BIAS REPORT

by

R. Gyurcsik, D. Cheung, F. Ma and T. Yee

Memorandum No. UCB/ERL M82/89

16 December 1982

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

BIAS REPORT

Ron Gyurcsik
Danny Cheung
Frank Ma
Tony Yee

ELECTRONICS RESEARCH LABORATORY
College of Engineering
University of California, Berkeley
Berkeley, California 94720

ABSTRACT

Integrated circuit simulation programs for use on desktop computers are more feasible due to improvements in processing speed and memory size. Three circuit simulation programs, BIASB.36, BIASP.36 and BIASD.36; have been written for the Hewlett-Packard 9826 and 9836 desktop computers. BIASB.36 and BIASP.36 are interactive MOS integrated circuit simulation programs written in BASIC and PASCAL, respectively. BIASD.36 is an interactive bipolar circuit simulation program written in BASIC. All three programs have DC operating point and time-domain transient analysis capabilities. DC transfer analysis is provided in BIASB.36; and small-signal AC gain and input impedance analysis are provided in BIASD.36. BIASB.36 and BIASP.36 can simulate a 27-node 50-mosfet circuit over 100 timepoints in 70 and 27 minutes , respectively, while BIASD.36 can simulate a 9-node 5-bipolar transistor circuit over 100 timepoints in 8.25 minutes.

BIAS REPORT

Ron Gyurcsik
Danny Cheung
Frank Ma
Tony Yee

ELECTRONICS RESEARCH LABORATORY
College of Engineering
University of California, Berkeley
Berkeley, California 94720

ACKNOWLEDGEMENTS

The development of the BIAS36 programs could not have been accomplished without the aid of faculty and co-workers. The continuing encouragement and support of our research advisor, Professor D. O. Pederson is greatly apreciated. We also acknowledge the contribution of Professor A. R. Newton, who developed the origional version of BIASL.25 from which BIASB.45, BIASB.36 and BIASP.36 were based, and Dr. B. Biehl, who wrote the original BIASD from which BIASD.36 was based. Discussions with staff of the CAD group of Hewlett-Packard is also acknowledged.

TABLE OF CONTENTS

1. Introduction....1
2. History and Background....2
3. Program Descriptions....3
   3.1. BIASB.36 & BIASP.36....3
      3.1.1. Program Structure....3
      3.1.2. Translation & Enhancements....4
   3.2. BIASD.36....5
      3.2.1. Program Structure....5
      3.2.2. Translation & Enhancements....6
4. Program Comparisons....8
   4.1. BIASB.36 & BIASP.36....8
   4.2. BIASD.36....8
5. Conclusions & Recommendations....9
   5.1. BIASB.36 & BIASP.36....9
   5.2. BIASD.36....9

Appendix A: BIASL.25 and BIASB.45....11
Appendix B: MOS & Bipolar Transistor Models....13
Appendix C: Solution Algorithm....17
Appendix D: Sparse Matrix....18
Appendix E: Users Manuals....20
Appendix F: Program Examples....47
Appendix G: Program Listings....72

1. INTRODUCTION

Integrated circuit simulation requires large amounts of computer memory and many computational steps. Until recently, simulation programs written for desktop computers were limited to small circuits(< 15 nodes) because of the desktop's limited memory. With recent increases in the amount of memory and processing speed available on desktop computers more powerful circuit simulators can be written for them. Three cirucit simulation programs, BIASB.36, BIASD.36 and BIASP.36, are enhancements of earlier versions and have been written to operate on the Hewlett-Packard 9826/36 desktop computers.

BIASB.36 and BIASP.36 are MOS integrated circuit simulation programs. BIASB.36 is written in interpretive BASIC and BIASP.36 is written in interpretive and compiled PASCAL. BIASD.36 is a bipolar circuit simulation program written in interpretive BASIC.

This report presents the history and description of each program and the conversion and enhancements from the preceeding versions. Also presented are the simulation results from benchmark circuits.

2. HISTORY

BIASB.36, BIASP.36 and BIASD.36 are the most recent programs in a series of related programs written at Berkeley dating back to 1967 [1]. In 1967 W. G. Howard wrote BIASX in FORTRAN for the IBM 1800. BIASX was a program written to simulate the change in DC operating point of a circuit with temperature. This program was rewritten in 1968 in a more consistent FORTRAN by G. Shaeffer and released as BIAS2. BIAS2 was run in batch on the CDC 6400 and used by students in classes at Berkeley in the winter and spring of 1969.

W. J. McCalla wrote BIASIII in the summer of 1969 and it was used by students at Berkeley through the spring quarter of 1970. A rewrite of BIAS3 called BIASN was done by S. P. Fan. BIASN included sparse-matrix methods and the formulation of adjunct networks used to calculate circuit noise. McCalla later combined BIASIII and a program called FRANK into SLIC which included transient time-domain analysis.

In the summer of 1972 Roger Cheung rewrote BIASN using threaded-list storage and LU decomposition. Ron Barham continued the work of Roger Cheung. Barham introduced new schemes for iteration termination on 32 bit machines, and was successful in simulating the DC operating point of a 741 op-amp.

Following the work of BIASIII and BIASN, Dr. Brian Biehl of the Harry Diamond labs wrote BIASD [2]. BIASD is a bipolar circuit simulator written in BASIC for the Hewlett-Packard 9830A calculator. Biehl later rewrote BIASD for use on the Hewlett-Packard 9845 computer. These two versions of BIASD used the matrix analysis ROM available on both the 9830A and 9845. Biehl then rewrote BIASD in minicomputer compatible FORTRAN<1>, incorporating sparse-matrix methods. BIASD.36 is based on the FORTRAN version of BIASD. Throughout the report, BIASD for the 9830A will be refered to as BIASD-1, BIASD-2 for the version for the 9845 and BIASD-3 for the FORTRAN version.

In 1975 Richard Newton wrote a new simulation program, BIASL.25 [3]. BIASL.25 is a MOS circuit simulation program written in Hewlett-Packared Language(HPL) for the Hewlett-Packard 9825 calculator. In 1981 Ron Gyurcsik wrote BIASB.45 [4], a rewrite of BIASL.25, in BASIC for use on the Hewlett-Packard 9845 computer. Both BIASB.36 and BIASP.36 are based on BIASB.45.

3. PROGRAM DESCRIPTION

3.1. BIASB and BIASP

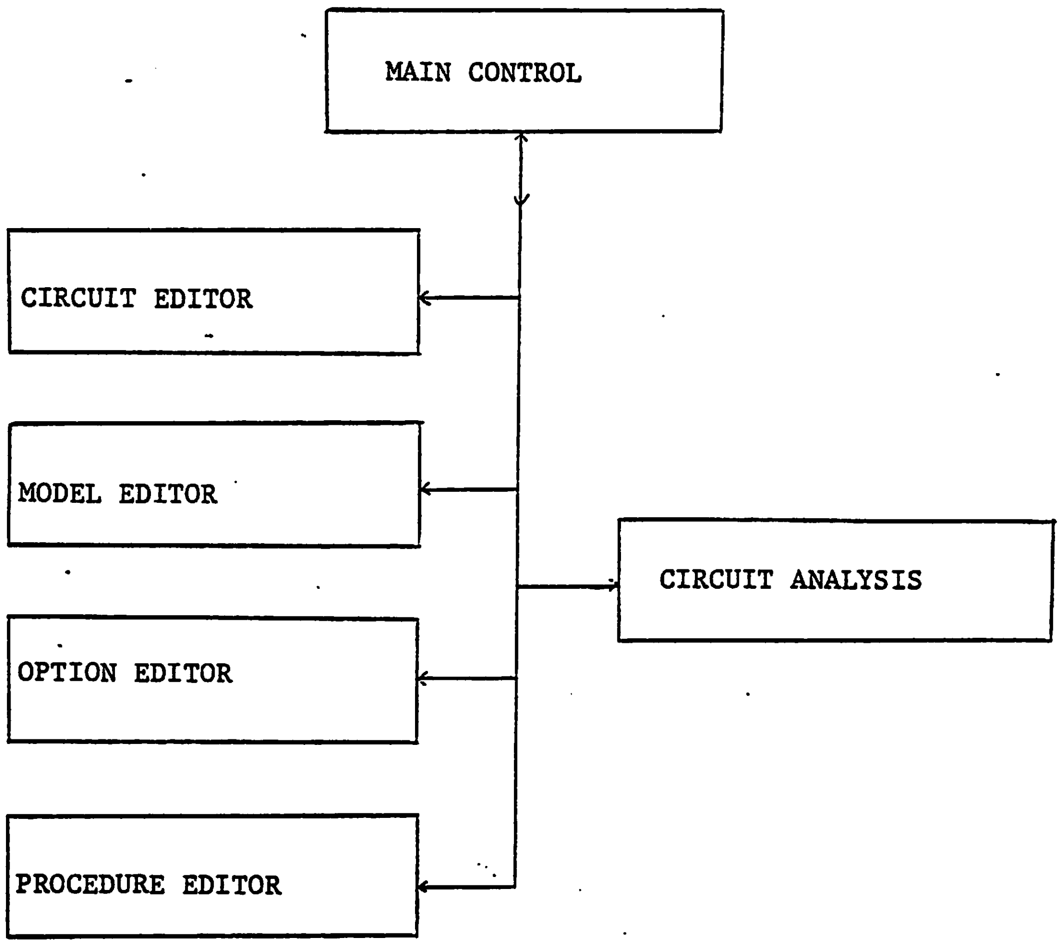
3.1.1. PROGRAM STRUCTURE

The program structure of BIASB.36 and BIASP.36 consists of a Main Controller, a series of editors: Circuit, Model, Option and Procedure, and an Analysis program section, as shown in Figure 1. Program operation begins in the Main Controller. From the Main Controller, the user may access any of the editors, define simulation conditions, initiate circuit simulation, and store and retrieve circuit and model descriptions from data files. When the use of an editor is ended or simulation is complete, program control is returned to the Main Controller.

Circuit-element descriptions and source-model descriptions are entered, changed or deleted through the Circuit Editor. Model descriptions for capacitors, diodes, and MOS transistors are entered and changed through the Model Editor. Program options are set and changed through the Option Editor. The Analysis component uses the descriptions entered from the editors to do the circuit simulation.

A procedure file is a list of BIASB.36 and BIASP.36 data and commands. The procedure file is entered and changed through the Procedure Editor. The program then executes the commands from the procedure file instead of commands given directly by the user, thus allowing for user independent operation.

BIASB.36 and BIASP.36 formulate the circuit equation using modified nodal analysis (MNA). Sparse-matrix techniques are used to store the modified nodal admittance matrix. The matrix is stored as a bi-directional threaded list [5]. Markowitz reordering is used to minimize the number of multiplications and divisions needed to solve the matrix equations. The Newton-Raphson algorithm is used to solve the circuit equations. The integration method used is the second order trapezoidal method.



BIASB & BIASP
PROGRAM STRUCTURE

FIGURE 1

3.1.2. TRANSLATION and ENHANCEMENTS

BIASB.36 is a line-by-line translation of BIASB.45 with slightly different BASIC syntax, and thus the data and program structures are similar. The difference lies in the operation of matrixes. BIASB.45 uses the advance matrix-operation functions available on the Hewelett-Packard 9845. Due to the absence of these functions on the Hewelett Packard 9836, the functions are implemented with special software routines.

The program structures of BIASP.36 and BIASB.45 are similar but their data structures differ due to differences between PASCAL and BASIC. PASCAL supports structures such as records which store mixed data types, primitives and dynamic pointers. BIASP.36 uses arrays of reçords to store circuit elements and model descriptions and dynamic linked-list to store the sparse-matrix. For example, the record structure used to store a resistor element is shown in Figure 2. It consists of the name and all of the parameters associated with the resistor.

Record Structure of a Resistor

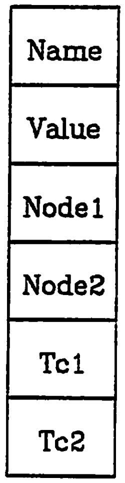


Figure 2

BIASB.36 and BIASP.36 maintain BIASB.45's functionality. To realize memory and speed savings, sparse-matrix techniques are implemented in BIASB.36 and BIASP.36.

Bi-directional linked lists are used to store the modifed nodal matrix and the Markowitz reordering algorithm is used to minimize the number of long operations. In addition, DC transfer characteristic analyses are implemented to allow the user to simulate I-V and V-V characteritics of the circuit. A simple Shichman-Hodges MOS transistor model is added to the programs to allow a less accurate but faster simulation compared with the SPICE MOS-II like model that is also available.

3.2. BIASD.36

3.2.1. Program Structure[2]

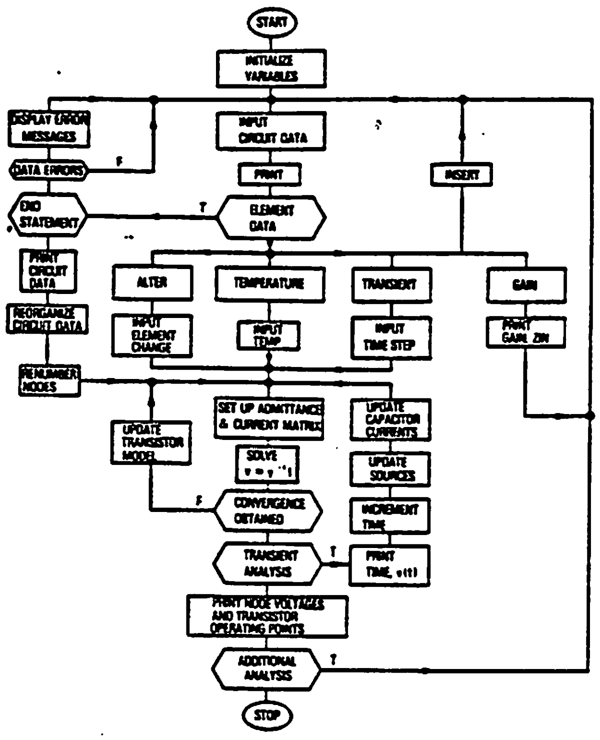
A flow diagram of input processing is shown in Figure 3. The input consists of data and control entries. The content of the first three columns of an input line determines the process path. A number from 1 ot 8 is assigned to a flag variable when a control statement is entered to indicate the desired function, whereas an input statement is processed as it is entered. An END card signals the start of the analysis, after which control is returned to the user for further commands. After the circuit has been entered, the data are rearranged for reprinting on screen in an ordered format, and then are restructured into a form suitable for analysis. The circuit nodes are first reordered<3> in the set-up procedure, and the voltage sources with series impedances are converted into their Norton equivalents.<4> At this point the restructured circuit and the pertinent circuit models are loaded as a definite admittance matrix for solving of the matrix equation:

Y*V=I

Convergence to a proper solution and termination of the iterative process are determined by the square of the node voltage changes from the previous iteration summed over all nodes, that is:

$$S_k = \sum_{n=0}^{n=N} (V_n - V_{n-1})^{\frac{1}{2}}$$

where k is the present iteration count, N is the total number of circuit nodes, and Vn the present node voltage at node n. Since S_k is determined after each iteration, the



BIASD

PROGRAM FLOW

FIGURE 3

values for S_k at the past two iterations, S_{k-1} and S_{k-2} , are available. If, during any three consecutive iterations, the values of $(S_k/N)^{½}$, $(S_{k-1}/N)^{½}$, and $(S_{k-2}/N)^{½}$ are less than 10uV, then convergence is assumed. If S_k has increased for three consecutive iterations and remains below 1 mV, the iteration process is also terminated with a possible error noted.<5> The final results of the analysis are printed in an ordered format.

3.2.2. TRANSLATION - BIASD.36

There is a fundamental difference between BIASD-1 and the present program. BIASD-1 used the matrix-operation ROM available on the HP 9830A to perform all matrix operations, and thus reduced the program size and complexity. This renders BIASD-1 completely incompatible to any machine without a matrix operation ROM. Therefore, BIASD.36 used sparse-matrix techniques to store and perform matrix operations.

Chart 1

	BIASD-1	BIASD.36
MACHINE	HP 9830A	HP 9836/26
SOLUTION ALGORITHM	MATRIX OPERATION ROM	SPARSE MATRIX LU DECOMPOSITION
DATA STORAGE TECHNIQUE	STANDARD MATRIX ARRAY	LINKED-LIST DYNAMIC ELEMENT ALLOCATION
COMPATIBLE WITH OTHERS	NO	YES

BIASD.36 is a line-to-line translation of BIASD-3. Incorporated in the program are speed- and memory- saving techniques including node reordering, sparse-matrix decomposition, sparse-matrix storage, and linked-list element storage. There is no special matrix function invoked in BIASD.36 since none is available on either the 9836/26 machines. Therefore BIASD.36 should be fully compatible to most BASIC compilers or interpreters, with minor syntatical modifications if necessary.

The present program retains similar data structure, variable names, program structure and I/O format. This is done for three reasons:

(1) To take advantage of the memory- and speed-saving techniques.

(2) To ensure proper functionality of the new version.

(3) To employ the most efficient method to acquire a realistic feel of the feasibility and potential of the project.

Subtle differences between FORTRAN and BASIC caused difficulties in I/O formatting, character-string manipulation, and system-related function conversions. Otherwise the translation was straight-forward because the structure of the two languages (BASIC/FORTRAN) is similar. The integration method implemented is the second order trapezoidal method.

4. PROGRAM COMPARISIONS

4.1. BIASB.36 & BIASP.36

The test circuits simulated by BIASB.36 & BIASP.36 are shown in Figure 4. The simulation times and results for the first three circuits are compared with those of BIASB.45 and BIASL.25 and are shown in Table 1. BIASP.36 is written in compiled PASCAL, BIASB.36 and BIASB.45 are written in interpretive BASIC and BIASL.25 is written in HPL. As we can see, BIASP.36 is about 2.5 times faster than BIASB.36; is about 5 times faster than BIASB.45A; and is about 7.5 times faster than BIASL.25.

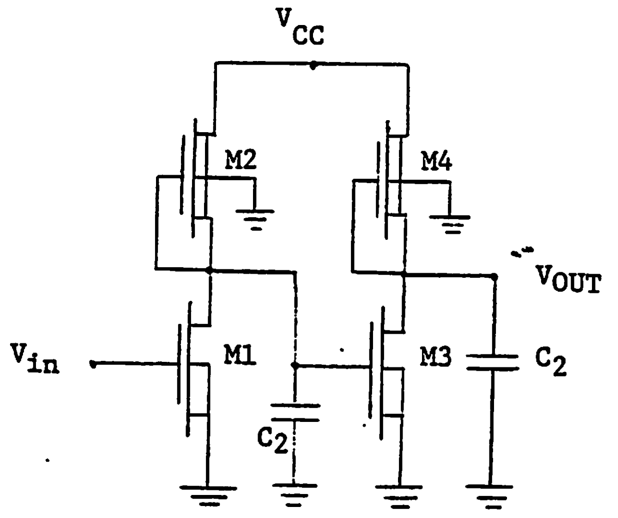
Results comparing the Shichman-Hodeges MOS transistor model and the SPICE-MOS II like MOS transistor model are shown in Table 2. We can see the former is about three times faster than the later model for large circuits.

Results comparing sparse matrix techniques and square matrix for BIASP.36 is shown in Table 3. As the circuit size increases, the speed differences also increases. For circuit 5, the sparse matrix version is runs about 5 times faster than the square matrix version.

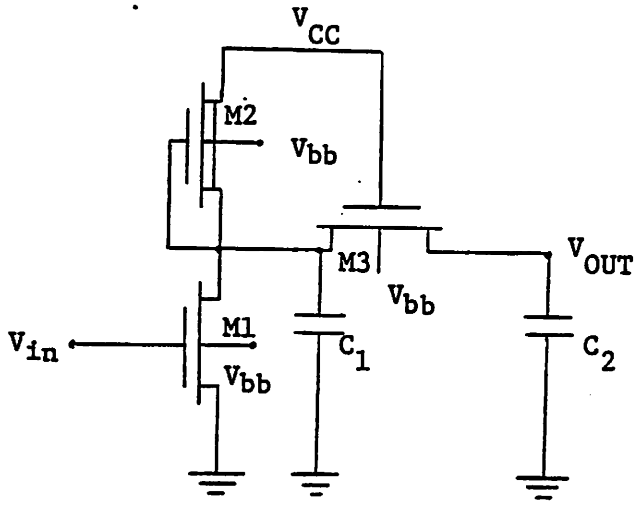
4.2. BIASD

The following tables compare the performance of the HP-9836 BASIC version to the HP-9845 version (Figure 5). As the tables indicate, the iteration time of the 9836 version is independent on the (node) size of the circuit (0.1 sec per iteration per node). Whereas, due to the square-matrix technique use on the 9845, the iteration time per node goes up as a function of the circuit size. Therefore, depending on the number of nodes contained in a circuit, the rate in which the simulation time goes up is about N**2.

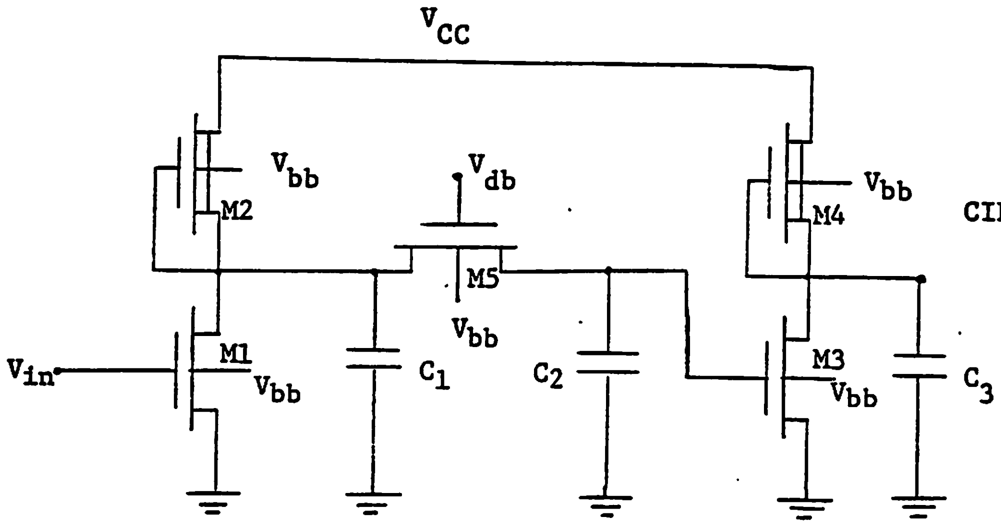
In the case of the benchmark circuits tested, BIASD.36 is anywhere from two to five times as fast as BIASD.45, with circuit sizes ranging from 9 to 19 nodes. For example, the 9-node circuit requires 12.6 seconds to converge to a DC operation point and 8.25 minutes for a 100-timepoint transient analysis. BIASD.45 required 23.4 seconds to calculate the DC operating point and 15.6 minutes for the transient analysis.



CIRCUIT 1



CIRCUIT 2



CIRCUIT 3

BIASB & BIASP
TEST CIRCUITS
FIGURE 4

CKT	BIASP ON HP9836		BIASB ON HP9836		BIASB ON HP9845A		BIASL ON HP9825	
	Time (sec)	Time/ Iter./ Tran.	Time (sec)	Time/ Iter/. Tran.	Time (sec)	Time/ Iter./ Tran.	Time (sec)	Time/ Iter/ Tran.
1	186	0.19	416	0.45	930	1.0	960	0*
2	90	0.15	309	0.51	490	0.8	1020	1.7
3	182	0.18	474	0.46	950	0.94	1530	1.36

TABLE 1

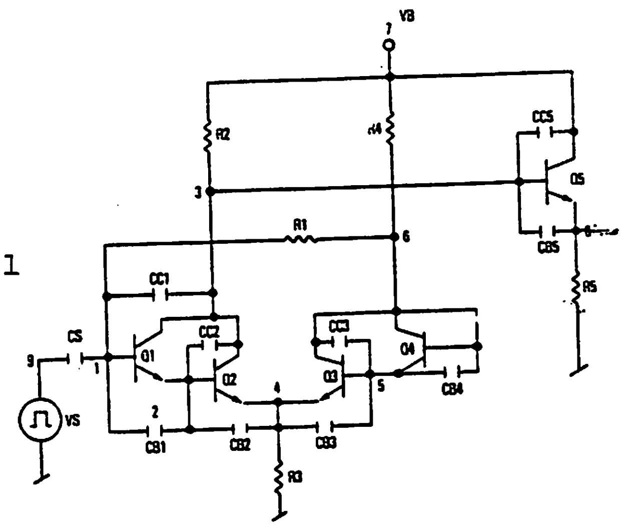
MOS TRANSISTOR MODELS PERFORMANCE OF BIASP ON 9836		
CKT	LEVEL I (s)	LEVEL II (s)
1	102	186
2	94	90
3	169	182
4	311	1100
5	3140	5327

TABLE 2

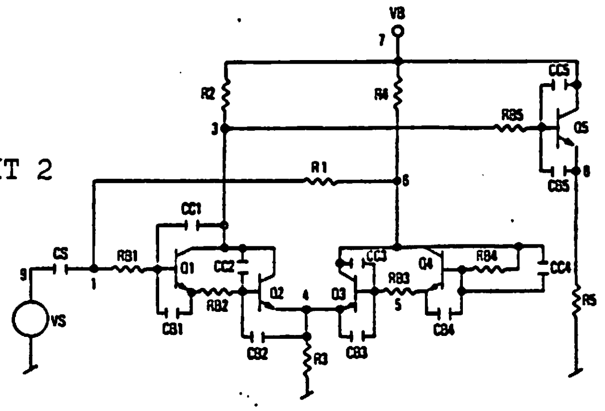
BIASP on 9836		
CKT	SPARSE MATRIX (second)	SQUARE MATRIX (second)
1	186	201
2	90	169
3	182	419
4	1100	1745
5	5327	19800

TABLE 3

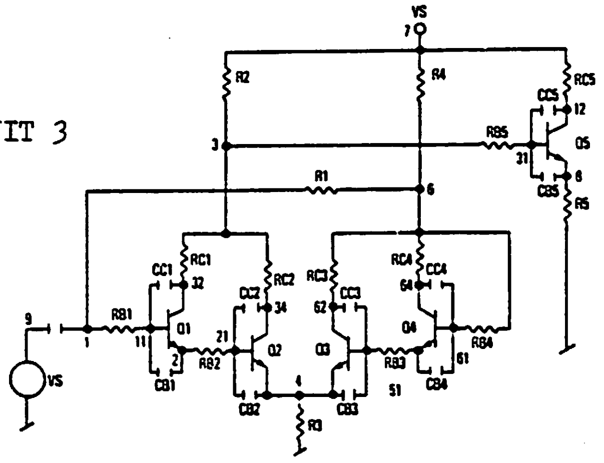
CIRCUIT 1



CIRCUIT 2



CIRCUIT 3



BIASD EXAMPLES

FIGURE 5

SIMULATION ON THE 9836					
TRANSIENT ANALYSIS					
CKT #	# OF ITERATIONS	TIME REQUIRED SEC.	TIME PER ITER. PER DEVICE SEC.	TIME PER ITER. PER NODE SEC.	# OF NODES
1	14	14	0.2	0.1	9
2	14	17.5	0.25	0.1	14
3	14	25	0.35	0.1	19

SIMULATION ON THE 9845					
TRANSIENT ANALYSIS					
CKT #	# OF ITERATIONS	TIME REQUIRED SEC.	TIME PER ITER. PER DEVICE SEC.	TIME PER ITER, PER NODE SEC.	# OF NODES
1	10	24	0.46	0.26	9
2	10	48	0.96	0.34	14
3	10	96	1.92	0.50	19

MEMORY USAGE ON 9836			
	BIASB	BIASD	BIASP
SOURCE CODE	120K	97K	140K
+ DIMENSIONS	197K*	128K*	140K

* This figure is a function of the array-sizes pre-set by the program and not a function of the size of the circuit being simulated. As the program is interpreted, memory is allocated according to the dimensions of the array declarations. However this figure can be increased or decreased by re-declaring the dimensions of the arrays.

5. CONCLUSIONS AND RECOMENDATIONS

5.1. BIASB & BIASP

Since BIASB.36 and BIASP.36 are translated from BIASB.45, a new MOS transitor model and the analysis of dc transfer characteristics are added along with many features that improve user interaction. Sparse-matrix techniques are implemented for matrix manipulation. Dynamic memory storage is employed in BIASP.36 for sparse-matrix storage.

The Hewlwtt-Packard 9836 running under interpretive BASIC is five times faster then the 9845 under interpretive BASIC, and the 9836 running compiled Pascal is five times faster then the 9836 under interpretive BASIC. These speed improvements are not achieved by the BIASB.36 and BIASP.36 programs(Figure 6), as BIASB.36 only showed a 2 times speed improvement over BIASB.45, and BIASP.36 a 2.5 times speed improvement over BIASB.36. Therefore, to improve the speed of BIASB.36 and BIASP.36, they must be optimized to run on the 9836. All of the proposed future improvements are listed below.

Future improvement:

1) Improved MOS transistor models
2) Graphic output options
3) Dynamic storage of all elements in BIASP.36
4) Interruption of transient simulation to change circuit and model descriptions.
5) Inclusion of a bipolar transistor model
6) Optimize program to machine

5.2. BIASD

BIASD.36 is structurally and functionally the same as BIASD-3. Although the program functions properly in its present form, the following could be incorporated to enhance its capability:

PROGRAM SPEED vs. RELATIVE MACHINE SPEED

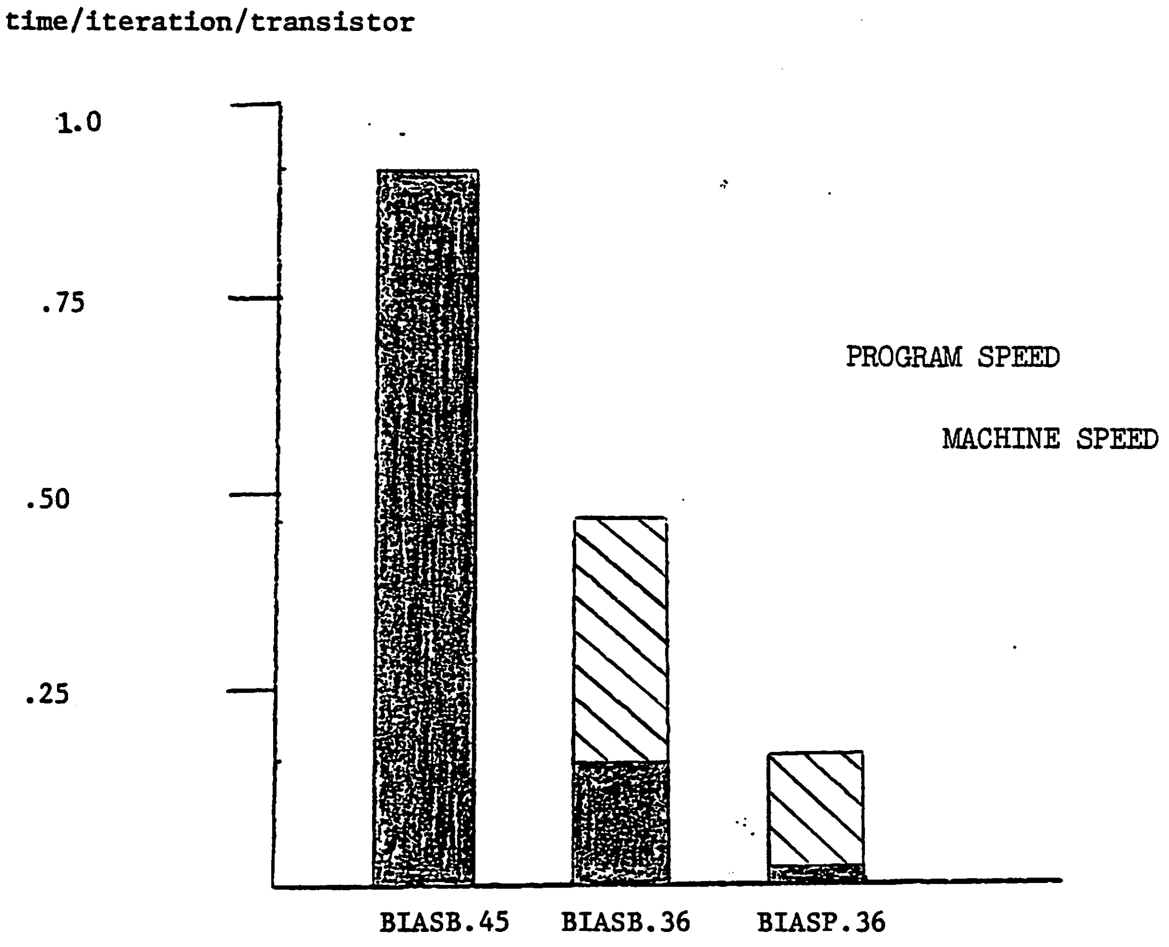


FIGURE 6

1. Optimize the program to the 9836/26 architecture.
2. Revise the existing device model.
3. Include a method to save the pertinent matrix and arrays. (for post-processing).
4. Implement the .GAIN and .GRAPH options.

APPENDIX A - BIASL.25 AND BIASB.45

BIASL.25 is an interactive circuit simulation program written in Hewlett-Packard Language(HPL) for the Hewlett-Packard 9825 desktop calculator [6]. BIASL.25 was written by Richard Newton of the University of California and Gary Taylor of Hewlett-Packard. BIASL.25 performs both DC operating point and transient analysis of circuits containing resistors, capacitors, voltage sources, current sources, diodes and mosfets.

BIASL.25 is composed of seven program sections, of which only one is resident in in the 9825's memory at any one time. BIASL.25 was partitioned as such to conserve the amount of available memory in the 9825. The seven segments consist of five editors, an analysis component and a controlling section. The controlling section links the analysis component and editors, directing the flow of program operation. The editors are used to enter circuit and model parameters, set program options and define procedure files for user independent operation. The analysis component provides the iterative solution analysis of the nonlinear circuit equations.

BIASL.25 uses modified nodal analysis to form the circuit equations. Newton-Raphson iteration to solve the circuit equations for the DC operating point and the discrete transient time points. For transient analysis, the second order trapezoidal method is used to represent the time dependent elements.

The MOS transistor model used in BIASL.25 is equivalent to the SPICE MOS-II model. This model includes the effects of subthreshold conduction, channel-length modulation, short-channel effects and capacitances.

BIASL.25 was rewritten in BASIC by Ron Gyurcsik in 1981 for use on the Hewlett-Packard 9845 desk-top computer. BIASB.45 retained the same program structure and analysis capabilities as BIASL.25. Both versions are limited in the the size of circuits that can be analyzed by the amount of memory available in the 9825 and 9845. Both simulators are able to analyze circuits containing a maximum of 15 nodes. The following table contains all of the program limitations.

Program Limitations	
Limitations	Maximum Amount
Nodes	15
Equations	25
Resistors	10
Capacitors	10
Voltage Sources	10
Current Sources	4
MOSFETs	15
Diodes	5
Capacitor Models	2
Source Models	5
MOSFET Models	3
Diode Models	1

Table 1

The limitations given for BIASB.45 are based on the worse case condition, 64 K-bytes of RAM.

Appendex B - Transistor Models

1.) MOS Transistors

a.) Shichman-Hodges Model

The Shichman-Hodges model[7] of the MOS transistor is a simplified representation of the Spice MOS-II like model. The Shichman-Hodges model is defined over the three regions of operation as the level-II like model, but the Shichman-Hodges model's boundary conditions are simplified. (only equations for the n-channel device will be given) (refer to Figure 1 - The MOS Transistor)

Resistive $V_{GS} > V_t \quad V_{DS} \leq V_{GS} - V_t$

Satsuration $V_{GS} > V_t \quad V_{DS} > V_{GS} - V_t$

Subthreshold $V_{GS} \leq V_t$

Shichman-Hodges approximates the drain-to-source saturation voltage to be the gate-to-source voltage minus a threshold voltage. The MOS equations for the three regions are:

Resistive $I_{DS} = (u_n C_{ox}/2)(W/L')[(V_{GS} - V_t)V_{DS} - V_{DS}^2/2]$ B.1

Saturation $I_{DS} = u_n C_{ox}/2(W/L')[(V_{GS} - V_t)^2]$ B.2

Subthreshold $I_{DS} = 0$ B.3

Variation in channel length with drain-to-source voltage is modeled by the following relationship,

$L' = L(1 - \lambda V_{DS})$ B.4

where λ is a constant. Threshold voltage variation with source-to-bulk voltage is modeled by

$V_t = V_{to} + [(V_{SB} + \varphi)^{1/2} - V_{SB}^{1/2}]$ B.5

V_{to} is the zero-biased threshold voltage, and φ is twice the bulk potential.

b.) **SPICE MOS-II Like Model**

The MOS model equations used in BIASB.36 and BIASP.36 are similar to SPICE Level-II model equations [8,9]. The MOS transistor model has three regions of operation: resistive, saturation and subthreshold. (For simplicity, only n-channel model equations are presented here.) The boundaries for the three regions are given by (refer to Figure 1 - the MOS transistor):

Resistive $V_{GS} > V_t \quad V_{DS} \leq V_{DSAT}$

Saturation $V_{GS} > V_t \quad V_{DS} > V_{DSAT}$

Subthreshold $V_{GS} \leq V_t$

where V_t is the device threshold voltage and V_{DSAT} is the drain-to-source saturation voltage.

Operation of the MOS transistor in the resistive region is given by Equation B.6.

$$I_{DS} = u_n C_{ox}(W/L') \Big\{[V_{GS} - V_{DS}/2 - V_{FB} - 2|\varphi_p|]V_{DS} - (2/3)\gamma\,[(2|\varphi_p| + V_{DS} + V_{SB})^{1.5} - (2|\varphi_p| + V_{SB})^{1.5}]\Big\} \qquad \text{B.6}$$

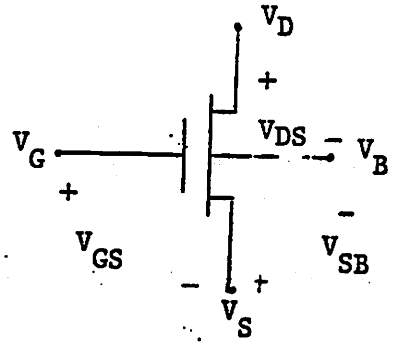
The drain-to-source current is a function of the drain-to-source, gate-to-source and source-to-bulk voltages. Operation of the MOS transistor in the saturation region is given by Equation B.7.

$$I_{DS} = u_n C_{ox}(W/L') \Big\{[V_t - V_{DSAT}/2 - V_{FB} - 2|\varphi_p|]V_{DSAT} - (2/3)\gamma\,[(2|\varphi_p| + V_{DSAT} + V_{SB})^{1.5} - (2|\varphi_p| + V_{SB})^{1.5}]\Big\} \qquad \text{B.7}$$

The drain-to-source current for the saturated device is independent of drain-to-source voltage. The drain-to-source current in the subthreshold region ideally is zero.

$$I_{DS} = 0 \qquad \text{B.8}$$

BIASB.36 and BIASP.36 MOS transistor models also account for subthreshold conduction, channel-length modulation, short-channel effects and mobility



MOSFET

FIGURE 1

variations with electric field.

Drain-to-source current in MOS transistors operating in the Subthreshld region is nonzero. Subthreshold conduction curent is modeled as a function of drain-to-source saturation voltages

$$I_{DS} = u_n C_{ox}(W/L') \{[V_t - V_{DS}/2 - V_{FB} - 2|\varphi_p|]V_{DS}$$
$$-(2/3)\gamma\,[(2|\varphi_p| + V_{DS} + V_{SB})^{1.5} - (2|\varphi_p| + V_{SB})^{1.5}$$
$$-(2|\varphi_p| + V_{SB})^{1.5}]\}\exp((V_{GS}-Vt)/C) \qquad \text{B.9}$$

where

$$C = f*T*[(q\varepsilon_{si}/(2(V_{SB} + |\varphi_p|)))^{1/2} + q*N_{fs}+C_{ox}]/(q*C_{ox}) \qquad \text{B.10}$$

The channel length of the MOS transistor is not constant, but it is a function of the drain-to-source voltage. To a first-order approximation,

$$I_{DS} = I_{DS}(1/(1 - \lambda V_{DS})) \qquad \text{B.11}$$

where lambda (λ) is the channel-length modulation parameter.

$$\lambda = (C_{ox}\,\gamma/V_{DS})\,(V_{DS}-V_{DSAT})/4 + [((V_{DS}-V_{DSAT})/4)^2 + 1]^{1/2} \qquad \text{B.12}$$

The threshold voltage is ideally dependent only on process parameters.

$$V_t = V_{FB} + |V_C| + 2|\varphi_p| + |Q_d'|/C_{ox} \qquad \text{B.13}$$

For narrow channel devices, the effects of geometry must be accounted for in the threshold voltage equation.

$$V_t = V_{FB} + |V_C| + 2|\varphi_p| + f|Q_d'|/C_{ox} \qquad \text{B.14}$$

$$f = 1 - (x_j/L')[(1 + 2x_{dmax}/x_j)^{1/2} - 1] \qquad \text{B.15}$$

For low levels of electric field in the MOS transistor, the carrier mobility is constant. When the electric field in the transistor's channel becomes large (>1e5 V/cm) the effective mobility decreases due to velocity saturation effects.

$$u_{eff} = u[1 + E/E_{cr} + E_{tr}V_{DS}]^{-E_{ex}} \quad 0<E_{ex}<1 \qquad \text{B.16}$$

c.) MOS Transistor Capacitors

The MOS transistor has several inherent capacitors. Their effect is nonlinear and depends on the MOS transistor voltages. These capacitors are shown in Figure 2, and represented by the model in Figure 3. The voltage dependence of the capacitors in Figure 3 are shown in Figure 4. C_{gsovl}, C_{gdovl} and C_{gbovl} are parasitic gate overlap capacitancess. C_{db} and C_{sb} are reverse bias p-n junction capacitances of the drain and source diffusions. C_{ox} is the gate-to-channel capacitance.

2.) Bipolar Model

The bipolar transistor equations used in BIASD are based on the modified Ebers-Moll model[2]. The Ebers-Moll model is valid over all operating regions of the bipolar transistor. (Because of the duality between NPN and PNP bipolar transistors only the NPN equations will be presented.) Refer to the Ebers-Moll circuit model in Figure 5.

$$I_E = -I_S(1 + 1/\beta_F)[\exp(V_{BE}/V_T) - 1] + I_S[\exp(V_{BC}/V_T) - 1]$$

$$+I_{RS}[\exp(V_{BE}/2V_T) - 1] \qquad \text{B.17}$$

$$I_C = I_S[\exp(V_{BE}/V_T) - 1] + I_S(1 + 1/\beta_R)[\exp(V_{BC}/V_T) - 1] \qquad \text{B.18}$$

$$I_B = I_E - I_C \qquad \text{B.19}$$

The model accounts for low collector current operation and base-width modulation effects.

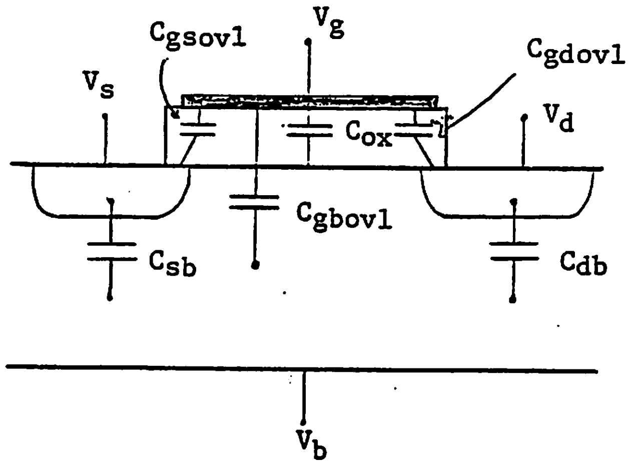
Low collector current results in a reduction of DC beta primarily due to minority carrier recombination. This effect is accounted for by the Irs term in the emitter current equation. The recombination-saturation current(Irs) is related to the foreward Beta knee current(Il) by the following relationship.

$$I_L = [(\beta_F I_{RS})^2]/I_S \qquad \text{B.20}$$

Base-width modulation is due to variations in collector-base voltage. Effects of base-width modulation is modeled by a voltage dependence of the foreward Beta and the saturation current.

