, or LES; "statement" is any DEFCOM primitive
GOTO label	Unconditional transfer to the statement preceded by "label"	"label" is any combination of six or fewer characters, beginning with a letter, which has been assigned to the statement to which a transfer is to be made.
PRINT n $var_1 \ldots var_n$	Prints the values of the variables "$var_1 \ldots var_n$" along with their names.	"n" is the number of variables whose values are to be printed; "$var_1 \ldots var_n$" is a list of the variable names.
var1 = var2 op var3 (assignment statement)	"var1" is assigned the value of the expression "var2 op var3"	"var1" is either a network parameter or DEFCOM argument; "var2" and "var3" are either network parameters, DEFCOM arguments, numbers, or specifications of analysis data; "op" is one of +, -, *, or /

ANALYSIS ROUTINES

Name of Routine	Type of Analysis	Elements Allowed	Output
FREQ	Linear frequency domain nodal analysis	LTI R's, L's, C's Independent current sources Voltage dependent current sources Nests (any level) No functions or functionals allowed	Real part of voltage at node "n" with respect to ground, "VR n" Imaginary part of voltage at node "n" with respect to ground, "VI n"
PRNTDS	Prints network data structure	----	Printout of the data structure created for the circuit being analyzed along with headings and values of pointers