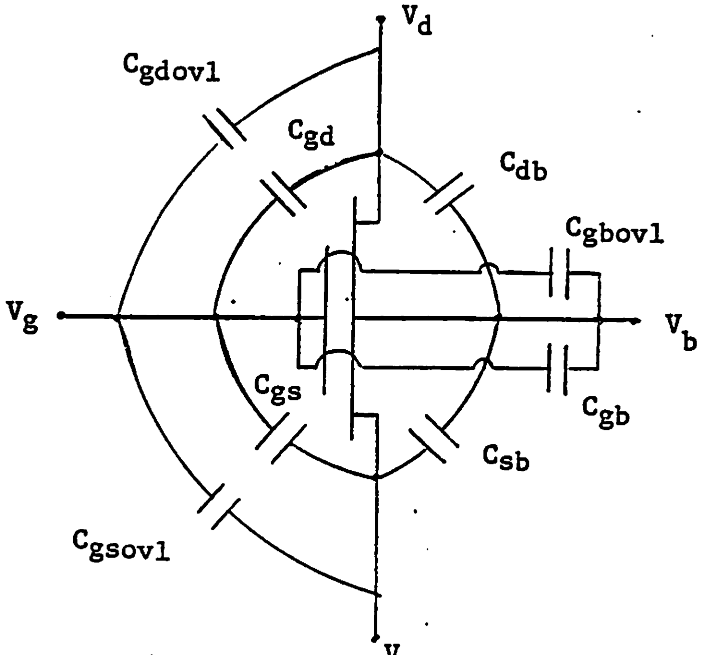
$$\beta_F = \beta_F/(1 - V_{CB}/V_A) \approx \beta_F\,(1 + V_{CB}/V_A) \qquad \text{B.21}$$

$$I_S = I_S/(1 - V_{CB}/V_A) \approx \beta_F\,(1 + V_{CB}/V_A) \qquad \text{B.22}$$



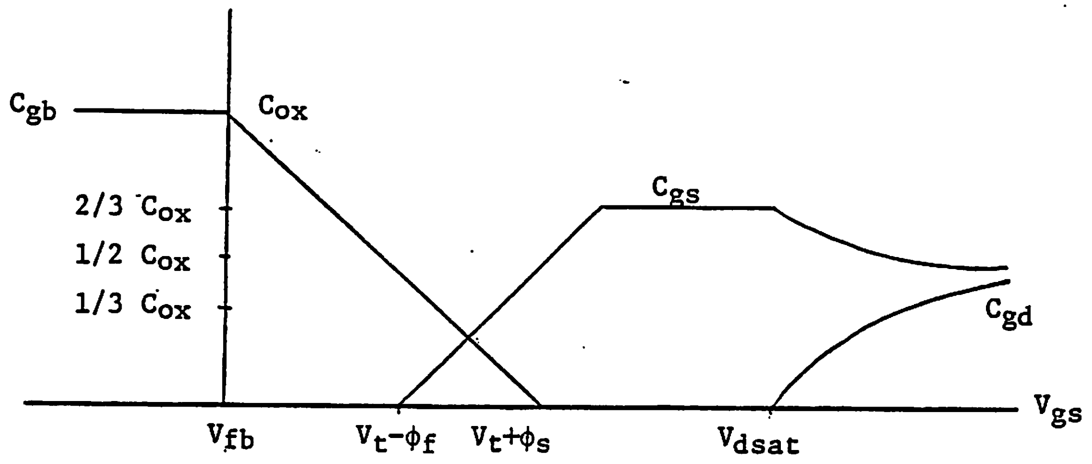
MOS CAPACITANCE

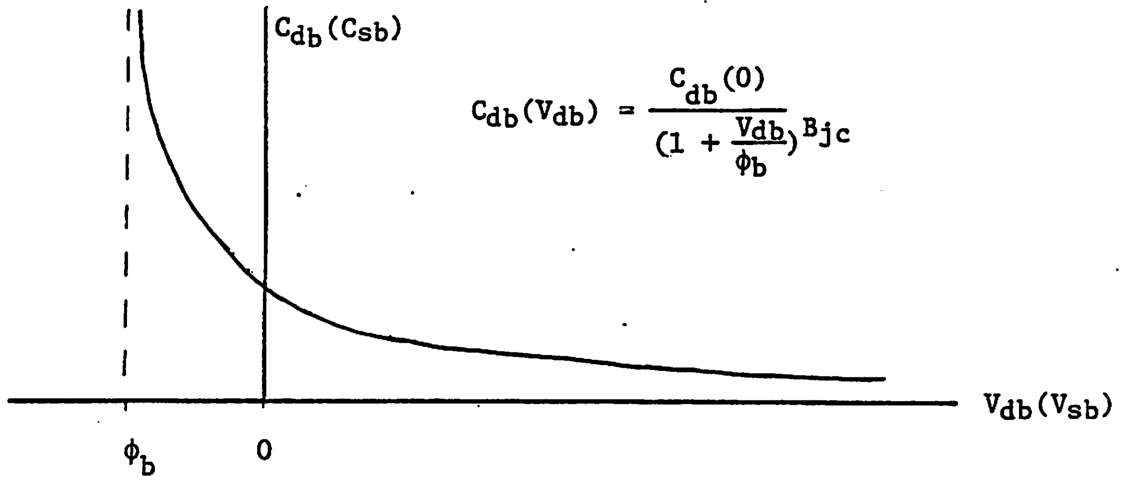
FIGURE 2



MOS CAPACITOR MODEL

FIGURE 3





MOS CAPACITORS VOLTAGE DEPENDENCE

FIGURE 4

Appendix C - Solution Algorithms

Circuit equations are generally nonlinear functions of voltage. Therefore, for a set of n such equations in n unknowns there is no guarantee of a direct solution. Without a direct solution, an iterative solution method must be used.

BIASB, BIASP and BIASD use the Newton-Raphson iteration method to solve the set of nonlinear equations[5]. Newton-Raphson linearizes the nonlinear equation by solving for the coefficent using the value for v from the previous iterations. For a single equation and unknown,

$$I_{n+1} = G_n(V_n,t)*V_{n+1} + K(V_n,t) \qquad \text{C.1}$$

$$V_{n+1} = [I_{n+1}-K(V_n,t)]/\, G_n(V_n,t) \qquad \text{C.2}$$

where $K(V_n\,,t)$ is the Norton equivalent intercept. The equation is then solved. The new solution of the equation is compared to the solution from the previous iteration.

$$\Delta V_{n+1} = |V_{n+1} - V_n| \qquad \text{C.3}$$

If the difference between them is greater than an allowed error the process is repeated. If the difference is less than the allowable error, the solutions is complete.

Newton-Raphson is further applied to the solution of a set of nonlinear circuit equations. The matrix representation of a circuit is

$$I' = G' \times V'$$

The column vector V' contains the unknowns. Matrices G' and I' are a function of the unknowns, but are solved for by values of the unknowns from the previous iteration. The error test is applied on every unknown. If any of the unknowns fails the error test, the iteration process is continued, otherwise the solution is considered complete. With Newton-Raphson systems of nonlinear equations can be solved.

Appendix D - Sparse Matrix

1. Sparse Matrix Storage

In BIASB.36 and BIASP.36, sparse matrix storage is implemented using a bi-directional threaded list[5]. The bi-directional threaded list method is used because it provides more flexibility than other methods. Furthermore, new non-zero elements can easily be appended into the matrix without much change in the data storage structure of the matrix.

In BIASB.36, arrays, governed by the same set of index numbers, are used to implement the bi-directional threaded list method (IPT(*), JPT(*), ROW(*),COL(*), and RHS(*)). Storage locations are provided for each non-zero element in the RHS(*). The arrays ROW(*) and COL(*) records the i,j co-ordinates of each non-zero element of the matrix. Finally, the arrays IPT(*) and JPT(*) point to the row 'i' of the next element in a column and the column 'j' of the next element in a row respectively. An example of this implementation is showed in Figure 1.

The bi-directional threaded list structure implemented in BIASP.36 consists two static arrays and dynamic elements. The two static arrays, ROW and COL, serve as the row and column pointers respectively. Each element in ROW and COL is a record consists of a pointer to the first element in its row or column, an integer which is the number of elements in that row or column and the old index row or column before reordering. Entries pointed by ROW and COL are the non-zero elements and are allocated dynamically in **procedure addelement** if not yet exist. Each entry consist of value, row and column index, and pointers to next row and column elements. The right hand side value is stored in BASR.

The sparse matrix storage technique used in BIASD is "row-column indexing" [2]. The arrays IUR and IUC are the upper triangular row element counter and element column indicator respectively, and the arrays ILC and ILR are the lower triangular column element counter and element row indicator respectively. Due the fact that IUR and ILC, and IUC and ILR are declared to be equivalent, these arrays are collapsed simply into IUR and ILR in the BASIC version. The "row-column indexing" scheme is similar to indirect addressing in assembly language program. The location of the first non-zero entry of the upper trangular row entries is given by IUR and that of the lower

$$\begin{bmatrix} 11 & 0 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 & -6 \\ 0 & -2 & 6 & -1 & -3 \\ -4 & 0 & 0 & 5 & 0 \\ 0 & 0 & -5 & 0 & 7 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 5 \\ 2 \\ 4 \end{bmatrix}$$

	IPT	ROW	COL	JPT	RHS	
(1)	6	-	-	6	1.0	b_1
(2)	8	-	-	8	3.0	b_2
(3)	11	-	-	9	5.0	b_3
(4)	13	-	-	7	2.0	b_4
(5)	10	-	-	12	4.0	b_5
(6)	7	1	1	0	11.0	a_{11}
(7)	0	4	1	15	-4.0	a_{41}
(8)	9	2	2	10	8.0	a_{22}
(9)	0	3	2	11	-2.0	a_{32}
(10)	14	2	5	0	-6.0	a_{25}
(11)	12	3	3	13	6.0	a_{33}
(12)	0	5	3	16	-5.0	a_{53}
(13)	15	3	4	14	-1.0	a_{34}
(14)	16	3	5	0	-3.0	a_{35}
(15)	0	4	4	0	5.0	a_{44}
(16)	0	5	5	0	7.0	a_{55}

Sparse Matrix Storage via Bi-directional Threaded List.

FIGURE 1

triangular column is given by ILC.

2. Re-ordering Algorithm:

The Markowitz re-ordering algorithm is employed in the BIASB.36, BIASP.36 and BIASD.36 program. Use of the Markowitz criterion in reordering the matrix approximately minimizes the multiplications needed to perform LU decomposition. Since Modified Nodal Analysis is used in the programs, the matrix is diagonally dominated. Therefore, the choice of the pivot element is restrained to the main diagonal elements in order to reduce the number of computations needed. Row and column exchange is used to eliminate zero diagonal elements before pivoting is done. Experimental results show that less fill-ins are generated and less computations are needed when using the Markowitz algorithm

3. LU Decomposition and Solution

Doolittle method is used for the LU decomposition in both programs. Since the diagonal elements of the L matrix are all ones', the same memory locations of the original matrix can be used to store data for both the L and U matrice. Therefore, no extra memory is needed for the L and U matrix storages. Matrix solution is obtained by performing forward and backward substitutions.

APPENDIX E - USER'S MANUAL

BIASB.36 AND BIASP.36 USER'S MANUAL

4. Introduction

BIASB.36 and BIASP.36 are interactive MOS circuit simulation programs developed for use on the Hewlett-Packard 9826/36 desktop computers. (For simplicity, BIASB.36 will be refered to as BIASB and BIASP.36 will be refered to as BIASP.) The programs are structured so that at start the user is in the Main-Control component of the program. From the Main-Control component the user can either access other program components or begin analysis on the circuit to be simulated. After the analysis is completed or use of one of the other components is ended, control is returned back to the Main-Control component.

The other accessible components of the program are the Circuit Editor, Model Editor, Option Editor and Procedure Editor. The Circuit Editor is used to enter the circuit element and source descriptions. The Model Editor is used to enter the device model parameters. The Option Editor is used to define certain program constants (convergence criterion) and to set program options (use initial conditions). The Procedure Editor is used to input program commands, allowing the program to run independently.

To simulate a circuit, the user enters the circuit and model descriptions and sets the appropriate options desired. The user sets the program to begin either DC or transient analysis and then initiates the simulation. With DC analysis, all node voltages, currents of voltage sources and MOSFETs are printed. With transient analysis only those specified are printed at each time point.

5. Start

To begin operation for BIASB, the computer must be turned on with the BASIC system disc in the right hand side disc drive. The user then replaces the BASIC system disc with the BIASB disc, and loads BIASB by executing **LOAD** for the HP 9836. By pressing **RUN** ,the user will see the prompt **BIASB** , and the program is in the Main Controller.

To begin operation for BIASP, the computer must be turned on with the PASCAL system BOOT disc in the right hand side drive of the 9836. If the SYSVOL volume is not on line, you must put it in as the computer tells you to. Next, prefixing the volume to save and to load circuit description files is recommended or else the computer will save and load files on the default volume. Generally, the circuit description files are in the same disc as the program. Therefore, only prefixing the volume with the program on it is needed. At last, load the program with the eXecute command.

6. Entering Commands and Data

Each program component displays a unique prompt. Data and commands are then entered from the keyboard and input to the program by pressing ENTER. All commands are exclusive to each program component. For BIASP, commands can be lower or upper case.

Element and model names entered to the program can be as long desidred. but only the first four letters of the name are stored by the program. In order to distinguish between elements of the same type, the first four letters of the element name must be different. Numeric inputs are stored as either integer or floating-point variables. The following numeric scale factors are also available.

G = 1E9	M = 1E6	K = 1E3	m = 1E-3
u = 1E-6	n = 1E-9	p = 1E-12	f = 1E-15

Due to differences between PASCAL operating system handling of I/O and BASIC, certain commands exist in BIASP but not in BIASB. These commands will be followed by a note indicating it is only implemented in BIASP.

7. Output

The standard output device of the 9836 at the start of the execution is the CRT display. The user may redefine the output device to be the printer simply by entering 'out prin' for BIASP and entering the appropriate 9836 commands for BIASB.

8. Main Control Commands

The Main-Control section allows the user to access either one of the four editors or initiate and run a simulation. The following is a description of the comands used in the Main-Control section. Only those letters that are capitalized are necessary to be entered. The prompt given by the Main-Control section is BIASP.

8.1. CIrcuit

Switches control from the Main-Control component to the Circuit Editor.

8.2. MOdel

Switches control from the Main-Control component to the Model Editor.

8.3. OPtion

Switches control from the Main-Control component to the Option Editor.

8.4. PC (Procedure Editor)

Switches control from the Main-Control component to the Procedure Editor.

8.5. NEw

Deletes the current circuit and model descriptions.

8.6. TEmp value

Enters a new operating temperature the circuit is to be simulated at. All values are input in degrees celsius, with 27 at the default at turn-on.

8.7. DC

Initiates the program for DC analysis.

8.8. TRan tstop (tincr) (tstart)

Initiates the program for transient analysis. The values for stop time, time increment, and start time are input here. The stop time is required. The start time defaults to 0 seconds, and time increment defaults to 0.01*tstop.

8.9. GO

Instructs the program to begin circuit simulation. It will simulate the DC operating point unless transient analysis is initiated.

8.10. PRint DC

Prints the DC node voltages, and MOSFET and voltage-source currents. This command is entered after the computer has completed DC analysis.

8.11. PRint TRan V(node #) I (MOSFET name) I (v-source name)

Prints the node voltages and device currents requested. The maximum number of outputs is 6. This command must be implemented prior to transient analysis.

8.12. OUt option

This command only works in BIASP. If the option is printer, the output device is the printer. If the option is crt or 1, the output device is the CRT display. The default is the CRT display.

8.13. CReate file name

Creates a data file (file name) on the mass storage medium.

8.14. CAtalog

Gives a listing of the data file names.

8.15. KIll file name

Removes a data file (file name) from the mass storage medium.

8.16. SAve file name

Saves circuit and model descriptions on a data file (file name).

8.17. LOad file name

Loads circuit and model descriptions from a data file (file name).

9. Circuit Editor

The circuit description is entered by way of the Circuit Editor. The prompt given while in the Circuit Editor is CIRCUIT. Only those letters capitalized need be entered. Bracketed items are optional.

9.1. Element Descriptions

9.1.1. Resistors: Rxxx n1 n2 value (TC1) (TC2)

TC1 and TC2 are optional temperature coefficients.

R(T) = R(27) * (1 + TC1 * T + TC2 * SQR(T))

The value of resistance given is that at 27 degrees celsius. The resistor name must be begin with R. Resistor value is in ohms. TC1 and TC2 are the first and second order temperature coefficients respectively.

9.1.2. Capacitors: Cxxx n1 n2 value (I.C.) (model)

The initial transient condition and model is optional. The capacitor name must be begin with a C. Capacitor value is in farads. When a model is used, value is the area in meters.

9.1.3. Diodes: Dxxx nc na model (scale)

Diode nodes, nc and na, refer to cathode and anode connections. Model refers to the diode-model name which must begin with a D. An optional scaling factor is allowed. This parameter scales the junction area of the diodes. The default scale factor is 1. The diode name must begin with a D.

9.1.4. Voltage Sources: Vxxx n+ n- value (model)

Nodes n+ and n- refer to the positive and negative terminals of the voltage source. The optional source model allows the voltage source to be modeled as a periodic pulse or as a piecewise-linear function. The source model name must begin with a S. The voltage-source name must begin with a V. The voltage-source value is in volts.

9.1.5. Current Sources: Ixxx nf nt value (model)

Nodes nf and nt refer to the direction of positive current flow, from node nf to node nt. The optional source model allows the current source to be modeled as a periodic pulse or as a piecewise-linear function. The source model name must begin with a S. The current-source name must begin with an I. The current-source value is in amperes.

9.1.6. MOS Transistors:

Mxxx nd ng ns nb model [(wc) (ad) (as) (vgsi) (vdsi) (vsbi)]

Nodes nd, ng, ns and nb refer to the drain, gate, source and bulk nodes of the MOS transistor. The MOS model name is required and must begin with a M. Wc and lc are the layout channel width and length of the transistor. If not entered the default for both is 10e-6 meters (10 microns). The drain and source areas, ad and as are optional and default to zero if not given. Optional initial transient conditions may also be entered. The transistor name must begin with a M. The values of lengths are in meters, areas in square meters and voltages in volts.

9.1.7. Source Models

9.1.7.1. Pulse: Sxxx iv pv td pw (per) (tr) (tf)

Inputs iv and pv refer to the initial and pulsed value of the source and are required. The delay time, td, and the pulse width, pw, must be entered. The period, rise time, and fall time are optional. The period defaults to twice the pulse width. The rise and fall times default to 1/10 the pulse width. Iv and pv are either in volts or amperes, and all times are in seconds.

9.1.7.2. Piece-Wise Linear: Sxxx

The source-model name need only be inputted. The program will respond by asking for an initial value and then ask for subsequent time points and values. To exit this mode, the user need only enter 0 when the program asks for a time.

9.2. Commands

9.2.1. New

Removes the present circuit description from the program.

9.2.2. Kill name

Removes the circuit element given by name from the present circuit description.

9.2.3. Present node value

Allows node voltages to be preset to a specified voltage for transient analysis.

9.2.4. List

Produces a listing of the circuit description.

9.2.5. End

Exits the user from the Circuit Editor, returning operation to the Main-Control component.

10. Model Editor Model Descriptions and Commands

The device model descriptions are entered by way of the Model Editor. The prompt give while in the Model Editor is MODEL. Only those letters capitalized need be entered into the program, and bracketed items are optional.

10.1. Model Descriptions

10.1.1. Cuuu c/a [(exp) (phi)]

The capacitor model name must begin with a C. The capacitor model represents a junction capacitance. The capacitance per unit area, c/a, junction coefficient, exp, and built in potential, phi, are the capacitor model input parameters.

10.1.2. Duuu is

The saturation current in amperes is the only diode model parameter.

10.1.3. Muuu type

To enter MOS transitor parameters, the user enters the model name and polarity of transistor (N-channel or P-channel). The program responds with the prompt PARAMETER. The user then enters a parameter and its value. Entering END removes ther user from the parameter input mode. The following table is a listing of parameters and their descriptions.

Either device or process parameters can be input to the MOS model. If process parameters are input, the program calculates the device parameters. If both process and device parameters are input, the program defaults to using teh device parameters entered. (NOTE: the precedence is opposite to that of SPICE2.)

DESCRIPTION	PARAMETER	UNITS
VTO	Zero-biased threshold voltage	V
K"	Transconductance prarmeter	A/V**2
GAM	Bulk threshold parameter	V**0.5
PHI	Surface potential	V
LAM	Channel length modulation	V**-1
CGS	Gate to source overlap capacitance	F/M**2
CGD	Gate to drain overlap capacitance	F/M**2
CGB	Gate to bulk overlap capacitance	F/M**2
CDB	Zero-biased drain to bulk junction capacitance per unit area	F/M**2
CSB	Zero-biased source to bulk junction capacitance per unit area	F/M**2
COX	Gate oxide capacitance per unit area	F/CM**2
JS	Drain and source junction leakage current per unit area	A/M**2
PB	Bulk junction potential	V
NSU	Substrate doping concentration	CM**-3
NSS	Surface state density	CM**-2
UO	Surface carrier mobility	CM**2/V-S
ECR	Critical field	V/CM
EEX	Critical field exponent	-
ETR	Transverse field coefficient	-
NFS	Fast surface state density	CM**-2
LDI	Lateral diffusion	M
XJ	Metallurgical junction depth	M
POL	Poly gate diffusion doping. A positive value indicates poly doping is opposite to substrate, wile a negative indicates it is the same.	CM**-3
BJC	Bulk junction grading coefficient	-

10.2. Commands

10.2.1. List

Lists the model's parameter values.

10.2.2. New

Removes all previous model and their parameter values.

10.2.3. End

Exits the Model Editor, returning control to the Main-Control component.

11. Option Editor-Program Options

The option choices are entered by way of the Option Editor. The prompt give while in the Option Editor is OPTION. Only those letters capitalized need be entered into the program.

11.1. TStep value

Inputs the value of the time step desired.

11.2. MInimum-iteration value

Enters the minimum number of iterations in transient analysis before the program increases the time step. The default value is 3 iterations. The maximum time step allowed is that given to the program by the user.

11.3. MAximum-iteration value

Enters the maximum number of iterations allowed in the transient analysis before teh program decreases the time step. The default value is 8 iterations. The minimum time step allowed is 1/80 the value of the time step given by the user.

11.4. COnvergence value

Enters the maximum aboslute error tolerance of the program for convergence. The default value is 0.0003.

11.5. DV/dt value

Enters the maximum allowed voltage change per iteration for the MOSFET. The defaultvalue is 0.2 volts.

11.6. DI/dt value

Enters the multiplicative constant for change in current with respect to time per iteration. The default value is 0.9.

11.7. MOdel level number

Enters the MOS transistors model to be used. Level 1 uses Shichman-Hodges MOS transistor model and level 2 uses SPICE MOS-II Like Model. The default is 2.

11.8. SHort channel

The MOSFET analysis accounts for short-channel effects.

11.9. CHannel-length modulation

The MOSFET analysis accounts for change in channel length as a function of drain-to-source voltages. Note, this option is neglected if the device models include a value for LAMBDA.

11.10. SUbthreshold conduction

The MOSFET analysis accounts for device conduction when gate-to-source voltage is less than the threshold voltage.

11.11. PReset

The program uses thepreset node voltages to initiate transient analysis in contrast to an initial DC result. A note of warning: be careful when using this option and the initial condition option at the same time.

11.12. UI-use initial conditions

The program uses the device initial conditions to initiate transient analysis in contrasttoan initial DC result.

11.13. CUtoff/Resistive

Prints the MOSFET name and time when it changes between the cutoff and resistive regions of transistor operation.

11.14. SAturation/Resistive

Prints the MOSFET nameand time when itchanges between the saturation and resistive regions of transistor operation.

11.15. End

Returns program control to the Main-Control component.

12. Procedure Editor

The procedure file is entered by the way of the Procedure Editor. Only the first character need to be entered.

12.1. Write

Enter a list of commands to be executed. The last command entered should be DONE. This will return control back to the Procedure Editor.

12.2. Insert line

Inserts a new line at the give line number.

12.3. Delete line

Deletes line #.

12.4. List

Lists procedure file.

12.5. Save

Stores procedure file on mass storage device.

12.6. Get

Loads the saved procedure file.

12.7. Use

Sets program to operate using the procedure file.

12.8. End

Returns program control to the Main-Control component.

13. Program Example

The CMOS inverter in Figure 1 is to be simulated using BIASB.36. The program operation begins in the Main-Control section. The following list of commands are executed to enter the circuit imformation and do the circuit simulation.

CIRCUIT - Enter the Circuit Editor and begin entering the circuit
description.

```
M1 2 1 0 0 MOD1 5u 5u 200p 200p
M2 2 1 3 3 MOD2 5u 5u 200p 200p
COUT 2 0 5p
VCC 3 0 5
VIN 1 0 0 SM1
SM1 0 5 10n 20n 50n 5n 5n
```

LIST - List circuit description. This is shown in Figure 3.
END - Exit the Circuit Editor
MODEL - Enter the Model Editor and begin entering model descriptions.
MOD1 N - Set for entering of the n-channel model, MOD1, parameters.

```
NSU 1E15
NSS -1.5E11
U0700
ECR 1E5
EEX .1
ETR .2
XJ 1E-6


 COX 3.48E-8
 COD 1.7E-10
 CGS 1.7E-10

 CDB 1W-4
 CSB 1E-4
 JS 1E-5
```
END - Discontinue entering MOD1 parameters.

MOD2 P - Set for entering of the p-channel model, MOD2, parameters.
```
NSU 2E15
NSS -1.5E11
UO 300
ECR 1E5
EEX .1
ETR .2
XJ 1E-6
COX 3.45E-8
CGD 1.7E-10
CGS 1.7E-10
CDB 1E-4
CSB 1E4
JS 1E5
```
END - Discontinue entering MOD2 parameters.

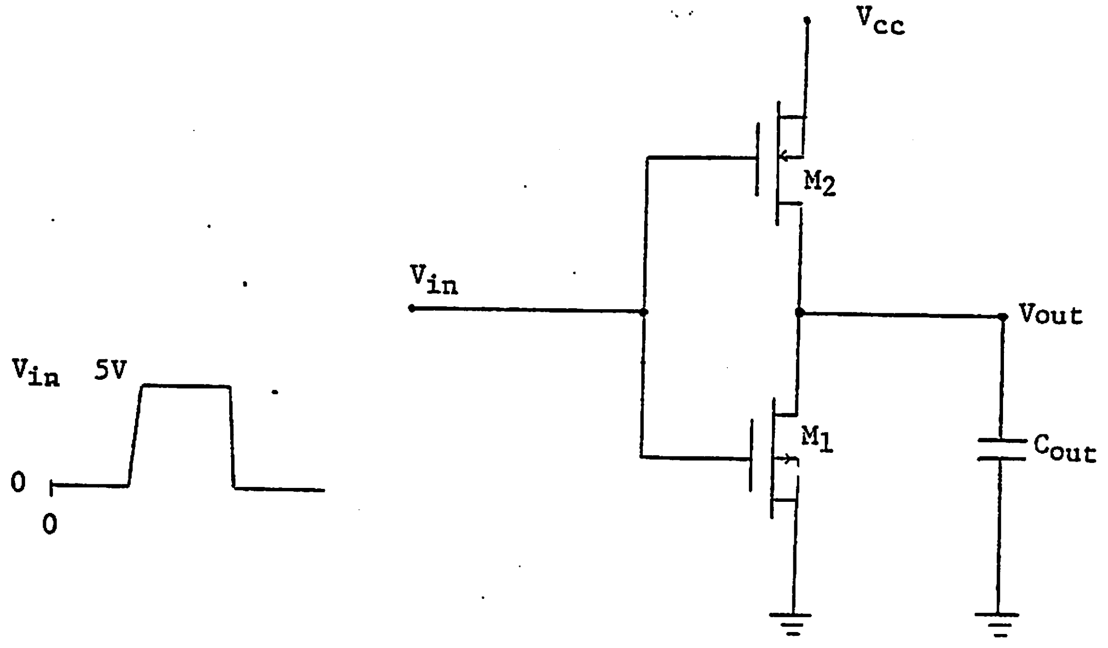
LIST - Lists the model parameters. This is shown in Figure 4.
END - Exit the Model Editor.
OPTION - Enter the Option Editor.

SHORT-CHANNEL - Include short-channel effects
CHANNEL-MODULATION - Include channel-length modulation.
SUBTHRESHOLD - Include subthreshold conduction.

LIST - Lists the options. This is shown in Figure 5.
END - Exit the Option Editor

DC - Initializes circuit for DC analysis.
GO - Simulate DC
PRINT DC - Print DC results. This is shown in Figure 6.

TRAN 50n 1n 0 - Initializes circuit for a transient analysis of
50 nsec in length at a 1 nsec time step.
PRINT TRAN V(1) V(2) - Print transient voltages of nodes 1 and 2.
GO - Begin transient analysis. This is shown in Figure 7.



CMOS INVERTER

FIGURE 1

```
LIST



                    CIRCUIT ELEMENTS


                    *CAPACITORS*
    NAME   N1  N2   VALUE       I. C.      MODEL
    COUT    2   0  1.00E-13


                    *VOLTAGE SOURCES*
    NAME   N+  N-   VALUE      MODEL
    VBB     3   0  5.00E+00
    VIN     1   0  0.00E+00     SM1


                   *MOSFETS*
    NAME   ND  NG  NS  NB    MODEL      WC          LC          AD          AS
    M1      2   1   0   0    MOD1    5.00E-06    5.00E-06    2.00E-10    2.00E-10
    M2      2   1   3   3    MOD2    5.00E-06    5.00E-06    2.00E-10    2.00E-10


                   *PRESET TRANSIENT VOLTAGES*
    NODE    VALUE     NODE     VALUE     NODE     VALUE     NODE     VALUE
      3   5.00E+00


                   *SOURCE MODEL SPECIFICATIONS*

    MODEL    SM1
                     TIME       VALUE
                   0.00E+00   0.00E+00
                   1.00E-08   0.00E+00
                   1.50E-08   5.00E+00
                   3.00E-08   5.00E+00
                   3.50E-08   0.00E+00
                   1.00E+00   0.00E+00
LIST
```

FIGURE 2

```
                 *DEVICE MODELS*


           MOS MODEL    'MOD2'  P-Channel
VTO > -8.04E-01      K'  >  1.04E-05      GAM >  7.47E-01
PHI >  6.12E-01      LAM >  0.00E+00      CGS >  1.70E-10
CGD >  1.70E-10      CGB >  0.00E+00      CDB >  1.00E-04
CSB >  1.00E-04      COX >  3.45E-08      JS  >  1.00E-05
PB  >  8.70E-01      NSU >  2.00E+15      NSS > -1.50E+11
UO  >  3.00E+02      VCR >  3.00E+00      EEX >  1.00E-01
ETR >  2.00E-01      NFS >  0.00E+00      LDI >  0.00E+00
XJ  >  1.00E-06      POL >  0.00E+00      BJC >  5.00E-01

           MOS MODEL    'MOD1'  N-Channel
VTO >  7.75E-01      K'  >  2.42E-05      GAM >  5.28E-01
PHI >  5.76E-01      LAM >  0.00E+00      CGS >  1.70E-10
CGD >  1.70E-10      CGB >  0.00E+00      CDB >  1.00E-04
CSB >  1.00E-04      COX >  3.45E-08      JS  >  1.00E-05
PB  >  8.70E-01      NSU >  1.00E+15      NSS > -1.50E+11
UO  >  7.00E+02      VCR >  3.00E+00      EEX >  1.00E-01
ETR >  2.00E-01      NFS >  0.00E+00      LDI >  0.00E+00
XJ  >  1.00E-06      POL >  0.00E+00      BJC >  5.00E-01
```

FIGURE 3

```
LIST


                 *ANALYSIS OPTIONS*

                  Tstep > 0.00E+00
                  Minit >   3
                  Maxit >   8
                  Convg > 3.00E-04
                  dV/dt > 2.00E-01
                  dI/dt > 9.00E-01

                  Shortch On
                  Chanlmd On
                  Subtcon On
                  Sat/Lin Off
                  Cut/Lin Off
                  Use I C Off
                  Preset  Off
```

FIGURE 4

```
   DC
   GO
                  6  ITERATIONS

   PRINT DC




                     *D.C. OPERATING POINT*


                        NODE VOLTAGES

    (  2) >   5.00E+00  (  1) >   0.00E+00  (  3) >   5.00E+00


                       MOSFET CURRENTS

   M1    9.08E-18  M2   -3.47E-09


                       VOLTAGE SOURCE CURRENTS

   VBB   1.43E-12  VIN   0.00E+00


  TRAN 50n 1n
  PRINT TRAN V(1) V(2)
  GO
```

FIGURE 5

```
                    *TRANSIENT ANALYSIS*


        TIME(Sec)     V( 1)       V( 2)
         1.00E-09   0.00E+00   5.00E+00
         2.00E-09   0.00E+00   5.00E+00
         3.00E-09   0.00E+00   5.00E+00
         4.00E-09   0.00E+00   5.00E+00
         5.00E-09   0.00E+00   5.00E+00
         6.00E-09   0.00E+00   5.00E+00
         7.00E-09   0.00E+00   5.00E+00
         8.00E-09   0.00E+00   5.00E+00
         9.00E-09   0.00E+00   5.00E+00
         1.00E-08   0.00E+00   5.00E+00
         1.10E-08   1.00E+00   5.03E+00
         1.20E-08   2.00E+00   4.97E+00
         1.30E-08   3.00E+00   4.62E+00
         1.40E-08   4.00E+00   3.77E+00
         1.50E-08   5.00E+00   2.37E+00
         1.60E-08   5.00E+00   1.11E+00
         1.70E-08   5.00E+00   4.56E-01
         1.80E-08   5.00E+00   1.77E-01
         1.90E-08   5.00E+00   6.68E-02
         2.00E-08   5.00E+00   2.50E-02
         2.10E-08   5.00E+00   9.30E-03
         2.20E-08   5.00E+00   3.46E-03
         2.30E-08   5.00E+00   1.29E-03
         2.40E-08   5.00E+00   4.78E-04
         2.50E-08   5.00E+00   1.78E-04
         2.60E-08   5.00E+00   6.60E-05
         2.70E-08   5.00E+00   2.45E-05
         2.80E-08   5.00E+00   9.11E-06
         2.90E-08   5.00E+00   1.33E-05
         3.00E-08   5.00E+00   4.94E-06
         3.10E-08   4.00E+00   2.12E-03
         3.20E-08   3.00E+00   4.89E-02
         3.30E-08   2.00E+00   2.05E-01
         3.40E-08   1.00E+00   5.75E-01
         3.50E-08   0.00E+00   1.28E+00
         3.60E-08   0.00E+00   2.11E+00
         3.70E-08   0.00E+00   2.86E+00
         3.80E-08   0.00E+00   3.46E+00
         3.90E-08   0.00E+00   3.93E+00
         4.00E-08   0.00E+00   4.26E+00
         4.10E-08   0.00E+00   4.50E+00
         4.20E-08   0.00E+00   4.67E+00
         4.30E-08   0.00E+00   4.78E+00
         4.40E-08   0.00E+00   4.85E+00
         4.50E-08   0.00E+00   4.90E+00
         4.60E-08   0.00E+00   4.94E+00
         4.70E-08   0.00E+00   4.96E+00
         4.80E-08   0.00E+00   4.97E+00
         4.90E-08   0.00E+00   4.98E+00
         5.00E-08   0.00E+00   4.99E+00
         5.10E-08   0.00E+00   4.99E+00
                 113  ITERATIONS
```

FIGURE 6

BIAS-D USER MANUAL (HP-9836 VERSION)

The procedure of operation outlined below is for the use of the BIAS-D simulation program on the HP-9836/26 desktop calculator. This program is suitable for use on any machine that supports compiled or interpreted BASIC, because only "generic" BASIC functions are used in this version, and the maximum allowable circuit size will depend on the size of the available memory.

A.TO PREPARE FOR SIMULATION:

1. Load the system BASIC interpreter disc.
2. When the "BASIC READY" prompt appears, remove the system disc, and load the BIASD disc on to the disc drive.
3. Type 'LOAD "BIASD"'.
4. Press the "RUN" button. When the words "INPUT DATA?" appear on the screen, BIASD is ready.

B.TO INPUT DATA:

1. Input data consist of two categories: circuit data and control statement data. The circuit element data are input by specifying the element symbol followed by the required data for that element. The control statement data are characterized by a dot (.) followed by the desured operation. Control statements do not affect the results of the analyses-- they only enable the user to direct the analysis procedure.
2. Certain general instructions must be followed to input circuit data:
   a. Each circuit element must begin in column 1.

b. Single spaces are used as delimiters between data fields (multiple spacing may result in errors).

c. Abbreviated notation cannot be used (i.e., 2u#2e-6)

d. Scientific notation may be used (i.e., 1000=1e3)

e. Decimal points are not required (i.e., 2=2.0)

f. The ground node must be node 0 (zero).

g. Compact node numbering is not required (i.e., node numbers may be skipped).

h. The maximum allowable node number is 99.

i. Element values are to be in basic units (i.e.,ohms, farads, volts, amperes, hertz, seconds).

3. Circuit Element Description:

a. Resistors, Capacitors

General form:

```
RX N1 N2 VALUE
CX N1 N2 VALUE
```

where X is any string up to three characters, N1 and N2 are node numbers (order not important), and VALUE is the resistor or capacitor value in ohms or farads. Note: VALUE cannot be zero.

b. Independent Sources: Volatge, Current

General form:

```
VX N+ N- VALUE M#
IX N+ N- VALUE M#
```

where X is any string up to three characters, N+ and N- are the positive and negative source nodes,and VALUE is the source value in volts or amperes. The letter M followed by an integer from 1 to 5 denotes the model name (see Models).

For voltage vources, either N+ and N- must be grounded (node 0). For example,

V+ 3 0 5 M1

and

V+ 0 3 -5 M1

are equivalent.

For current sources, current flows from the positive node through the negative node. The letter M followed by the model name may be omitted. However, a default number of zero is assigned.

c. Transistors

General form:

QX NC NB NE M#

where X is any string up to three characters, and NC, NB and NE are the collector, base and emitter node numbers, respectively. The letter M followed by an integer from 1 to 5 denotes the model name (see Model). The letter M followed by the model name may be omitted. However, a default number of zero is assigned.

d. Model

General form:

M# YYY F1 F2 F3 F4 F5 F6

where # is an integer from 1 ot 5 corresponding to the model number desugnated by the source or element. YYY is three-letter name designating one of five available model types as follows:

i.	NPN	npn transistor parameters
ii.	PNP	pnp transistor parameters
iii.	PUL	pulse source specifications
iv.	SIN	sinusoidal source specifications
v.	TEM	element temperature coefficients

F1, F2, ..., F6 are the data fields for specfying the above model parameters. These fields are defined below.

Field	**Parameter**	**Default**
	i. NPN–transistor parameters	
F1	Forward dc beta (Bf)	100
F2	Reverse dc beta (Br)	1
F3	Saturation current (Is)	1E-15
F4	Early voltage (Va)	1E+12
F5	Recombination current parameter (collector current at which beta=Bf/2)	0
F6	Not used	

ii. PNP–transistor parameters (same as NPN)

iii. PUL–pulse source specifications

F1	Initial source value at t=0	0
F2	Pulsed value	0
F3	Pulse delay time	0
F4	Pulse rise time	0
F5	Pulse duration (width)	0
F6	Pulse fall time	0

iv. SIN—sinusoidal source specification

F1	dc source value (offset)	0
F2	Source amplitude (0-P)	0
F3	Source frequency (Hz)	0
F4	Time delay	Tstep
F5	Phase shift (deg)	0
F6	Not used	-

The value of the sinusoidal source is determined by the equation

$$F(t) = F1 + F2\sin[2\pi F3(t-F4)+F5]$$

v. TEM—element temperature coefficients

F1	Resistor coefficient (Tc1)	0
F2	Tesistor coefficient (Tc2)	0
F3	Capacitor coefficient (Tc1)	0
F4	Capacitor coefficient (Tc2)	0
F5	Transistor beta (Tc1)	0
F6	Transistor beta (Tc2)	0

The element value at temperature T is determined by the equation

$$E(T) = E(To)[1+(T-To)Tc1+(T-To)^{2}Tc2]$$

where To=300K. Tc1 and Tc2 are the element's first- and second-order temperature coefficients, respectively. The dimensions of Tc1 and Tc2 are in decimal percentages per degree Celsius (a decimal percent of 0.002 = 2000ppm/C)

e. Comment statement

General form:

```
*any comment
```

A comment may be inseted at any line in the input circuit by using an asterik (*) in column 1 followed by any message up to 80 characters long.

f. End statement

END terminates the inputting of circuit data. If a default transistor model is used, it may be necessary to use END twice in succession. (Note: this is not the same as the END key.)

4. Control Commmands

After each type of analysis is completed, program control is returned to the operator. This is indicated by "INPUT CARD" appearing on the display. At this time it is possible to initiate anew analysis. This is done by using one of the control commands described in the following sections; all control commmands are prefixed by a dot (.).

5. .ALTER

The .ALTER command enables element values, models, and models parameters to altered. This is done as follows.

```
RX VALUE
VX VALUE
.
.
.
END
```

where X is a valid element name (i.e., has been previously defined) and VALUE is the new element using a single .ALTER command. An END statement terminates the alter operation. Models and model parameters can be altered in the same manner as entering the elements. Model types may be changed by entering a different three-letter designation (see Model). For example, a pulse source PUL can be changed to a sinusoidal source, SIN, etc. All model parameters must be entered or they are set to their default values. Both models and elements can be altered at the same time.

6. .INSERT

The .INSERT command permits elements or models to be inserted into an existing circuit. The use of this command is limited to insertion of elements and current sources between existing nodes which are not connected to a voltage source (except node 0). Any type of model may be inserted. The .INSERT command is used as follows.

```
.INSERT
RX N1 N2 VALUE
QX NC NB NE M#
M# YYY F1 F2 F3 F4 F5 F6
 .
 .
 .
END
```

The format for the elements and models is the same as described in the previous sections.

7. .TEMP

The analysis of the circuit at a tempereature other than 27 C is obtained as follows.

```
.TEMP
"TEMPERATURE (DEG C)?"
(enter temperature)
```

This procedure is repeated for each new temperature. If a TEM model has not been defined, "ILLEGAL CHARACTER" will be displayed. This model can be inserted using the .INSERT command. (Note: any aubsequent analysis is performed at the last temperature specified.)

8. .TRAN

A transient analysis analysis can be obtained using the .TRAN command as follows:

```
.TRAN
"TR T0 TSTOP TSTEP"
(enter start time, stop time, and time step)
"VXX PRT/PLO XMIN XMAX VMIN VMAX"
(enter output node and PRT.  PLO is not implemented )
```

where T0 specifies the start time for the transient analysis, TSTOP denotes the stopping point. TSTEP specifies the interval between each analysis point. In order for the transient analysis to meaningful, one or more source models (SIN, PUL) must have been specified. Voltage sources cannot be added once the initial circuit has been entered; however, source models can be inserted or altered, except for M0 . A dc transfer curve can be obtained using the .TRAN command. This done using the PUL model with such parameters that the pulse rise-time is long compared to the circuit time constants.

9. .AC

.AC is an ac small-signal analysis that computes the ac output variables as a function of frequency. The .AC commmand is used as follows.

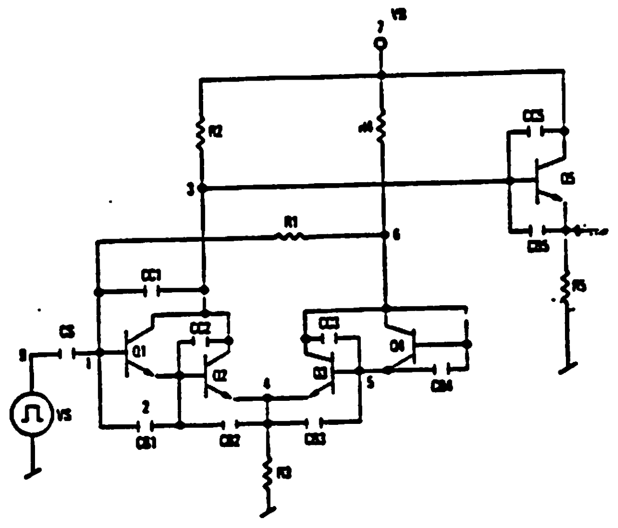
```
.AC
"VIN FSTRT FSTOP PTS/DEC"
(enter "V"-the input node, the starting frequency, final frequency
and points per decade)
"VXX PRT/PLO XMIN XMAX VMIN VMAX"
(enter "V"-the output node, "PRT"-print)
```

C. Termination

To terminate simply type END or press the dial PAUSE. It is sometimes necessary to suspend the analysis to read the output on the screen. To do this, again use the PAUSE dial, which will temporarily stop execution until the CONTINUE dial is pressed.

D. Example

The circuit in Figure 1 is simulated using BIASD. The circuit input listing and simulation results are listed as follows.



INTEGRATED PREAMPLIFIER

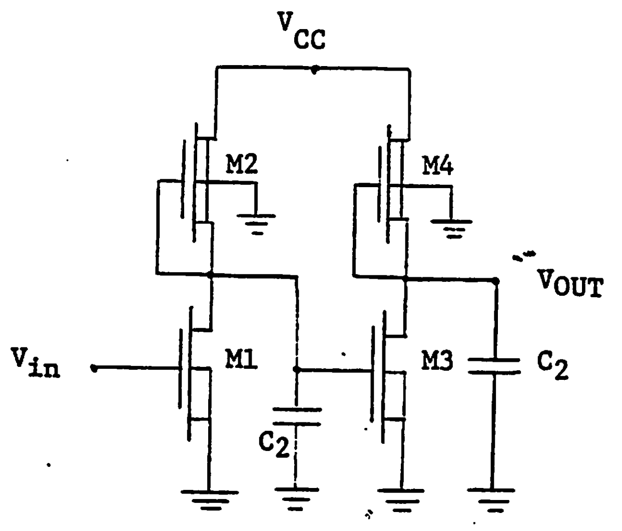
```
BIAS-D . JUNE 1982
*TEST CIRCUIT 10 (9 NODES)
   INTEGRATED PREAMPLIFIER **
* RESISTORS
R1 6 1 12K
R2 7 3 7.5K
R3 4 0 680
R4 7 6 9K
R5 8 0 5K
* TRANSISTORS
Q1 3 1 2 M2
Q2 3 2 4 M2
Q3 6 5 4 M2
Q4 6 6 5 M2
Q5 7 3 8 M2
* VOLTAGE SOURCES
V+ 7 0 6.1
VS 9 0 1.0 M1
CS 9 1 1U
* JUNCTION CAPACITORS
CB1 1 2 2P
CB2 2 4 2P
CB5 5 4 2P
CB3 3 8 2P
CC1 3 1 2P
CC2 3 2 2P
CC3 6 5 2P
(    7 3 2P
* MODELS
M1 PUL 0 -1 .5U .5U 5U .5U
M2 NPN 100 1 5E-15
END

 RESISTORS:
 NAME      Nodes        VALUE        MODEL
 R1     6       1       1.200E+04     M 0
 R2     7       3       7.500E+03     M 0
 R3     4       0       6.800E+02     M 0
 R4     7       6       9.000E+03     M 0
 R5     8       0       5.000E+03     M 0

 CAPACITORS:
 NAME      Nodes        VALUE        MODEL
 CS     9       1       1.000E-06     M 0
 CB     1       2       2.000E-12     M 0
 CB     2       4       2.000E-12     M 0
 CB     5       4       2.000E-12     M 0
 CB     3       8       2.000E-12     M 0
 CC     3       1       2.000E-12     M 0
 CC     3       2       2.000E-12     M 0
 CC     6       5       2.000E-12     M 0
 CC     7       3       2.000E-12     M 0

   ANSISTORS:
 NAME     C       B       E      MODEL
 Q1       3       1       2      M 2
 Q2       3       2       4      M 2
 Q3       6       5       4      M 2
 Q4      6      6      5    M 2
 Q5      7      3      8    M 2

 VOLTAGE SOURCES:
 NAME   + Nodes-      VALUE        MODEL
 V+     7      0     6.100E+00     M 0
 VS     9      0     1.000E+00     M 1

 MODELS:
 NAME    TYPE
 M1      PUL     0    -1   5.000E-07  5.000E-07  5.000E-06  5.000E-07  0.000E+00
 M2      NPN   100     1   5.000E-15  1.000E+12  0.000E+00  0.000E+00  0.000E+00

NODES:   9

**** END OF INPUT DATA ****

   NNODE = 7              IU= 11
    NOPS = 63      FILL-INS= 2
SPARSITY = 55.1020408163 %   MXPOS= 284    MXYPOS= 33


      S=3.957E+01
      S=2.477E+00
      S=1.782E-01
      S=7.010E-03
      S=2.488E-03
      S=1.746E-03
      S=1.685E-03
      S=1.586E-03
      S=9.312E-04
      S=1.366E-04
      S=1.263E-06
      S=7.692E-11
      S=1.473E-18
      S=6.626E-26

ITERATIONS: 14


 T=      27DEG C


 NODE VOLTAGES:
 V 1         1.8283
 V 2         1.2948
 V 3         2.5509
 V 4          .6419
 V 5         1.2950
 V 6         1.8289
 V 8         1.9035
 V 7         6.1000
 V 9         1.0000


 RANSISTOR OPERATING POINTS:
NAME      IB         IC            VBE       VBC       BETA        GM         RPI
Q1    4.602E-08  4.602E-06        .5335     -.7226    100.00    1.780E-04  5.617E+05
Q2    4.648E-06  4.648E-04        .6528    -1.2562    100.00    1.798E-02  5.561E+03
Q3    4.698E-06  4.698E-04        .6531     -.5338    100.00    1.818E-02  5.502E+03
```

```
Q4    4.652E-08  4.652E-06     .5338    0.0000    100.00    1.800E-04  5.557E+05
Q5    3.769E-06  3.769E-04     .6474   -3.5491    100.00    1.458E-02  6.858E+03
.TR
TR TO TSTOP TSTEP LINE 1950
 UTPUTS:
VXX PRT/PLO XMIN XMAX VMIN VMAX LINE7270
            T                   V 8
       +++++++++++++++++++++++++++++
        0.000E+00          1.904E+00                        3
        1.000E-07          1.904E+00                        3
        2.000E-07          1.904E+00                        3
        3.000E-07          1.904E+00                        3
        4.000E-07          1.904E+00                        3
        5.000E-07          1.904E+00                        3
        6.000E-07          2.103E+00                        6
        7.000E-07          2.438E+00                        5
        8.000E-07          2.720E+00                        5
        9.000E-07          3.014E+00                        5
        1.000E-06          3.269E+00                        5
        1.100E-06          3.416E+00                        4
        1.200E-06          3.490E+00                        5
        1.300E-06          3.612E+00                        4
        1.400E-06          3.686E+00                        5
        1.500E-06          3.788E+00                        4
        1.600E-06          3.860E+00                        5
        1.700E-06          3.947E+00                        4
        1.800E-06          4.014E+00                        5
        1.900E-06          4.089E+00                        4
        2.000E-06          4.152E+00                        5
        2.100E-06          4.217E+00                        5
        2.200E-06          4.274E+00                        5
        2.300E-06          4.331E+00                        5
        2.400E-06          4.383E+00                        5
        2.500E-06          4.433E+00                        5
        2.600E-06          4.480E+00                        5
        2.700E-06          4.525E+00                        5
        2.800E-06          4.566E+00                        5
        2.900E-06          4.607E+00                        5
        3.000E-06          4.644E+00                        5
        3.100E-06          4.680E+00                        5
        3.200E-06          4.713E+00                        4
        3.300E-06          4.745E+00                        5
        3.400E-06          4.775E+00                        4
        3.500E-06          4.803E+00                        5
        3.600E-06          4.830E+00                        4
        3.700E-06          4.855E+00                        5
        3.800E-06          4.879E+00                        4
        3.900E-06          4.902E+00                        5
        4.000E-06          4.923E+00                        4
        4.100E-06          4.944E+00                        5
        4.200E-06          4.963E+00                        4
        4.300E-06          4.981E+00                        5
        4.400E-06          4.998E+00                        4
        4.500E-06          5.014E+00                        5
        4.600E-06          5.030E+00                        4
        4.700E-06          5.044E+00                        5
        4.800E-06          5.058E+00                        4
        4.900E-06          5.071E+00                        5
        5.000E-06          5.083E+00                        4
        5.100E-06          5.095E+00                        5
```

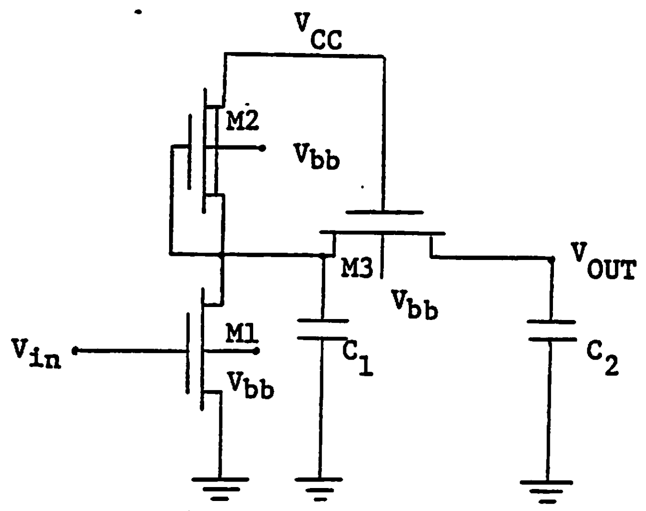
```
            5.200E-06           5.106E+00                                   4
            5.300E-06           5.116E+00                                   5
            5.400E-06           5.126E+00                                   4
            5.500E-06           5.136E+00                                   5
            5.600E-06           5.145E+00                                   4
            5.700E-06           5.153E+00                                   5
            5.800E-06           5.161E+00                                   4
            5.900E-06           5.169E+00                                   4
            6.000E-06           5.176E+00                                   4
            6.100E-06           5.111E+00                                   6
            6.200E-06           4.975E+00                                   5
            6.300E-06           4.877E+00                                   5
            6.400E-06           4.473E+00                                   8
            6.500E-06           3.179E+00                                   7
            6.600E-06           2.090E+00                                   7
            6.700E-06           1.989E+00                                   5
            6.800E-06           1.942E+00                                   4
            6.900E-06           1.924E+00                                   5
            7.000E-06           1.910E+00                                   5
            7.100E-06           1.907E+00                                   5
            7.200E-06           1.902E+00                                   5
            7.300E-06           1.903E+00                                   5
            7.400E-06           1.901E+00                                   4
            7.500E-06           1.901E+00                                   4
            7.600E-06           1.900E+00                                   4
            7.700E-06           1.901E+00                                   4
            7.800E-06           1.900E+00                                   4
            7.900E-06           1.901E+00                                   4
            8.000E-06           1.900E+00                                   4
            8.100E-06           1.900E+00                                   4
            8.200E-06           1.900E+00                                   4
            8.300E-06           1.900E+00                                   4
            8.400E-06           1.900E+00                                   4
            8.500E-06           1.900E+00                                   4
            8.600E-06           1.900E+00                                   4
            8.700E-06           1.900E+00                                   4
            8.800E-06           1.900E+00                                   4
            8.900E-06           1.900E+00                                   4
            9.000E-06           1.900E+00                                   4
            9.100E-06           1.900E+00                                   4
            9.200E-06           1.900E+00                                   4
            9.300E-06           1.900E+00                                   4
            9.400E-06           1.900E+00                                   4
            9.500E-06           1.900E+00                                   4
            9.600E-06           1.900E+00                                   4
            9.700E-06           1.900E+00                                   4
            9.800E-06           1.900E+00                                   4
            9.900E-06           1.900E+00                                   4
TOTAL ITERATIONS= 506
V8 PRT
END
```

APPENDIX F - PROGRAM EXAMPLES

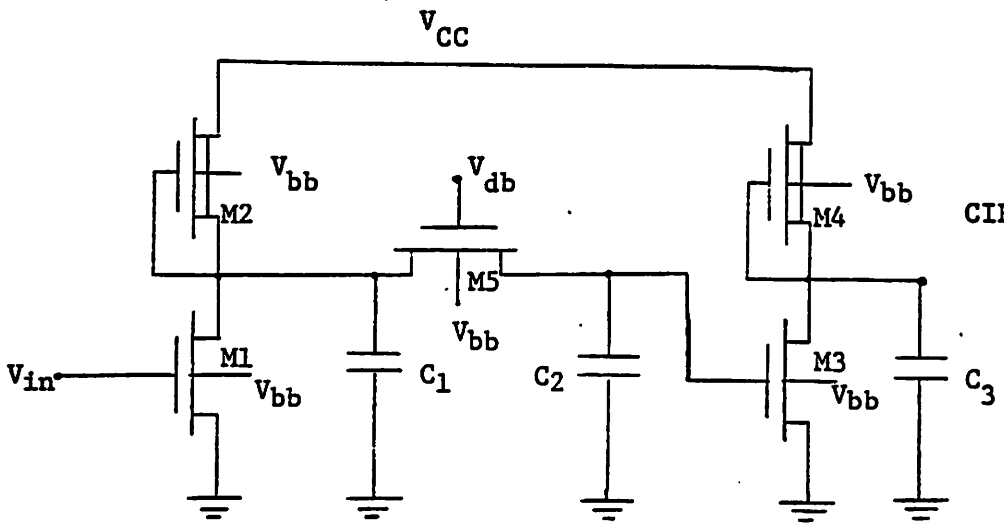
The following are circuit examples (Figure 1) from BIASB.36 and BIASP.36. The circuit input file, ciruit and model descriptions, and the simulation results are shown. Option editor information is also shown for circuit 2 and 3. The example shown for BIASD.36 is an integrated preamplifier (Figure 2). The circuit listing and simulation results are presented.



CIRCUIT 1



CIRCUIT 2



CIRCUIT 3

BIASB & BIASP
TEST CIRCUITS
FIGURE 1

```
circuit
c1 2 0 5.0e-14
c2 4 0 5.0e-14
vin 1 0 1.5 sm1
vcc 3 0 12
m1 2 1 0 0 mod1 10u 10u
m2 3 2 2 0 mod2 10u 50u
m3 4 2 0 0 mod1 10u 10u
m4 3 4 4 0 mod2 10u 50u
sm1
10
100n
0
1
0
0
list
end
model
mod1 n
vto 3.79e-1
gam 0.987
phi 0.639
cox 3.38e-8
uo 475
nsu 3.35e15
nss 2.5e10
end
mod2 n
vto -6.23
gam 0.987
phi 0.639
cox 3.38e-8
uo 475
nsu 3.35e15
nss 1.42e12
end
list
end
dc
go
print dc
tran 100n 1n 0
print tran v(1) v(2) v(4)
go
end
```

Circuit 1 Input File

```
                    CIRCUIT ELEMENT


                    *CAPACITOR*
NAME  N1  N2        VALUE       I. C.       MODEL
c1     2   0      5.00E-14
c2     4   0      5.00E-14

                    *VOLTAGES SOURCES*
NAME  N+  N-        VALUE       MODEL
vin    1   0      1.50E+00       sm1
vcc    3   0      1.20E+01



                    *MOSFETS*
NAME  ND  NG  NS  NB  MODEL      WC          LC          AD          AS
m1     2   1   0   0   mod1   1.00E-05    1.00E-05    0.00E+00    0.00E+00
m2     3   2   2   0   mod2   1.00E-05    5.00E-05    0.00E+00    0.00E+00
m3     4   2   0   0   mod1   1.00E-05    1.00E-05    0.00E+00    0.00E+00
m4     3   4   4   0   mod2   1.00E-05    5.00E-05    0.00E+00    0.00E+00

                    *PRESET TRANSIENT VOLTAGES*
NODE    VALUE       NODE    VALUE       NODE    VALUE       NODE    VALUE
  1   1.50E+00       3    1.20E+01

                    *SOURCE MODEL SPECIFICATIONS*

MODEL:    sm1
                      TIME        VALUE
                    0.00E+00    1.00E+01
                    1.00E-07    0.00E+00
                    1.00E+00    0.00E+00
CIRCUIT
```

Circuit Description - Circuit 1

```
                  *DEVICE MODELS*


            MOS MODEL   "mod1"  N-CHANNEL
vto >   3.79E-01      k    >  1.60E-05      gam >   9.86E-01
phi >   6.38E-01      lam  >  0.00E+00      cgs >   0.00E+00
cgd >   0.00E+00      cgb  >  0.00E+00      cdb >   0.00E+00
csb >   0.00E+00      cox  >  3.38E-08      js  >   0.00E+00
pb  >   8.69E-01      nsu  >  3.34E+15      nss >   2.50E+10
uo  >   4.75E+02      ecr  >  0.00E+00      eex >   0.00E+00
etr >   0.00E+00      nfs  >  0.00E+00      ldi >   0.00E+00
xj  >   0.00E+00      pol  >  0.00E+00      bjc >   5.00E-01

            MOS MODEL   "mod2"  N-CHANNEL
vto >  -6.23E+00      k    >  1.60E-05      gam >   9.86E-01
phi >   6.38E-01      lam  >  0.00E+00      cgs >   0.00E+00
cgd >   0.00E+00      cgb  >  0.00E+00      cdb >   0.00E+00
csb >   0.00E+00      cox  >  3.38E-08      js  >   0.00E+00
pb  >   8.69E-01      nsu  >  3.34E+15      nss >   1.41E+12
uo  >   4.75E+02      ecr  >  0.00E+00      eex >   0.00E+00
etr >   0.00E+00      nfs  >  0.00E+00      ldi >   0.00E+00
xj  >   0.00E+00      pol  >  0.00E+00      bjc >   5.00E-01
```

MOS Model Descriptions - Circuit 1

```
                    *D.C. OPERATION POINT*

                      NODE VOLTAGES

(2) > 1.13E+01  (4) > 2.56E-01  (1) > 1.50E+00  (3) > 1.20E+01


                      MOSFET CURRENTS

 m1     6.58E-06  m2     6.58E-06  m3     4.43E-05  m4     4.43E-05


                      VOLTAGE SOURCE CURRENT

 vin    0.00E+00  vcc   -5.09E-05

BIASP.45
```

DC Operation Point Information - Circuit 1

```
                 *TRANSIENT ANALYSIS*

TIME(SEC)      V( 1)       V( 2)      V( 4)
 1.00E-09     9.89E+00    2.95E-01    1.20E+01
 2.00E-09     9.80E+00    2.98E-01    1.19E+01
 3.00E-09     9.69E+00    3.01E-01    1.20E+01
 4.00E-09     9.59E+00    3.04E-01    1.20E+01
 5.00E-09     9.50E+00    3.07E-01    1.19E+01
 6.00E-09     9.39E+00    3.11E-01    1.20E+01
 7.00E-09     9.29E+00    3.14E-01    1.20E+01
 8.00E-09     9.19E+00    3.18E-01    1.19E+01
 9.00E-09     9.09E+00    3.22E-01    1.20E+01
 1.00E-08     9.00E+00    3.25E-01    1.20E+01
 1.10E-08     8.89E+00    3.29E-01    1.19E+01
 1.20E-08     8.79E+00    3.33E-01    1.20E+01
 1.30E-08     8.70E+00    3.37E-01    1.20E+01
 1.40E-08     8.59E+00    3.42E-01    1.19E+01
 1.50E-08     8.49E+00    3.46E-01    1.20E+01
 1.60E-08     8.39E+00    3.50E-01    1.20E+01
 1.70E-08     8.29E+00    3.55E-01    1.19E+01
 1.80E-08     8.19E+00    3.59E-01    1.20E+01
 1.90E-08     8.10E+00    3.64E-01    1.20E+01
 2.00E-08     8.00E+00    3.69E-01    1.20E+01
 2.09E-08     7.89E+00    3.74E-01    1.19E+01
 2.20E-08     7.80E+00    3.79E-01    1.19E+01
 2.29E-08     7.69E+00    3.85E-01    1.19E+01
 2.39E-08     7.59E+00    3.90E-01    1.19E+01
 2.50E-08     7.50E+00    3.96E-01    1.19E+01
 2.59E-08     7.39E+00    4.02E-01    1.19E+01
 2.69E-08     7.30E+00    4.08E-01    1.19E+01
 2.79E-08     7.20E+00    4.14E-01    1.19E+01
 2.90E-08     7.09E+00    4.20E-01    1.19E+01
 2.99E-08     7.00E+00    4.27E-01    1.19E+01
 3.09E-08     6.90E+00    4.34E-01    1.19E+01
 3.19E-08     6.80E+00    4.41E-01    1.19E+01
 3.29E-08     6.70E+00    4.48E-01    1.19E+01
 3.39E-08     6.60E+00    4.55E-01    1.19E+01
 3.49E-08     6.50E+00    4.63E-01    1.19E+01
 3.59E-08     6.40E+00    4.71E-01    1.19E+01
 3.69E-08     6.30E+00    4.80E-01    1.19E+01
 3.79E-08     6.20E+00    4.88E-01    1.19E+01
 3.89E-08     6.10E+00    4.98E-01    1.19E+01
 3.99E-08     6.00E+00    5.07E-01    1.19E+01
 4.09E-08     5.90E+00    5.17E-01    1.19E+01
 4.19E-08     5.80E+00    5.27E-01    1.19E+01
 4.29E-08     5.70E+00    5.37E-01    1.19E+01
 4.39E-08     5.60E+00    5.48E-01    1.19E+01
 4.49E-08     5.50E+00    5.60E-01    1.19E+01
 4.59E-08     5.40E+00    5.72E-01    1.19E+01
 4.69E-08     5.30E+00    5.84E-01    1.19E+01
 4.79E-08     5.20E+00    5.98E-01    1.19E+01
 4.89E-08     5.10E+00    6.12E-01    1.19E+01
 4.99E-08     5.00E+00    6.26E-01    1.19E+01
 5.09E-08     4.90E+00    6.42E-01    1.19E+01
```

5.19E-08	4.80E+00	6.58E-01	1.19E+01
5.29E-08	4.70E+00	6.75E-01	1.19E+01
5.39E-08	4.60E+00	6.94E-01	1.19E+01
5.49E-08	4.50E+00	7.13E-01	1.19E+01
5.59E-08	4.40E+00	7.34E-01	1.19E+01
5.69E-08	4.30E+00	7.55E-01	1.19E+01
5.79E-08	4.20E+00	7.79E-01	1.19E+01
5.89E-08	4.10E+00	8.04E-01	1.19E+01
5.99E-08	4.00E+00	8.31E-01	1.19E+01
6.09E-08	3.90E+00	8.60E-01	1.19E+01
6.19E-08	3.80E+00	8.92E-01	1.19E+01
6.29E-08	3.70E+00	9.26E-01	1.19E+01
6.39E-08	3.60E+00	9.63E-01	1.19E+01
6.50E-08	3.50E+00	1.00E+00	1.19E+01
6.60E-08	3.40E+00	1.05E+00	1.19E+01
6.70E-08	3.30E+00	1.10E+00	1.19E+01
6.79E-08	3.20E+00	1.15E+00	1.18E+01
6.90E-08	3.10E+00	1.22E+00	1.18E+01
7.00E-08	3.00E+00	1.29E+00	1.18E+01
7.10E-08	2.89E+00	1.38E+00	1.18E+01
7.20E-08	2.79E+00	1.48E+00	1.18E+01
7.30E-08	2.69E+00	1.60E+00	1.17E+01
7.40E-08	2.59E+00	1.75E+00	1.17E+01
7.50E-08	2.49E+00	1.92E+00	1.16E+01
7.60E-08	2.39E+00	2.12E+00	1.15E+01
7.70E-08	2.29E+00	2.34E+00	1.14E+01
7.80E-08	2.19E+00	2.58E+00	1.13E+01
7.90E-08	2.09E+00	2.84E+00	1.11E+01
8.00E-08	1.99E+00	3.11E+00	1.09E+01
8.10E-08	1.89E+00	3.40E+00	1.07E+01
8.20E-08	1.79E+00	3.70E+00	1.04E+01
8.30E-08	1.69E+00	4.01E+00	1.00E+01
8.40E-08	1.59E+00	4.32E+00	9.54E+00
8.50E-08	1.49E+00	4.65E+00	8.98E+00
8.60E-08	1.39E+00	4.98E+00	8.31E+00
8.70E-08	1.29E+00	5.31E+00	6.80E+00
8.80E-08	1.19E+00	5.72E+00	4.31E+00
8.90E-08	1.09E+00	6.12E+00	1.97E+00
9.00E-08	9.99E-01	6.51E+00	7.45E-01
9.10E-08	8.99E-01	6.89E+00	4.70E-01
9.20E-08	7.99E-01	7.27E+00	4.26E-01
9.30E-08	6.99E-01	7.64E+00	4.01E-01
9.31E-08	6.87E-01	7.68E+00	3.98E-01
9.33E-08	6.62E-01	7.77E+00	3.93E-01
9.38E-08	6.12E-01	7.95E+00	3.83E-01
9.43E-08	5.62E-01	8.03E+00	3.75E-01
9.48E-08	5.12E-01	8.11E+00	3.70E-01
9.53E-08	4.62E-01	8.19E+00	3.66E-01
9.58E-08	4.12E-01	8.26E+00	3.62E-01
9.63E-08	3.62E-01	8.34E+00	3.58E-01
9.68E-08	3.12E-01	8.42E+00	3.55E-01
9.73E-08	2.62E-01	8.49E+00	3.51E-01
9.78E-08	2.12E-01	8.57E+00	3.48E-01
9.83E-08	1.62E-01	8.64E+00	3.44E-01

```
9.88E-08     1.12E-01  8.71E+00   3.41E-01
9.93E-08     6.24E-02  8.79E+00   3.38E-01
9.98E-08     1.24E-02  8.86E+00   3.35E-01
1.00E-07    -1.92E-19  8.93E+00   3.32E-01

              249  iterations
```

Transient Result - Circuit 1

```
circuit
c1 2 0 1p
c2 4 0 1p
vin 1 0  0 sm1
vcc 3 0 10
vbb 5 0 -5
m1 2 1 0 5 mod1 10u 10u
m2 3 2 2 5 mod2 10u 10u
m3 4 3 2 5 mod1 10u 10u
sm1
0
10n
0
20n
10
1
10
0
pr 4 10
pr 2 10
list
end
model
mod1 n
vto 0.521
k  1.68e-5
gam 1.29
phi 6.62e-1
end
mod2 n
vto -4.97
k  1.68e-5
gam 1.29
phi 0.662
end
list
end
op
pr
li
en
tran 100n 1n 0
print tran v(1) v(2) v(4)
go
end
```

Circuit 2 Input File

```
                    CIRCUIT ELEMENT


                    *CAPACITOR*
NAME    N1  N2      VALUE        I. C.        MODEL
c1       2   0    1.00E-12
c2       4   0    1.00E-12

                    *VOLTAGES SOURCES*
NAME    N+  N-      VALUE        MODEL
vin      1   0    0.00E+00        sm1
vcc      3   0    1.00E+01
vbb      5   0   -5.00E+00



                    *MOSFETS*
NAME    ND  NG  NS  NB  MODEL      WC          LC          AD          AS
m1       2   1   0   5  mod1   1.00E-05    1.00E-05    0.00E+00    0.00E+00
m2       3   2   2   5  mod2   1.00E-05    1.00E-05    0.00E+00    0.00E+00
m3       4   3   2   5  mod1   1.00E-05    1.00E-05    0.00E+00    0.00E+00

                    *PRESET TRANSIENT VOLTAGES*
NODE    VALUE      NODE    VALUE      NODE    VALUE      NODE    VALUE
  2   1.00E+01       4   1.00E+01       3   1.00E+01       5   -5.00E+00


                    *SOURCE MODEL SPECIFICATIONS*

MODEL:    sm1
                      TIME        VALUE
                    0.00E+00    0.00E+00
                    1.00E-08    0.00E+00
                    2.00E-08    1.00E+01
                    1.00E+00    1.00E+01
CIRCUIT
```

Circuit Description - Circuit 2

```
                        *DEVICE MODELS*


                  MOS MODEL   "mod1"   N-CHANNEL
vto  >   5.21E-01      k    >  1.67E-05      gam >  1.29E+00
phi  >   6.62E-01      lam  >  0.00E+00      cgs >  0.00E+00
cgd  >   0.00E+00      cgb  >  0.00E÷00      cdb >  0.00E+00
csb  >   0.00E+00      cox  >  0.00E÷00      js  >  0.00E+00
pb   >   8.69E-01      nsu  >  0.00E÷00      nss >  0.00E+00
uo   >   0.00E÷00      ecr  >  0.00E÷00      eex >  0.00E+00
etr  >   0.00E+00      nfs  >  0.00E÷00      ldi >  0.00E+00
xj   >   0.00E+00      pol  >  0.00E÷00      bjc >  5.00E-01

                  MOS MODEL   "mod2"   N-CHANNEL
vto  >  -4.96E+00      k    >  1.67E-05      gam >  1.29E+00
phi  >   6.62E-01      lam  >  0.00E÷00      cgs >  0.00E+00
cgd  >   0.00E+00      cgb  >  0.00E÷00      cdb >  0.00E+00
csb  >   0.00E+00      cox  >  0.00E÷00      js  >  0.00E+00
pb   >   8.69E-01      nsu  >  0.00E÷00      nss >  0.00E+00
uo   >   0.00E+00      ecr  >  0.00E÷00      eex >  0.00E+00
etr  >   0.00E+00      nfs  >  0.00E÷00      ldi >  0.00E+00
xj   >   0.00E+00      pol  >  0.00E÷00      bjc >  5.00E-01


MODELS
BIASP.45
```

MOS Model Description - Circuit 2

```
*ANALYSIS OPTIONS*

 TSTEP >  0.00E+00
 MINIT >    3
 MAXIT >    8
 CONVG >  3.00E-04
 DV/DT >  2.00E-01
 DI/DT >  9.00E-01

 SHORTCH  OFF
 CHANLMD  OFF
 SUBTCON  OFF
 SAT/LIN  OFF
 CUT/LIN  OFF
 USE I C  OFF
 PRESET   ON
```

Options - Circuit 2

```
                    *TRANSIENT ANALYSIS*

TIME(SEC)       V( 1)        V( 2)        V( 4)
   1.00E-09    0.00E+00     9.99E+00     1.00E+01
   2.00E-09    0.00E+00     9.99E+00     9.99E+00
   3.00E-09    0.00E+00     9.99E+00     9.99E+00
   4.00E-09    0.00E+00     9.99E+00     9.99E+00
   5.00E-09    0.00E+00     9.99E+00     9.99E+00
   6.00E-09    0.00E+00     9.99E+00     9.99E+00
   7.00E-09    0.00E+00     9.99E+00     9.99E+00
   8.00E-09    0.00E+00     9.99E+00     9.99E+00
   9.00E-09    0.00E+00     9.99E+00     9.99E+00
   1.00E-08    4.44E-16     9.99E+00     9.99E+00
   1.10E-08    1.00E+00     9.99E+00     9.99E+00
   1.20E-08    2.00E+00     9.99E+00     9.99E+00
   1.30E-08    3.00E+00     9.99E+00     9.99E+00
   1.40E-08    4.00E+00     9.99E+00     9.99E+00
   1.50E-08    5.00E+00     9.96E+00     9.99E+00
   1.60E-08    6.00E+00     9.90E+00     9.99E+00
   1.70E-08    7.00E+00     9.80E+00     9.99E+00
   1.80E-08    8.00E+00     9.63E+00     9.99E+00
   1.90E-08    9.00E+00     9.39E+00     9.99E+00
   2.00E-08    1.00E+01     9.07E+00     9.99E+00
   2.09E-08    1.00E+01     8.70E+00     9.99E+00
   2.20E-08    1.00E+01     8.33E+00     9.99E+00
   2.29E-08    1.00E+01     7.97E+00     9.99E+00
   2.39E-08    1.00E+01     7.60E+00     9.99E+00
   2.50E-08    1.00E+01     7.23E+00     9.99E+00
   2.59E-08    1.00E+01     6.87E+00     9.99E+00
   2.69E-08    1.00E+01     6.51E+00     9.99E+00
   2.79E-08    1.00E+01     6.15E+00     9.99E+00
   2.90E-08    1.00E+01     5.79E+00     9.99E+00
   2.99E-08    1.00E+01     5.44E+00     9.99E+00
   3.09E-08    1.00E+01     5.09E+00     9.98E+00
   3.19E-08    1.00E+01     4.77E+00     9.97E+00
   3.29E-08    1.00E+01     4.46E+00     9.95E+00
   3.39E-08    1.00E+01     4.17E+00     9.92E+00
   3.49E-08    1.00E+01     3.91E+00     9.87E+00
   3.59E-08    1.00E+01     3.67E+00     9.82E+00
   3.69E-08    1.00E+01     3.45E+00     9.75E+00
   3.79E-08    1.00E+01     3.25E+00     9.68E+00
   3.89E-08    1.00E+01     3.08E+00     9.59E+00
   3.99E-08    1.00E+01     2.93E+00     9.50E+00
   4.09E-08    1.00E+01     2.80E+00     9.39E+00
   4.19E-08    1.00E+01     2.68E+00     9.28E+00
   4.29E-08    1.00E+01     2.58E+00     9.16E+00
   4.39E-08    1.00E+01     2.50E+00     9.03E+00
   4.49E-08    1.00E+01     2.42E+00     8.89E+00
   4.59E-08    1.00E+01     2.36E+00     8.76E+00
   4.69E-08    1.00E+01     2.31E+00     8.61E+00
   4.79E-08    1.00E+01     2.26E+00     8.47E+00
   4.89E-08    1.00E+01     2.22E+00     8.32E+00
   4.99E-08    1.00E+01     2.19E+00     8.16E+00
```

```
5.09E-08    1.00E+01    2.16E+00    8.01E+00
5.19E-08    1.00E+01    2.14E+00    7.85E+00
5.29E-08    1.00E+01    2.12E+00    7.69E+00
5.39E-08    1.00E+01    2.10E+00    7.53E+00
5.49E-08    1.00E+01    2.09E+00    7.37E+00
5.59E-08    1.00E+01    2.07E+00    7.21E+00
5.69E-08    1.00E+01    2.06E+00    7.04E+00
5.79E-08    1.00E+01    2.05E+00    6.88E+00
5.89E-08    1.00E+01    2.05E+00    6.71E+00
5.99E-08    1.00E+01    2.04E+00    6.55E+00
6.09E-08    1.00E+01    2.04E+00    6.38E+00
6.19E-08    1.00E+01    2.03E+00    6.22E+00
6.29E-08    1.00E+01    2.03E+00    6.05E+00
6.39E-08    1.00E+01    2.02E+00    5.88E+00
6.50E-08    1.00E+01    2.02E+00    5.72E+00
6.60E-08    1.00E+01    2.02E+00    5.55E+00
6.70E-08    1.00E+01    2.01E+00    5.39E+00
6.79E-08    1.00E+01    2.01E+00    5.22E+00
6.90E-08    1.00E+01    2.00E+00    5.06E+00
7.00E-08    1.00E+01    1.99E+00    4.91E+00
7.10E-08    1.00E+01    1.98E+00    4.75E+00
7.20E-08    1.00E+01    1.96E+00    4.60E+00
7.30E-08    1.00E+01    1.95E+00    4.45E+00
7.40E-08    1.00E+01    1.93E+00    4.31E+00
7.50E-08    1.00E+01    1.91E+00    4.16E+00
7.60E-08    1.00E+01    1.89E+00    4.03E+00
7.70E-08    1.00E+01    1.87E+00    3.89E+00
7.80E-08    1.00E+01    1.84E+00    3.73E+00
7.90E-08    1.00E+01    1.82E+00    3.64E+00
8.00E-08    1.00E+01    1.79E+00    3.51E+00
8.10E-08    1.00E+01    1.76E+00    3.40E+00
8.20E-08    1.00E+01    1.73E+00    3.28E+00
8.30E-08    1.00E+01    1.70E+00    3.17E+00
8.40E-08    1.00E+01    1.67E+00    3.06E+00
8.50E-08    1.00E+01    1.64E+00    2.96E+00
8.60E-08    1.00E+01    1.61E+00    2.86E+00
8.70E-08    1.00E+01    1.57E+00    2.76E+00
8.80E-08    1.00E+01    1.54E+00    2.67E+00
8.90E-08    1.00E+01    1.51E+00    2.58E+00
9.00E-08    1.00E+01    1.47E+00    2.49E+00
9.10E-08    1.00E+01    1.44E+00    2.41E+00
9.20E-08    1.00E+01    1.41E+00    2.33E+00
9.30E-08    1.00E+01    1.38E+00    2.25E+00
9.40E-08    1.00E+01    1.35E+00    2.18E+00
9.50E-08    1.00E+01    1.32E+00    2.10E+00
9.60E-08    1.00E+01    1.29E+00    2.04E+00
9.70E-08    1.00E+01    1.26E+00    1.97E+00
9.80E-08    1.00E+01    1.23E+00    1.90E+00
9.90E-08    1.00E+01    1.20E+00    1.84E+00
1.00E-07    1.00E+01    1.17E+00    1.78E+00
              201   iterations
```

Transient Result - Circuit 2

```
circuit
c1 2 0 5e-13
c2 5 0 5e-13
c3 6 0 1p
vin 1 0 0 sm11
vcl 7 0 0 sm21
vd 4 0 12
vb 3 0 -5
m1 2 1 0 3 mod1 10u 10u 0 0
m2 4 2 2 3 mod2 10u 10u 0 0
m3 6 5 0 3 mod1 10u 10u 0 0
m4 4 6 6 3 mod2 10u 10u 0 0
m5 5 7 2 3 mod1 10u 10u 0 0
pr 5 12
pr 2 12
sm11
0
10n
0
20n
10
60n
10
70n
0
1
0
0
sm21
0
40n
0
50n
10
1
10
0
list
end
model
mod1 n
phi 0.638
vto -0.191
k   1.54e-5
gam 0.985
end
mod2 n
phi 0.638
vto -6.92
k   1.54e-5
gam 0.985
end
list
end
```

```
list
end
op
pr
li
en
tran 100n 1n 0
print tran v(1) v(2) v(6) v(7) v(5)
go
end
```

Circuit 3 Input File

```
                    CIRCUIT ELEMENT

                    *CAPACITOR*
NAME   N1  N2      VALUE       I. C.       MODEL
c1      2   0    5.00E-13
c2      5   0    5.00E-13
c3      6   0    1.00E-12

                    *VOLTAGES SOURCES*
NAME   N+  N-      VALUE     MODEL
vin     1   0    0.00E+00     sm11
vcl     7   0    0.00E+00     sm21
vd      4   0    1.20E+01
vb      3   0   -5.00E+00


                    *MOSFETS*
NAME   ND  NG  NS  NB   MODEL      WC          LC          AD          AS
m1      2   1   0   3   mod1   1.00E-05   1.00E-05   0.00E+00   0.00E+00
m2      4   2   2   3   mod2   1.00E-05   1.00E-05   0.00E+00   0.00E+00
m3      6   5   0   3   mod1   1.00E-05   1.00E-05   0.00E+00   0.00E+00
m4      4   6   6   3   mod2   1.00E-05   1.00E-05   0.00E+00   0.00E+00
m5      5   7   2   3   mod1   1.00E-05   1.00E-05   0.00E+00   0.00E+00

                    *PRESET TRANSIENT VOLTAGES*
NODE    VALUE      NODE    VALUE      NODE    VALUE      NODE    VALUE
  2   1.20E+01      5    1.20E+01      4    1.20E+01      3   -5.00E+00

                    *SOURCE MODEL SPECIFICATIONS*

MODEL:    sm11
                      TIME        VALUE
                    0.00E+00    0.00E+00
                    1.00E-08    0.00E+00
                    2.00E-08    1.00E+01
                    6.00E-08    1.00E+01
                    7.00E-08    0.00E+00
                    1.00E+00    0.00E+00

MODEL:    sm21
                      TIME        VALUE
                    0.00E+00    0.00E+00
                    4.00E-08    0.00E+00
                    5.00E-08    1.00E+01
                    1.00E+00    1.00E+01
```

Circuit Description - Circuit 3

```
                    *DEVICE MODELS*

              MOS MODEL  "mod1"  N-CHANNEL
vto > -1.91E-01     k    >  1.54E-05     gam >  9.85E-01
phi >  6.38E-01     lam  >  0.00E+00     cgs >  0.00E+00
cgd >  0.00E+00     cgb  >  0.00E+00     cdb >  0.00E+00
csb >  0.00E+00     cox  >  0.00E+00     js  >  0.00E+00
pb  >  8.69E-01     nsu  >  0.00E+00     nss >  0.00E+00
uo  >  0.00E+00     ecr  >  0.00E+00     eex >  0.00E+00
etr >  0.00E+00     nfs  >  0.00E+00     ldi >  0.00E+00
xj  >  0.00E+00     pol  >  0.00E+00     bjc >  5.00E-01

              MOS MODEL  "mod2"  N-CHANNEL
vto > -6.92E+00     k    >  1.54E-05     gam >  9.85E-01
phi >  6.38E-01     lam  >  0.00E+00     cgs >  0.00E+00
cgd >  0.00E+00     cgb  >  0.00E+00     cdb >  0.00E+00
csb >  0.00E+00     cox  >  0.00E+00     js  >  0.00E+00
pb  >  8.69E-01     nsu  >  0.00E+00     nss >  0.00E+00
uo  >  0.00E+00     ecr  >  0.00E+00     eex >  0.00E+00
etr >  0.00E+00     nfs  >  0.00E+00     ldi >  0.00E+00
xj  >  0.00E+00     pol  >  0.00E+00     bjc >  5.00E-01
```

MOS Model Descriptions - Circuit 2

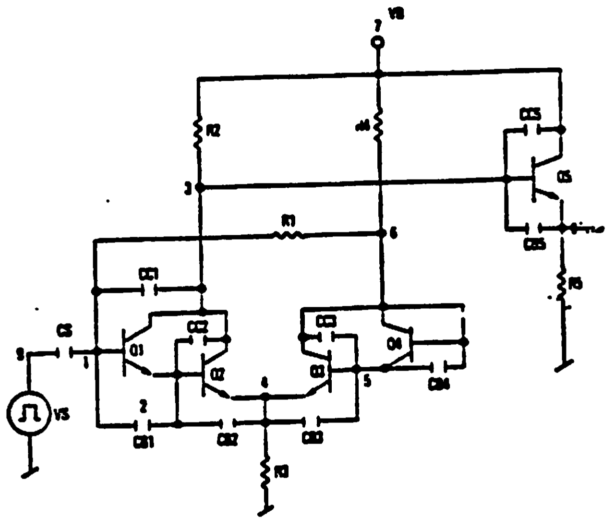
```
       *ANALYSIS OPTIONS*

        TSTEP >  0.00E+00
        MINIT >    3
        MAXIT >    8
        CONVG >  3.00E-04
        DV/DT >  2.00E-01
        DI/DT >  9.00E-01

        SHORTCH  OFF
        CHANLMD  OFF
        SUBTCON  OFF
        SAT/LIN  OFF
        CUT/LIN  OFF
        USE I C  OFF
        PRESET   ON
OPTIONS
```

Options - Circuit 3

```
                          *TRANSIENT ANALYSIS*

TIME(SEC)     V( 1)        V( 2)        V( 6)        V( 7)        V( 5)
1.00E-09    0.00E+00     1.20E+01     1.59E-01     0.00E+00     1.20E+01
2.00E-09    0.00E+00     1.19E+01     3.79E-01     0.00E+00     1.20E+01
3.00E-09    0.00E+00     1.19E+01     4.91E-01     0.00E+00     1.20E+01
4.00E-09    0.00E+00     1.19E+01     5.86E-01     0.00E+00     1.20E+01
5.00E-09    0.00E+00     1.19E+01     6.67E-01     0.00E+00     1.20E+01
6.00E-09    0.00E+00     1.19E+01     7.35E-01     0.00E+00     1.20E+01
7.00E-09    0.00E+00     1.19E+01     7.93E-01     0.00E+00     1.20E+01
8.00E-09    0.00E+00     1.19E+01     8.43E-01     0.00E+00     1.20E+01
9.00E-09    0.00E+00     1.19E+01     8.85E-01     0.00E+00     1.20E+01
1.00E-08    4.44E-16     1.19E+01     9.21E-01     0.00E+00     1.20E+01
1.10E-08    1.00E+00     1.19E+01     9.51E-01     0.00E+00     1.20E+01
1.20E-08    2.00E+00     1.19E+01     9.77E-01     0.00E+00     1.19E+01
1.30E-08    3.00E+00     1.19E+01     1.00E+00     0.00E+00     1.19E+01
1.40E-08    4.00E+00     1.19E+01     1.01E+00     0.00E+00     1.19E+01
1.50E-08    5.00E+00     1.18E+01     1.03E+00     0.00E+00     1.19E+01
1.60E-08    6.00E+00     1.16E+01     1.04E+00     0.00E+00     1.19E+01
1.70E-08    7.00E+00     1.13E+01     1.06E+00     0.00E+00     1.19E+01
1.80E-08    8.00E+00     1.09E+01     1.07E+00     0.00E+00     1.19E+01
1.90E-08    9.00E+00     1.03E+01     1.08E+00     0.00E+00     1.19E+01
2.00E-08    1.00E+01     9.66E+00     1.08E+00     0.00E+00     1.19E+01
2.09E-08    1.00E+01     8.88E+00     1.09E+00     0.00E+00     1.19E+01
2.20E-08    1.00E+01     8.12E+00     1.10E+00     0.00E+00     1.19E+01
2.29E-08    1.00E+01     7.37E+00     1.10E+00     0.00E+00     1.19E+01
2.39E-08    1.00E+01     6.64E+00     1.10E+00     0.00E+00     1.19E+01
2.50E-08    1.00E+01     5.94E+00     1.11E+00     0.00E+00     1.19E+01
2.59E-08    1.00E+01     5.28E+00     1.11E+00     0.00E+00     1.19E+01
2.69E-08    1.00E+01     4.68E+00     1.11E+00     0.00E+00     1.19E+01
2.79E-08    1.00E+01     4.15E+00     1.11E+00     0.00E+00     1.20E+01
2.90E-08    1.00E+01     3.68E+00     1.12E+00     0.00E+00     1.20E+01
2.99E-08    1.00E+01     3.27E+00     1.12E+00     0.00E+00     1.20E+01
3.09E-08    1.00E+01     2.93E+00     1.12E+00     0.00E+00     1.20E+01
3.19E-08    1.00E+01     2.64E+00     1.12E+00     0.00E+00     1.20E+01
3.29E-08    1.00E+01     2.41E+00     1.12E+00     0.00E+00     1.20E+01
3.39E-08    1.00E+01     2.21E+00     1.12E+00     0.00E+00     1.20E+01
3.49E-08    1.00E+01     2.05E+00     1.12E+00     0.00E+00     1.20E+01
3.59E-08    1.00E+01     1.93E+00     1.12E+00     0.00E+00     1.20E+01
3.69E-08    1.00E+01     1.82E+00     1.12E+00     0.00E+00     1.20E+01
3.79E-08    1.00E+01     1.74E+00     1.13E+00     0.00E+00     1.20E+01
3.89E-08    1.00E+01     1.67E+00     1.13E+00     0.00E+00     1.20E+01
3.99E-08    1.00E+01     1.62E+00     1.13E+00     0.00E+00     1.20E+01
4.09E-08    1.00E+01     1.58E+00     1.13E+00     9.99E-01     1.20E+01
4.19E-08    1.00E+01     1.54E+00     1.13E+00     1.99E+00     1.20E+01
4.29E-08    1.00E+01     1.52E+00     1.13E+00     2.99E+00     1.20E+01
4.39E-08    1.00E+01     1.50E+00     1.13E+00     3.99E+00     1.19E+01
4.49E-08    1.00E+01     1.51E+00     1.13E+00     4.99E+00     1.19E+01
4.59E-08    1.00E+01     1.55E+00     1.13E+00     5.99E+00     1.18E+01
4.69E-08    1.00E+01     1.65E+00     1.13E+00     6.99E+00     1.17E+01
4.79E-08    1.00E+01     1.80E+00     1.14E+00     7.99E+00     1.15E+01
4.89E-08    1.00E+01     2.01E+00     1.15E+00     8.99E+00     1.12E+01
4.99E-08    1.00E+01     2.26E+00     1.16E+00     9.99E+00     1.08E+01
```

```
5.09E-08   1.00E+01   2.50E+00   1.18E+00   1.00E+01   1.03E+01
5.19E-08   1.00E+01   2.66E+00   1.20E+00   1.00E+01   9.94E+00
5.29E-08   1.00E+01   2.78E+00   1.23E+00   1.00E+01   9.55E+00
5.39E-08   1.00E+01   2.86E+00   1.26E+00   1.00E+01   9.18E+00
5.49E-08   1.00E+01   2.91E+00   1.30E+00   1.00E+01   8.82E+00
5.59E-08   1.00E+01   2.95E+00   1.34E+00   1.00E+01   8.46E+00
5.69E-08   1.00E+01   2.98E+00   1.38E+00   1.00E+01   8.12E+00
5.79E-08   1.00E+01   3.00E+00   1.43E+00   1.00E+01   7.77E+00
5.89E-08   1.00E+01   3.02E+00   1.48E+00   1.00E+01   7.43E+00
5.99E-08   1.00E+01   3.03E+00   1.53E+00   1.00E+01   7.09E+00
6.09E-08   9.00E+00   3.07E+00   1.59E+00   1.00E+01   6.76E+00
6.19E-08   8.00E+00   3.18E+00   1.65E+00   1.00E+01   6.45E+00
6.29E-08   7.00E+00   3.35E+00   1.72E+00   1.00E+01   6.17E+00
6.39E-08   6.00E+00   3.56E+00   1.78E+00   1.00E+01   5.92E+00
6.50E-08   5.00E+00   3.83E+00   1.86E+00   1.00E+01   5.72E+00
6.60E-08   4.00E+00   4.14E+00   1.93E+00   1.00E+01   5.57E+00
6.70E-08   3.00E+00   4.47E+00   2.00E+00   1.00E+01   5.46E+00
6.79E-08   2.00E+00   4.80E+00   2.08E+00   1.00E+01   5.40E+00
6.90E-08   9.99E-01   5.10E+00   2.15E+00   1.00E+01   5.36E+00
7.00E-08   0.00E+00   5.38E+00   2.22E+00   1.00E+01   5.35E+00
7.10E-08   0.00E+00   5.63E+00   2.29E+00   1.00E+01   5.36E+00
7.20E-08   0.00E+00   5.87E+00   2.36E+00   1.00E+01   5.39E+00
7.30E-08   0.00E+00   6.09E+00   2.42E+00   1.00E+01   5.42E+00
7.40E-08   0.00E+00   6.30E+00   2.48E+00   1.00E+01   5.46E+00
7.50E-08   0.00E+00   6.50E+00   2.54E+00   1.00E+01   5.51E+00
7.60E-08   0.00E+00   6.70E+00   2.59E+00   1.00E+01   5.56E+00
7.70E-08   0.00E+00   6.89E+00   2.64E+00   1.00E+01   5.61E+00
7.80E-08   0.00E+00   7.08E+00   2.69E+00   1.00E+01   5.66E+00
7.90E-08   0.00E+00   7.27E+00   2.73E+00   1.00E+01   5.72E+00
8.00E-08   0.00E+00   7.45E+00   2.77E+00   1.00E+01   5.76E+00
8.10E-08   0.00E+00   7.64E+00   2.81E+00   1.00E+01   5.81E+00
8.20E-08   0.00E+00   7.82E+00   2.84E+00   1.00E+01   5.85E+00
8.30E-08   0.00E+00   8.01E+00   2.88E+00   1.00E+01   5.90E+00
8.40E-08   0.00E+00   8.19E+00   2.91E+00   1.00E+01   5.94E+00
8.50E-08   0.00E+00   8.38E+00   2.93E+00   1.00E+01   5.97E+00
8.60E-08   0.00E+00   8.56E+00   2.96E+00   1.00E+01   6.01E+00
8.70E-08   0.00E+00   8.75E+00   2.98E+00   1.00E+01   6.04E+00
8.80E-08   0.00E+00   8.93E+00   3.00E+00   1.00E+01   6.07E+00
8.90E-08   0.00E+00   9.10E+00   3.02E+00   1.00E+01   6.11E+00
9.00E-08   0.00E+00   9.27E+00   3.03E+00   1.00E+01   6.14E+00
9.10E-08   0.00E+00   9.44E+00   3.05E+00   1.00E+01   6.16E+00
9.20E-08   0.00E+00   9.60E+00   3.06E+00   1.00E+01   6.19E+00
9.30E-08   0.00E+00   9.76E+00   3.07E+00   1.00E+01   6.22E+00
9.40E-08   0.00E+00   9.91E+00   3.08E+00   1.00E+01   6.24E+00
9.50E-08   0.00E+00   1.00E+01   3.09E+00   1.00E+01   6.27E+00
9.60E-08   0.00E+00   1.01E+01   3.09E+00   1.00E+01   6.29E+00
9.70E-08   0.00E+00   1.03E+01   3.10E+00   1.00E+01   6.31E+00
9.80E-08   0.00E+00   1.04E+01   3.10E+00   1.00E+01   6.33E+00
9.90E-08   0.00E+00   1.05E+01   3.11E+00   1.00E+01   6.35E+00
1.00E-07   0.00E+00   1.06E+01   3.11E+00   1.00E+01   6.37E+00
                   203  iterations
```

Transient Result - Circuit 3



INTEGRATED PREAMPLIFIER

FIGURE 2

```
BIAS-D . JUNE 1982
-TEST CIRCUIT 10 (9 NODES)
   INTEGRATED PREAMPLIFIER **
* RESISTORS
R1 6 1 12K
R2 7 3 7.5K
R3 4 0 680
R4 7 6 9K
R5 8 0 5K
* TRANSISTORS
Q1 3 1 2 M2
Q2 3 2 4 M2
Q3 6 5 4 M2
Q4 6 6 5 M2
Q5 7 3 8 M2
* VOLTAGE SOURCES
V+ 7 0 6.1
VS 9 0 1.0 M1
CS 9 1 1U
* JUNCTION CAPACITORS
CB1 1 2 2P
CB2 2 4 2P
CB5 5 4 2P
CB3 3 8 2P
CC1 3 1 2P
CC2 3 2 2P
CC3 6 5 2P
(    7 3 2P
* MODELS
M1 PUL 0 -1 .5U .5U 5U .5U
M2 NPN 100 1 5E-15
END

 RESISTORS:
 NAME     Nodes        VALUE       MODEL
 R1     6       1      1.200E+04    M 0
 R2     7       3      7.500E+03    M 0
 R3     4       0      6.800E+02    M 0
 R4     7       6      9.000E+03    M 0
 R5     8       0      5.000E+03    M 0

 CAPACITORS:
 NAME     Nodes        VALUE       MODEL
 CS     9       1      1.000E-06    M 0
 CB     1       2      2.000E-12    M 0
 CB     2       4      2.000E-12    M 0
 CB     5       4      2.000E-12    M 0
 CB     3       8      2.000E-12    M 0
 CC     3       1      2.000E-12    M 0
 CC     3       2      2.000E-12    M 0
 CC     6       5      2.000E-12    M 0
 CC     7       3      2.000E-12    M 0

  .ANSISTORS:
 NAME     C       B       E      MODEL
 Q1       3       1       2      M 2
 Q2       3       2       4      M 2
 Q3       6       5       4      M 2
 Q4      6      6      5     M 2
 Q5      7      3      8     M 2

 VOLTAGE SOURCES:
 NAME   + Nodes-     VALUE       MODEL
 V+      7      0    6.100E+00    M 0
 VS      9      0    1.000E+00    M 1

 MODELS:
 NAME    TYPE
 M1      PUL     0    -1   5.000E-07  5.000E-07  5.000E-06  5.000E-07  0.000E+00
 M2      NPN   100     1   5.000E-15  1.000E+12  0.000E+00  0.000E+00  0.000E+00

NODES:    9

**** END OF INPUT DATA ****

   NNODE = 7                IU= 11
    NOPS = 63       FILL-INS= 2
SPARSITY = 55.1020408163 %    MXPOS= 284     MXYPOS= 33


     S=3.957E+01
     S=2.477E+00
     S=1.782E-01
     S=7.010E-03
     S=2.488E-03
     S=1.746E-03
     S=1.685E-03
     S=1.586E-03
     S=9.312E-04
     S=1.366E-04
     S=1.263E-06
     S=7.692E-11
     S=1.473E-18
     S=6.626E-26

ITERATIONS: 14


 T=       27DEG C


 NODE VOLTAGES:
 V 1          1.8283
 V 2          1.2948
 V 3          2.5509
 V 4           .6419
 V 5          1.2950
 V 6          1.8289
 V 8          1.9035
 V 7          6.1000
 V 9          1.0000


 TRANSISTOR OPERATING POINTS:
 NAME       IB          IC             VBE        VBC       BETA        GM          RPI
 Q1     4.602E-08   4.602E-06       .5335     -.7226     100.00    1.780E-04   5.617E+05
 Q2     4.648E-06   4.648E-04       .6528    -1.2562     100.00    1.798E-02   5.561E+03
 Q3     4.698E-06   4.698E-04       .6531     -.5338     100.00    1.818E-02   5.502E+03
 Q4     4.652E-08  4.652E-06      .5338    0.0000    100.00   1.800E-04  5.557E+05
 Q5     3.769E-06  3.769E-04      .6474   -3.5491    100.00   1.458E-02  6.858E+03
.TR
TR TO TSTOP TSTEP LINE 1950
 UTPUTS:
 VXX PRT/PLO XMIN XMAX VMIN VMAX LINE7270
               T                 V 8
          +++++++++++++++++++++++++++++
           0.000E+00         1.904E+00                                   3
           1.000E-07         1.904E+00                                   3
           2.000E-07         1.904E+00                                   3
           3.000E-07         1.904E+00                                   3
           4.000E-07         1.904E+00                                   3
           5.000E-07         1.904E+00                                   3
           6.000E-07         2.103E+00                                   6
           7.000E-07         2.438E+00                                   5
           8.000E-07         2.720E+00                                   5
           9.000E-07         3.014E+00                                   5
           1.000E-06         3.269E+00                                   5
           1.100E-06         3.416E+00                                   4
           1.200E-06         3.490E+00                                   5
           1.300E-06         3.612E+00                                   4
           1.400E-06         3.686E+00                                   5
           1.500E-06         3.788E+00                                   4
           1.600E-06         3.860E+00                                   5
           1.700E-06         3.947E+00                                   4
           1.800E-06         4.014E+00                                   5
           1.900E-06         4.089E+00                                   4
           2.000E-06         4.152E+00                                   5
           2.100E-06         4.217E+00                                   5
           2.200E-06         4.274E+00                                   5
           2.300E-06         4.331E+00                                   5
           2.400E-06         4.383E+00                                   5
           2.500E-06         4.433E+00                                   5
           2.600E-06         4.480E+00                                   5
           2.700E-06         4.525E+00                                   5
           2.800E-06         4.566E+00                                   5
           2.900E-06         4.607E+00                                   5
           3.000E-06         4.644E+00                                   5
           3.100E-06         4.680E+00                                   5
           3.200E-06         4.713E+00                                   4
           3.300E-06         4.745E+00                                   5
           3.400E-06         4.775E+00                                   4
           3.500E-06         4.803E+00                                   5
           3.600E-06         4.830E+00                                   4
           3.700E-06         4.855E+00                                   5
           3.800E-06         4.879E+00                                   4
           3.900E-06         4.902E+00                                   5
           4.000E-06         4.923E+00                                   4
           4.100E-06         4.944E+00                                   5
           4.200E-06         4.963E+00                                   4
           4.300E-06         4.981E+00                                   5
           4.400E-06         4.998E+00                                   4
           4.500E-06         5.014E+00                                   5
           4.600E-06         5.030E+00                                   4
           4.700E-06         5.044E+00                                   5
           4.800E-06         5.058E+00                                   4
           4.900E-06         5.071E+00                                   5
           5.000E-06         5.083E+00                                   4
           5.100E-06         5.095E+00                                   5
```

```
               5.200E-06              5.106E+00                                        4
               5.300E-06              5.116E+00                                        5
               5.400E-06              5.126E+00                                        4
               5.500E-06              5.136E+00                                        5
               5.600E-06              5.145E+00                                        4
               5.700E-06              5.153E+00                                        5
               5.800E-06              5.161E+00                                        4
               5.900E-06              5.169E+00                                        4
               6.000E-06              5.176E+00                                        4
               6.100E-06              5.111E+00                                        6
               6.200E-06              4.975E+00                                        5
               6.300E-06              4.877E+00                                        5
               6.400E-06              4.473E+00                                        8
               6.500E-06              3.179E+00                                        7
               6.600E-06              2.090E+00                                        7
               6.700E-06              1.989E+00                                        5
               6.800E-06              1.942E+00                                        4
               6.900E-06              1.924E+00                                        5
               7.000E-06              1.910E+00                                        5
               7.100E-06              1.907E+00                                        5
               7.200E-06              1.902E+00                                        5
               7.300E-06              1.903E+00                                        5
               7.400E-06              1.901E+00                                        4
               7.500E-06              1.901E+00                                        4
               7.600E-06              1.900E+00                                        4
               7.700E-06              1.901E+00                                        4
               7.800E-06              1.900E+00                                        4
               7.900E-06              1.901E+00                                        4
               8.000E-06              1.900E+00                                        4
               8.100E-06              1.900E+00                                        4
               8.200E-06              1.900E+00                                        4
               8.300E-06              1.900E+00                                        4
               8.400E-06              1.900E+00                                        4
               8.500E-06              1.900E+00                                        4
               8.600E-06              1.900E+00                                        4
               8.700E-06              1.900E+00                                        4
               8.800E-06              1.900E+00                                        4
               8.900E-06              1.900E+00                                        4
               9.000E-06              1.900E+00                                        4
               9.100E-06              1.900E+00                                        4
               9.200E-06              1.900E+00                                        4
               9.300E-06              1.900E+00                                        4
               9.400E-06              1.900E+00                                        4
               9.500E-06              1.900E+00                                        4
               9.600E-06              1.900E+00                                        4
               9.700E-06              1.900E+00                                        4
               9.800E-06              1.900E+00                                        4
               9.900E-06              1.900E+00                                        4
TOTAL ITERATIONS= 506
V8 PRT
END
```

APPENDIX G - PROGRAM LISTINGS, BIASB.36, BIASP.36 AND BIASD.36

BIASB.36 LISTING

```
5      ! ---------------------------------------------------------------------
10     !                               BIASB.26 PROGRAM
11     !
20     ! MINICOMPUTER AIDED CIRCUIT SIMULATOR
40     ! BIASB.26 IS WRITTEN BY DANNY Y. CHEUNG
50     ! FOR THE HEWLETT-PACKARD 9826/36 SERIES DESKTOP COMPUTER
51     ! USER INTERACTIVE
52     ! SIMULATE DC, DC TRANSFER AND TRANSCIENT ANALYSIS OF MOS CIRCUIT ONLY
53     ! SPARSE MATRIX ROUTINE -- MARKOWITZ REORDERING ALGORITHM
54     ! SPARSE MATRIX STORAGE -- BI-DIRECTIONAL THREADED LIST
60     ! DATE : NOV 8TH, 1982      VERSION #8  (REDIMENSION ARRAYS)
70     !----------------------------------------------------------------------
80     !
90     !
100    ! BIASB.26 INITIALIZATION
110      R_(1)=80 ! # OF EQUATIONS
120      R_(2)=60 ! # OF MOSFETS
130      R_(3)=10 ! # OF RESISTORS
140      R_(4)=30 ! # OF CAPACITORS
150      R_(5)=10 ! # OF VOLTAGE SOURCES
160      R_(6)=4  ! # OF CURRENT SOURCES
170      R_(7)=5  ! # OF DIODES
180      R_(8)=5  ! # OF MOS MODELS
190      R_(9)=1  ! # OF DIODE MODELS
200      R_(10)=5 ! # OF SOURCE MODELS
210      R_(31)=2 ! # OF CAPACITOR MODELS
220      COM /Blk1/ Vb$(10)[4],Mb$(60)[4],Ib$(4)[4],Eb$(1)[4]
230      COM /Blk2/ Fb$(3)[4],Sb$(5)[4],Lb$(2)[4]
240      COM /Blk3/ Ra,C,M,D,I,V,F,S,L,E
250      COM /Blk4/ J(1:100),H(100),G_(12)
260      COM /Blk5/ Me(1:60,1:32),Ce(1:40,1:8),Rb$(10)[4],Cb$(40)[4]
270      COM /Blk6/ De(5,7),Ee(1,2),Re(10,5),Se(5,23),Ve(10,3)
280      COM Ipt(350),Jpt(350),Row(350),Col(350),Rhs(350),Buf(350),Inzero(350
ero(150)
290      COM /Blk7/ N,Loc_ptr,Min_index,Newrow(150),Newcol(150),R_(60)
300      COM /Blk8/ Del_elem(3,100),Newtemp(150)
301      COM /Blk9/ Flag
310      DIM A_(1:25,1:25),B(1:100),O(1:100),P(1:100),Q_(1:100),N_(1:100)
320      DIM Va_(10),Ia(4),Db$(5)[4],O$(40)[70],A$[70],B$[80],N$[30]
330      DIM H$[40],Fl(16),Fi$(1:10)[4],Le(2,3),Ie(4,3),Fe(3,27)
340      DIM C_(1:40,1:2),M_(1:60,1:15)
341      DIM D_(1:6),I_flag(2,2),Node(2,2)
342      DIM Dt$(1:2)[20]
350      R_(19)=2.585E-2
360      R_(28)=2.585E-2
370      R_(20)=1.45E+10
380      R_(30)=1.45E+10
390      R_(21)=8.854E-14
400      R_(23)=R_(21)*3.9
410    R_(22)=R_(21)*11.7
420    R_(18)=1.12
430    R_(24)=1.602E-19
440    R_(14)=8
```

BIASB.36 LISTING

```
450   R_(15)=3
460   R_(16)=.0003
470   R_(25)=.0002
480   R_(17)=.2
490   W=.9
500   Q=27
503   !
504   !------------------MAIN PROGRAM-------------------------
506   Flag=0
510 Main:Fl(12)=0
520 Linkm:    IF Fl(12)<>0 THEN GOTO Work
530   B$="CIMOOPPCNEDCTRPRKILOTEGOCRCASADT"
540   R_(32)=1
550   IF Fl(0)<>0 THEN CALL Proc(A$,Pn,O$(*))
560   IF Fl(0)<>0 THEN 590
571   INPUT "BIASB.26",A$
580   PRINT A$
590   CALL Dcod(R_(34),R_(32),R_(33),0,0,0,A$,B$,2)
600   IF (R_(33)<0) AND (R_(34)<>0) THEN GOTO Goti
610   CALL Errr(R_(32),1,A$)
620   GOTO Linkm
630 Goti:    R_(34)=(R_(34)+1)/2
640   IF R_(34)=1 THEN GOTO Circ
650   IF R_(34)=2 THEN GOTO Moded
660   IF R_(34)=3 THEN GOTO Option
670   IF R_(34)=4 THEN GOTO Proc
680   IF R_(34)=5 THEN GOSUB Newc
690   IF R_(34)=6 THEN GOSUB Dcop
700   IF R_(34)=7 THEN CALL Tran(R_(*),A$,B$,G,Fl(*),O(*),J(*),T,N,A)
710   IF R_(34)=8 THEN CALL Prin(R_(*),A,N,H(*),O(*),M,Mb$(*),V,Vb$(*),A$,G
Me(*),M_(*))
720   IF R_(34)=9 THEN CALL Kill(A$,R_(*),Fi$(*))
730   IF R_(34)=10 THEN CALL Load(A$,Fi$(*),Va_(*),Ia(*),M_(*),Db$(*),Ie(*)
),Le(*),N$)
740  IF R_(34)=11 THEN CALL Temp(R_(*),A$,B$,Q,F,Fb$(*),Fe(*))
750   IF R_(34)=12 THEN CALL Runc(A,N,V,A_(*),B(*),O(*),P(*),Q_(*),N_(*),Fl
,U,R_(*))
760   IF R_(34)=13 THEN CALL Create(A$,R_(*),Fi$(*))
770   IF R_(34)=14 THEN CALL Catf(Fi$(*))
780   IF R_(34)=15 THEN CALL Save(A$,Fi$(*),Va_(*),Ia(*),M_(*),Db$(*),Ie(*)
),Le(*),N$)
781   IF R_(34)=16 THEN CALL Dtran(R_(*),A$,B$,Dt$(*),D_(*),N,A,Fl(*),Ve(*)
790   GOTO Linkm
791   !
792   !------------ ------------CIRCUIT EDITOR-------------------------
800 Circ: Flag=0
810 Linkc:    B$="RCMDVISELNKP"
820  R_(32)=1
830  IF Fl(0)<>0 THEN CALL Proc(A$,Pn,O$(*))
840  IF Fl(0)<>0 THEN 870
850  INPUT "CIRCUIT",A$
860  PRINT A$
870  CALL Dcod(R_(34),R_(32),R_(33),0,0,0,A$,B$,1)
880  IF (R_(33)<0) AND (R_(34)<>0) THEN GOTO Gotc
890  CALL Errr(R_(32),6,A$)
900  GOTO Linkc
```

BIASB.36 LISTING

```
910 Gotc:      IF R_(34)=1 THEN CALL Resrc(R_(*),Rb$(*),Ra,N$,A$,B$,H(*),N,F
920   IF R_(34)=2 THEN CALL Caprc(R_(*),Cb$(*),C,Lb$(*),L,N$,A$,B$,H(*),K,M
),Le(*)
930   IF R_(34)=3 THEN CALL Mosfc(R_(*),Mb$(*),M,Fb$(*),F,N$,A$,B$,H(*),K,M
),Fe(*))
940   IF R_(34)=4 THEN CALL Diodc(R_(*),Db$(*),D,Eb$(*),E,N$,A$,B$,H(*),N,I
950   IF R_(34)=5 THEN CALL Voltc(R_(*),Vb$(*),Va_(*),V,Sb$(*),S,N$,A$,B$,T
960   IF R_(34)=6 THEN CALL Curnc(R_(*),Ib$(*),Ia(*),I,Sb$(*),S,N$,A$,B$,H(
970  IF R_(34)=7 THEN CALL Smodc(R_(*),S,Sb$(*),N$,A$,B$,Se(*))
980  IF R_(34)=8 THEN CALL Veryc(H(*),Db$(*),Ie(*))
990  IF R_(34)=8 THEN GOTO Main
1000 IF R_(34)=9 THEN CALL Liscc(A$,B$,Va_(*),Ia(*),Db$(*),Le(*),Ie(*),Fe(*
1010 IF R_(34)<>10 THEN 1130
1020 CALL Matzv(25,H(*))
1030 H(0)=0
1040 CALL Matzv(25,J(*))
1050 Ra=0
1060 M=0
1070 C=0
1080 D=0
1090 I=0
1100 N=0
1110 S=0
1120 V=0
1130 IF R_(34)=11 THEN CALL Kilec(A$,B$,N$,Db$(*),Va_(*),Ia(*),K,Ie(*))
1140 IF R_(34)=12 THEN CALL Presc(R_(*),J(*),N,H(*),A$,B$)
1150 GOTO Linkc
1151 !
1152 !-----------------MODELS EDITOR--------------------------------
1160 Moded:   !
1170 Linko: B$="MDCLEN"
1180 R_(32)=1
1190 IF F1(0)<>0 THEN CALL Proc(A$,Pn,O$(*))
1200 IF F1(0)<>0 THEN 1230
1210 INPUT "MODELS",A$
1220 PRINT A$
1230 CALL Dcod(R_(34),R_(32),R_(33),0,0,0,A$,B$,1)
1240 IF (R_(33)<0) AND (R_(34)<>0) THEN 1270
1250 CALL Errr(R_(32),13,A$)
1260 GOTO Linko
1270 IF R_(34)=1 THEN CALL Mmod(A$,B$,N$,F,Fb$(*),R_(*),Q,Fe(*),O$(*))
1280 IF R_(34)=2 THEN CALL Dmod(A$,N$,Q,E,Eb$(*),R_(*),Ee(*))
1290 IF R_(34)=3 THEN CALL Cmod(A$,N$,L,Lb$(*),R_(*),Le(*))
1300 IF R_(34)=4 THEN CALL Lism(F,Fb$(*),E,Eb$(*),L,Lb$(*),B$,Fe(*),Ee(*),L
1310 IF R_(34)=5 THEN GOTO Main
1320 IF R_(34)=6 THEN F=0
1330 IF R_(34)=6 THEN E=0
1340 IF R_(34)=6 THEN L=0
1350 GOTO Linko
1351 !
1352 !--------------- --OPTION EDITOR--------------------------------------
1360 Option:   !
1370 Link: B$="TSMIMACODVDISHCHSUSACUUILIENPR"
1380 R_(32)=1
1390 IF F1(0)<>0 THEN CALL Proc(A$,Pn,O$(*))
1400 IF F1(0)<>0 THEN 1430
```

BIASB.36 LISTING

```
1410 INPUT "OPTIONS",A$
1420 PRINT A$
1430 CALL Dcod(R_(34),R_(32),R_(33),0,0,0,A$,B$,2)
1440 IF (R_(33)<0) AND (R_(34)>0) THEN 1470
1450 CALL Errr(R_(32),10,A$)
1460 GOTO Link
1470 R_(34)=1+(R_(34)-1)/2
1480 IF R_(34)=1 THEN CALL Dcod(R_(13),R_(32),R_(33),0,0,0,A$,B$,0)
1490 IF R_(34)=2 THEN CALL Dcod(R_(15),R_(32),R_(33),0,0,0,A$,B$,0)
1500 IF R_(34)=3 THEN CALL Dcod(R_(14),R_(32),R_(33),0,0,0,A$,B$,0)
1510 IF R_(34)=4 THEN CALL Dcod(R_(16),R_(32),R_(33),0,0,0,A$,B$,0)
1520 IF R_(34)=5 THEN CALL Dcod(R_(17),R_(32),R_(33),0,0,0,A$,B$,0)
1530 IF R_(34)=6 THEN CALL Dcod(W,R_(32),R_(33),0,0,0,A$,B$,0)
1540 IF R_(34)=7 THEN F1(1)=ABS(F1(1)-1)
1550 IF R_(34)=8 THEN F1(2)=ABS(F1(2)-1)
1560 IF R_(34)=9 THEN F1(3)=ABS(F1(3)-1)
1570 IF R_(34)=10 THEN F1(8)=ABS(F1(8)-1)
1580 IF R_(34)=11 THEN F1(6)=ABS(F1(6)-1)
1590 IF R_(34)=12 THEN F1(5)=ABS(F1(5)-1)
1600 IF R_(34)=13 THEN GOTO Liop
1610 IF R_(34)=14 THEN GOTO Main
1620 IF R_(34)=15 THEN F1(16)=ABS(F1(16)-1)
1630 GOTO Link
1631 ! Option Listing
1632 ! --------------
1640 Liop: IMAGE 30X,20A
1650 IMAGE 31X,7A,X,D.DDE
1660 IMAGE 31X,7A,X,3D
1670 IMAGE 31X,7A,X,3A
1680 PRINT FNLin$(2)
1690 PRINT USING 1640;"*ANALYSIS OPTIONS*"
1700 PRINT
1710 PRINT USING 1650;"Tstep >",R_(13)
1720 PRINT USING 1660;"Minit >",R_(15)
1730 PRINT USING 1660;"Maxit >",R_(14)
1740 PRINT USING 1650;"Convg >",R_(16)
1750 PRINT USING 1650;"dV/dt >",R_(17)
1760 PRINT USING 1650;"dI/dt >",W
1770 PRINT
1780  N$="Off"
1790  IF F1(1)<>0 THEN N$="On"
1800  PRINT USING 1670;"Shortch",N$
1810  N$="Off"
1820  IF F1(2)<>0 THEN N$="On"
1830  PRINT USING 1670;"Chanlmd",N$
1840  N$="Off"
1850  IF F1(3)<>0 THEN N$="On"
1860  PRINT USING 1670;"Subtcon",N$
1870  N$="Off"
1880  IF F1(8)<>0 THEN N$="On"
1890  PRINT USING 1670;"Sat/Lin",N$
1900  N$="Off"
1910  IF F1(6)<>0 THEN N$="On"
1920  PRINT USING 1670;"Cut/Lin",N$
1930  N$="Off"
1940  IF F1(5)<>0 THEN N$="On"
```

BIASB.36 LISTING

```
1950  PRINT USING 1670;"Use I C",N$
1960  N$="Off"
1970  IF F1(16)<>0 THEN N$="On"
1980  PRINT USING 1670;"Preset ",N$
1990  PRINT
2000  GOTO Link
2001  !
2002  !-----------------PROCEDURE EDITOR---------------------------
2010 Proc: !
2020 Linkp:   B$="INDELIKIWRUSENSAGE"
2030   R_(32)=1
2040   INPUT "PROCEDURE",A$
2050   PRINT A$
2060   CALL Dcod(R_(34),R_(32),R_(33),0,0,0,A$,B$,2)
2070   IF (R_(33)<0) AND (R_(34)<>0) THEN GOTO Gotp
2080   CALL Errr(R_(32),1,A$)
2090   GOTO Linkp
2100 Gotp:   R_(34)=(R_(34)+1)/2
2110   IF R_(34)=1 THEN CALL Insert(R_(*),A$,O$(*))
2120  IF R_(34)=2 THEN CALL Delete(R_(*),A$,O$(*))
2130  IF R_(34)=3 THEN CALL Listp(O$(*))
2140  IF R_(34)=4 THEN CALL Killp(O$(*))
2150  IF R_(34)=5 THEN CALL Write(O$(*))
2160  IF R_(34)=6 THEN F1(0)=ABS(F1(0)-1)
2170  IF R_(34)=7 THEN GOTO Main
2180  IF R_(34)=8 THEN CALL Savep(O$(*))
2190  IF R_(34)=9 THEN CALL Getp(O$(*))
2200  GOTO Linkp
2201  !
2202  !-----------------WORK SECTION-------------------------------
2210 Work: !
2220 A=N+V
2230 Loc_ptr=A+1
2240 T=0
2250 Anal: IF F1(4)<>0 THEN GOTO Tran
2260 ! D.C. ANALYSIS
2261 ! ------------
2262 ! CALCULATE DC OPERATING POINT
2270 R_(26)=0
2280 R_(29)=0
2290 Dcan: CALL Rstr(Ra,Rb$(*),A_(*),B(*),Q,Re(*))
2300 CALL Vsrc(V,Vb$(*),Va_(*),Sb$(*),A_(*),B(*),F1(*),T,U,Ve(*),Se(*))
2310 CALL Isrc(I,Ib$(*),Ia(*),Sb$(*),A_(*),B(*),F1(*),T,U,Ie(*),Se(*))
2320 CALL Diod(D,Db$(*),Eb$(*),A_(*),B(*),O(*),F1(*),W,De(*),Ee(*))
2330 CALL Mosf(M,Mb$(*),Fb$(*),A_(*),B(*),O(*),T,U,F1(*),W,Q_(*),N_(*),M_(*
2340 CALL Mate(O(*),P(*),A)
2350 GOSUB Solv
2360 F1(7)=0
2370 GOSUB Conv
2380 IF (F1(7)<>0) AND (R_(26)>2) THEN GOTO Relo
2390 ON KEY 5 GOTO Leave!
2400 R_(26)=R_(26)+1
2410 R_(29)=R_(26)
2420 GOTO Dcan
2421 ! TRANSCIENT ANALYSIS
2422 ! -----------------
```

BIASB.36 LISTING

```
2430 Tran:  IF T=0 THEN R_(29)=0
2431 IF F1(4)=2 THEN GOTO Dctran      ! GO TO DC TRANSFER ANALYSIS
2440 IF T=0 THEN T=R_(12)
2450 IF R_(29)=0 AND Flag=0 THEN CALL Initial_buf(A)
2460 U=R_(13)
2461 R_(27)=U
2470 R_(26)=0
2480 Ltop:  CALL Rstr(Ra,Rb$(*),A_(*),B(*),Q,Re(*))
2490 CALL Vsrc(V,Vb$(*),Va_(*),Sb$(*),A_(*),B(*),F1(*),T,U,Ve(*),Se(*))
2500 CALL Isrc(I,Ib$(*),Ia(*),Sb$(*),A_(*),B(*),F1(*),T,U,Ie(*),Se(*))
2510 CALL Capr(C,Cb$(*),Lb$(*),A_(*),B(*),F1(*),Q_(*),N_(*),T,U,C_(*),Ce(*)
2520 CALL Diod(D,Db$(*),Eb$(*),A_(*),B(*),O(*),F1(*),W,De(*),Ee(*))
2530 CALL Mosf(M,Mb$(*),Fb$(*),A_(*),B(*),O(*),T,U,F1(*),W,Q_(*),N_(*),M_(*
2540 CALL Mate(O(*),P(*),A)
2550 GOSUB Solv
2560 F1(7)=0
2570 GOSUB Conv
2580 IF (F1(7)<>0) AND (R_(26)>1) OR (R_(26)>R_(14)) THEN GOTO Yes
2590 R_(26)=R_(26)+1
2600 GOTO Ltop
2610 Yes:  F1(5)=0
2620  F1(16)=0
2630 IF R_(26)>R_(14) THEN GOTO Cuts
2640 T=T+U
2650 R_(27)=U
2660 CALL Mate(Q_(*),N_(*),A)
2670 CALL Mate(P(*),Q_(*),A)
2680 IF R_(26)<R_(15) THEN CALL Min2(R_(13),2*U,U)
2690 R_(29)=R_(29)+R_(26)
2700 IF (F1(8)<>0) OR (F1(6)<>0) THEN CALL Chkf(M,Mb$(*),Fb$(*),F1(*),T,U,M
2710 GOSUB Outa
2720 ON KEY 5 GOTO Leave! THIS TO BE REMOVED
2730 IF T>R_(11) THEN GOTO Relo
2740 R_(26)=0
2750 GOTO Ltop
2760 Leave:  BEEP
2770 R_(12)=T-U
2780 PRINT "**INTERUPT**"
2790 F1(0)=0
2800 GOTO Relo
2810 Cuts:  CALL Max2(R_(13)/80,U/8,U)
2820 R_(29)=R_(26)+R_(29)
2830 R_(26)=0
2840 IF R_(29)<1500 THEN GOTO Ltop
2850 BEEP
2860 PRINT "THAT'S ALL SHE WROTE"
2870 STOP
2871 ! DC TRANSFER ANALYSIS
2872 ! ------------------
2874 Dctran:R_(26)=0
2875 R_(29)=0
2876 R_(52)=D_(1)
2877 R_(53)=D_(4)
2878 Repeat:  IF Converge<>0 THEN
2879           Converge=0
2880           GOTO 2887
```

```
2881          END IF
2882          IF R_(29)<>0 AND R_(26)=0 THEN
2883          FOR Pp=1 TO N
2884       !  O(Pp)=Q_(Pp)+(P(Pp)-Q_(Pp))*2
2885          NEXT Pp
2886          END IF
2887 CALL Rstr(Ra,Rb$(*),A_(*),B(*),Q,Re(*))
2888 CALL Vsrc(V,Vb$(*),Va_(*),Sb$(*),A_(*),B(*),Fl(*),T,U,Ve(*),Se(*))
2889 CALL Isrc(I,Ib$(*),Ia(*),Sb$(*),A_(*),B(*),Fl(*),T,U,Ie(*),Se(*))
2890 CALL Diod(D,Db$(*),Eb$(*),A_(*),B(*),O(*),Fl(*),W,De(*),Ee(*))
2891  CALL Mosf(M,Mb$(*),Fb$(*),A_(*),B(*),O(*),T,U,Fl(*),W,Q_(*),N_(*),M_(
2892  IF R_(26)=0 AND R_(29)<>0 THEN GOTO 2894
2893  CALL Mate(O(*),P(*),A)
2894  CALL Re_adjust(Node(*),Rhs(*),Va_(*),Ia(*),I_flag(*),N,D_(*),R_(*))
2895  GOSUB Solv
2896  Fl(7)=0
2897  GOSUB Conv
2902  IF (R_(26)>R_(14)) AND Zflag=0 THEN GOTO Zero
2904  IF R_(26)>(3*R_(14)) THEN
2905  PRINT
2907         PRINT "***** THE FOLLOWING PARTICULAR POINT DOES NOT CONVERGE *
2908         Converge=1
2909         Matza(15,M_(*))
2911         GOTO Yeah
2912         END IF
2913  IF (Fl(7)<>0) AND (R_(26)>1) THEN GOTO Yeah
2914  R_(26)=R_(26)+1
2915  GOTO Repeat
2916 Yeah: Fl(16)=0
2917       Fl(5)=0
2918       Zflag=0
2919       CALL Mate(Q_(*),N_(*),A)
2920  CALL Mate(P(*),Q_(*),A)
2921  CALL Mate(O(*),P(*),A)
2922  R_(29)=R_(29)+R_(26)
2923 !PRINT "ITERATION =";R_(26)
2924  R_(26)=0
2925  GOSUB Outa
2926  ON KEY 5 GOTO Leave
2927  R_(52)=R_(52)+D_(3)
2928  IF R_(52)>(D_(2)+D_(3)) AND D_(1)<=D_(2) THEN GOTO 2935
2929  IF R_(52)<(D_(2)+D_(3)) AND D_(1)>=D_(2) THEN GOTO 2935
2930  GOTO Repeat
2935  R_(52)=D_(1)
2936  PRINT
2937  PRINT
2938  PRINT
2940  R_(53)=R_(53)+D_(6)
2941  IF D_(4)=0 AND D_(5)=0 THEN GOTO Relo
2942  IF R_(53)>D_(5) AND D_(4)<=D_(5) THEN GOTO Relo
2943  IF R_(53)<D_(5) AND D_(4)>=D_(5) THEN GOTO Relo
2944  GOTO Repeat
2945 Zero: !
2946       Fl(5)=0
2947       Fl(16)=0
2948       Zflag=1
```

BIASB.36 LISTING

```
2949        CALL Matzv(A,O(*))
2950        CALL Matzv(A,P(*))
2951        CALL Matzv(A,Q_(*))
2952        CALL Matzv(A,N_(*))
2953   !    CALL Matza(15,M_(*))     ! SHOULD BE REMOVED
2954        GOTO Repeat
2955 Relo:  IMAGE 10X,4D,2X,10A
2956 PRINT USING 2955;R_(29),"ITERATIONS"
2957 PRINT
2958 GOTO Main
2959 Solv:  CALL Solve(O(*),A)
2960 CALL Matzv(A,B(*))
2961 RETURN
2962 Conv:  X=0
2963 FOR Z=1 TO A
2970 IF X<ABS(O(Z)-P(Z)) THEN X=ABS(O(Z)-P(Z))
2980 NEXT Z
2990 IF X<R_(16) THEN F1(7)=1
3000 RETURN
3010 Outa:  IMAGE #,2X,MD.DDE
3020 IF G_(1)<>0 THEN !PRINT USING 3010;T
3021       IF F1(4)=1 THEN PRINT USING 3010;T
3022       IF F1(4)=2 THEN PRINT USING 3010;R_(52)
3023       END IF
3030 FOR Z=1 TO 6
3040 IF G_(Z)=0 THEN 3080
3050 IF G_(Z)<=A THEN PRINT USING 3010;O(G_(Z))
3060 IF G_(Z)>A THEN PRINT USING 3010;Me(G_(Z)-A,6)
3070 NEXT Z
3080 PRINT
3090 RETURN
3091 ! RE-INITIALIZATION FOR NEW CIRCUIT
3092 ! --------------------------------
3100 Newc:    C=0
3110 D=0
3120 E=0
3130 F=0
3140 G=0
3150 I=0
3160 L=0
3170 M=0
3180 N=0
3190 Ra=0
3200 S=0
3210 V=0
3220 IF A=0 THEN A=25
3230 CALL Matzv(A,O(*))
3240 CALL Matzv(A,P(*))
3250 CALL Matzv(A,Q_(*))
3260 CALL Matzv(A,N_(*))
3270 CALL Matzv(A,H(*))
3280 H(0)=0
3290 CALL Matzv(A,J(*))
3291 Flag=0
3300 A=0
3310 RETURN
```

```
3311 ! D.C. ANALYSIS -- PART II
3312 ! ----------------------
3317 Dcop:    FOR Twice=1 TO 2
3320          Fl(4)=0
3321          A=N+V
3322          Loc_ptr=A+1
3325          GOTO 3350
3327          IF Flag=0 THEN
3328 CALL Rstr(Ra,Rb$(*),A_(*),B(*),Q,Re(*))
3329 CALL Vsrc(V,Vb$(*),Va_(*),Sb$(*),A_(*),B(*),Fl(*),T,U,Ve(*),Se(*))
3330 CALL Isrc(I,Ib$(*),Ia(*),Sb$(*),A_(*),B(*),Fl(*),T,U,Ie(*),Se(*))
3331 CALL Capr(C,Cb$(*),Lb$(*),A_(*),B(*),Fl(*),Q_(*),N_(*),T,U,C_(*),Ce(*
3332 CALL Diod(D,Db$(*),Eb$(*),A_(*),B(*),O(*),Fl(*),W,De(*),Ee(*))
3333 CALL Mosf(M,Mb$(*),Fb$(*),A_(*),B(*),O(*),T,U,Fl(*),W,Q_(*),N_(*),M_(
3337 CALL Chkzero(A)
3338 CALL Reorder(A)
3339 CALL Marko(A)
3342 Flag=1
3348 END IF
3349 GOTO 3560
3350  T=0
3351  U=0
3352  R_(11)=0
3360  R_(12)=0
3370  R_(13)=0
3380  R_(27)=0
3390  IF A=0 THEN A=25
3400 CALL Matzv(A,P(*))
3410 CALL Matzv(A,O(*))
3420  IF Flag=0 THEN CALL Initial_buf(A)
3430 CALL Initial_mat(A)
3440 CALL Matza(15,M_(*))
3450 FOR Z=1 TO C
3460 C_(Z,1)=0
3470 C_(Z,2)=0
3480 NEXT Z
3490 CALL Matzv(A,Q_(*))
3500 CALL Matzv(A,N_(*))
3510 FOR Z=1 TO D
3520 De(Z,5)=0
3530 De(Z,6)=0
3540 De(Z,7)=0
3550 NEXT Z
3551 GOTO 3327
3560 NEXT Twice
3561 RETURN
3570 END
3580 !-----------------MAIN SUBROUTINE-----------------------------
3581 !
3583 ! TRANSCIENT ANALYSIS SUBROUTINE
3584 ! ----------------------------
3585 ! SET UP TSTART,TSTOP,TSTEP AND PRESET NODE VOLTAGE
3590  SUB Tran(R_(*),A$,B$,G,Fl(*),O(*),J(*),T,N,A)
3600  IF A=0 THEN A=25
3610  CALL Dcod(R_(11),R_(32),R_(33),0,0,0,A$,B$,0)
3620  IF R_(33)>0 THEN 3650
```

BIASB.36 LISTING

```
3630  CALL Errr(R_(32),2,A$)
3640  SUBEXIT
3650  CALL Dcod(R_(13),R_(32),R_(33),0,0,0,A$,B$,0)
3660  IF (R_(33)=0) AND (R_(13)=0) THEN R_(13)=R_(11)/50
3670  IF R_(33)<=0 THEN SUBEXIT
3680  CALL Dcod(R_(12),R_(32),R_(33),0,0,0,A$,B$,0)
3690  IF R_(33)<=0 THEN R_(12)=0
3700  IF R_(13)=0 THEN R_(13)=(R_(11)-R_(12))/50
3710  IF Fl(16)<>0 THEN CALL Mate(J(*),O(*),A)
3720  Fl(4)=1
3730  SUBEND
3731  ! SUBROUTINE ADJUSTS THE TEMPERATURE FACTOR
3732  ! ---------------------------------------
3740  SUB Temp(R_(*),A$,B$,Q,F,Fb$(*),Fe(*))
3750  CALL Dcod(Q,R_(32),R_(33),0,0,0,A$,B$,0)
3760  IF R_(33)=0 THEN Q=27
3770  PRINT "TEMP:",Q
3780  P1=Q+273
3790  R_(18)=1.16-7.02*1.E-4*P1*P1/(P1+1108)
3800  R_(28)=R_(19)*P1/300
3810  R_(30)=R_(20)*(P1/300)^1.5*EXP(-.5*R_(18)*R_(24)/1.38E-23*(1/P1-1/300
3820  CALL Mosu(F,Fb$(*),R_(*),Q,Fe(*))
3830  SUBEND
3831 ! SUBROUTINE PRINTS THE D.C.,DC TRANSFER OR TRANSCIENT ANALYSIS RESULT
3832 ! -------------------------------------------------------------------
3840 SUB Prin(R_(*),A,N,H(*),O(*),M,Mb$(*),V,Vb$(*),A$,G_(*),Me(*),M_(*))
3850 B$="DCTR"
3860 CALL Dcod(R_(34),R_(32),R_(33),0,0,0,A$,B$,2)
3870 IF R_(33)>=0 THEN CALL Errr(R_(32),3,A$)
3880 IF R_(33)>=0 THEN SUBEXIT
3890 IF R_(34)=3 THEN GOTO Prtr
3900 IF A=0 THEN SUBEXIT
3910 IMAGE 20X,25A
3920 IMAGE #,2X,A,3D,3A
3930 IMAGE #,2X,MD.DDE
3940 IMAGE #,2X,4A,X,MD.DDE
3950 IMAGE 18X,25A
3960 PRINT FNLin$(5)
3970 PRINT USING 3950;"*D.C.  OPERATING POINT*"
3980 PRINT FNLin$(2)
3990 PRINT USING 3910;"NODE VOLTAGES"
4000 PRINT
4010 FOR Z=1 TO N
4020 PRINT USING 3920;"(",H(Z),") >"
4030 PRINT USING 3930;O(Z)
4040 IF Z=INT(Z/4)*4 THEN PRINT
4050 NEXT Z
4060 PRINT
4070 IF M=0 THEN GOTO Prso
4080 PRINT FNLin$(2)
4090 PRINT USING 3910;"MOSFET CURRENTS"
4100 PRINT
4110 FOR Z=1 TO M
4120 PRINT USING 3940;Mb$(Z)[1,4],M_(Z,3)
4130 IF Z=INT(Z/4)*4 THEN PRINT
4140 NEXT Z
4150 PRINT
4160 Prso:  IF V=0 THEN SUBEXIT
4170 PRINT FNLin$(2)
4180 PRINT USING 3910;"VOLTAGE SOURCE CURRENTS"
4190 PRINT
4200 FOR Z=1 TO V
4210 PRINT USING 3940;Vb$(Z)[1,4],O(N+Z)
4220 IF Z=INT(Z/4)*4 THEN PRINT
4230 NEXT Z
4240 PRINT FNLin$(3)
4250 SUBEXIT
4260 Prtr: B$="VI"
4270 FOR Z=1 TO 6
4280 G_(Z)=0
4290 NEXT Z
4300 P5=1
4310 Trlo: CALL Dcod(R_(34),R_(32),R_(33),0,0,0,A$,B$,1)
4320 IF R_(33)=0 THEN SUBEXIT
4330 IF (R_(34)=0) AND (R_(33)>=0) THEN CALL Errr(R_(32),4,A$)
4340 IF R_(34)=1 THEN GOTO Prnv
4350 R_(32)=R_(32)+1
4360 P2=POS(A$[R_(32),R_(32)+5],")")
4370 IF P2=0 THEN CALL Errr(R_(32),5,A$)
4380 IF P2=0 THEN SUBEXIT
4390 N$="     "
4400 N$[1,P2-1]=A$[R_(32),R_(32)+P2-2]
4410 IF N$[1,1]="V" THEN GOTO Pris
4420 FOR Z=1 TO M
4430 IF Mb$(Z)[1,4]=N$[1,4] THEN 4470
4440 NEXT Z
4450 CALL Errr(R_(32),6,A$)
4460 SUBEXIT
4470 G_(P5)=N+Z+V
4480 P5=P5+1
4490 R_(32)=R_(32)+P2+1
4500 IF P5=7 THEN SUBEXIT
4510 GOTO Trlo
4520 Pris: FOR Z=1 TO V
4530 IF N$[1,4]=Vb$(Z)[1,4] THEN 4570
4540 NEXT Z
4550 CALL Errr(R_(32),6,A$)
4560 SUBEXIT
4570 G_(P5)=N+Z
4580 P5=P5+1
4590 R_(32)=R_(32)+P2+1
4600 IF P5=7 THEN SUBEXIT
4610 GOTO Trlo
4620 Prnv: CALL Dcod(R_(34),R_(32),R_(33),0,0,0,A$,B$,0)
4630 IF R_(33)<=0 THEN CALL Errr(R_(32),6,A$)
4640 IF R_(33)<0 THEN SUBEXIT
4650 FOR Z=1 TO N
4660 IF H(Z)=R_(34) THEN 4700
4670 NEXT Z
4680 CALL Errr(R_(32),6,A$)
4690 SUBEXIT
4700 G_(P5)=Z
```

BIASB.36 LISTING

```
4710 P5=P5+1
4720 R_(32)=R_(32)+P2+1
4730 IF P5=7 THEN SUBEXIT
4740 GOTO Trlo
4750 SUBEND
4751 ! GO SUBROUTINE
4752 ! ------------
4753 ! PRINT DC TRANSFER OR TRANSCIENT ANALYSIS HEADINGS
4760 SUB Runc(A,N,V,A_(*),B(*),O(*),P(*),Q_(*),N_(*),F1(*),T,U,R_(*),Dt$(*)
4770 COM /Blk1/ Vb$(*),Mb$(*),Ib$(*),Eb$(*)
4780 COM /Blk4/ J(*),H(*),G_(*)
4790 A=N+V
4810 F1(12)=1
4820 IF F1(4)<>0 THEN GOTO Runt
4830 T=0
4840 F1(11)=0
4850 SUBEXIT
4860 Runt: IF F1(16)=0 THEN CALL Mate(P(*),O(*),A)
4870 CALL Mate(O(*),Q_(*),A)
4880 CALL Mate(Q_(*),N_(*),A)
4890 R_(26)=0
4900 IF F1(11)=0 THEN 4930
4910 F1(11)=0
4920 SUBEXIT
4930 T=0
4940 R_(29)=0
4950 PRINT FNLin$(4)
4960 IMAGE 18X,28A
4970 IMAGE #,2X,9A,3X
4980 IMAGE #,2X,2A,DD,A,4X
4990 IMAGE #,2A,4A,A,4X
4995 IF F1(4)=1 THEN
5000 PRINT USING 4960;"*TRANSIENT ANALYSIS*"
5010 PRINT FNLin$(2)
5020 PRINT USING 4970;"TIME(Sec)"
5021 ELSE
5022 PRINT USING 4960;"*DC TRANSFER ANALYSIS*"
5023 PRINT FNLin$(2)
5025 PRINT USING 4970;Dt$(1)
5026 END IF
5030 FOR Z=1 TO 6
5040 IF G_(Z)=0 THEN GOTO Endp
5050 IF G_(Z)<=N THEN PRINT USING 4980;"V(",H(G_(Z)),")"
5060 IF (G_(Z)>N) AND (G_(Z)<=A) THEN PRINT USING 4990;"I(",Vb$(G_(Z)-N)[1,
5070 IF G_(Z)>A THEN PRINT USING 4990;"I(",Mb$(G_(Z)-A)[1,4],")"
5080 NEXT Z
5090 Endp: !
5100 PRINT
5110 SUBEND
5111 ! CATALOG SUBROUTINE
5112 ! ----------------
5120  SUB Catf(Fi$(*))
5130  ASSIGN @File1 TO "FILES"
5140  ENTER @File1;Fi,Fi$(*)
5150  IMAGE 15X,15A
5160  IMAGE 16X,DD,4X,4A
```

BIASB.36 LISTING

```
5170  PRINT FNLin$(3)
5180  PRINT USING 5150; "*FILE NAMES*"
5190  PRINT
5200  FOR Z=1 TO Fi
5210  PRINT USING 5160; Z, Fi$(Z)[1,4]
5220  NEXT Z
5230  PRINT FNLin$(2)
5240  SUBEND
5241  ! CREATE SUBROUTINE
5242  !-----------------
5243  ! CREATE A STORAGE FILE
5250  SUB Create(A$,R_(*),Fi$(*))
5260  IF R_(32)>=LEN(A$) THEN CALL Errr(R_(32),8,A$)
5270  IF R_(32)>=LEN(A$) THEN SUBEXIT
5280  ASSIGN @File1 TO "FILES"
5290  ENTER @File1;Fi,Fi$(*)
5300  Pos=NUM(A$[R_(32),R_(32)])
5310  IF Pos=32 THEN R_(32)=R_(32)+1
5320  IF Pos=32 THEN 5300
5330  CALL Gnam(R_(32),0,A$,N$)
5340  FOR Z=1 TO Fi
5350  IF Fi$(Z)[1,4]=N$[1,4] THEN CALL Errr(R_(32),14,A$)
5360  IF Fi$(Z)[1,4]=N$[1,4] THEN SUBEXIT
5370  NEXT Z
5380  Z=Fi+1
5390  Fi=Z
5400  Fi$(Z)[1,4]=N$[1,4]
5410  ASSIGN @File1 TO "FILES"
5420  OUTPUT @File1;Fi,Fi$(*)
5430  CREATE BDAT N$[1,4],150
5440  SUBEND
5441  ! LOAD SUBROUTINE
5442  ! ---------------
5443  ! LOADING A FILE FROM MEMORY STORAGE
5450    SUB Load(A$,Fi$(*),Va_(*),Ia(*),M_(*),Db$(*),Ie(*),Fe(*),Le(*),N$)
5460    COM /Blk1/ Vb$(*),Mb$(*),Ib$(*),Eb$(*)
5470    COM /Blk2/ Fb$(*),Sb$(*),Lb$(*)
5480    COM /Blk3/ Ra,C,M,D,I,V,F,S,L,E
5490    COM /Blk4/ J(*),H(*),G_(*)
5500    COM /Blk5/ Me(*),Ce(*),Rb$(*),Cb$(*)
5510    COM /Blk6/ De(*),Ee(*),Re(*),Se(*),Ve(*)
5520    COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
5521    COM /Blk9/ Flag
5530    IF R_(32)>=LEN(A$) THEN CALL Errr(R_(32),8,A$)
5540    IF R_(32)>=LEN(A$) THEN SUBEXIT
5550    ASSIGN @File1 TO "FILES"
5560    ENTER @File1;Fi,Fi$(*)
5570    Pos=NUM(A$[R_(32),R_(32)])
5580    IF Pos=32 THEN R_(32)=R_(32)+1
5590    IF Pos=32 THEN 5570
5600    CALL Gnam(R_(32),0,A$,N$)
5610    FOR Z=1 TO Fi
5620    IF Fi$(Z)[1,4]=N$[1,4] THEN 5660
5630    NEXT Z
5640    CALL Errr(R_(32),12,A$)
5650    SUBEXIT
```

```
5660   ASSIGN @File1 TO N$[1,4]
5670   ENTER @File1;Ra,C,M,D,I,V,F,S,L,E,N
5671   ENTER @File1;Va_(*),Ia(*),M_(*),G_(*),H(*),J(*)
5680   ENTER @File1;Rb$(*),Cb$(*),Db$(*),Mb$(*),Vb$(*)
5681   ENTER @File1;Ib$(*),Fb$(*),Eb$(*),Sb$(*),Lb$(*)
5690   ENTER @File1;Re(*),Ce(*),De(*),Me(*),Ve(*)
5691   ENTER @File1;Ie(*),Fe(*),Ee(*),Se(*),Le(*)
5692   Flag=0
5700   SUBEND
5701   ! SAVE SUBROUTINE
5702   ! --------------
5703   ! SAVE A FILE IN MEMORY STORAGE
5710   SUB Save(A$,Fi$(*),Va_(*),Ia(*),M_(*),Db$(*),Ie(*),Fe(*),Le(*),N$)
5720   COM /Blk1/ Vb$(*),Mb$(*),Ib$(*),Eb$(*)
5730   COM /Blk2/ Fb$(*),Sb$(*),Lb$(*)
5740   COM /Blk3/ Ra,C,M,D,I,V,F,S,L,E
5750   COM /Blk4/ J(*),H(*),G_(*)
5760   COM /Blk5/ Me(*),Ce(*),Rb$(*),Cb$(*)
5770   COM /Blk6/ De(*),Ee(*),Re(*),Se(*),Ve(*)
5780   COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
5790   IF R_(32)>=LEN(A$) THEN CALL Errr(R_(32),8,A$)
5800   IF R_(32)>=LEN(A$) THEN SUBEXIT
5810   ASSIGN @File1 TO "FILES"
5820   ENTER @File1;Fi,Fi$(*)
5830   Pos=NUM(A$[R_(32),R_(32)])
5840   IF Pos=32 THEN R_(32)=R_(32)+1
5850   IF Pos=32 THEN 5830
5860   CALL Gnam(R_(32),0,A$,N$)
5870   FOR Z=1 TO Fi
5880   IF Fi$(Z)[1,4]=N$[1,4] THEN 5920
5890   NEXT Z
5900   CALL Errr(R_(32),12,A$)
5910   SUBEXIT
5920   ASSIGN @File1 TO N$[1,4]
5930   OUTPUT @File1;Ra,C,M,D,I,V,F,S,L,E,N,Va_(*),Ia(*),M_(*),G_(*),H(*),
5940   OUTPUT @File1;Rb$(*),Cb$(*),Db$(*),Mb$(*),Vb$(*),Ib$(*),Fb$(*),Eb$(
5950   OUTPUT @File1;Re(*),Ce(*),De(*),Me(*),Ve(*),Ie(*),Fe(*),Ee(*),Se(*)
5960   SUBEND
5961 ! KILL SUBROUTINE
5962 ! --------------
5963 ! KILL A PARTICULAR FILE IN THE MEMORY STORAGE
5970 SUB Kill(A$,R_(*),Fi$(*))
5980 IF R_(32)>=LEN(A$) THEN CALL Errr(R_(32),8,A$)
5990 IF R_(32)>=LEN(A$) THEN SUBEXIT
6000 ASSIGN @File1 TO "FILES"
6010 ENTER @File1;Fi,Fi$(*)
6020 Pos=NUM(A$[R_(32),R_(32)])
6030 IF Pos=32 THEN R_(32)=R_(32)+1
6040 IF Pos=32 THEN 6020
6050 CALL Gnam(R_(32),0,A$,N$)
6060 FOR Z=1 TO Fi
6070 IF Fi$(Z)[1,4]=N$[1,4] THEN 6100
6080 NEXT Z
6090 SUBEXIT
6100 PURGE N$[1,4]
6110 FOR Y=Z TO Fi
```

```
6120  Fi$(Y)[1,4]=Fi$(Y+1)[1,4]
6130  NEXT Y
6140  Fi=Fi-1
6150  ASSIGN @File1 TO "FILES"
6160  OUTPUT @File1;Fi,Fi$(*)
6170  SUBEND
6180  !-------------------CIRCUIT EDITOR SUBROUTINES------------------
6181  !
6182  ! SUBROUTINE FOR RESISTOR
6183  ! ----------------------
6190 SUB Resrc(R_(*),Rb$(*),Ra,N$,A$,B$,H(*),N,Re(*))
6200 CALL Gnam(1,0,A$,N$)
6210 CALL Gnum(R_(32),3,P1,P2,P3,0,0,A$,B$,R_(*))
6220 IF R_(32)=0 THEN SUBEXIT
6230 CALL Gnum(R_(32),-2,P4,P5,0,0,0,A$,B$,R_(*))
6240 FOR Z=1 TO Ra
6250 IF Rb$(Z)[1,4]=N$[1,4] THEN 6310
6260 NEXT Z
6270 IF Ra=R_(3) THEN CALL Errr(R_(32),15,A$)
6280 IF Ra=R_(3) THEN SUBEXIT
6290 Ra=Ra+1
6300 Z=Ra
6310 Rb$(Z)[1,4]=N$[1,4]
6320 CALL Gnod(P1,N,H(*))
6330 CALL Gnod(P2,N,H(*))
6340 Re(Z,1)=P1
6350 Re(Z,2)=P2
6360 Re(Z,3)=1/P3
6370 Re(Z,4)=P4
6380 Re(Z,5)=P5
6390 R_(34)=0
6400 SUBEND
6401 ! SUBROUTINE FOR CAPACITOR
6402 ! -----------------------
6410 SUB Caprc(R_(*),Cb$(*),C,Lb$(*),L,N$,A$,B$,H(*),K,N,Ce(*),Le(*))
6420 Temp=0
6430 P5=Temp
6440 Ui=Temp
6450 CALL Gnam(1,0,A$,N$)
6460 CALL Gnum(R_(32),3,P1,P2,P3,0,0,A$,B$,R_(*))
6470 IF R_(32)=0 THEN SUBEXIT
6480 FOR Z=1 TO C
6490 IF Cb$(Z)[1,4]=N$[1,4] THEN 6550
6500 NEXT Z
6510 IF C=R_(4) THEN CALL Errr(R_(32),15,A$)
6520 IF C=R_(4) THEN SUBEXIT
6530 Z=C+1
6540 C=Z
6550 Cb$(Z)[1,4]=N$[1,4]
6560 CALL Gnod(P1,N,H(*))
6570 CALL Gnod(P2,N,H(*))
6580 Ce(Z,3)=P1
6590 Ce(Z,4)=P2
6600 CALL Dcod(P5,R_(32),P4,0,0,0,A$,B$,0)
6610 IF P4=1 THEN Vi=P5
6620 IF P4=1 THEN Ui=1
```

```
6630 IF P4<>1 THEN 6650
6640 CALL Dcod(P5,R_(32),P4,0,0,0,A$,B$,0)
6650 IF (P4>=0) OR (P5<>0) THEN X=0
6660 IF (P4>=0) OR (P5<>0) THEN 6790
6670 CALL Gnam(ABS(P4),0,A$,N$)
6680 FOR X=1 TO L
6690 IF Lb$(X)[1,4]=N$[1,4] THEN 6790
6700 NEXT X
6710 IF L=R_(31) THEN CALL Errr(R_(32),16,A$)
6720 IF L=R_(31) THEN SUBEXIT
6730 L=L+1
6740 X=L
6750 Lb$(X)[1,4]=N$[1,4]
6760 Le(X,1)=0
6770 Le(X,2)=0
6780 Le(X,3)=0
6790 Ce(Z,5)=X
6800 Ce(Z,6)=P3
6810 Ce(Z,1)=0
6820 Ce(Z,2)=0
6830 IF Ui<>0 THEN Ce(Z,1)=Vi
6840 IF Ui<>0 THEN Ce(Z,2)=Ui
6850 R_(34)=0
6860 SUBEND
6861 ! SUBROUTINE FOR MOSFET
6862 ! ---------------------
6870 SUB Mosfc(R_(*),Mb$(*),M,Fb$(*),F,N$,A$,B$,H(*),K,N,Me(*),Fe(*))
6880 CALL Gnam(1,0,A$,N$)
6890 CALL Gnum(R_(32),4,P1,P2,P3,P4,0,A$,B$,R_(*))
6900 IF R_(32)=0 THEN SUBEXIT
6910 FOR Z=1 TO M
6920 IF N$[1,4]=Mb$(Z)[1,4] THEN 6990
6930 NEXT Z
6940 IF M=R_(2) THEN CALL Errr(R_(32),15,A$)
6950 IF M=R_(2) THEN SUBEXIT
6960 M=M+1
6970 Z=M
6980 Mb$(Z)[1,4]=N$[1,4]
6990 CALL Gnod(P1,N,H(*))
7000 CALL Gnod(P2,N,H(*))
7010 CALL Gnod(P3,N,H(*))
7020 CALL Gnod(P4,N,H(*))
7030 Me(Z,16)=P1
7040 Me(Z,17)=P2
7050 Me(Z,18)=P3
7060 Me(Z,19)=P4
7070 CALL Dcod(P6,R_(32),P5,0,0,0,A$,B$,0)
7080 IF (P5=0) OR (P6<>0) THEN CALL Errr(ABS(P5),13,A$)
7090 IF (P5=0) OR (P6<>0) THEN R_(34)=0
7100 IF (P5=0) OR (P6<>0) THEN SUBEXIT
7110 CALL Gnam(ABS(P5),0,A$,N$)
7120 FOR X=1 TO F
7130 IF Fb$(X)[1,4]=N$[1,4] THEN 7240
7140 NEXT X
7150 IF F=R_(8) THEN CALL Errr(R_(32),16,A$)
7160 IF F=R_(8) THEN SUBEXIT
```

BIASB.36 LISTING

```
7170 X=F+1
7180 F=X
7190 Fb$(X)[1,4]=N$[1,4]
7200 Fe(X,1)=0
7210 FOR Y=2 TO 27
7220 Fe(X,Y)=0
7230 NEXT Y
7240 Me(Z,20)=X
7250 CALL Gnum(R_(32),-4,P6,P7,P8,P9,0,A$,B$,R_(*))
7260 IF P6=0 THEN P6=1.0E-5
7270 IF P7=0 THEN P7=1.0E-5
7280 Me(Z,21)=0
7290 Me(Z,22)=Z
7300 Me(Z,23)=0
7310 Me(Z,24)=P7
7320 Me(Z,25)=P6
7330 Me(Z,26)=P8
7340 Me(Z,27)=P9
7350 P10=0
7360 P11=0
7370 P12=0
7380 Pos=R_(32)
7390 CALL Dcod(P6,R_(32),P5,0,0,0,A$,B$,0)
7400 CALL Gnum(Pos,-3,P10,P11,P12,0,0,A$,B$,R_(*))
7410 Me(Z,28)=P10
7420 Me(Z,29)=P11
7430  Me(Z,30)=P12
7440  Me(Z,31)=0
7450  Me(Z,32)=0
7460  IF (P5<>0) OR (P6<>0) THEN Me(Z,31)=1
7470  R_(34)=0
7480  SUBEND
7481  ! SUBROUTINE FOR DIODE
7482  ! --------------------
7490  SUB Diodc(R_(*),Db$(*),D,Eb$(*),E,N$,A$,B$,H(*),N,De(*),Ee(*))
7500  CALL Gnam(1,0,A$,N$)
7510  CALL Gnum(R_(32),2,P1,P2,0,0,0,A$,B$,R_(*))
7520  IF R_(32)=0 THEN SUBEXIT
7530  FOR Z=1 TO D
7540  IF N$[1,4]=Db$(Z)[1,4] THEN 7610
7550  NEXT Z
7560  IF D=R_(7) THEN CALL Errr(R_(32),15,A$)
7570  IF D=R_(7) THEN SUBEXIT
7580  Z=D+1
7590  D=Z
7600  Db$(Z)[1,4]=N$[1,4]
7610  CALL Gnod(P1,N,H(*))
7620  CALL Gnod(P2,N,H(*))
7630  De(Z,1)=P1
7640  De(Z,2)=P2
7650  CALL Dcod(P4,R_(32),P3,P5,0,0,A$,B$,0)
7660  IF (P3=0) OR (P4<>0) THEN CALL Errr(ABS(P4),13,A$)
7670  IF (P3=0) OR (P4<>0) THEN R_(34)=0
7680  IF (P3=0) OR (P4<>0) THEN SUBEXIT
7690  CALL Gnam(ABS(P3),0,A$,N$)
7700  FOR Y=1 TO E
```

```
7710  IF N$[1,4]=Eb$(Y)[1,4] THEN 7790
7720  NEXT Y
7730  IF E=R_(9) THEN CALL Errr(R_(32),16,A$)
7740  IF E=R_(9) THEN SUBEXIT
7750  E=E+1
7760  Y=E
7770  Eb$(Y)[1,4]=N$[1,4]
7780  Ee(Y,1)=0
7790  Ee(Y,2)=Y
7800  CALL Gnum(R_(32),-1,P3,0,0,0,0,A$,B$,R_(*))
7810  IF P3=0 THEN P3=1
7820  De(Z,3)=Y
7830  De(Z,4)=P3
7840  De(Z,5)=0
7850  De(Z,6)=0
7860  De(Z,7)=0
7870  R_(34)=0
7880  SUBEND
7881  ! SUBROUTINE FOR VOLTAGE SOURCE
7882  ! ----------------------------
7890  SUB Voltc(R_(*),Vb$(*),Va_(*),V,Sb$(*),S,N$,A$,B$,H(*),N,J(*),Ve(*),S
7900  CALL Gnam(1,0,A$,N$)
7910  CALL Gnum(R_(32),3,P1,P2,P3,0,0,A$,B$,R_(*))
7920  IF R_(32)=0 THEN SUBEXIT
7930  FOR Z=1 TO V
7940  IF N$[1,4]=Vb$(Z)[1,4] THEN 8010
7950  NEXT Z
7960  IF V=R_(5) THEN CALL Errr(R_(32),15,A$)
7970  IF V=R_(5) THEN SUBEXIT
7980  Z=V+1
7990  V=Z
8000  Vb$(Z)[1,4]=N$[1,4]
8010  CALL Dcod(R_(34),R_(32),R_(33),0,0,0,A$,B$,0)
8020  IF R_(33)>=0 THEN X=0
8030  IF R_(33)>=0 THEN 8170
8040  CALL Gnam(ABS(R_(33)),0,A$,N$)
8050  FOR X=1 TO S
8060  IF Sb$(X)[1,4]=N$[1,4] THEN 8170
8070  NEXT X
8080  IF S=R_(10) THEN CALL Errr(R_(32),16,A$)
8090  IF S=R_(10) THEN SUBEXIT
8100  X=S+1
8110  S=X
8120  Sb$(X)[1,4]=N$[1,4]
8130  Se(X,1)=0
8140  FOR Y=2 TO 23
8150  Se(X,Y)=0
8160  NEXT Y
8170  CALL Gnod(P1,N,H(*))
8180  CALL Gnod(P2,N,H(*))
8190  Ve(Z,1)=P1
8200  Ve(Z,2)=P2
8210  Ve(Z,3)=X
8220  Va_(Z)=P3
8230  IF P2=0 THEN J(P1)=P3
8240  IF P1=0 THEN J(P2)=-P3
```

```
8250  R_(34)=0
8260  SUBEND
8261  ! SUBROUTINE FOR CURRENT SOURCE
8262  ! ----------------------------
8270  SUB Curnc(R_(*),Ib$(*),Ia(*),I,Sb$(*),S,N$,A$,B$,H(*),N,J(*),Ie(*),Se
8280  CALL Gnam(1,0,A$,N$)
8290  CALL Gnum(R_(32),3,P1,P2,P3,0,0,A$,B$,R_(*))
8300  IF R_(32)=0 THEN SUBEXIT
8310  FOR Z=1 TO I
8320  IF N$[1,4]=Ib$(Z)[1,4] THEN 8390
8330  NEXT Z
8340  IF I=R_(6) THEN CALL Errr(R_(32),15,A$)
8350  IF I=R_(6) THEN SUBEXIT
8360  I=I+1
8370  Z=I
8380  Ib$(Z)[1,4]=N$[1,4]
8390  CALL Gnod(P1,N,H(*))
8400  CALL Gnod(P2,N,H(*))
8410  Ie(Z,1)=P1
8420  Ie(Z,2)=P2
8430  CALL Dcod(R_(34),R_(32),R_(33),0,0,0,A$,B$,0)
8440  IF R_(33)>=0 THEN X=0
8450  IF R_(33)>=0 THEN 8590
8460  CALL Gnam(ABS(R_(33)),0,A$,N$)
8470  FOR X=1 TO S
8480  IF Sb$(X)[1,4]=N$[1,4] THEN 8590
8490  NEXT X
8500  IF S=R_(10) THEN CALL Errr(R_(32),16,A$)
8510  IF S=R_(10) THEN SUBEXIT
8520  X=S+1
8530  S=X
8540  Sb$(X)[1,4]=N$[1,4]
8550  Se(X,1)=0
8560  FOR Y=2 TO 23
8570  Se(X,Y)=0
8580  NEXT Y
8590  Ie(Z,3)=X
8600  Ia(Z)=P3
8610  R_(34)=0
8620  SUBEND
8621  ! SUBROUTINE FOR PRE-SETTING THE CIRCUIT DESCRIPTION
8622  ! -----------------------------------------------
8630  SUB Presc(R_(*),J(*),N,H(*),A$,B$)
8640  CALL Gnum(R_(32),-2,P1,P2,0,0,0,A$,B$,R_(*))
8650  IF (R_(32)<=0) OR (P1<=0) THEN SUBEXIT
8660  CALL Gnod(P1,N,H(*))
8670  J(P1)=P2
8680  GOTO 8640
8690  SUBEND
8691  ! SUBROUTINE TO LIST ALL CIRCUIT DESCRIPTION
8692  ! ----------------------------------------
8700  SUB Liscc(A$,B$,Va_(*),Ia(*),Db$(*),Le(*),Ie(*),Fe(*))
8710  COM /Blk1/ Vb$(*),Mb$(*),Ib$(*),Eb$(*)
8720  COM /Blk2/ Fb$(*),Sb$(*),Lb$(*)
8730  COM /Blk3/ Ra,C,M,D,I,V,F,S,L,E
8740  COM /Blk4/ J(*),H(*),G_(*)
```

```
8750  COM /Blk5/ Me(*),Ce(*),Rb$(*),Cb$(*)
8760  COM /Blk6/ De(*),Ee(*),Re(*),Se(*),Ve(*)
8770  COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
8780  IMAGE 20X,30A
8790  IMAGE 5X,70A
8800  IMAGE #,5X,4A,1X  !PRINTS ELEMENT NAME
8810  IMAGE #,5X,4A,2X
8820  IMAGE #,1X,2D,1X,2D !PRINTS NODES
8830  IMAGE #,11X  !USED AS A SPACER
8840  IMAGE 5X,5A,3X,5A
8850  IMAGE 18X,15A,X,MD.DDE
8860  IMAGE 18X,2(MD.DDE,2X)
8870  IMAGE #,2X,MD.DDE   !PRINTS VALUES
8880  IMAGE #,4X,D.DD,1X
8890  PRINT FNLin$(5)
8900  A$="CIRCUIT ELEMENTS"
8910  PRINT USING 8780;A$
8920  PRINT FNLin$(1)
8921  ! PRINT RESISTORS DESCRIPTION
8922  ! -------------------------
8930  IF Ra<=0 THEN Lstc
8940  A$="*RESISTORS*"
8950  B$="NAME  N1 N2    VALUE       TC1         TC2"
8960  PRINT USING 8780;A$
8970  PRINT USING 8790;B$
8980  FOR Z=1 TO Ra
8990  P1=Re(Z,1)
9000  P2=Re(Z,2)
9010  P3=Re(Z,3)
9020  P4=Re(Z,4)
9030  P5=Re(Z,5)
9040  PRINT USING 8800;Rb$(Z)[1,4]
9050  PRINT USING 8820;H(P1),H(P2)
9060  PRINT USING 8870;1/P3
9070  IF P4<>0 THEN PRINT USING 8870;P4
9080  IF P4=0 THEN PRINT USING 8830
9090  IF P5<>0 THEN PRINT USING 8870;P5
9100  PRINT
9110  NEXT Z
9120  PRINT FNLin$(1)
9121  ! PRINT CAPACITORS DESCRIPTION
9122  ! --------------------------
9130 Lstc:  IF C<=0 THEN Lstv
9140  A$="*CAPACITORS*"
9150  B$="NAME  N1 N2    VALUE      I. C.       MODEL"
9160  PRINT USING 8780;A$
9170  PRINT USING 8790;B$
9180  FOR Z=1 TO C
9190  P1=Ce(Z,3)
9200  P2=Ce(Z,4)
9210  P3=Ce(Z,5)
9220  P4=Ce(Z,6)
9230  P5=Ce(Z,1)
9240  P6=Ce(Z,2)
9250  PRINT USING 8800;Cb$(Z)[1,4]
9260  PRINT USING 8820;H(P1),H(P2)
```

```
9270  PRINT USING 8870;P4
9280  IF P6<>0 THEN PRINT USING 8870;P5
9290  IF P6=0 THEN PRINT USING 8810;"    "
9300  IF P3<>0 THEN PRINT USING 8800;Lb$(P3)[1,4]
9310  PRINT
9320  NEXT Z
9330  PRINT FNLin$(1)
9331  ! PRINT VOLTAGE SOURCE DESCRIPTION
9332  ! --------------------------------
9340 Lstv: IF V<=0 THEN GOTO Lsti
9350  A$="*VOTLAGE SOURCES*"
9360  B$="NAME  N+ N-    VALUE      MODEL"
9370  PRINT USING 8780;A$
9380  PRINT USING 8790;B$
9390  FOR Z=1 TO V
9400  P1=Ve(Z,1)
9410  P2=Ve(Z,2)
9420  P3=Ve(Z,3)
9430  PRINT USING 8800;Vb$(Z)[1,4]
9440  PRINT USING 8820;H(P1),H(P2)
9450  PRINT USING 8870;Va_(Z)
9460  IF P3<>0 THEN PRINT USING 8800;Sb$(P3)[1,4]
9470  PRINT
9480  NEXT Z
9490  PRINT FNLin$(1)
9491  ! PRINT CURRENT SOURCE DESCRIPTION
9492  ! --------------------------------
9500 Lsti: IF I<=0 THEN GOTO Lstd
9510  A$="*CURRENT SOURCES*"
9520  B$="NAME  NF NT    VALUE      MODEL"
9530  PRINT USING 8780;A$
9540  PRINT USING 8790;B$
9550  FOR Z=1 TO I
9560  P1=Ie(Z,1)
9570  P2=Ie(Z,2)
9580  P3=Ie(Z,3)
9590  PRINT USING 8800;Ib$(Z)[1,4]
9600  PRINT USING 8820;H(P1),H(P2)
9610  PRINT USING 8870;Ia(Z)
9620  IF P3<>0 THEN PRINT USING 8800;Sb$(P3)[1,4]
9630  PRINT
9640  NEXT Z
9650  PRINT FNLin$(1)
9651  ! PRINT DIODE DESCRIPTION
9652  ! ----------------------
9660 Lstd: IF D<=0 THEN GOTO Lstm
9670  A$="*DIODES*"
9680  B$="NAME  N+ N-    AREA     MODEL"
9690  PRINT USING 8780;A$
9700  PRINT USING 8790;B$
9710  FOR Z=1 TO D
9720  P1=De(Z,1)
9730  P2=De(Z,2)
9740  P3=De(Z,4)
9750  PRINT USING 8800;Db$(Z)[1,4]
9760  PRINT USING 8820;H(P1),H(P2)
```

BIASB.36 LISTING

```
9770  PRINT USING 8880;P3
9780  PRINT USING 8800;Eb$(1)[1,4]
9790  PRINT
9800  NEXT Z
9810  PRINT FNLin$(1)
9811  ! PRINT MOSFET DESCRIPTION
9812  ! ----------------------
9820 Lstm: IF M<=0 THEN GOTO Lstp
9830  A$="*MOSFETS*"
9840  B$="NAME  ND NG NS NB    MODEL       WC           LC           AD
9850  PRINT USING 8780;A$
9860  PRINT USING 8790;B$
9870  FOR Z=1 TO M
9880  P1=Me(Z,16)
9890  P2=Me(Z,17)
9900  P3=Me(Z,18)
9910  P4=Me(Z,19)
9920  P5=Me(Z,20)
9930  IF Me(Z,31)<>0 THEN Ii=1
9940  PRINT USING 8800;Mb$(Z)[1,4]
9950  PRINT USING 8820;H(P1),H(P2)
9960  PRINT USING 8820;H(P3),H(P4)
9970  PRINT USING 8800;Fb$(P5)[1,4]
9980  PRINT USING 8870;Me(Z,25)
9990  PRINT USING 8870;Me(Z,24)
10000 PRINT USING 8870;Me(Z,26)
10010 PRINT USING 8870;Me(Z,27)
10020 PRINT
10030 NEXT Z
10040 IF Ii=0 THEN 10180
10050 PRINT FNLin$(1)
10060 A$="*MOSFET INITIAL CONDITIONS*"
10070 B$="NAME     Vgsi       Vdsi       Vsbi"
10080 PRINT USING 8780;A$
10090 PRINT USING 8790;B$
10100 FOR Z=1 TO M
10110 Vgsi=Me(Z,28)
10120 Vdsi=Me(Z,29)
10130 Vsbi=Me(Z,30)
10140 IF Me(Z,31)<>0 THEN PRINT USING 8800;Mb$(Z)[1,4]
10150 IF Me(Z,31)<>0 THEN PRINT USING 8870;Vgsi,Vdsi,Vsbi
10160 IF Me(Z,31)<>0 THEN PRINT
10170 NEXT Z
10180 PRINT FNLin$(1)
10181 ! PRINT PRESET VOLTAGE DESCRIPTION
10182 ! ------------------------------
10190 Lstp: FOR Z=1 TO N
10200       X=0
10210 IMAGE #,4X
10220 IMAGE 4X
10230 IMAGE #,XX,DD,XX
10240 IMAGE #,MD.DDE,X
10250 IF J(Z)<>0 THEN 10280
10260 NEXT Z
10270 GOTO Lsts
10280 PRINT USING 8780;"*PRESET TRANSIENT VOLTAGES*"
```

```
10290 PRINT USING 8790;"NODE   VALUE    NODE   VALUE    NODE   VALUE    NOD
10300 PRINT USING 10210
10310 FOR Z=1 TO N
10320 IF J(Z)=0 THEN 10370
10330 PRINT USING 10230;H(Z)
10340 PRINT USING 10240;J(Z)
10350 W=W+1
10360 IF INT(W/4)*4=W THEN PRINT USING 10220
10370 NEXT Z
10380 PRINT FNLin$(1)
10381 ! PRINT SOURCE MODEL SPECIFICATIONS
10382 ! ------------------------------
10390 Lsts: IF S<=0 THEN SUBEXIT
10400 A$="*SOURCE MODEL SPECIFICATIONS*"
10410 PRINT
10420 PRINT USING 8780;A$
10430 FOR Z=1 TO S
10440 PRINT
10450 PRINT USING 8840;"MODEL:",Sb$(Z)[1,4]
10460 IF Se(Z,1)<>0 THEN Liss
10470 PRINT USING 8790;"     **MODEL UNDEFINED**"
10480 GOTO Exit
10490 Liss: IF Se(Z,1)>0 THEN Ltim
10500 P1=Se(Z,6)-Se(Z,4)
10510 P2=Se(Z,10)-Se(Z,8)
10520 P3=Se(Z,8)-Se(Z,6)
10530 PRINT USING 8850;"INITIAL VALUE >",Se(Z,3)
10540 PRINT USING 8850;"PULSED VALUE  >",Se(Z,7)
10550 PRINT USING 8850;"PULSED DELAY  >",Se(Z,4)
10560 PRINT USING 8850;"PULSE  WIDTH  >",P3
10570 PRINT USING 8850;"RISE  TIME    >",P1
10580 PRINT USING 8850;"FALL  TIME    >",P2
10590 PRINT USING 8850;"PERIOD        >",Se(Z,22)/2
10600 GOTO Exit
10610 Ltim: PRINT USING 8780;"  TIME      VALUE"
10620 FOR Y=1 TO 6
10630 IF (Se(Z,Y*2)=0) AND (Y<>1) THEN GOTO Exit
10640 PRINT USING 8860;Se(Z,2*Y),Se(Z,2*Y+1)
10650 NEXT Y
10660 Exit: NEXT Z
10670 SUBEND
10671 ! KILL SUBROUTINE IN CIRCUIT EDITOR
10672 ! ------------------------------
10680 SUB Kilec(A$,B$,N$,Db$(*),Va_(*),Ia(*),K,Ie(*))
10690 COM /Blk1/ Vb$(*),Mb$(*),Ib$(*),Eb$(*)
10700 COM /Blk3/ Ra,C,M,D,I,V,F,S,L,E
10710 COM /Blk5/ Me(*),Ce(*),Rb$(*),Cb$(*)
10720 COM /Blk6/ De(*),Ee(*),Re(*),Se(*),Ve(*)
10730 COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
10740 A$[1,4]=A$[R_(32)+1]
10750 CALL Dcod(P3,P1,P2,0,0,0,A$,B$,1)
10760 IF P2<0 THEN 10790
10770 CALL Errr(R_(32),6,A$)
10780 SUBEXIT
10790 N$="    "
10800 N$=A$
```

```
10810 IF P3=1 THEN GOSUB Popr
10820 IF P3=2 THEN GOSUB Popc
10830 IF P3=3 THEN GOSUB Popm
10840 IF P3=4 THEN GOSUB Popd
10850 IF P3=5 THEN GOSUB Popv
10860 IF P3=6 THEN GOSUB Popi
10870 SUBEXIT
10880 Popr: FOR Z=1 TO Ra
10890 IF Rb$(Z)[1,4]=N$[1,4] THEN 10930
10900 NEXT Z
10910 CALL Errr(R_(32),6,A$)
10920 RETURN
10930 Ra=Ra-1
10940 FOR Y=Z TO Ra
10950 Rb$(Y)=Rb$(Y+1)
10960 FOR X=1 TO 5
10970 Re(Y,X)=Re(Y+1,X)
10980 NEXT X
10990 NEXT Y
11000 RETURN
11010 Popc: FOR Z=1 TO C
11020 IF Cb$(Z)[1,4]=N$[1,4] THEN 11060
11030 NEXT Z
11040 CALL Errr(R_(32),6,A$)
11050 RETURN
11060 C=C-1
11070 FOR Y=Z TO C
11080 Cb$(Y)=Cb$(Y+1)
11090 FOR X=1 TO 8
11100 Ce(Y,X)=Ce(Y+1,X)
11110 NEXT X
11120 NEXT Y
11130 RETURN
11140 Popm: FOR Z=1 TO M
11150 IF Mb$(Z)[1,4]=N$[1,4] THEN 11190
11160 NEXT Z
11170 CALL Errr(R_(32),6,A$)
11180 RETURN
11190 M=M-1
11200 FOR Y=Z TO M
11210 Mb$(Y)=Mb$(Y+1)
11220 FOR X=1 TO 32
11230 Me(Y,X)=Me(Y+1,X)
11240 NEXT X
11250 NEXT Y
11260 RETURN
11270 Popd: FOR Z=1 TO D
11280 IF Db$(Z)[1,4]=N$[1,4] THEN 11320
11290 NEXT Z
11300 CALL Errr(R_(32),6,A$)
11310 RETURN
11320 D=D-1
11330 FOR Y=Z TO D
11340 Db$(Y)=Db$(Y+1)
11350 FOR X=1 TO 7
11360 De(Y,X)=De(Y+1,X)
```

```
11370 NEXT X
11380 NEXT Y
11390 RETURN
11400 Popv: FOR Z=1 TO V
11410 IF Vb$(Z)[1,4]=N$[1,4] THEN 11450
11420 NEXT Z
11430 CALL Errr(R_(32),6,A$)
11440 RETURN
11450 V=V-1
11460 FOR Y=Z TO V
11470 Vb$(Y)=Vb$(Y+1)
11480 FOR X=1 TO 3
11490 Ve(Y,X)=Ve(Y+1,X)
11500 NEXT X
11510 NEXT Y
11520 RETURN
11530 Popi: FOR Z=1 TO I
11540 IF Ib$(Z)[1,4]=N$[1,4] THEN 11580
11550 NEXT Z
11560 CALL Errr(R_(32),6,A$)
11570 RETURN
11580 I=I-1
11590 FOR Y=Z TO I
11600 Ib$(Y)=Ib$(Y+1)
11610 FOR X=1 TO 3
11620 Ie(Y,X)=Ie(Y+1,X)
11630 NEXT X
11640 NEXT Y
11650 RETURN
11660 SUBEND
11661 ! SUBROUTINE PRINTS WHAT EACH NODE IS CONNECTED TO
11662 ! ---------------------------------------------------
11670 SUB Veryc(H(*),Db$(*),Ie(*))
11680 COM /Blk1/ Vb$(*),Mb$(*),Ib$(*),Eb$(*)
11690 COM /Blk3/ Ra,C,M,D,I,V,F,S,L,E       !*
11700 COM /Blk5/ Me(*),Ce(*),Rb$(*),Cb$(*)
11710 COM /Blk6/ De(*),Ee(*),Re(*),Se(*),Ve(*)
11720 COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
11730 DIM Aa$(4)[70]
11740 IMAGE 20X,12A
11750 Temp=0
11760 P1=Temp
11770 A$="*NODE TABLE*"
11780 PRINT FNLin$(1)
11790 PRINT USING 11740;A$
11800 PRINT
11810 FOR Z=1 TO N
11820 PRINT
11830 IMAGE 15X,4A,1X,DD
11840 PRINT USING 11830;"NODE";H(Z)
11850 PRINT
11860 Aa$(1)="Drain"
11870 Aa$(2)="Gate"
11880 Aa$(3)="Source"
11890 Aa$(4)="Bulk"
11900 CALL Nprin(M,Mb$(*),4,Z,P2,P3,Aa$(*),Me(*),15)
```

```
11910 Aa$(1)="+Node"
11920 Aa$(2)="-Node"
11930 CALL Nprin(V,Vb$(*),2,Z,P2,P3,Aa$(*),Ve(*),0)
11940 Aa$(1)="From"
11950 Aa$(2)="To"
11960 CALL Nprin(I,Ib$(*),2,Z,P2,P3,Aa$(*),Ie(*),0)
11970 Aa$(1)="Anode"
11980 Aa$(2)="Cathode"
11990 CALL Nprin(D,Db$(*),2,Z,P2,P3,Aa$(*),De(*),0)
12000 Aa$(1)=" "
12010 Aa$(2)=" "
12020 CALL Nprin(Ra,Rb$(*),2,Z,P2,P3,Aa$(*),Re(*),0)
12030 IF P3<2 THEN BEEP
12040 IF P3<2 THEN PRINT "WARNING:  DC"
12050 CALL Nprin(C,Cb$(*),2,Z,P2,P3,Aa$(*),Ce(*),2)
12060 IF P3<2 THEN BEEP
12070 IF P3<2 THEN PRINT "WARNING:  TR"
12080 NEXT Z
12090 SUBEND
12091 !
12100 SUB Nprin(A,Zb$(*),N,Z,P2,P3,Aa$(*),Ze(*),Dan)
12110 IMAGE 20X,4A,2X,10A
12120 FOR X=1 TO A
12130 FOR Y=1 TO N
12140 IF Z<>Ze(X,Y+Dan) THEN 12170
12150 PRINT USING 12110;Zb$(X)[1,4],Aa$(Y)
12160 P3=P3+1
12170 NEXT Y
12180 NEXT X
12190 SUBEND
12191 ! SOURCE MODEL SUBROUTINE
12192 ! ----------------------
12193 ! GET AND PROCESS THE SOURCE MODEL DATA
12200 SUB Smodc(R_(*),S,Sb$(*),N$,A$,B$,Se(*))
12210 CALL Gnam(1,0,A$,N$)
12220 FOR Z=1 TO S
12230 IF N$[1,4]=Sb$(Z)[1,4] THEN 12290
12240 NEXT Z
12250 S=S+1
12260 Z=S
12270 Sb$(Z)[1,4]=N$[1,4]
12280 Se(Z,1)=-1
12290 CALL Dcod(P3,R_(32),P2,0,0,0,A$,B$,0)
12300 IF P2<=0 THEN GOTO Input
12310 CALL Dcod(P5,R_(32),P2,0,0,0,A$,B$,0)
12320 IF P2>0 THEN 12350
12330 CALL Errr(R_(32),7,A$)
12340 SUBEXIT
12350 CALL Dcod(P6,R_(32),P2,0,0,0,A$,B$,0)
12360 IF P2>0 THEN 12390
12370 CALL Errr(R_(32),7,A$)
12380 SUBEXIT
12390 CALL Dcod(P7,R_(32),P2,0,0,0,A$,B$,0)
12400 IF P2>0 THEN 12430
12410 CALL Errr(R_(32),7,A$)
12420 SUBEXIT
```

```
12430 CALL Dcod(P10,R_(32),P2,0,0,0,A$,B$,0)
12440 IF P2<=0 THEN P10=2*P7
12450 P8=P7/10
12460 P9=P8
12470 CALL Dcod(Pa,R_(32),P2,0,0,0,A$,B$,0)
12480 IF P2>0 THEN P8=Pa
12490 CALL Dcod(Pa,R_(32),P2,0,0,0,A$,B$,0)
12500 IF P2>0 THEN P9=Pa
12510 Se(Z,2)=0
12520 Se(Z,3)=P3
12530 Se(Z,4)=P6
12540 Se(Z,5)=P3
12550 Se(Z,6)=P6+P8
12560 Se(Z,7)=P5
12570 Se(Z,8)=P6+P7+P8
12580 Se(Z,9)=P5
12590 Se(Z,10)=P6+P7+P8+P9
12600 Se(Z,11)=P3
12610 Se(Z,12)=P10
12620 Se(Z,13)=P3
12630 Se(Z,14)=P10+P6
12640 Se(Z,15)=P3
12650 Se(Z,16)=P10+P6+P8
12660 Se(Z,17)=P5
12670 Se(Z,18)=P10+P6+P7+P8
12680 Se(Z,19)=P5
12690 Se(Z,20)=P10+P6+P7+P8+P9
12700 Se(Z,21)=P3
12710 Se(Z,22)=2*P10
12720 Se(Z,23)=P3
12730 SUBEXIT
12740 Input:Y=2
12750 Se(Z,1)=1
12760 INPUT "Initial Value",A$
12770 CALL Dcod(P3,1,P4,0,0,0,A$,B$,0)
12780 IF P4>0 THEN GOTO 12810
12790 SUBEXIT
12800 Se(Z,2)=0
12810 Se(Z,3)=P3
12820 INPUT "Time:",A$
12830 CALL Dcod(P3,1,P4,0,0,0,A$,B$,0)
12840 IF P4>0 THEN GOTO 12860
12850 SUBEXIT
12860 IF P3=0 THEN SUBEXIT
12870 Se(Z,Y+2)=P3
12880 INPUT "Value:",A$
12890 CALL Dcod(P3,1,P4,0,0,0,A$,B$,0)
12900 IF P4>0 THEN GOTO 12920
12910 SUBEXIT
12920 Se(Z,Y+3)=P3
12930 Y=Y+2
12940 IF Y<22 THEN 12820
12950 SUBEND
12960 !----------------MODEL SUBROUTINES--------------------------------
12961 !
12962 ! SUBROUTINE FOR MOSFET MODEL
```

```
12963 ! -------------------------
12970 SUB Mmod(A$,B$,N$,F,Fb$(*),R_(*),Q,Fe(*),O$(*))
12980 CALL Gnam(1,0,A$,N$)
12990 FOR Z=1 TO F
13000 IF (N$[1,4]=Fb$(Z)[1,4]) AND (Fe(Z,1)<>0) THEN 13110
13010 IF N$[1,4]=Fb$(Z)[1,4] THEN 13060
13020 NEXT Z
13030 Z=F+1
13040 F=Z
13050 Fb$(Z)[1,4]=N$[1,4]
13060 B$="NP"
13070 CALL Dcod(P2,R_(32),P1,0,0,0,A$,B$,1)
13080 P1=1
13090 IF P2>1 THEN P1=-1
13100 Fe(Z,1)=P1
13110 B$="VTOK' GAMPHILAMCGSCGDCGBCDBCSBCOXJS PB NSUNSSUO ECREEXETRNFSLDIXJ
13120 IF F1(0)<>0 THEN CALL Proc(A$,Pn,O$(*))
13130 IF F1(0)<>0 THEN GOTO 13150
13140 Mose: INPUT "PARAMETER",A$
13150 P1=1
13160 CALL Dcod(P3,P1,P2,0,0,0,A$,B$,3)
13170 IF (P2<0) AND (P3<>0) THEN 13200
13180 CALL Errr(P1,9,A$)
13190 GOTO Mose
13200 IF P3<>73 THEN 13230
13210 CALL Mosu(F,Fb$(*),R_(*),Q,Fe(*))
13220 SUBEXIT
13230 CALL Dcod(P4,P1,P2,0,0,0,A$,B$,3)
13240 IF P4=0 THEN 13280
13250 PRINT B$[P3,P3+2];P4
13260 P3=(P3-1)/3+2
13270 Fe(Z,P3)=P4
13280 GOTO Mose
13290 SUBEND
13300 ! SUBROUTINE FOR DIODE MODEL
13301 ! -------------------------
13310 SUB Dmod(A$,N$,Q,E,Eb$(*),R_(*),Ee(*))
13320 CALL Gnam(1,0,A$,N$)
13330 FOR Z=1 TO E
13340 IF N$[1,4]=Eb$(Z)[1,4] THEN 13390
13350 NEXT Z
13360 E=E+1
13370 Z=E
13380 Eb$(Z)[1,4]=N$[1,4]
13390 CALL Dcod(P2,R_(32),P1,0,0,0,A$,B$,0)
13400 IF (P1>0) AND (P2<>0) THEN 13430
13410 CALL Errr(R_(32),7,A$)
13420 SUBEXIT
13430 Ee(Z,1)=P2*((Q+273)/300)^2*EXP(7.54E+3*(1/300-1/(Q+273)))
13440 Ee(Z,2)=Z
13450 SUBEND
13451 ! SUBROUTINE FOR CAPACITOR MODEL
13452 ! ----------------------------
13460 SUB Cmod(A$,N$,L,Lb$(*),R_(*),Le(*))
13470 CALL Gnam(1,0,A$,N$)
13480 FOR Z=1 TO L
```

```
13490 IF N$[1,4]=Eb$(Z)[1,4] THEN 13540 !Eb OR Lb?B
13500 NEXT Z
13510 L=L+1
13520 Z=L
13530 Lb$(Z)[1,4]=N$[1,4]
13540 CALL Gnum(R_(32),-3,P1,P2,P3,0,0,A$,B$,R_(*))
13550 Le(Z,1)=P1
13560 Le(Z,2)=P2
13570 Le(Z,3)=P3
13580 SUBEND
13581 ! SUBROUTINE LISTS ALL THE MODEL PARAMETERS
13582 ! ------------------------------------------
13590 SUB Lism(F,Fb$(*),E,Eb$(*),L,Lb$(*),B$,Fe(*),Ee(*),Le(*))
13600 B$="VTOK' GAMPHILAMCGSCGDCGBCDBCSBCOXJS PB NSUNSSUO VCREEXETRNFSLDIXJ
13610 IMAGE 23X,15A
13620 IMAGE 18X,10A,2X,A,4A,A,2X,9A
13630 IMAGE #,5X,3A,X,A,X,MD.DDE
13640 IMAGE 5X,3(5A,X,MD.DDE,5X)
13650 PRINT FNLin$(5)
13660 PRINT USING 13610;"*DEVICE MODELS*"
13670 PRINT FNLin$(2)
13680 FOR Z=1 TO F
13690 P1=Fe(Z,1)
13700 N$="N-Channel"
13710 IF P1<0 THEN N$="P-Channel"
13720 PRINT USING 13620;"MOS MODEL","'",Fb$(Z)[1,4],"'",N$
13730 FOR Y=1 TO 24
13740 PRINT USING 13630;B$[3*Y-2,3*Y],">",Fe(Z,Y+1)
13750 IF Y=INT(Y/3)*3 THEN PRINT
13760 NEXT Y
13770 PRINT FNLin$(1)
13780 NEXT Z
13790 PRINT FNLin$(1)
13800 IF E=0 THEN GOTO Cmli
13810 FOR Z=1 TO E
13820 PRINT USING 13620;"DIO MODEL","'",Eb$(Z)[1,4],"'"
13830 PRINT USING 13640;" IS >",Ee(Z,1)
13840 NEXT Z
13850 PRINT FNLin$(1)
13860 Cmli:FOR Z=1 TO L
13870 PRINT USING 13620;"CAP MODEL","'",Lb$(Z)[1,4],"'"
13880 PRINT USING 13640;"C/A >",Le(Z,1),"Exp >",Le(Z,2),"Phi >",Le(Z,3)
13890 PRINT
13900 NEXT Z
13910 SUBEND
13920 ! ------------------PROCEDURE SUBROUTINES-------------------------
13921 !
13922 ! KILL SUBROUTINE IN PROCEDURE EDITOR
13923 ! ----------------------------------
13924 ! KILL A PARTICULAR LINE IN THE PROCEDURE FILE
13930 SUB Killp(O$(*))
13940 FOR Z=1 TO 40
13950 O$(Z)=" "
13960 NEXT Z
13970 SUBEND
13980 SUB Savep(O$(*))
```

```
13990 ASSIGN @File1 TO "PROCF"
14000 OUTPUT @File1;O$(*)
14010 SUBEND
14011 ! GET SUBROUTINE
14012 ! --------------
14013 ! GET THE PROCEDURE FILE FROM MASS STORAGE
14020 SUB Getp(O$(*))
14030 ASSIGN @File1 TO "PROCF"
14040 ENTER @File1;O$(*)
14050 SUBEND
14051 ! LIST SUBROUTINE
14052 ! ---------------
14053 ! LIST ALL THE LINES IN THE PROCEDURE FILE
14060 SUB Listp(O$(*))
14070 IMAGE 2X,30A
14080 IMAGE 4X,DD,4X,60A
14090 PRINT FNLin$(3)
14100 PRINT USING 14070;"     *PROCEDURE FILE*"
14110 PRINT
14120 PRINT USING 14070;"Line #       Command"
14130 FOR Z=1 TO 40
14140 IF O$(Z)[1,4]="DONE" THEN SUBEXIT
14150 PRINT USING 14080;Z,O$(Z)
14160 NEXT Z
14170 SUBEND
14171 ! WRITE SUBROUTINE
14172 ! ----------------
14173 ! WRITE STATEMENTS ON THE PROCEDURE FILE
14180 SUB Write(O$(*))
14190 FOR Z=1 TO 40
14200 INPUT "COMMAND",O$(Z)
14210 IF O$(Z)[1,4]="DONE" THEN SUBEXIT
14220 NEXT Z
14230 SUBEND
14231 ! INSERT LINE SUBROUTINE
14232 ! ----------------------
14233 ! INSERT LINE IN THE PROCEDURE FILE
14240 SUB Insert(R_(*),A$,O$(*))
14250 CALL Dcod(Temp,R_(32),R_(33),0,0,0,A$,B$,0)
14260 IF O$(40)[1,4]="DONE" THEN O$(39)="DONE"
14270 FOR Z=39 TO Temp STEP -1
14280 O$(Z+1)=O$(Z)
14290 NEXT Z
14300 INPUT "COMMAND",O$(Temp)
14310 SUBEND
14311 ! DELETE LINE SUBROUTINE
14312 ! ----------------------
14313 ! DELETE A LINE FROM THE PROCEDURE FILE
14320 SUB Delete(R_(*),A$,O$(*))
14330 CALL Dcod(Temp1,R_(32),R_(33),0,0,0,A$,B$,0)
14340 FOR Z=Temp1 TO 39
14350 O$(Z)=O$(Z+1)
14360 NEXT Z
14370 SUBEND
14371 ! ------------------DEVICES MODELLING-----------------------------
14372 !
```

```
14380 ! RESISTOR DEVICE MODEL
14381 ! --------------------
14390 SUB Rstr(Ra,Rb$(*),A_(*),B(*),Q,Re(*))
14400 COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
14410 COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
14420 FOR Z=1 TO Ra
14430 N1=Re(Z,1)
14440 N2=Re(Z,2)
14450 Ro=Re(Z,3)
14460 T1=Re(Z,4)
14470 T2=Re(Z,5)
14480 IF (T1<>0) OR (T2<>0) THEN Ro=Ro/(1+(Q-300)*(T1+(Q-300)*T2))
14490 IF N1<>0 THEN CALL Add_elem(N1,N1,Ro)
14500 IF N2<>0 THEN CALL Add_elem(N2,N2,Ro)
14510 IF (N1<>0) AND (N2<>0) THEN CALL Add_elem(N1,N2,-Ro)
14520 IF (N1<>0) AND (N2<>0) THEN CALL Add_elem(N2,N1,-Ro)
14530 NEXT Z
14540 SUBEND
14541 ! CAPACITOR DEVICE MODEL
14542 ! ---------------------
14550 SUB Capr(C,Cb$(*),Lb$(*),A_(*),B(*),Fl(*),Q_(*),N_(*),T,U,C_(*),Ce(*))
14560 COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
14570 COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
14580 FOR Z=1 TO C
14590 N1=Ce(Z,3)
14600 N2=Ce(Z,4)
14610 Icn=C_(Z,1)
14620 Gc=C_(Z,2)
14630 IF Fl(10)<>0 THEN GOTO Caps
14640 Cm=Ce(Z,5)
14650 V1=0
14660 V2=0
14670 Va=0
14680 Vb=0
14690 IF N1<>0 THEN V1=Q_(N1)
14700 IF N1<>0 THEN Va=N_(N1)
14710 IF N2<>0 THEN V2=Q_(N2)
14720 IF N2<>0 THEN Vb=N_(N2)
14730 Vc=V1-V2
14740 Ovc=Va-Vb
14750 IF (Fl(5)<>0) AND (Ce(Z,1)<>0) THEN Ovc=Ce(Z,2)
14760 IF (Fl(5)<>0) AND (Ce(Z,1)<>0) THEN Vc=Ce(Z,2)
14770 Cap=Ce(Z,6)
14780 IF Cm=0 THEN GOTO Cape
14790 Cex=Le(Cm,2)
14800 Cpa=Le(Cm,1)
14810 Phi=Le(Cm,3)
14820 Cap=Cap*Cpa/(1-Vc/Phi)^Cex
14830 Cape:CALL Int8(Cap,Vc,Ovc,Icn,Gc,Ic,R_(*),T,U)
14840 IF N1=0 THEN 14870
14850 CALL Add_elem(N1,N1,Gc)
14860 Rhs(N1)=Rhs(N1)+Ic
14870 IF N2=0 THEN 14930
14880 CALL Add_elem(N2,N2,Gc)
14890 Rhs(N2)=Rhs(N2)-Ic
14900 IF (N1=0) OR (N2=0) THEN 14930
```

```
14910 CALL Add_elem(N1,N2,-Gc)
14920 CALL Add_elem(N2,N1,-Gc)
14930 C_(Z,1)=Icn
14940 C_(Z,2)=Gc
14950 GOTO Cap1
14960 Caps:IF N1<>0 THEN Rhs(N1)=Rhs(N1)+Icn
14970 IF N2<>0 THEN Rhs(N2)=Rhs(N2)-Icn
14980 Cap1:  NEXT Z
14990 SUBEND
14991 ! VOLTAGE SOURCE DEVICE MODEL
14992 ! -------------------------
15000 SUB Vsrc(V,Vb$(*),Va_(*),Sb$(*),A_(*),B(*),Fl(*),T,U,Ve(*),Se(*))
15010 COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
15020 COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
15030 FOR Z=1 TO V
15040 Np=Ve(Z,1)
15050 Nm=Ve(Z,2)
15060 Vo=Va_(Z)
15070 IF Fl(4)=1 THEN GOTO Supv
15080 GOTO Lodv
15090 Supv:Vm=Ve(Z,3)
15100 IF Vm<>0 THEN GOSUB Sup8
15110 Lodv:Rhs(N+Z)=Vo
15120 IF Fl(10)<>0 THEN GOTO Ldve
15130 IF Np<>0 THEN CALL Add_elem(Np,N+Z,1)
15140 IF Np<>0 THEN CALL Add_elem(N+Z,Np,1)
15150 IF Nm<>0 THEN
15160   CALL Add_elem(Nm,N+Z,-1)
15170   CALL Add_elem(N+Z,Nm,-1)
15180 END IF
15190 Ldve:NEXT Z
15200 SUBEXIT
15210 Sup8:  Ti=0     !UPDATE THE VOLTAGE SOURCE VALUE
15220 Th=0
15230 Tn=0
15240 Supt:  FOR Y=2 TO 20 STEP 2
15250 Th=Ti
15260 Ti=Se(Vm,Y+2)
15270 IF Ti=0 AND Y<>2 THEN 15310
15280 IF Ti+Tn>T+U THEN GOTO Sul8
15290 IF Ti+Tn=T+U THEN GOTO Exac
15300 NEXT Y
15310 Tn=Tn+Th
15320 GOTO Supt
15330 Sul8:  Vn=Se(Vm,Y+3)
15340 Vp=Se(Vm,Y+1)
15350 Tp=Se(Vm,Y)
15360 Vo=Vn-(Vn-Vp)*(Ti+Tn-T-U)/(Ti-Tp)
15370 RETURN
15380 Exac:Vo=Se(Vm,Y+3)
15390 RETURN
15400 SUBEND
15401 ! CURRENT SOURCE DEVICE
15402 ! -------------------
15410 SUB Isrc(I,Ib$(*),Ia(*),Sb$(*),A_(*),B(*),Fl(*),T,U,Ie(*),Se(*))
15420 COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
```

```
15430 COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
15440 FOR Z=1 TO I
15450 Nf=Ie(Z,1)
15460 Nt=Ie(Z,2)
15470 Io=Ia(Z)
15480 IF Fl(4)=1 THEN GOTO Supi
15490 GOTO Lodi
15500 Supi:  Im=Ie(Z,3)
15510 IF Im<>0 THEN GOSUB Sup8
15520 Lodi:IF Nf<>0 THEN Rhs(Nf)=Rhs(Nf)-Io
15530 IF Nt<>0 THEN Rhs(Nt)=Rhs(Nt)+Io
15540 NEXT Z
15550 SUBEXIT
15560 Sup8:  Ti=0   ! UPDATE THE CURRENT SOURCE VALUE
15570 Th=0
15580 Tn=0
15590 Supt:  FOR Y=2 TO 10 STEP 2
15600 Th=Ti
15610 Ti=Se(Im,Y+2)
15620 IF Ti=0 THEN 15660
15630 IF Ti+Tn>T+U THEN GOTO Sul8
15640 IF Ti+Tn=T+U THEN GOTO Exac
15650 NEXT Y
15660 Tn=Tn+Th
15670 GOTO Supt
15680 Sul8:In=Se(Im,Y+3)
15690 Ip=Se(Im,Y+1)
15700 Tp=Se(Im,Y)
15710 Io=In-(In-Ip)*(Ti+Tn-T-U)/(Ti-Tp)
15720 RETURN
15730 Exac:Io=Se(Im,Y+3)
15740 RETURN
15750 SUBEND
15751 ! DIODE DEVICE MODEL
15752 ! ------------------
15760 SUB Diod(D,Db$(*),Eb$(*),A_(*),B(*),O(*),Fl(*),W,De(*),Ee(*))
15770 COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
15780 COM /Blk7/ N,Lptor,Min_index,Newrow(*),Newcol(*),R_(*)
15790 FOR Z=1 TO D
15800 Np=De(Z,1)
15810 Nm=De(Z,2)
15820 Ido=De(Z,6)
15830 Gm=De(Z,7)
15840 IF Fl(10)<>0 THEN GOTO Daps
15850 Mn=1!De(Z,3)
15860 Ad=De(Z,4)
15870 Is=Ad*Ee(1,1)
15880 Vp=0
15890 Vm=0
15900 IF Np<>0 THEN Vp=O(Np)
15910 IF Nm<>0 THEN Vm=O(Nm)
15920 Vd=Vp-Vm
15930 Ovd=De(Z,5)
15940 IF (Ovd>=0) OR (Vd>=0) THEN 15970
15950 CALL Limt(Vd,Ovd,Vd-Ovd,5,0,50,R_(*),T,U)
15960 GOTO Deqn
```

```
15970 Id=Ido+Is*Gm
15980 P7=W*Id/(Is*EXP(Ovd/(1.02*R_(28)))/(1.02*R_(28)))
15990 P7=FNMax((P7),.1)
16000 CALL Limt(Vd,Ovd,Vd-Ovd,P7,-.1,50,R_(*),T,U)
16010 Deqn: !
16020 De(Z,5)=Vd
16030 Id=Is*(EXP(Vd/(1.02*R_(28)))-1)
16040 Gd=Id/(1.02*R_(28))
16050 Ido=Id-Vd*Gd
16060 De(Z,6)=Ido
16070 De(Z,7)=Gd
16080 IF Np=0 THEN 16110
16090 CALL Add_elem(Np,Np,Gd)
16100 Rhs(Np)=Rhs(Np)-Ido
16110 IF Nm=0 THEN GOTO Dioe
16120 CALL Add_elem(Nm,Nm,Gd)
16130 Rhs(Nm)=Rhs(Nm)+Ido
16140 IF Np=0 THEN GOTO Dioe
16150 CALL Add_elem(Np,Nm,-Gd)
16160 CALL Add_elem(Nm,Np,-Gd)
16170 GOTO Dioe
16180 Daps: IF Np<>0 THEN Rhs(Np)=Rhs(Np)-Ido
16190 IF Nm<>0 THEN Rhs(Nm)=Rhs(Nm)+Ido
16200 Dioe: NEXT Z
16210 SUBEND
16211 ! MOSFET DEVICE MODLE
16212 ! ------------------
16220  SUB Mosf(M,Mb$(*),Fb$(*),A_(*),B(*),O(*),T,U,F1(*),W,Q_(*),N_(*),M_(
16230  COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
16240  COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
16250  FOR Z=1 TO M
16260  Nd=Me(Z,16) !DRAIN NODE
16270  Ng=Me(Z,17)  !GATE NODE
16280  Ns=Me(Z,18)  !SOURCE NODE
16290  Nb=Me(Z,19)  !BULK NODE
16300  Mn=Me(Z,20)  !MODEL #
16310  Tt=Fe(Mn,1)  !TYPE N OR P
16320  Vgs=0  !INITIALIZE
16330  Vds=0
16340  Vsb=0
16350  Vb=0
16360  IF Nb<>0 THEN Vb=O(Nb)  !VB
16370  IF Ns<>0 THEN Vsb=O(Ns) !VS
16380  IF Ng<>0 THEN Vgs=O(Ng) !VG
16390  IF Nd<>0 THEN Vds=O(Nd) !VD
16400  Vgs=Tt*(Vgs-Vsb)        !VGS
16410  Vds=Tt*(Vds-Vsb)        !VDS
16420  Vsb=Tt*(Vsb-Vb)         !VSB
16430 IF (F1(5)<>0) AND (Me(Z,31)<>0) THEN GOTO Uicm
16440 Ovgs=M_(Z,4)             !OLD VGS
16450 Ovds=M_(Z,5)             !OLD VDS
16460 Ovsb=M_(Z,6)             !OLD VSB
16470 Dvgs=Vgs-Ovgs            !DELTA VGS
16480 Dvds=Vds-Ovds            !DELTA VDS
16490 Dvsb=Vsb-Ovsb            !DELTA VSB
16500 Ovt=Tt*M_(Z,1)
```

```
16510 Ovsa=M_(Z,2)
16520 Oids=Tt*M_(Z,3)
16530 Gds=M_(Z,12)
16540 Gm=M_(Z,13)
16550 GOTO Check
16560 Vicm:Ovgs=Me(Z,28)
16570 Vgs=Ovgs
16580 Ovds=Me(Z,29)
16590 Vds=Ovds
16600 Ovsb=Me(Z,30)
16610 Vsb=Ovsb
16620 Dvgs=0
16630 Dvds=0
16640 Dvsb=0
16650 GOTO Leng
16660 Check:Fl(9)=0
16670 Leng:  Wc=Me(Z,25)        !WIDTH
16680 Lc=Me(Z,24)               !LENGTH
16690 Ad=Me(Z,26)               !A D
16700 As=Me(Z,27)               !A S
16710 Modl:  Vto=Fe(Mn,2)       !VTO
16720 Kp=Fe(Mn,3)               !K'
16730 Gam=Fe(Mn,4)              !GAM
16740 Phi=Fe(Mn,5)              !PHI
16750 Lam=Fe(Mn,6)              !LAM
16760 Is=Fe(Mn,13)              !IS
16770 Cox=Fe(Mn,12)             !COX
16780 Ldi=Fe(Mn,22)             !LDI
16790 Bjc=Fe(Mn,25)             !BJC
16800 Vbi=Fe(Mn,27)             !VBI
16810 Xd=Fe(Mn,26)              !XD
16820 Lc=Lc-2*Ldi               !L'
16830 IF (Ovds>0) OR (Ovds=0) AND (Vds>0) THEN GOTO Norm!FOREWARD BIASED
16840 Fl(9)=1
16850 Vgs=Vgs-Vds               !VGS TO VGD
16860 Ovgs=Ovgs-Ovds
16870 Dvgs=Dvgs-Dvds
16880 Vsb=Vsb+Vds               !VSB TO VDB
16890 Ovsb=Ovsb+Ovds
16900 Dvsb=Dvsb+Dvds
16910 Vds=-Vds                  !VDS TO VSD
16920 Ovds=-Ovds
16930 Dvds=-Dvds
16940 Norm:CALL Limy(Vgs,Ovgs,Dvgs,Ovt,R_(*),U)
16950 CALL Limt(Vsb,Ovsb,Dvsb,5,0,50,R_(*),T,U)
16960 IF (Nd-Ng=0) OR (Ng-Ns=0) AND (Fl(9)<>0) THEN Vds=Vgs
16970 IF (Nd-Ng=0) OR (Ng-Ns=0) AND (Fl(9)<>0) THEN GOTO Dcmd
16980 IF Ovgs>Ovt THEN GOTO Lili
16990 CALL Limt(Vds,Ovds,Dvds,5,0,50,R_(*),T,U)
17000 GOTO Dcmd
17010 Lili:IF Vds>1.001*Ovsa THEN GOTO Lisa
17020 Va=0
17030 IF Ovgs-Ovt-Ovds<>0 THEN Va=W*Oids/(Kp*(Vgs-Ovt-Ovds))
17040 IF Va<R_(17) THEN Va=R_(17)
17050 CALL Limt(Vds,Ovds,Dvds,Va,0,50,R_(*),T,U)
17060 GOTO Dcmd
```

```
17070 Lisa:Va=0
17080 Va=W*Oids/(Kp*(Vgs-Ovt))
17090 IF Va<R_(17) THEN Va=R_(17)
17100 Va=Ovsa-Va-1.E-3
17110 CALL Limt(Vds,Ovds,Dvds,5,Va,50,R_(*),T,U)
17120 Dcmd:P37=0
17130 IF Fl(1)=0 THEN GOTO Vthr !SHORT CHANNEL CHECK
17140 Xj=Fe(Mn,23)!XJ
17150 P37=Xd*SQR(Vsb+Phi)
17160 P37=Xj*(SQR(1+2*P37/Xj)-1)/Lc
17170 Vthr:Ovt=Vto+Gam*(1-P37)*(SQR(Vsb+Phi)-SQR(Phi)) !VT
17180 M_(Z,1)=Ovt  !STORE VT
17190 Ovt=Tt*Ovt
17200 P39=0
17210 IF (Fl(3)=0) OR (Cox=0) THEN GOTO Vlow!SUBTHRESH CHECK
17220 P39=R_(24)*Fe(Mn,21)  !NFS*Q
17230 P40=Gam/SQR(4*(Vsb+Phi))
17240 P39=R_(28)*(Cox*P40+P39+Cox)/Cox
17250 Vlow:P40=Ovt+P39  !=VT+DEL1
17260 IF Vgs<P40 THEN GOTO Subt!OFF
17270 GOSUB Idsg
17280 GOTO Junc
17290 Subt: IF Fl(3)<>0 THEN GOTO Subv !SUBTHRESHOLD
17300 Oids=0
17310 Gds=0
17320 Gm=0
17330 Gmb=0
17340 GOTO Junc
17350 Subv:P54=Vgs
17360 Vgs=P40
17370 GOSUB Idsg
17380 Vgs=P54
17390 P54=EXP((Vgs-P40)/P39)
17400 Gds=Gds*P54
17410 Gm=P54*(Gm+Oids/P39)
17420 Gmb=Gmb*P54
17430 Oids=Oids*P54
17440 Junc:IF Fl(9)<>0 THEN GOTO Jinv
17450 Gsb=Is*As/R_(28)
17460 Gdb=Is*Ad/R_(28)
17470 GOTO Mobl
17480 Jinv: Gsb=Is*Ad/R_(28)
17490 Gdb=Is*As/R_(28)
17500 Mobl: Eex=Fe(Mn,19)  !C FIELD EX
17510 P40=1
17520 IF Eex=0 THEN GOTO Chnl
17530 Vcr=Fe(Mn,18)
17540 Ttr=Fe(Mn,20)
17550 IF Vcr>Vgs-Ovt-Ttr*Vds THEN GOTO Chnl
17560 P40=(Vcr/(Vgs-Ovt-Ttr*Vds))^Eex
17570 Chnl: P60=0
17580 IF (Lam<>0) OR (Fl(2)=0) THEN GOTO Gdsl
17590 P59=(Vds-Ovsa)/4
17600 P60=SQR(P59*P59+1)
17610 P61=SQR(P59+P60)
17620 IF Vds<1.E-5 THEN Kp=0
```

```
17630 IF Vds<1.E-5 THEN GOTO Gdsl
17640 Lam=Xd*P61/(Lc*Vds)
17650 P60=Xd*(1+P59/P60)/(8*Lc*Vds*P61)
17660 Gdsl:Gds=Gds+P60*Vds*Oids/(1-Lam*Vds)
17670 Beta:Kp=Kp*P40*Wc/(Lc*(1-Lam*Vds))
17680 Oids=Oids*Kp*Tt
17690 M_(Z,3)=Oids
17700 Gds=Gds*Kp+1.E-8
17710 M_(Z,12)=Gds
17720 Gm=Gm*Kp
17730 M_(Z,13)=Gm
17740 Gmb=Gmb*Kp
17750 Oids=(Oids*Tt-Gm*Vgs-Gds*Vds-Gmb*Vsb)*Tt
17760 Lodd:IF (Ns<>0) AND (Nb<>0) THEN CALL Add_elem(Nb,Ns,-Gsb)
17770 IF (Nd<>0) AND (Nb<>0) THEN CALL Add_elem(Nb,Nd,-Gdb)
17780 IF Nb<>0 THEN CALL Add_elem(Nb,Nb,-Gdb)
17790 IF Fl(9)<>0 THEN GOTO Ldin
17800 IF (Nd<>0) AND (Ng<>0) THEN CALL Add_elem(Nd,Ng,Gm)
17810 IF (Ng<>0) AND (Ns<>0) THEN CALL Add_elem(Ns,Ng,-Gm)
17820 IF (Nd<>0) AND (Ns<>0) THEN CALL Add_elem(Nd,Ns,-Gm+Gmb-Gds)
17830 IF (Nd<>0) AND (Ns<>0) THEN CALL Add_elem(Ns,Nd,-Gds)
17840 IF (Ns<>0) AND (Nb<>0) THEN CALL Add_elem(Ns,Nb,Gmb-Gsb)
17850 IF (Nd<>0) AND (Nb<>0) THEN CALL Add_elem(Nd,Nb,-Gmb-Gdb)
17860 IF Nd<>0 THEN CALL Add_elem(Nd,Nd,Gds+Gdb)
17870  IF Nd<>0 THEN Rhs(Nd)=Rhs(Nd)-Oids
17880  IF Ns<>0 THEN CALL Add_elem(Ns,Ns,Gm-Gmb+Gds+Gsb)
17890  IF Ns<>0 THEN Rhs(Ns)=Rhs(Ns)+Oids
17900  GOTO Acan
17910 Ldin:  IF (Nd<>0) AND (Ng<>0) THEN CALL Add_elem(Nd,Ng,-Gm)
17920  IF (Ng<>0) AND (Ns<>0) THEN CALL Add_elem(Ns,Ng,Gm)
17930  IF (Nd<>0) AND (Ns<>0) THEN CALL Add_elem(Nd,Ns,-Gds)
17940  IF (Nd<>0) AND (Ns<>0) THEN CALL Add_elem(Ns,Nd,-Gm+Gmb-Gds)
17950  IF (Ns<>0) AND (Nb<>0) THEN CALL Add_elem(Ns,Nb,-Gmb-Gsb)
17960  IF (Nd<>0) AND (Nb<>0) THEN CALL Add_elem(Nd,Nb,Gmb-Gdb)
17970  IF Nd<>0 THEN CALL Add_elem(Nd,Nd,Gm-Gmb+Gds+Gdb)
17980  IF Nd<>0 THEN Rhs(Nd)=Rhs(Nd)+Oids
17990  IF Ns<>0 THEN CALL Add_elem(Ns,Ns,Gds+Gsb)
18000  IF Ns<>0 THEN Rhs(Ns)=Rhs(Ns)-Oids
18010 Acan:  IF Fl(4)<>1 THEN GOTO Putv  !****CHANGED
18020  Cgs=Fe(Mn,7) !CGSOVL
18030  Cgd=Fe(Mn,8) !CGDOVL
18040  Cgb=Fe(Mn,9) !CGB
18050  Cdb=Fe(Mn,10)!CDB
18060  Csb=Fe(Mn,11) !CSB
18070  Pb=Fe(Mn,14) !PB
18080  IF Pb=0 THEN Pb=.87
18090  P40=Vgs-Vds   !VGD
18100  Vdb=Vds+Vsb!VDB
18110  Vgb=P40+Vdb  !VGB
18120  IF (Cgs<=0) AND (Cgd<=0) AND (Cgb<=0) AND (Cdb<=0) AND (Csb<=0) AND
18130  Cgs=Wc*Cgs !COVL*W
18140  Cgd=Cgd*Wc  !COVL*W
18150  Cgb=Cgb*Wc*Lc !CGB*W*L
18160  Cdb=Cdb*Ad !CDB
18170  Csb=Csb*As !CSB
18180  IF Fl(9)=0 THEN GOTO Ecox
```

```
18190  P40=Cgs
18200  Cgs=Cgd
18210  Cgd=P40
18220 Ecox:  IF Cox=0 THEN GOTO Cjunc !COX
18230  Cox=Cox*Wc*Lc*1.E+4
18240 IF Vgs>=Ovt-Phi THEN GOTO Cut1
18250 Cgb=Cgb+Cox
18260 GOTO Cjunc
18270 Cut1:  IF Vgs>Ovt-Phi/2 THEN GOTO Cut2
18280 Cgb=Cgb+Cox*(Ovt-Vgs)/Phi
18290 GOTO Cjunc
18300 Cut2: IF Vgs>=Ovt THEN GOTO Sat1
18310 Cgb=Cgb+Cox*(Ovt-Vgs)/Phi
18320 Cgs=Cgs+2*Cox*(Vgs-Ovt+Phi/2)/(1.5*Phi)
18330 GOTO Cjunc
18340 Sat1: IF Vgs>=Ovt+Vds THEN GOTO Lin1
18350 Cgs=Cgs+2*Cox/3
18360 GOTO Cjunc
18370 Lin1:  P45=Ovsa-Vgs
18380 P47=2*Ovsa-Vgs
18390 Cgs=Cgs+2*Cox*(1-P45*P45/(P47*P47))/3
18400 Cgd=Cgd+2*Cox*(1-Ovsa*Ovsa/(P47*P47))/3
18410 Cjunc: IF (Vdb/Pb<=-1) OR (Vsb/Pb<=-1) THEN GOTO Cevl
18420 IF F1(9)<>0 THEN 18460
18430 Cdb=Cdb/(1+Vdb/Pb)^Bjc
18440 Csb=Csb/(1+Vsb/Pb)^Bjc
18450 GOTO Cevl
18460 Cdb=Cdb/(1+Vsb/Pb)^Bjc
18470 Csb=Csb/(1+Vdb/Pb)^Bjc
18480 Cevl: Vdq=0
18490 Vgq=0
18500 Vsq=0
18510 Vbq=0
18520 Vdn=0
18530 Vgn=0
18540 Vsn=0
18550 Vbn=0
18560 IF Nd<>0 THEN Vdq=Q_(Nd)
18570 IF Nd<>0 THEN Vdn=N_(Nd)
18580 IF Ng<>0 THEN Vgq=Q_(Ng)
18590 IF Ng<>0 THEN Vgn=N_(Ng)
18600 IF Ns<>0 THEN Vsq=Q_(Ns)
18610 IF Ns<>0 THEN Vsn=N_(Ns)
18620 IF Nb<>0 THEN Vbq=Q_(Nb)
18630 IF Nb<>0 THEN Vbn=N_(Nb)
18640 Dvgsq=Tt*(Vgq-Vsq)
18650 Dvgsn=Tt*(Vgn-Vsn)
18660 Dvgdq=Tt*(Vgq-Vdq)
18670 Dvgdn=Tt*(Vgn-Vdn)
18680 Dvgbq=Tt*(Vgq-Vbq)
18690 Dvgbn=Tt*(Vgn-Vbn)
18700 Dvdbq=Tt*(Vdq-Vbq)
18710 Dvdbn=Tt*(Vdn-Vbn)
18720 Dvsbq=Tt*(Vsq-Vbq)
18730 Dvsbn=Tt*(Vsn-Vbn)
18740 Oi=M_(Z,7)
```

```
18750 CALL Int8(Cgs,Dvgsq,Dvgsn,Oi,Gcgs,Icgs,R_(*),T,U)
18760 M_(Z,7)=Oi
18770 Oi=M_(Z,8)
18780 CALL Int8(Cgd,Dvgdq,Dvgdn,Oi,Gcgd,Icgd,R_(*),T,U)
18790 M_(Z,8)=Oi
18800 Oi=M_(Z,11)
18810 CALL Int8(Cgb,Dvgbq,Dvgbn,Oi,Gcgb,Icgb,R_(*),T,U)
18820 M_(Z,11)=Oi
18830 IF Fl(9)<>0 THEN GOTO Invc
18840 Oi=M_(Z,10)
18850 CALL Int8(Cdb,Dvdbq,Dvdbn,Oi,Gcdb,Icdb,R_(*),T,U)
18860 M_(Z,10)=Oi
18870 Oi=M_(Z,9)
18880 CALL Int8(Csb,Dvsbq,Dvsbn,Oi,Gcsb,Icsb,R_(*),T,U)
18890 M_(Z,9)=Oi
18900 GOTO Lodc
18910 Invc:Oi=M_(Z,10)
18920 CALL Int8(Cdb,Dvdbq,Dvdbn,Oi,Gcdb,Icdb,R_(*),T,U)
18930 M_(Z,10)=Oi
18940 Oi=M_(Z,9)
18950 CALL Int8(Csb,Dvsbq,Dvsbn,Oi,Gcsb,Icsb,R_(*),T,U)
18960 M_(Z,9)=Oi
18970 Lodc:IF (Ng<>0) AND (Nb<>0) THEN CALL Add_elem(Ng,Nb,-Gcgb)
18980 IF (Ng<>0) AND (Nb<>0) THEN CALL Add_elem(Nb,Ng,-Gcgb)
18990 IF (Ns<>0) AND (Nb<>0) THEN CALL Add_elem(Ns,Nb,-Gcsb)
19000 IF (Ns<>0) AND (Nb<>0) THEN CALL Add_elem(Nb,Ns,-Gcsb)
19010 IF (Nd<>0) AND (Nb<>0) THEN CALL Add_elem(Nd,Nb,-Gcdb)
19020 IF (Nd<>0) AND (Nb<>0) THEN CALL Add_elem(Nb,Nd,-Gcdb)
19030 IF Ng<>0 THEN CALL Add_elem(Ng,Ng,Gcgb+Gcgd+Gcgs)
19040 IF Ng<>0 THEN Rhs(Ng)=Rhs(Ng)+(Icgs+Icgd+Icgb)*Tt
19050 IF Nb<>0 THEN CALL Add_elem(Nb,Nb,Gcgb+Gcdb+Gcsb)
19060 IF Nb<>0 THEN Rhs(Nb)=Rhs(Nb)-(Icgb+Icdb+Icsb)*Tt
19070 IF Fl(9)<>0 THEN GOTO Incl
19080 IF (Nd<>0) AND (Ng<>0) THEN CALL Add_elem(Ng,Nd,-Gcgd)
19090 IF (Nd<>0) AND (Ng<>0) THEN CALL Add_elem(Nd,Ng,-Gcgd)
19100 IF (Ng<>0) AND (Ns<>0) THEN CALL Add_elem(Ng,Ns,-Gcgs)
19110 IF (Ng<>0) AND (Ns<>0) THEN CALL Add_elem(Ns,Ng,-Gcgs)
19120 IF Nd<>0 THEN CALL Add_elem(Nd,Nd,Gcgd+Gcdb)
19130 IF Nd<>0 THEN Rhs(Nd)=Rhs(Nd)-(Icgd-Icdb)*Tt
19140 IF Ns<>0 THEN CALL Add_elem(Ns,Ns,Gcgs+Gcsb)
19150 IF Ns<>0 THEN Rhs(Ns)=Rhs(Ns)-(Icgs-Icsb)*Tt
19160 GOTO Putv
19170 Incl:IF (Nd<>0) AND (Ng<>0) THEN CALL Add_elem(Ng,Nd,-Gcgs)
19180 IF (Nd<>0) AND (Ng<>0) THEN CALL Add_elem(Nd,Ng,-Gcgs)
19190 IF (Ng<>0) AND (Ns<>0) THEN CALL Add_elem(Ng,Ns,-Gcgd)
19200 IF (Ng<>0) AND (Ns<>0) THEN CALL Add_elem(Ns,Ng,-Gcgd)
19210 IF Nd<>0 THEN CALL Add_elem(Nd,Nd,Gcgs+Gcdb)
19220 IF Nd<>0 THEN Rhs(Nd)=Rhs(Nd)-(Icgs-Icdb)*Tt
19230 IF Ns<>0 THEN CALL Add_elem(Ns,Ns,Gcgd+Gcsb)
19240 IF Ns<>0 THEN Rhs(Ns)=Rhs(Ns)-(Icgd-Icsb)*Tt
19250 Putv:IF Fl(9)=0 THEN GOTO Pute
19260 Vgs=Vgs-Vds
19270 Vsb=Vsb+Vds
19280 Vds=-Vds
19290 Pute:M_(Z,4)=Vgs
19300 M_(Z,5)=Vds
```

```
19310 M_(Z,6)=Vsb
19320 Fl(9)=0
19330 NEXT Z
19340 GOTO End
19350 Idsg:P41=Gam*(1-P37)    !GAMMA PRIME
19360 P44=P41*P41 !GAMMA SQUARED
19370 IF P44=0 THEN P42=1.E-20
19380 IF P44=0 THEN 19420
19390 IF 1+4*(Vgs-Vbi+Phi+Vsb)/P44<0 THEN P42=1.E-20
19400 IF 1+4*(Vgs-Vbi+Phi+Vsb)/P44<0 THEN 19420
19410 P42=SQR(1+4*(Vgs-Vbi+Phi+Vsb)/P44)
19420 P43=P44*(1-P42)/2
19430 P45=Vgs-Vbi
19440 Ovsa=P45+P43
19450 IF Ovsa<=0 THEN Ovsa=0
19460 M_(Z,2)=Ovsa
19470 P47=Vsb+Phi
19480 P48=Vds+P47
19490 IF Vds>Ovsa THEN GOTO Satn
19500 Linr:IF P47<0 THEN P47=0
19510 IF P48<0 THEN P48=0
19520 P49=2*P41*(P48^1.5-P47^1.5)/3
19530 Oids=Vds*(P45-Vds/2)-P49
19540 Gds=P45-Vds-P41*SQR(P48)
19550 Gm=Vds
19560 Gmb=-P41*(SQR(P48)-SQR(P47))
19570 RETURN
19580 Satn:IF (P47>=0) AND (Ovsa+P47>=0) THEN 19610
19590 P47=0
19600 Ovsa=0
19610 P49=2*P41*((Ovsa+P47)^1.5-P47^1.5)/3
19620 Oids=(P45-Ovsa/2)*Ovsa-P49
19630 Gds=0
19640 P44=P45-Ovsa-P41*SQR(Ovsa+P47)
19650 Gm=Ovsa-(1-P42)*P44/P42
19660 Gmb=P41*(SQR(Ovsa+P47)-SQR(P47))-P44/P42
19670 RETURN
19680 End:SUBEND
19681 !
19690 SUB Limt(P1,P2,P3,P4,P5,P6,R_(*),T,U)
19700 IF (ABS(P3)<P4) OR (R_(26)=1) AND (U-R_(27)<>0) THEN 19720
19710 P1=P2+P4*SGN(P1-P2)
19720 IF P1<P5 THEN P1=P5
19730 IF P1>P6 THEN P1=P6
19740 SUBEND
19750 SUB Int8(P1,P2,P3,P4,P5,P6,R_(*),T,U)
19760 IF P1<=0 THEN
19770   Temp=0
19780   P5=Temp
19790   P6=Temp
19800 END IF
19810 IF P1<=0 THEN SUBEXIT
19820 IF T=0 THEN GOTO Beul
19830 IF R_(26)=0 THEN P4=2*P1*(P2-P3)/R_(27)-P4
19840 P5=2*P1/U
19850 P6=P2*P5+P4
```

BIASB.36 LISTING

```
19860 SUBEXIT
19869 Beul:IF U=0 THEN U=1.E-12
19870       P5=P1/U
19880 P6=P2*P5
19890 SUBEND
19891 !
19900 SUB Chkf(M,Mb$(*),Fb$(*),Fl(*),T,U,Me(*),M_(*))
19910 FOR Z=1 TO M
19920 Mn=Me(Z,20)
19930 Tt=Fe(Mn,1)
19940 IF Fl(8)=0 THEN GOTO Chkc
19950 Sat=Me(Z,21)
19960 Lin=SGN(M_(Z,2)*Tt-M_(Z,5))
19970 Lin=SGN(Lin+1)
19980 IF Sat-Lin<>0 THEN GOTO Chkc
19990 Me(Z,21)=ABS(Lin-1)
20000 A$="  Linear   "
20010 IF Lin<>0 THEN A$="Saturation"
20020 IF T-U<>0 THEN GOSUB Prtm
20030 Chkc:IF Fl(6)=0 THEN GOTO Chkp
20040 Off=Me(Z,23)
20050 Lin=SGN(M_(Z,1)-Tt*M_(Z,4))*Tt
20060 Lin=SGN(Lin+1)
20070 IF Off-Lin<>0 THEN GOTO Chkb
20080 Me(Z,23)=ABS(Lin-1)
20090 A$="  Linear   "
20100 IF Lin<>0 THEN A$="  Cutoff   "
20110 IF T-U<>0 THEN GOSUB Prtm
20120 Chkb:IF M_(Z,6)<-.2 THEN A$="Source"
20130 IF M_(Z,6)<-.2 THEN GOSUB Prtn
20140 IF M_(Z,6)+M_(Z,5)<-.2 THEN A$="Drain"
20150 IF M_(Z,6)+M_(Z,5)<-.2 THEN GOSUB Prtn
20160 NEXT Z
20170 SUBEXIT
20180 Prtm:  IMAGE 2X,4A,17A,10A,10A,MD.DDE,X,3A
20190 PRINT USING 20180;Mb$(Z)[1,4]," has entered the ",A$," region at ",T-
20200 RETURN
20210 Prtn:  IMAGE 2X,4A,3A,6A,21A,MD.DDE,X,3A
20220 PRINT USING 20210;Mb$(Z)[1,4],"'s ",A$," is forward biased at ",T-U,"
20230 RETURN
20240 SUBEND
20241 !
20250 SUB Limi(Vn,Ov,Dv,G,Moi,Mii,R_(*),T,U)
20260 IF (R_(26)=1) AND (U-R_(27)<>0) THEN SUBEXIT
20280 IF Moi<Mii THEN Moi=Mii
20290 IF G=0 THEN SUBEXIT
20300 CALL Min2(ABS(Moi/G),ABS(Dv),Dd)
20310 Ndvn=SGN(Vn-Ov)*Dd
20320 Vn=Ov+Ndvn
20330 IF SGN(Vn)<>SGN(Ov) THEN Vn=0
20340 SUBEND
20341 !
20350 SUB Limy(Vnew,Vold,Delv,Vto,R_(*),U)
20360 IF (R_(26)=1) AND (U-R_(27)<>0) THEN SUBEXIT
20370 Vtsthi=ABS(2*(Vold-Vto))+2
20380 Vtstlo=Vtsthi/2+2
```

```
20390 Vtox=Vto+3.5
20400 IF Vold<Vto THEN 20570
20410 IF Vold<Vtox THEN 20520
20420 IF Delv>0 THEN 20490
20430 IF Vnew<Vtox THEN 20470
20440 IF -Delv<=Vtstlo THEN SUBEXIT
20450 Vnew=Vold-Vtstlo
20460 SUBEXIT
20470 IF Vnew<Vto+2 THEN Vnew=Vto+2
20480 SUBEXIT
20490 IF Delv<Vtsthi THEN SUBEXIT
20500 Vnew=Vold+Vtsthi
20510 SUBEXIT
20520 IF Delv>0 THEN 20550
20530 IF Vnew<Vto-.5 THEN Vnew=Vto-.5
20540 SUBEXIT
20550 IF Vnew>Vto+4 THEN Vnew=Vto+4
20560 SUBEXIT
20570 IF Delv>0 THEN 20610
20580 IF -Delv<=Vtsthi THEN SUBEXIT
20590 Vnew=Vold-Vtsthi
20600 SUBEXIT
20610 Vtemp=Vto+.5
20620 IF Vnew>Vtemp THEN 20660
20630 IF Delv<=Vtstlo THEN SUBEXIT
20640 Vnew=Vold+Vtstlo
20650 SUBEXIT
20660 Vnew=Vtemp
20670 SUBEND
20680 ! -----------------COMMON SUBROUTINES---------------------
20681 !
20682 ! SUBROUTINE WHICH EXACTS THE NUMBER FROM ARRAY A$ OR FIND LOCATION OF
20690 SUB Dcod(P3,P1,P2,P4,P5,P6,A$,B$,K)
20700 IF P1<1 THEN P1=1
20710 P6=LEN(A$)
20720 IF P1<=P6 THEN 20790
20730 Temp=0
20740 P2=Temp
20750 P3=Temp
20760 P1=P1-1
20770 IF P1<1 THEN P1=1
20780 SUBEXIT
20790 P4=NUM(A$[P1,P1])
20800 IF (P4<>32) AND (P4<>61) AND (P4<>44) AND (P4<>40) THEN GOTO 20830
20810 P1=P1+1
20820 GOTO 20720
20830 IF (P4>47) AND (P4<58) OR (P4=45) OR (P4=43) OR (P4=46) THEN GOTO Dcnc
20840 P3=POS(B$,A$[P1,K+P1-1])
20850 P2=-P1
20860 GOTO Dcst
20870 Dcno:P5=P1
20880 P5=P5+1
20890 IF P5<=P6 THEN 20920
20900 P5=P5-1
20910 GOTO 20940
20920 P4=NUM(A$[P5,P5])
```

```
20930 IF (P4>47) AND (P4<58) OR (P4=45) OR (P4=43) OR (P4=46) OR (P4=69) THE
20940 P2=1
20950 P3=VAL(A$[P1,P5])
20960 IF P4=71 THEN P3=P3*1.E+9
20970 IF P4=77 THEN P3=P3*1.E+6
20980 IF P4=75 THEN P3=P3*1.E+3
20990 IF P4=109 THEN P3=P3*1.E-3
21000 IF P4=117 THEN P3=P3*1.E-6
21010 IF P4=110 THEN P3=P3*1.E-9
21020 IF P4=112 THEN P3=P3*1.E-12
21030 IF P4=102 THEN P3=P3*1.E-15
21040 Dcst: IF P1>P6 THEN SUBEXIT
21050 P4=NUM(A$[P1,P1])
21060 IF (P4=32) OR (P4=61) OR (P4=44) OR (P4=40) THEN SUBEXIT
21070 P1=P1+1
21080 GOTO Dcst
21090 SUBEND
21091 ! ERROR SUBROUTINE
21092 ! ---------------
21100 SUB Errr(P1,E,A$)
21110 BEEP
21120 IF E=1 THEN N$="UNKNOWN COMMAND"
21130 IF E=2 THEN N$="NEED STOP TIME"
21140 IF E=3 THEN N$="DC or TRANSIENT"
21150 IF E=4 THEN N$="CANNOT PLOT DC"
21160 IF E=5 THEN N$="NAME IS TOO LONG"
21170 IF E=6 THEN N$="NO SUCH ELEMENT"
21180 IF E=7 THEN N$="NEED NUMBER, #0"
21190 IF E=8 THEN N$="WHAT FILENAME"
21200 IF E=9 THEN N$="UNKNOWN PARAMETER"
21210 IF E=10 THEN N$="UNKNOWN OPTION"
21220 IF E=11 THEN N$="WRONG FILE TYPE"
21230 IF E=12 THEN N$="CANNOT FIND FILE"
21240 IF E=13 THEN N$="ILLEGAL MODEL NAME"
21250 IF E=14 THEN N$="ALREADY EXISTS"
21260 IF E=15 THEN N$="> MAX # OF ELEMENTS"
21270 IF E=16 THEN N$="> MAX # OF MODELS"
21280 IF P1<1 THEN P1=1
21290 IF P1>LEN(A$) THEN P1=LEN(A$)
21300 PRINT N$
21310 PRINT A$
21320 PRINT A$[1,P1]
21330 SUBEND
21331 ! SUBROUTINE EXTRACTS NAMES OF VARIABLE FROM A$ ARRAY
21332 ! -------------------------------------------------
21340 SUB Gnam(P1,P2,A$,N$)
21350 CALL Min2(4,LEN(A$[P1])-1,Dd)
21360 FOR Y=P1 TO P1+Dd
21370 P2=NUM(A$[Y,Y])
21380 IF (P2=32) OR (P2=61) OR (P2=44) OR (P2=40) THEN 21410
21390 N$[Y-P1+1,Y-P1+1]=A$[Y,Y]
21400 NEXT Y
21410 P2=Y-P1+1
21420 FOR Y=P2 TO 4
21430 N$[Y,Y]=" "
21440 NEXT Y
```

```
21450 SUBEND
21451 ! SUBROUTINE FINDS THE CORRESPONDING NODE INDEX
21453 ! ---------------------------------------
21460 SUB Gnod(P1,N,H(*))
21470 IF P1=0 THEN SUBEXIT
21480 IF P1>0 THEN 21520
21490 CALL Min2(ABS(P1),N,Dd)
21500 P1=H(Dd)
21510 SUBEXIT
21520 FOR Y=1 TO N
21530 IF P1<>H(Y) THEN 21560
21540 P1=Y
21550 SUBEXIT
21560 NEXT Y
21570 N=N+1
21580 H(N)=P1
21590 P1=N
21600 SUBEND
21601 ! SUBROUTINE EXTRACTS THE CORRESPONDING NO. OF AN INPUT
21602 ! ---------------------------------------------
21610 SUB Gnum(P1,P2,P3,P4,P5,P6,P7,A$,B$,R(*))
21620 DIM P(5)
21630 FOR X=1 TO ABS(P2)
21640 CALL Dcod(P(X),P1,R(33),0,0,0,A$,B$,0)
21650 IF (R(33)>0) OR (P2<=0) THEN 21690
21660 CALL Errr(P1,7,A$)
21670 P1=-1
21680 SUBEXIT
21690 IF R(33)<>0 THEN 21720
21700 P(X)=0
21710 P1=LEN(A$)+1
21720 NEXT X
21730 IF R(32)=0 THEN P1=0
21740 P3=P(1)
21750 P4=P(2)
21760 P5=P(3)
21770 P6=P(4)
21780 P7=P(5)
21790 SUBEND
21791 ! SUBROUTINE CALCULATES THE MOSFETS MODEL PARAMETERS
21792 ! ------------------------------------------
21800 SUB Mosu(F,Fb$(*),R(*),Q,Fe(*))
21810 FOR Z=1 TO F
21820 P26=Fe(Z,1) !N OR P CHAN
21830 P14=Fe(Z,15) !NSU
21840 IF P14=0 THEN GOTO Mobt
21850 P11=Fe(Z,12)    !COX
21860 P4=2*R(29)*LOG(P14/R(30))  !PHI
21870 Fe(Z,5)=P4
21880 P1=3.2
21890 P23=Fe(Z,24)!POLY DOPING
21900 IF P23=0 THEN 21920
21910 P1=3.25+R(18)/2-SGN(P23)*LOG(ABS(P23)/R(30))*P26*R(28)
21920 P1=P1-(3.25+R(18)/2+P26*P4/2) !Metel or Poly to Si diff
21930 P15=Fe(Z,16)!NSS
21940 P24=P1-P15*R(24)/P11+P4*P26
```

```
21950 Fe(Z,27)=P24
21960 P3=SQR(2*R(22)*R(24)*P14)/P11!GAMMA
21970 Fe(Z,4)=P3
21980 P1=P24+P3*SQR(P4)*P26   !VTO
21990 Fe(Z,2)=P1
22000 P25=SQR(2*R(22)/(R(24)*P14))*.01
22010 Fe(Z,26)=P25
22020 P17=Fe(Z,18)   !U CRITICAL
22030 IF P17<100 THEN 22060
22040 P17=P17*R(22)/P11
22050 Fe(Z,18)=P17
22060 P16=Fe(Z,17)
22070 P2=P11*P16*(300/(Q+273))^1.5
22080 Fe(Z,3)=P2
22090 GOTO 22150
22100 Mobt:P1=Fe(Z,2)
22110 P4=Fe(Z,5)
22120 P3=Fe(Z,4)
22130 P24=P26*P1-P3*SQR(P4)!GAM
22140 Fe(Z,27)=P24
22150 IF Fe(Z,25)=0 THEN Fe(Z,25)=.5
22160 IF Fe(Z,5)=0 THEN Fe(Z,5)=.66
22170 IF Fe(Z,14)=0 THEN Fe(Z,14)=.87
22180 IF (P26>0) AND (Fe(Z,3)=0) THEN Fe(Z,3)=2.42E-5
22190 IF (P26<0) AND (Fe(Z,3)=0) THEN Fe(Z,3)=1.02E-5
22200 NEXT Z
22210 SUBEND
22211 !
22212 ! PROCEDURE SUBROUTINE
22213 !--------------------
22220 SUB Proc(A$,Pn,O$(*))
22230 Pn=Pn+1
22240 IF O$(Pn)[1,4]="DONE" THEN STOP
22250 A$=O$(Pn)
22260 SUBEND
22270 ! SUBROUTINE TO ZERO ARRAYS
22271 ! -----------------------
22280 SUB Matza(B,A_(*))
22290    FOR Z=1 TO B
22300      FOR Y=1 TO B
22310        A_(Y,Z)=0
22320      NEXT Y
22330    NEXT Z
22340 SUBEND
22350 !
22360 ! SUBROUTINE TO ZERO VECTORS
22361 ! ------------------------
22370 SUB Matzv(B,A_(*))
22380    FOR Z=1 TO B
22390      A_(Z)=0
22400    NEXT Z
22410 SUBEND
22411 ! SUBROUTINE FOR MATRIX MULTIPLICATION
22412 ! ------------------------------------
22413 ! THIS SUBROUTINE CAN BE REMOVED
22420 SUB Matm(A_(*),B_(*),C_(*),D)
```

```
22430 CALL Matzv(D,C_(*))
22440   FOR X=1 TO D
22450     FOR Y=1 TO D
22460        C_(X)=C_(X)+A_(X,Y)*B_(Y)
22470     NEXT Y
22480   NEXT X
22490 SUBEND
22491 ! SUBROUTINE INITIALIZING ALL MATRIX ENTRIES TO ZEROS
22492 ! ------------------------------------------------
22500 SUB Initial_mat(A)
22510 COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
22520 COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
22530 Loc_ptr=A+1
22540 FOR I=0 TO A
22550 Ipt(I)=0
22560 Jpt(I)=0
22570 Row(I)=0
22580 Col(I)=0
22590 Rhs(I)=0
22600 Inzero(I)=0
22610 Jnzero(I)=0
22620 NEXT I
22630 SUBEND
22640 !
22650 ! SUBROUTINE FINDS THE MINIMUM OF TWO NUMBERS
22651 ! -----------------------------------------
22660 SUB Min2(A,B,C)
22670 IF A>=B THEN C=B
22680 IF A<B THEN C=A
22690 SUBEND
22700 !
22710 ! SUBROUTINE FINDS  A MINIMUM OF THREE NUMBERS
22711 ! -----------------------------------------
22720 SUB Min3(A,B,C,D)
22730 IF (A>=B) OR (A>=C) THEN GOTO 22760
22740 IF (B>=A) OR (B>=C) THEN GOTO 22790
22750 IF (C>=A) OR (C>=B) THEN GOTO 22820
22760 IF B>=C THEN D=C
22770 IF B<C THEN D=B
22780 SUBEXIT
22790 IF A>=C THEN D=C
22800 IF A<C THEN D=A
22810 SUBEXIT
22820 IF A>=B THEN D=B
22830 IF A<B THEN D=A
22840 SUBEND
22841 ! SUBROUTINE TO EQUALIZE TWO VECTORS
22842 ! --------------------------------
22850 SUB Mate(A_(*),B_(*),C)
22860 FOR Z=1 TO C
22870 B_(Z)=A_(Z)
22880 NEXT Z
22890 SUBEND
22891 !
22900 DEF FNLin$(INTEGER X)
22910   INTEGER I
```

```
22920   IF X=0 THEN RETURN CHR$(13)
22930   Eol$=CHR$(13)&CHR$(10)
22940   IF X<0 THEN Eol$=CHR$(10)
22950   ALLOCATE R$[X*LEN(Eol$)]
22960   R$=""
22970   FOR I=1 TO X
22980     R$=R$&Eol$
22990   NEXT I
23000   RETURN R$
23010   FNEND !
23011 !
23020 DEF FNY_to_x(Y,INTEGER X)
23030   INTEGER Sign
23040   IF Y<0 THEN
23050     Sign=ABS(X) MOD 2
23060     Sign=(Sign=0)-(Sign=1)
23070     RETURN Sign*ABS(Y)^X
23080   ELSE
23090     RETURN Y^X
23100   END IF
23110 FNEND !
23111 !
23120 DEF FNMax(A0,A1,OPTIONAL A2,A3,A4,A5,A6,A7,A8,A9)
23130   Max=A0
23140   IF NPAR=1 THEN RETURN Max
23150   IF A1>Max THEN Max=A1
23160   IF NPAR=2 THEN RETURN Max
23170   IF A2>Max THEN Max=A2
23180   IF NPAR=3 THEN RETURN Max
23190   IF A3>Max THEN Max=A3
23200   IF NPAR=4 THEN RETURN Max
23210   IF A4>Max THEN Max=A4
23220   IF NPAR=5 THEN RETURN Max
23230   IF A5>Max THEN Max=A5
23240   IF NPAR=6 THEN RETURN Max
23250   IF A6>Max THEN Max=A6
23260   IF NPAR=7 THEN RETURN Max
23270   IF A7>Max THEN Max=A7
23280   IF NPAR=8 THEN RETURN Max
23290   IF A8>Max THEN Max=A8
23300   IF NPAR=9 THEN RETURN Max
23310   IF A9>Max THEN Max=A9
23320   RETURN Max
23330   FNEND
23340   SUB Max2(A,B,C)
23350   IF A>=B THEN C=A
23360   IF A<B THEN C=B
23370   SUBEND
23371   ! -----------------SPARCE MATRIX SUBROUTINE------------------
23372   !
23373   ! FUNCTION FINDS THE LOCATION OF MATRIX ENTRY A_(Ii,Jj)
23374   ! ----------------------------------------------------------
23380   DEF FNFindelem1(Ii,Jj)
23390   COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
23400   COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
23410     I=Ii
```

```
23420     J=Jj
23430     I=Jpt(I)
23440 Repeat:  IF Col(I)=J THEN GOTO Done
23450          I=Jpt(I)
23460          IF I=0 THEN
23470             Findelem1=0
23480             GOTO Finish
23490          END IF
23500          GOTO Repeat
23510 Finish:  RETURN Findelem1
23520 Done:    Findelem1=I
23530          RETURN Findelem1
23540   FNEND
23541   ! FUNCTION FINDS THE LOCATIONS OF MATRIX ENTRY
23542   ! ----------------------------------------
23550   DEF FNFindelem2(Lloc1,Lloc2)
23560   COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
23570   COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
23580   Loc1=Lloc1
23590   Loc2=Lloc2
23600 Loop:IF Col(Loc1)=Col(Loc2) THEN
23610          Findelem2=Loc1
23620          GOTO Finish
23630       END IF
23640       Loc1=Jpt(Loc1)
23650       IF Loc1=0 THEN
23660          Findelem2=0
23670          GOTO Finish
23680       END IF
23690       GOTO Loop
23700 Finish:RETURN Findelem2
23710   FNEND
23750   ! ADD MATRIX ELEMENT SUBROUTINE
23751   ! ---------------------------
23760   SUB Add_elem(Ii,Jj,X)
23770   COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
23780   COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
23781   COM /Blk9/ Flag
23790   I=Ii
23800   J=Jj
23810   IF Flag=0 THEN GOTO 23840
23820   I=Newrow(I)
23830   J=Newcol(J)
23840   Piv=Jpt(I)
23850 Repeat:IF Col(Piv)=J THEN GOTO Done
23860   IF Col(Piv)>J OR Piv=0 THEN GOTO Cont1
23870   Piv=Jpt(Piv)
23880   GOTO Repeat
23890 Cont1:  CALL Fillin_ij(I,J)
23900   Piv=Loc_ptr-1
23910   Inzero(I)=Inzero(I)+1
23920   Jnzero(J)=Jnzero(J)+1
23930 Done:  Rhs(Piv)=Rhs(Piv)+X
23940   SUBEND
23950   ! CREATE FILL IN SUBROUTINE WITH ROW AND COLUMN INDEX ARE GIVEN
23951   ! -------------------------------------------------------------
```

```
23960   SUB Fillin_ij(Ii,Jj)
23970   COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
23980   COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
23990   I=Ii
24000   J=Jj
24010   IF Jpt(I)=0 THEN
24020     Jpt(I)=Loc_ptr
24030     Jpt(Loc_ptr)=0
24040     GOTO Done1
24050   END IF
24060   Tmpa=Jpt(I)
24070   IF Col(Tmpa)>J THEN
24080     Temp=Jpt(I)
24090     Jpt(I)=Loc_ptr
24100     Jpt(Loc_ptr)=Temp
24110     GOTO Done1
24120   END IF
24130 Repeat1:Tmpb=Jpt(Tmpa)
24140   IF Col(Tmpb)>J OR Tmpb=0 THEN GOTO Cont1
24150   Tmpa=Tmpb
24160   GOTO Repeat1
24170 Cont1:  Jpt(Tmpa)=Loc_ptr
24180   IF Tmpb=0 THEN
24190     Jpt(Loc_ptr)=0
24200     ELSE
24210     Jpt(Loc_ptr)=Tmpb
24220   END IF
24230 Done1:  IF Ipt(J)=0 THEN
24240     Ipt(J)=Loc_ptr
24250     Ipt(Loc_ptr)=0
24260     GOTO Done2
24270   END IF
24280   Tmpa=Ipt(J)
24290   IF Row(Tmpa)>I THEN
24300     Temp=Ipt(J)
24310     Ipt(J)=Loc_ptr
24320     Ipt(Loc_ptr)=Temp
24330     GOTO Done2
24340   END IF
24350 Repeat2:Tmpb=Ipt(Tmpa)
24360   IF Row(Tmpb)>I OR Tmpb=0 THEN GOTO Cont2
24370   Tmpa=Tmpb
24380   GOTO Repeat2
24390 Cont2:  Ipt(Tmpa)=Loc_ptr
24400   IF Tmpb=0 THEN
24410     Ipt(Loc_ptr)=0
24420     ELSE
24430     Ipt(Loc_ptr)=Tmpb
24440   END IF
24450 Done2:  Row(Loc_ptr)=I
24460   Col(Loc_ptr)=J
24470   Rhs(Loc_ptr)=0
24480   Loc_ptr=Loc_ptr+1
24490   SUBEND
24491   ! SUBROUTINE CREATES FILL_INs
24492   ! -------------------------
```

```
24500   SUB Fillin_loc(Lloc1,Lloc2)
24510   COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
24520   COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
24530   Loc1=Lloc1
24540   Loc2=Lloc2
24550   I=Row(Loc1)
24560   J=Col(Loc2)
24570   CALL Fillin_ij(I,J)
24580   SUBEND
24590   ! CHECK AND SOLVE DIAGONAL ZERO ELEMENT PROBLEM SUBROUTINE
24591   ! ----------------------------------------------------------
24600   SUB Chkzero(A)
24610   COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
24620   COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
24630   FOR I=1 TO A
24640     Temp1=FNFindelem1(I,I)
24650     IF Temp1<>0 AND Rhs(Temp1)<>0 THEN GOTO Done
24660     Tmpb=I
24670 Loop1: Tmpc=Ipt(Tmpb)
24680     Varr2=Row(Tmpc)
24690     IF Row(Tmpc)>I AND Rhs(Tmpc)<>0 THEN
24700       CALL Swap_row(I,Varr2,A)
24710       Tmpn=Inzero(I)
24720       Inzero(I)=Inzero(Row(Tmpc))
24730       Inzero(Row(Tmpc))=Tmpn
24740       GOTO Done
24750     END IF
24760     IF Tmpc=0 THEN GOTO Cont1
24770     Tmpb=Tmpc
24780     GOTO Loop1
24790 Cont1: Tmpb=Ipt(I)
24800 Loop2: Temp=Row(Tmpb)
24810     Temp2=FNFindelem1(I,Temp)
24820     IF Temp2<>0 AND Rhs(Temp2)<>0 THEN
24830       Varr1=Row(Tmpb)
24840       CALL Swap_row(Varr1,I,A)
24850       Tmpn=Inzero(I)
24860       Inzero(I)=Inzero(Row(Tmpb))
24870       Inzero(Row(Tmpb))=Tmpn
24880       GOTO Done
24890     END IF
24900     IF Temp=0 THEN GOTO Chkcol
24910     Tmpb=Ipt(Tmpb)
24920     GOTO Loop2
24930 Chkcol: Ctmpb=I
24940 Loop3:  Ctmpc=Jpt(Ctmpb)
24950     Varr3=Col(Ctmpc)
24960     IF Col(Ctmpc)>I THEN
24970       CALL Swap_col(I,Varr3,A)
24980       Tmpn=Jnzero(I)
24990       Jnzero(I)=Jnzero(Col(Ctmpc))
25000       Jnzero(Col(Ctmpc))=Tmpn
25010       GOTO Done
25020     END IF
25030     IF Ctmpc=0 THEN GOTO Cont3
25040     Ctmpb=Ctmpc
```

```
25050     GOTO Loop3
25060 Cont3:  Ctmpb=Jpt(I)
25070 Loop4: Temp=Col(Ctmpb)
25080     Temp3=FNFindelem1(Temp, I)
25090     IF Temp3<>0 AND Rhs(Temp3)<>0 THEN
25100       Varr5=Col(Ctmpb)
25110       CALL Swap_col(Varr5, I, A)
25120       Tmpn=Jnzero(I)
25130       Jnzero(I)=Jnzero(Col(Ctmpb))
25140       Jnzero(Col(Ctmpb))=Tmpn
25150       GOTO Done
25160     END IF
25170     IF Temp=0 THEN CALL Dia_zero
25180     Ctmpb=Jpt(Ctmpb)
25190     GOTO Loop4
25200 Done:  NEXT I
25210   SUBEND
25220   ! SUBROUTINE FINDS THE ROW HAVING THE MINIMUM NON-ZERO ELEMENTS
25221   ! ------------------------------------------------------------
25222   ! FIND PIVOT ELEMENT USING MARKOWITZ ALGORITHM
25230   SUB Min_index(I, A)
25240   COM Ipt(*), Jpt(*), Row(*), Col(*), Rhs(*), Buf(*), Inzero(*), Jnzero(*)
25250   COM /Blk7/ N, Loc_ptr, Min_index, Newrow(*), Newcol(*), R_(*)
25260   Min_index=I
25270   Minvalue=(Inzero(I)-1)*(Jnzero(I)-1)
25280   FOR J=I+1 TO A
25290     Product=(Inzero(J)-1)*(Jnzero(J)-1)
25300     IF Product<Minvalue THEN
25310       Minvalue=Product
25320       Min_index=J
25330     END IF
25340   NEXT J
25350   SUBEND
25360   ! SUBROUTINE DOES MARKOWITZ RE-ORDERING
25361   ! -----------------------------------
25370   SUB Reorder(A)
25380   COM Ipt(*), Jpt(*), Row(*), Col(*), Rhs(*), Buf(*), Inzero(*), Jnzero(*)
25390   COM /Blk7/ N, Loc_ptr, Min_index, Newrow(*), Newcol(*), R_(*)
25400   FOR I=1 TO A
25410     CALL Min_index(I, A)
25420     IF Min_index<>I THEN
25430       CALL Swap_row(I, Min_index, A)
25440       CALL Swap_col(I, Min_index, A)
25450       Tmp=Inzero(I)
25460       Inzero(I)=Inzero(Min_index)
25470       Inzero(Min_index)=Tmp
25480       Tmp=Jnzero(I)
25490       Jnzero(I)=Jnzero(Min_index)
25500       Jnzero(Min_index)=Tmp
25510     END IF
25520     Piv=FNFindelem1(I, I)
25530     Dpiv=Piv
25540 Loop1:  Dpiv=Ipt(Dpiv)
25550     IF Dpiv=0 THEN GOTO Rnext
25560     Inzero(Row(Dpiv))=Inzero(Row(Dpiv))-1
25570     Rpiv=Piv
```

```
25580 Loop2:  Rpiv=Jpt(Rpiv)
25590      IF Rpiv=0 THEN GOTO Loop1
25600      Jnzero(Col(Rpiv))=Jnzero(Col(Rpiv))-1
25610      IF FNFindelem2(Dpiv,Rpiv)=0 THEN
25620        CALL Fillin_loc(Dpiv,Rpiv)
25630        Inzero(Row(Dpiv))=Inzero(Row(Dpiv))+1
25640        Jnzero(Col(Rpiv))=Jnzero(Col(Rpiv))+1
25650      END IF
25660      GOTO Loop2
25670 Rnext:  NEXT I
25680    SUBEND
25681    ! SINGULAR MATRIX SUBROUTINE
25682    ! ----------------------
25690    SUB Dia_zero
25700    PRINT "DIAGONAL ZERO ELEMENT EXISTS - SINGULAR MATRIX"
25710    GOTO Outt
25720    SUBEND
25730    ! SUBROUTINE DOES LU DECOMPOSITION
25731    ! ------------------------------
25740    SUB Lu_decom(A)
25750    COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
25760    COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
25770    FOR I=1 TO A
25780    Piv=FNFindelem1(I,I)
25790    Dpiv=Piv
25800 Loop1:Dpiv=Ipt(Dpiv)
25810    IF Dpiv=0 THEN GOTO Unext
25811    IF Rhs(Piv)=0 THEN Rhs(Piv)=1.E-12
25820    Rhs(Dpiv)=Rhs(Dpiv)/Rhs(Piv)
25830    Rpiv=Piv
25840 Loop2:Rpiv=Jpt(Rpiv)
25850    IF Rpiv=0 THEN GOTO Loop1
25860    Tpiv=FNFindelem2(Dpiv,Rpiv)
25870    Rhs(Tpiv)=Rhs(Tpiv)-Rhs(Dpiv)*Rhs(Rpiv)
25880    GOTO Loop2
25890 Unext:NEXT I
25900    SUBEND
25910    ! SUBROUTINE DOES FORWARD SUBSTITUTION
25911    ! ----------------------------------
25920    SUB For_sub(A)
25930    COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
25940    COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
25950    FOR I=1 TO A
25960      Piv=FNFindelem1(I,I)
25970      Dpiv=Piv
25980 Loop3:Dpiv=Ipt(Dpiv)
25990      IF Dpiv=0 THEN GOTO Nextf
26000      Rhs(Row(Dpiv))=Rhs(Row(Dpiv))-Rhs(Dpiv)*Rhs(I)
26010      GOTO Loop3
26020 Nextf:NEXT I
26030    SUBEND
26040    ! SUBROUTINE DOES BACKWARD SUBSTITUTION
26041    ! -----------------------------------
26050    SUB Back_sub(A)
26060    COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
26070    COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
```

```
26080   FOR I=A TO 1 STEP -1
26090   Sum=0
26100   Piv=FNFindelem1(I,I)
26110   Rpiv=Piv
26120 Loopb: Rpiv=Jpt(Rpiv)
26130   IF Rpiv=0 THEN GOTO Calb
26140   Sum=Sum+Rhs(Rpiv)*Rhs(Col(Rpiv))
26150   GOTO Loopb
26159 Calb: IF Rhs(Piv)=0 THEN Rhs(Piv)=1.E-12
26160       Rhs(I)=(Rhs(I)-Sum)/Rhs(Piv)
26170   NEXT I
26180   SUBEND
26181   ! SOLVE SUBROUTINE
26182   ! ----------------
26190   SUB Solve(O(*),A)
26200   COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
26210   COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
26220   COM /Blk8/ Del_elem(*),Newtemp(*)
26221   COM /Blk9/ Flag
26223   ! DO Chkzero AND Re-order ON 1ST ITERATIONONLY
26230   IF R_(29)=0 AND R_(26)=0 AND Flag=0 THEN CALL Chkzero(A)
26240   IF R_(29)=0 AND R_(26)=0 AND Flag=0 THEN CALL Reorder(A)
26250   IF R_(29)=0 AND R_(26)=0 AND Flag=0 THEN GOTO 26350
26260   FOR J=1 TO A                  ! REARRANGE THE RIGHT HAND SIDE VECTOR
26270     Newtemp(J)=Rhs(J)           ! ACCORDING TO THE SWAP ROW
26280   NEXT J
26290   FOR J=1 TO A
26300     Temp=Newrow(J)
26310     Rhs(Temp)=Newtemp(J)
26320   NEXT J
26330   !
26340   IF R_(26)=0 AND R_(29)=0 AND Flag=0 THEN
26350     FOR J=1 TO A
26360      Newtemp(J)=Newrow(J)
26370     NEXT J
26380     FOR J=1 TO A
26390      Newrow(Newtemp(J))=J
26400     NEXT J
26410     FOR J=1 TO A
26420      Newtemp(J)=Newcol(J)
26430     NEXT J
26440     FOR J=1 TO A
26450      Newcol(Newtemp(J))=J
26460     NEXT J
26461     Flag=1
26470   END IF
26480   CALL Lu_decom(A)
26490   CALL For_sub(A)
26500   CALL Back_sub(A)
26510   ! RESORT THE ORDER OF ANSWERS AFTER SWAP COLUMN
26511   ! ---------------------------------------------
26520   FOR I=1 TO A
26530     Stemp=Rhs(I)
26540     Adjust=Buf(I)
26550     O(Adjust)=Stemp
26560   NEXT I
```

```
26570   CALL Initial_mat(A)
26580   Loc_ptr=A+1
26590   SUBEND
26591   !
26600   SUB Initial_buf(A)
26610   COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
26620   COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
26630   FOR I=0 TO A
26640     Buf(I)=I
26650     Newrow(I)=I
26660     Newcol(I)=I
26670   NEXT I
26680   SUBEND
26690   ! SWAP ROW I1 & I2 SUBROUTINE
26691   ! -------------------------
26700   SUB Swap_row(I1,I2,A)
26710   COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
26720   COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
26730   COM /Blk8/ Del_elem(*),Newtemp(*)
26740   FOR J=1 TO 100
26750     Del_elem(1,J)=0
26760     Del_elem(2,J)=0
26770     Del_elem(3,J)=0
26780   NEXT J
26790   CALL Del_row(I1,1)
26800   CALL Del_row(I2,A+1)
26810   CALL Insert_row(I1,I2,1,A)
26820   Temp=Newrow(I1)
26830   Newrow(I1)=Newrow(I2)
26840   Newrow(I2)=Temp
26850   SUBEND
26860   !
26870   ! DELETE ALL ELEMENTS IN ROW Ii
26871   ! ----------------------------
26880   SUB Del_row(Ii,X)
26890   COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
26900   COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
26910   COM /Blk8/ Del_elem(*),Newtemp(*)
26920   I=X
26930   Temp=Jpt(Ii)
26940   IF Temp=0 THEN GOTO 27180
26950 Redo: Del_elem(1,I)=Temp
26960   Del_elem(2,I)=Col(Temp)
26970   Del_elem(3,I)=Rhs(Temp)
26980   Jnzero(Col(Temp))=Jnzero(Col(Temp))-1
26990   Temp=Jpt(Temp)
27000   I=I+1
27010   IF Temp<>0 THEN GOTO Redo
27020   !
27030   FOR J=X TO I-1
27040   Tmpa=Del_elem(2,J)
27050 Again: Tmpb=Ipt(Tmpa)
27060   IF Tmpb=Del_elem(1,J) THEN GOTO Cont1
27070   Tmpa=Tmpb
27080   GOTO Again
27090 Cont1: Ipt(Tmpa)=Ipt(Tmpb)
```

```
27100    Rhs(Tmpb)=0
27110    Jpt(Tmpb)=0
27120    Ipt(Tmpb)=0
27130    Row(Tmpb)=0
27140    Col(Tmpb)=0
27150    NEXT J
27160    Inzero(Ii)=0
27170    Jpt(Ii)=0
27180    SUBEND
27190    !
27200    ! RE-INSERT ALL ELEMENTS AT ROW I1 & I2 AFTER THE ROW SWAPPING
27201    ! ---------------------------------------------------------------
27210    SUB Insert_row(I1,I2,J,A)
27220    COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
27230    COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
27240    COM /Blk8/ Del_elem(*),Newtemp(*)
27250    Tmp=Loc_ptr
27260    Tmpa=I2
27270    Loc_ptr=Del_elem(1,J)
27280    Tmpb=Del_elem(2,J)
27290    Tmpc=Del_elem(3,J)
27300    CALL Add_elem(Tmpa,Tmpb,Tmpc)
27310    IF Del_elem(1,J+1)=0 AND J>A THEN GOTO Done
27320    IF Del_elem(1,J+1)=0 THEN
27330      J=A+1
27340      Tmpa=I1
27350      ELSE
27360      J=J+1
27370    END IF
27380    GOTO 27270
27390 Done: Loc_ptr=Tmp
27400    SUBEND
27401    ! SUBROUTINE SWAPS COLUMN J1 AND J2
27410    ! -----------------------------------
27420    SUB Swap_col(J1,J2,A)
27430    COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
27440    COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
27450    COM /Blk8/ Del_elem(*),Newtemp(*)
27460    FOR J=1 TO 100
27470      Del_elem(1,J)=0
27480      Del_elem(2,J)=0
27490      Del_elem(3,J)=0
27500    NEXT J
27510    CALL Del_col(J1,1)
27520    CALL Del_col(J2,A+1)
27530    CALL Insert_col(J1,J2,1,A)
27540    Temp=Newcol(J1)
27550    Newcol(J1)=Newcol(J2)
27560    Newcol(J2)=Temp
27570    Temp=Buf(J1)
27580    Buf(J1)=Buf(J2)
27590    Buf(J2)=Temp
27600    SUBEND
27610    !
27620    ! DELELE COLUMN SUBROUTINE
27621    ! -----------------------
```

```
27630    SUB Del_col(Jj,X)
27640    COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
27650    COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
27660    COM /Blk8/ Del_elem(*),Newtemp(*)
27670    I=X
27680    Temp=Ipt(Jj)
27690    IF Temp=0 THEN GOTO 27930
27700 Redo: Del_elem(1,I)=Temp
27710    Del_elem(2,I)=Row(Temp)
27720    Del_elem(3,I)=Rhs(Temp)
27730    Inzero(Row(Temp))=Inzero(Row(Temp))-1
27740    Temp=Ipt(Temp)
27750    I=I+1
27760    IF Temp<>0 THEN GOTO Redo
27770    !
27780    FOR J=X TO I-1
27790    Tmpa=Del_elem(2,J)
27800 Again: Tmpb=Jpt(Tmpa)
27810    IF Tmpb=Del_elem(1,J) THEN GOTO Cont1
27820    Tmpa=Tmpb
27830    GOTO Again
27840 Cont1: Jpt(Tmpa)=Jpt(Tmpb)
27850    Rhs(Tmpb)=0
27860    Jpt(Tmpb)=0
27870    Ipt(Tmpb)=0
27880    Row(Tmpb)=0
27890    Col(Tmpb)=0
27900    NEXT J
27910    Jnzero(Jj)=0
27920    Ipt(Jj)=0
27930    SUBEND
27940    ! SUBROUTINE RE-INSERT ALL THE ELEMENTS IN THE COLUMN J1 & J2
27941    ! -------------------------------------------------------------
27950    SUB Insert_col(J1,J2,J,A)
27960    COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
27970    COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
27980    COM /Blk8/ Del_elem(*),Newtemp(*)
27990    Tmp=Loc_ptr
28000    Tmpa=J2
28010    Loc_ptr=Del_elem(1,J)
28020    Tmpb=Del_elem(2,J)
28030    Tmpc=Del_elem(3,J)
28040    CALL Add_elem(Tmpb,Tmpa,Tmpc)
28050    IF Del_elem(1,J+1)=0 AND J>A THEN GOTO Done
28060    IF Del_elem(1,J+1)=0 THEN
28070      J=A+1
28080      Tmpa=J1
28090      ELSE
28100      J=J+1
28110    END IF
28120    GOTO 28010
28130 Done:   Loc_ptr=Tmp
28140    SUBEND
28150    ! SUBROUTINE  MATCHES THE INPUT NODE NO. WITH THE MATRIX ENTRY INDEX
28151    ! ------------------------------------------------------------------
28160    SUB Marko(A)
```

```
28170    COM Ipt(*),Jpt(*),Row(*),Col(*),Rhs(*),Buf(*),Inzero(*),Jnzero(*)
28180    COM /Blk7/ N,Loc_ptr,Min_index,Newrow(*),Newcol(*),R_(*)
28190    COM /Blk8/ Del_elem(*),Newtemp(*)
28191    COM /Blk9/ Flag
28320      FOR J=1 TO A
28330       Newtemp(J)=Newrow(J)
28340      NEXT J
28350      FOR J=1 TO A
28360       Newrow(Newtemp(J))=J
28370      NEXT J
28380      FOR J=1 TO A
28390       Newtemp(J)=Newcol(J)
28400      NEXT J
28410      FOR J=1 TO A
28420       Newcol(Newtemp(J))=J
28430      NEXT J
28431      Flag=1
28450    SUBEND
28451    !
28452    ! DC TRANSFER SUBROUTINE
28453    ! ---------------------
28454    ! SUBROUTINE GETS DC TRANSFER COMMANDS
28460   SUB Dtran(R_(*),A$,B$,Dt$(*),D_(*),N,A,Fl(*),Ve(*),Ie(*),Node(*),I_f
28462   COM /Blk1/ Vb$(*),Mb$(*),Ib$(*),Eb$(*)
28463   COM /Blk3/ Ra,C,M,D,I,V,F,S,L,E
28470    FOR Ii=1 TO 6
28480    D_(Ii)=0
28490    NEXT Ii
28500    Dt$(1)=""
28510    Dt$(2)=""
28520    Bb$="VI"
28530    R_(32)=R_(32)+1
28540    IF R_(32)<=LEN(A$) THEN GOTO Lab1
28550    CALL Errr(R_(32),7,A$)
28560    SUBEXIT
28570 Lab1: CALL Gnam(R_(32),0,A$,N$)
28580    CALL Dcod(R_(34),R_(32),R_(33),0,0,0,A$,Bb$,1)
28590    R_(50)=R_(34)   !R(50)=FLAG OF SOURCE 1
28600    IF R_(33)<=0 AND R_(34)>0 THEN GOTO Lab2
28610    CALL Errr(R_(32),8,A$)
28620    SUBEXIT
28630 Lab2: CALL Gnum(R_(32),3,P1,P2,P3,0,0,A$,B$,R_(*))
28640   Dt$(1)[1,4]=N$[1,4]
28650   D_(1)=P1
28660   D_(2)=P2
28670   D_(3)=P3
28680   Fl(4)=2
28690   R_(32)=R_(32)+1
28700   IF R_(32)<=LEN(A$) THEN GOTO Lab3
28710   R_(51)=0
28720   GOTO 28822
28730 Lab3: CALL Gnam(R_(32),0,A$,N$)
28740   CALL Dcod(R_(34),R_(32),R_(33),0,0,0,A$,Bb$,1)
28750   IF R_(33)<>0 AND R_(34)>0 THEN GOTO Lab4
28751   CALL Errr(R_(32),8,A$)
28760   SUBEXIT
```

BIASB.36 LISTING

```
28770 Lab4: R_(51)=R_(34)
28780  CALL Gnum(R_(32),3,P4,P5,P6,0,0,A$,B$,R_(*))
28790  Dt$(2)[1,4]=N$[1,4]
28800  D_(4)=P4
28810  D_(5)=P5
28820  D_(6)=P6
28822  CALL Match_node(R_(*),Dt$(*),D_(*),N,A,V,I,Ve(*),Vb$(*),Ie(*),Ib$(*)
28830  SUBEND
29000  ! SUBROUTINE MATCHES THE NODE NO.  WITH RHS ARRAY INDEX NO.
29001  ! -----------------------------------------------------------
29010  SUB Match_node(R_(*),Dt$(*),D_(*),N,A,V,I,Ve(*),Vb$(*),Ie(*),Ib$(*),
29020  FOR Nu=1 TO 2
29030  IF R_(Nu+49)=1 THEN
29031  FOR Z=1 TO V
29040     IF Dt$(Nu)[1,4]<>Vb$(Z) THEN GOTO Lab1
29050     Node(Nu,1)=Z+N
29070     GOTO Lab2
29080 Lab1: NEXT Z
29090  END IF
29100 Lab2: IF R_(Nu+49)=2 THEN
29110  FOR Z=1 TO I
29120  IF Dt$(Nu)[1,4]<>Ib$(Z) THEN GOTO Lab3
29130  IF Ie(Z,1)<>0 THEN
29140     Node(Nu,1)=Ie(Z,1)
29150     I_flag(Nu,1)=1
29160  END IF
29170  IF Ie(Z,2)<>0 THEN
29180     Node(Nu,2)=Ie(Z,2)
29190     I_flag(Nu,2)=1
29200  END IF
29210 Lab3: NEXT Z
29220 END IF
29230 NEXT Nu
29240 SUBEND
29250 !
29251 ! SUBROUTINE ADJUSTS THE SOURCE SUPPLY VALUE AT RHS ARRAY
29252 ! -------------------------------------------------------
29260 SUB Re_adjust(Node(*),Rhs(*),Va_(*),Ia(*),I_flag(*),N,D_(*),R_(*))
29270 FOR Nu=1 TO 2
29280 IF Node(Nu,1)>N THEN
29290 IF Nu=1 THEN Rhs(Node(Nu,1))=Rhs(Node(Nu,1))-Va_(Node(Nu,1)-N)+R_(52)
29291 IF Nu=2 THEN Rhs(Node(Nu,1))=Rhs(Node(Nu,1))-Va_(Node(Nu,1)-N)+R_(53)
29300 END IF
29310 IF Node(Nu,1)<N THEN
29320    IF I_flag(Nu,1)<>0 THEN
29330 IF Nu=1 THEN Rhs(Node(Nu,1))=Rhs(Node(Nu,1))+Ia(Node(Nu,1))-R_(52)
29331 IF Nu=2 THEN Rhs(Node(Nu,1))=Rhs(Node(Nu,1))+Ia(Node(Nu,1))-R_(53)
29340    END IF
29341    IF I_flag(Nu,2)<>0 THEN
29350 IF Nu=1 THEN Rhs(Node(Nu,2))=Rhs(Node(Nu,2))-Ia(Node(Nu,2))+R_(52)
29351 IF Nu=2 THEN Rhs(Node(Nu,2))=Rhs(Node(Nu,2))-Ia(Node(Nu,2))+R_(53)
29353    END IF
29361 END IF
29371 NEXT Nu
29381 SUBEND
29391 ! END OF BIASB.26 PROGRAM
```

BIASD.36 LISTING

```
10      !.......................BIAS-D.......................
20      ! MINICOMPUTER AIDED ELECTRONIC CIRCUIT ANALYSIS PROGRAM
30      ! ORIGINAL WRITTEN IN FORTRAN BY:
40      !                           BRIAN L. BIEHL
41      !                           HARRY DIAMON LABS.
42      !                           2800 POWDER MILL RD.
43      !                           ADELPHI, MD 20783
44      !                           LATEST VERSION FEB., 1982
45      ! THE FOLLOWING VERSION IS TRANSLATED BY:
46      !                           FRANK H. MA
47      !                           PROF. D. O. PEDERSON
48      !                           DEPT. OF E.E.C.S.
50      !                           UNIVERSITY OF CALIF., BERKELEY
51      !                           CA 94702
52      !                           JUNE, 1982
53      !
55      ! DYNAMIC ELEMENT ALLOCATION   ( LINKED-LIST )
56      ! LU DECOMPOSITION
57      ! AC ANALYSIS: STANDARD METHOD USING COMPLEX MATRIX
58      ! SPARSE MATRIX INVERISON
59      ! ELEMENT MODELS
60      ! SPARSE STORAGE OF Y MATRIX
61      !
62      ! ...........................................................
63      !
64      OPTION BASE 1
65      DIM B$[12],M$[6],D$[8],Alen(9)
70      COM U(30),C(60),Y(800)
71      COM Ds,Delt,Delta
80      COM TO,Temp,Dtemp
90      COM Tm(6),A(8),Csat,Vt,Vct,Ptype
100     COM Iel,Jj,Kk,Ll,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
110     COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
120     COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
130     COM A$[80],Ni(30,2),Ieln(9),Ifrst(9),Ilast(9),Ielm(1000)
140     COM Iur(30),Iuc(120),Ipos(400),Ncon(30),Iy(30,30)
198     B$="RCLQIVME*.+ "
199     D$="AIPTESGL"
200     M$="NTSEPU"
201     Alen(1)=8
202     Alen(2)=12
203     Alen(3)=12
210     Alen(4)=16
220     Alen(5)=8
230     Alen(6)=8
240     Alen(7)=20
250     Alen(8)=0
260     Alen(9)=0
261     !
262     !               Iflg VALUES
264     ! 1- INITIAL DC ANALYSIS  6- SWEPT ALTER
265     ! 2- ALTER                7- SAVE CIRCUIT
266     ! 3- PRINT CIRCUIT        8- SMALL SIGNAL GAIN
267     ! 4- PRINT CIRCUIT        9- AC ANALYSIS
268     ! 5- TRANSIENT ANALYSIS  10- TEMPERATURE ANALYSIS
269     !
```

BIASD.36 LISTING

```
270   !                    Iel VALUES
272   ! 1- RESISTOR                  5- CURRENT SOURCE
273   ! 2- CAPACITOR                 6- VOLTAGE SOURCE
274   ! 3- INDUCTOR                  7- MODEL
275   ! 4- BJT
276   !
277   !.. MAIN PROGRAMME:
278   CALL Initl
280   PRINT "BIAS-D , JUNE 1982"
290   DISP "INPUT DATA";
300   Itoil=0
310   Nunit=Ir
320   READ A$
330   PRINT A$
340   L1=1
350   IF A$[1;1]=B$[12;1] THEN L1=2
360   FOR Iel=1 TO 12
370   IF A$[L1;1]=B$[Iel;1] THEN 450
380   NEXT Iel
390   DISP "ILLEGAL CHARACTER : RE-";
400   GOTO 290
410   Iflg=4
420   GOTO 290
440   !.. DETERMINE ELEMENT TYPE
450   IF Iel<8 THEN 480
460   I=Iel-7
470   ON I GOTO 2080,320,1201,320,320
480   IF Iflg=2 THEN 1490
490   Itt=Lpos+1
500   CALL Pinput
510   Ilast(Iel)=Lpos
520   IF Lpos<Mxlst THEN 550
530   DISP "ELEMENT ARRAY OVERFLOW";
540   GOTO 410
550   IF Iel=7 THEN 720
560   Lpos=Lpos+Alen(Iel)
580   !.. REPROCESS UNGROUNDED OF NEGATIVE VOLTAGE SOURCES
590   IF Iel<>6 THEN 320
600   IF A(1)<>0 THEN 660
610   Ielm(Itt+6)=A(2)
620   Ielm(Itt+7)=0
630   Itt=Itt/2
640   Ielm(2*(Itt+3))=-A(3)
650   GOTO 320
660   IF A(2)=0 THEN 320
670   PRINT "SOURCE UNGROUNDED : RE-";
681   Ieln(6)=Ieln(6)-1
690   GOTO 290
710   !PROCESS MODELS
720   Itt=Lpos
730   Mnum=FNPval(L1+1)
740   Ielm(Lpos+3)=Mnum
760   !.. ENTRY POINT FOR ALTERED MODEL
770   !.. SKIP LEADING BLANKS
780   IF A$[L1;1]<>B$[12;1] THEN 820
790   L1=L1+1
```

```
800    IF L1>12 THEN 390
810    GOTO 780
820    I=L1
830    L1=L1+4
840    CALL Rdfld
850    IF Mm<0 THEN 910
870    !CHECK IF LEGAL MODEL TYPE
880    FOR M=1 TO 5
890    IF A$[I;1]=M$[M;1] THEN 930
900    NEXT M
910    IF Iflg<>2 THEN Ieln(7)=Ieln(7)-1
920    GOTO 390
930    Mpos=Itt/2
940    FOR K=1 TO 7
950    Kk=Mpos+K+3
960    Ielm(2*Kk)=A(K)
970    NEXT K
973    Gx=Gx+1
974    IF Gx<2 THEN 1000
990    !DETERMINE MODEL TYPE
1000   I=I+1
1010   IF M<5 THEN 1050
1020   M=6
1030   IF A$[I;1]=M$[6;1] THEN M=5
1040   IF A$[I;1]=M$[1;1] THEN M=-1
1050   Ielm(2*(Mpos+3))=M
1060   IF M=-1 THEN 1120
1070   IF M=1 THEN 1120
1080   IF M=2 THEN Itemp=Itt/2
1090   GOTO 1160
1110   !.. BJT MODEL DEFUALT PARAMETER
1120   IF A(1)=0 THEN Ielm(2*(Mpos+4))=100.0
1130   IF A(2)=0 THEN Ielm(2*(Mpos+5))=1.0
1140   IF A(3)=0 THEN Ielm(2*(Mpos+6))=1.0E-15
1150   IF A(4)=0 THEN Ielm(2*(Mpos+7))=1.0E+12
1160   IF Iflg=2 THEN 320
1170   Lpos=Lpos+Alen(7)
1180   GOTO 320
1200   ! ******CIRCUIT UPDATES******
1201   IF A$[L1+1;1]<>D$[8;1] THEN 1210
1202   GOTO 320
1210   IF Iflg=1 THEN 390
1220   !
1230   !DETERMINE UPDATE TYPE
1240   FOR J=1 TO 7
1250   IF A$[L1+1;1]=D$[J;1] THEN 1280
1260   NEXT J
1270   GOTO 390
1280   Iflg=J+1
1290   ON J GOTO 1320,1680,1870,1740,278,410,1910
1310   !***.AC
1320   IF A$[L1+2;1]<>B$[2;1] THEN 290
1330   Iflg=9
1340   DISP "VIN FSTRT FSTOP PTS/DEC"
1350   INPUT A$
1360   Iel=6
```

```
1370  IF A$[L1;1]=B$[5;1] THEN Iel=5
1380  CALL Alter
1390  IF Ierr=0 THEN 410
1410  !INPT CONTAINS STRATING LOCATION OF SOURCE VALUE IN IELN()
1420  Inpt=Itt/2
1430  FOR M=1 TO 4
1440  Tm(M)=A(M)
1450  NEXT M
1460  GOTO 2040
1480  !** .ALTER
1490  CALL Alter
1500  IF Ierr=0 THEN 290
1510  IF Iel=7 THEN 780
1520  Mpos=Itt/2
1530  Tm(5)=Ielm(2*(Mpos+3))
1540  Ielm(2*(Mpos+3))=A(1)
1550  IF Iel=4 THEN 390
1560  IF A(3)=0 THEN 320
1580  !PROCESS SWEPTED ALTER
1590  FOR M=1 TO 3
1600  Tm(M)=A(M)
1610  NEXT M
1620  Tm(4)=Mpos
1630  Ltype=Iel
1640  Iflg=6
1650  GOTO 2040
1670  !***. INSERT
1680  IF Jflg=1 THEN 290
1690  CALL Renum(1)
1700  Ieln(9)=0
1710  GOTO 290
1730  !***. TEMPERATURE
1740  IF A$[L1+2;1]=B$[1;1] THEN 1950
1750  IF Itemp<>0 THEN 1780
1760  PRINT "TEMP MODEL NOT SPECIFIED : RE-";
1770  GOTO 290
1780  PRINT "T(DEG C)"
1790  INPUT A$
1800  Iflg=10
1810  Temp=FNPval(1)+273
1830  Dtemp=Temp-300
1840  GOTO 290
1860  !***. PRINT CKT
1870  CALL Prckt
1880  GOTO 290
1900  !***. GAIN
1910  DISP "THIS OPTION NOT IMPLEMENTED IN THIS VERSION: RE-";
1920  GOTO 410
1950  PRINT "TR TO TSTOP TSTEP"
1960  INPUT A$
1970  L1=4
1980  CALL Rdfld
1990  IF Ierr=-1 THEN 410
2000  IF A(3)=0 THEN A(3)=(A(2)-A(1))/50
2010  Delta=A(3)
2020  Tm(2)=A(2)
```

```
2030  TO=A(1)
2040  CALL Pout
2050  Jout=FNNconv(Iout,0,Ni(*),Node)
2060  GOTO 2370
2080  IF Iflg=2 THEN 2370
2090  IF Iflg>=5 THEN 2370
2100  IF Iflg=4 THEN 2500
2110  CALL Mchek
2120  IF Ierr=0 THEN 410
2130  Nnode=Node-Ieln(6)
2140  IF Nnode<=Ndmax THEN 2180
2150  DISP "NODE LIMIT EXCEEDED";
2160  GOTO 410
2170  !.. CHECK FOR UNCONNECTED NODES
2180  J=1
2190  FOR I=1 TO Node
2200  IF Ni(I,1)>0 THEN 2230
2210  PRINT "ONLY ONE CONNECTION AT NODE";Ni(I,2)
2220  J=0
2230  NEXT I
2240  IF J=0 THEN 410
2250  CALL Pout
2260  CALL Prckt
2261  PRINT
2270  PRINT "NODES:  ";Node
2271  PRINT
2280  PRINT "**** END OF INPUT DATA ****"
2281  PRINT
2290  !
2300  CALL Setup
2310  IF Ierr=-2 THEN 530
2320  Temp=300
2330  FOR I=1 TO Nnode
2340  U(I)=0.
2350  C(I)=0.
2360  NEXT I
2370  Delt=1.0E+12
2380  IF Iflg<>9 THEN 2410
2390  CALL Acsol
2400  GOTO 2420
2410  CALL Analy
2420  IF Iplt=0 THEN 2440
2421  !...CALL EXITGR
2422  !...K=IWAIT$(2)
2430  Iplt=0
2440  IF Jflg=1 THEN 2460
2450  PRINT "TOTAL ITERATIONS=";Itotl
2460  TO=0
2470  Jflg=Iflg
2480  Iunit=Iw
2490  GOTO 410
2500  STOP
2510  END
2511  !
2520  !.. START SUBPROGRAMS:
2521  !
```

```
2530  DEF FNIndx(Nr,Nc)
2531  OPTION BASE 1
2560  COM U(*),C(*),Y(*)
2561  COM Ds,Delt,Delta
2570  COM TO,Temp,Dtemp
2580  COM Tm(6),A(8),Csat,Vt,Vct,Ptype
2590  COM Iel,Jj,Kk,Ll,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
2600  COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Uout,Inpt,Iform
2610  COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
2620  COM A$[80],Ni(30,2),Ieln(9),Ifrst(9),Ilast(9),Ielm(1000)
2630  COM Iur(30),Iuc(120),Ipos(400),Ncon(30),Iy(30,30)
2640  !
2650  !...DETERMINE LINEAR Y ADDRESS LOCATION FROM Y(I,J)
2660  !   Y(.,)  ADDRESS
2670  !   MODIFIED FROM SINC-S 6-6-77
2680  IF Nr=Nc THEN 2940
2690  Pis=Ncon(Nr)
2700  Js=Ncon(Nc)
2710  IF Js>Pis THEN 2830
2720  !...LOWER TRIANGLE
2730  N=Iur(Js)
2740  Ne=Iur(Js+1)
2750  IF N>Ne THEN 2910
2760  IF Nr=Iuc(N) THEN 2790
2770  N=N+1
2780  GOTO 2750
2790  Indx=N+Node
2800  Nn=N+Mloc
2810  GOTO 2960
2820  !...UPPER TRIANGLE
2830  N=Iur(Pis)
2840  Ne=Iur(Pis+1)
2850  IF N>Ne THEN 2910
2860  IF Nc=Iuc(N) THEN 2890
2870  N=N+1
2880  GOTO 2850
2890  Indx=N+Mloc
2900  Nn=N+Node
2910  GOTO 2960
2930  !...DIAGONAL LOCATION
2940  Indx=Nr
2950  Nn=Nc
2960  RETURN Indx
2970  FNEND
2980  !
2990  SUB Initl
2991  OPTION BASE 1
3000  DIM Ilc(1),Ilr(1),Relm(1)
3010  COM U(*),C(*),Y(*)
3011  COM Ds,Delt,Delta
3020  COM TO,Temp,Dtemp
3030  COM Tm(6),A(8),Csat,Vt,Vct,Ptype
3040  COM Iel,Jj,Kk,Ll,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
3050  COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
3060  COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
3070  COM A$[80],Ni(30,2),Ieln(9),Ifrst(9),Ilast(9),Ielm(1000)
```

```
3080   COM Iur(30),Iuc(120),Ipos(400),Ncon(30),Iy(30,30)
3090   !
3100   !...INITIALIZE READ/WRITE UNITS
3110   Iw=1
3120   Ir=1
3130   Iunit=Iw
3140   Lpos=0
3150   Mxlst=1000
3160   Ndmax=30
3180   !...INITIALIZE ELEMENT COUNTERS
3190   FOR K=1 TO 9
3200   Ieln(K)=0
3210   NEXT K
3230   FOR K=1 TO Mxlst
3240   Ielm(K)=0
3250   NEXT K
3270   Node=0
3280   Itemp=0
3290   Dtemp=0
3300   Iter=0
3310   Iplt=0
3320   Ni(1,2)=0
3330   Iflg=1
3340   Jflg=1
3350   TO=0
3360   SUBEND
3370   !
3380   SUB Mchek
3390   !...CHECK FOR UNDEFINED MODELS AND STORE LOCATION OF MODEL WITH ELEMEN
3401   OPTION BASE 1
3420   COM U(*),C(*),Y(*)
3421   COM Ds,Delt,Delta
3430   COM TO,Temp,Dtemp
3440   COM Tm(6),A(8),Csat,Vt,Vct,Ptype
3450   COM Iel,Jj,Kk,L1,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
3460   COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
3470   COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
3480   COM A$[80],Ni(30,2),Ieln(9),Ifrst(9),Ilast(9),Ielm(1000)
3490   COM Iur(30),Iuc(120),Ipos(400),Ncon(30),Iy(30,30)
3500   !
3510   Ierr=1
3520   FOR Iel=1 TO 6
3530   K1=Ieln(Iel)
3540   IF K1=0 THEN 3710
3550   Kpos=Ifrst(Iel)
3560   FOR J=1 TO K1
3570   M=Ielm(Kpos+3)
3580   IF M=0 THEN 3690
3590   Mpos=Ifrst(7)
3600   FOR K=1 TO Ieln(7)
3610   N=Ielm(Mpos+3)
3620   IF N<>M THEN 3650
3630   Ielm(Kpos+4)=Mpos/2
3640   GOTO 3690
3650   Mpos=Ielm(Mpos+1)
3660   NEXT K
```

```
3670  DISP "MODEL: M";M;"NOT DEFINED, ";
3680  Ierr=0
3690  Kpos=Ielm(Kpos+1)
3700  NEXT J
3710  NEXT Iel
3720  SUBEND
3730  !
3740  SUB Pinput
3760  !.. CONTROLS READING OF INPUT DATA
3761  OPTION BASE 1
3780  COM U(*),C(*),Y(*)
3781  COM Ds,Delt,Delta
3790  COM TO,Temp,Dtemp
3800  COM Tm(6),A(8),Csat,Vt,Vct,Ptype
3810  COM Iel,Jj,Kk,L1,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
3820  COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
3830  COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
3840  COM A$[80],Ni(30,2),Ieln(9),Ifrst(9),Ilast(9),Ielm(1000)
3850  COM Iur(30),Iuc(120),Ipos(400),Ncon(30),Iy(*)
3851  !
3860  K1=Ieln(Iel)+1
3870  IF K1>1 THEN 3900
3880  Ifrst(Iel)=Lpos
3890  GOTO 3920
3900  Kpos=Ilast(Iel)
3910  Ielm(Kpos+1)=Lpos
3920  L=FNIpack(A$,L1+1)
3930  Ielm(Lpos+2)=L
3940  IF Iel=7 THEN 4280
3950  !
3960  !...READ IN DATA
3970  L1=L1+3
3980  CALL Rdfld
3990  IF Mm>=0 THEN 4020
4000  K1=K1-1
4010  GOTO 4280
4020  Pis=2
4030  IF Iel<>4 THEN 4050
4040  Pis=3
4050  FOR L=1 TO Pis
4060  Ii=A(L)
4070  IF Ii=0 THEN 4170
4090  !.. DETERMINE UNIQUE NODE NUMBERS
4100  FOR M=1 TO Node
4110  IF Ii=Ni(M,2) THEN 4160
4120  NEXT M
4130  Node=Node+1
4140  Ni(Node,2)=Ii
4150  GOTO 4170
4160  Ni(M,1)=1
4170  NEXT L
4180  Ielm(Lpos+7)=A(1)
4190  Ielm(Lpos+8)=A(2)
4200  IF Iel<>4 THEN 4230
4210  Ielm(Lpos+5)=A(3)
4220  GOTO 4270
```

```
4230  IF A(3)>0 THEN 4250
4240  IF Iel<5 THEN A(3)=-A(3)
4250  Mpos=Lpos/2
4260  Ielm(2*(Mpos+3))=A(3)
4270  Ielm(Lpos+3)=A(4)
4280  Ieln(Iel)=K1
4320  SUBEND
4330  !
4340  SUB Rdfld
4350  !...READ DATA FIELD
4351  OPTION BASE 1
4370  COM U(*),C(*),Y(*)
4371  COM Ds,Delt,Delta
4380  COM TO,Temp,Dtemp
4390  COM Tm(6),A(8),Csat,Vt,Vct,Ptype
4400  COM Iel,Jj,Kk,L1,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
4410  COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
4420  COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
4430  COM A$[80],Ni(30,2),Ieln(9),Ifrst(9),Ilast(9),Ielm(1000)
4431  COM Iur(30),Iuc(120),Ipos(400),Ncon(30),Iy(*)
4441  !
4451  !.. KK IS FIELD POINTER
4461  !.. LL IS COLUMN POINTER
4462  !
4470  FOR Kk=1 TO 8
4480  A(Kk)=0.
4490  NEXT Kk
4500  FOR Kk=1 TO 8
4510  A(Kk)=FNPval(L1)
4520  IF Mm<=1 THEN 4550
4530  L1=L1+1
4540  NEXT Kk
4550  SUBEND
4560  !
4570  SUB Prckt
4590  !..  PRINT INPUT DATA
4591  OPTION BASE 1
4610  DIM Itype$[28],Name$[10]
4620  COM U(*),C(*),Y(*)
4621  COM Ds,Delt,Delta
4630  COM TO,Temp,Dtemp
4631  COM Tm(*),A(*),Csat,Vt,Vct,Ptype
4640  COM Iel,Jj,Kk,L1,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
4650  COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
4660  COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
4670  COM A$[80],Ni(30,2),Ieln(9),Ifrst(*),Ilast(9),Ielm(1000)
4680  COM Iur(30),Iuc(120),Ipos(400),Ncon(30),Iy(*)
4690  Itype$[1,2]=" P"
4700  Itype$[3,4]="NP"
4710  Itype$[5,6]="  "
4720  Itype$[7,8]="  "
4730  Itype$[9,10]=" N"
4740  Itype$[11,12]="PN"
4750  Itype$[13,14]=" T"
4760  Itype$[15,16]="EM"
4770  Itype$[17,18]=" S"
```

BIASD.36 LISTING

```
4780  Itype$[19,20]="IN"
4790  Itype$[21,22]=" E"
4800  Itype$[23,24]="XT"
4810  Itype$[25,26]=" P"
4820  Itype$[27,28]="UL"
4830  Name$="RCLQIVM +-"
4840  FOR I=1 TO 7
4850  K1=Ieln(I)
4860  IF K1=0 THEN GOTO 5260
4870  Kpos=Ifrst(I)
4880  Iplus$[1]=Name$[8;1]
4890  Minus$[1]=Name$[8;1]
4891  PRINT
4900  ON I GOTO 4910,4930,4950,5090,4970,4990,5160
4910  PRINT " RESISTORS:"
4920  GOTO 5020
4930  PRINT " CAPACITORS:"
4940  GOTO 5020
4950  PRINT " INDUCTORS:"
4960  GOTO 5020
4970  PRINT " CURRENT SOURCES:"
4980  GOTO 5000
4990  PRINT " VOLTAGE SOURCES:"
5000  Iplus$[1]=Name$[9;1]
5010  Minus$[1]=Name$[10;1]
5020  PRINT " NAME   ";Iplus$[1;1];" Nodes";Minus$[1];"    VALUE        MODEL"
5030  FOR J=1 TO K1
5040  Mpos=Kpos/2
5041  Qq=Ielm(Kpos+2)
5050  PRINT USING 5051;Name$[I],CHR$(Qq),Ielm(Kpos+7),Ielm(Kpos+8),Ielm(2*(M
5051  IMAGE X,A,A,3X,DD,5X,DD,4X,D.3DE,4X,"M",DD
5060  Kpos=Ielm(Kpos+1)
5070  NEXT J
5080  GOTO 5250
5081  PRINT
5090  PRINT " TRANSISTORS:"
5100  PRINT " NAME   C     B     E    MODEL"
5110  FOR J=1 TO K1
5111  Sb=Ielm(Kpos+2)
5120  PRINT USING 5121;Name$[I],CHR$(Sb),Ielm(Kpos+7),Ielm(Kpos+8),Ielm(Kpos
5121  IMAGE X,A,A,4X,DD,4X,DD,4X,DD,4X,"M",DD
5130  Kpos=Ielm(Kpos+1)
5140  NEXT J
5150  GOTO 5250
5151  PRINT
5160  PRINT " MODELS:"
5170  PRINT " NAME   TYPE"
5180  FOR J=1 TO K1
5190  Mpos=Kpos/2
5200  K=2*Ielm(2*(Mpos+3))+3
5210  Kk=2*(Mpos+4)
5221  Tt=1
5224  Rr(Tt)=Ielm(Kk)
5225  Tt=Tt+1
5226  Kk=Kk+2
5227  IF Tt>7 THEN 5231
```

```
5228  GOTO 5224
5231  Q$[1]=Name$[7]
5232  Z$[1]=Itype$[2*K-1;2]
5233  W$[1]=Itype$[2*(K+1)-1;2]
5234  Kq$[1]=CHR$(Ielm(Kpos+2))
5236  PRINT USING 5237;Q$,Kq$[1],Z$[1],W$[1],Rr(1),Rr(2),Rr(3),Rr(4),Rr(5),R
5237  IMAGE X,A,A,XXXX,AA,AA,5D,5D,5(X,MD.3DE)
5240  Kpos=Ielm(Kpos+1)
5250  NEXT J
5260  NEXT I
5270  SUBEND
5280  !
5290  SUB Alter
5310  !.. FIND ALTER ELEMENT
5311  OPTION BASE 1
5330  COM U(*),C(*),Y(*)
5331  COM Ds,Delt,Delta
5340  COM TO,Temp,Dtemp
5350  COM Tm(6),A(8),Csat,Vt,Vct,Ptype
5360  COM Iel,Jj,Kk,L1,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
5370  COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
5380  COM Mx1st,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
5390  COM A$[80],Ni(30,2),Ieln(9),Ifrst(9),Ilast(9),Ielm(1000)
5400  COM Iur(30),Iuc(120),Ipos(400),Ncon(30),Iy(*)
5420  !Itt CONTAINS FIRST LOCATION OF ALTERED ELEMENT IN IELM
5430  !
5440  Ierr=1
5450  K1=Ieln(Iel)
5460  Itt=Ifrst(Iel)
5470  K=FNIpack(A$,L1+1)
5480  FOR I=1 TO K1
5490  IF Ielm(Itt+2)=K THEN 5550
5500  Itt=Ielm(Itt+1)
5510  NEXT I
5520  DISP " ELEMENT NOT FOUND: RE-";
5530  Ierr=0
5540  GOTO 5580
5550  L1=L1+3
5560  IF Iel=7 THEN 5580
5570  CALL Rdfld
5580  SUBEND
5590  !
5600  DEF FNPval(Loc1)
5601  !.. DETERMINE VALUE OF FIELD
5603  OPTION BASE 1
5620  DIM Sufix(5),Ichar$[22],Y$[100]
5630  COM U(*),C(*),Y(*)
5631  COM Ds,Delt,Delta
5640  COM TO,Temp,Dtemp
5650  COM Tm(6),A(8),Csat,Vt,Vct,Ptype
5660  COM Iel,Jj,Kk,L1,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
5670  COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
5680  COM Mx1st,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Loc,Kpos,Lpos
5690  COM A$[80],Ni(30,2),Ieln(9),Ifrst(9),Ilast(9),Ielm(1000)
5700  COM Iur(30),Iuc(120),Ipos(400),Ncon(30),Iy(*)
5710  Ichar$="0123456789 .E-+,MPNUKG"
```

```
5720  Sufix(1)=1.E-12
5730  Sufix(2)=1.E-9
5740  Sufix(3)=1.E-6
5750  Sufix(4)=1.E+3
5760  Sufix(5)=1.E+9
5770  !
5780  !        VALUE OF MM (RETURNED)
5790  !    -1 ILLEGAL CHARACTER   1 MODEL
5800  !     0 BLANK FIELD         2 VALID FIELD
5801  !
5803  Y$=A$
5812  A$[LEN(A$)+1;10]="          "
5820  Sign=1.0
5830  Mp=1
5840  Pis=0
5850  Ic=0
5860  Iint=0
5870  Iexp=0
5880  Frac=0.
5890  Emult=1.0
5900  Nblnk=0
5910  Mm=0
5920  J1=0
5930  Pval=0.
5940  FOR L1=Loc1 TO LEN(A$)
5950  Ii$[1;1]=A$[L1;1]
5960  !...DETERMINE CHARACTER
5970  FOR J=1 TO 22
5980  IF Ii$[1;1]=Ichar$[J;1] THEN 6010
5990  NEXT J
6000  GOTO 6510
6010  N=1
6020  J=J-1
6030  IF J<=9 THEN 6060
6040  IF J>16 THEN 6630
6050  N=J-8
6060  ON N GOTO 6070,6550,6270,6310,6380,6690,6720,6450,6510
6070  J1=1
6080  IF Pis<0 THEN 6130
6090  IF Pis=0 THEN 6170
6100  IF Pis>0 THEN 6210
6120  !...EXPONENT PART
6130  Iexp=Iexp*10+J
6140  GOTO Cc
6160  !...INTEGER PART
6170  Iint=Iint*10+J
6180  GOTO Cc
6200  !...FRACTION PART
6210  Ic=Ic+1
6220  S=J
6230  Frac=Frac+S/FNPw10(Ic)
6240  GOTO Cc
6260  !...DECIMAL POINT
6270  Pis=1
6280  GOTO Cc
6290  !...E
```

```
6310  IF Emult=1.0 THEN 6340
6320  Emult=1.E+6
6330  GOTO Cc
6340  Pis=-1
6350  GOTO Cc
6370  !...MINUS
6380  IF J1<>0 THEN 6410
6390  Sign=-1.0
6400  GOTO Cc
6410  Mp=-1
6420  GOTO Cc
6440  !...MODEL
6450  IF J1>0 THEN 6480
6460  Mm=1
6470  GOTO Cc
6480  Emult=1.E-3
6481  GOTO Cc
6500  !...ERROR
6501  PRINT "!"
6510  DISP "ILLEGAL CHARACTER-";Ii$[1;2];
6520  Mm=-1
6530  GOTO 6760
6540  !  ALLOW FORM 1EXX
6550  IF Pis<0 AND Iexp=0 THEN Cc
6570  IF J1>0 THEN 6720
6590  !...COUNT LEADING BLANKS
6600  IF Nblnk>7 THEN 6670
6610  Nblnk=Nblnk+1
6620  GOTO Cc
6630  IF J1=0 THEN 6510
6640  Nn=J-16
6650  Emult=Sufix(Nn)
6660  GOTO Cc
6670  Mm=0
6680  GOTO 6760
6690 Cc: NEXT L1
6700  DISP " MAXIMUM FIELD LENGTH EXCEEDED: RE-INPUT DATA";
6710  GOTO 6760
6720  J=Mp*Iexp
6730  Pval=Iint
6740  Pval=Sign*(Pval+Frac)*FNPw10(J)*Emult
6750  IF Mm<>1 THEN Mm=2
6760  A$=Y$
6763  RETURN Pval
6770  FNEND
6780  !
6790  DEF FNPw10(K)
6800  !...GENERATE POWER OF TEN
6810  Pw10=1.0
6820  IF K<0 THEN 6870
6830  IF K=0 THEN 6900
6840  FOR I=1 TO K
6850  Pw10=Pw10*10.0
6851  NEXT I
6860  GOTO 6900
6870  K=-K
```

```
6880  FOR I=1 TO K
6890  Pw10=Pw10/10.0
6891  NEXT I
6900  RETURN Pw10
6910  FNEND
6920  !
6930  DEF FNNconv(K,M,Ni(*),Node)
6940  !...DETERMINE ELEMENT NODE FROM TABLE
6941  OPTION BASE 1
6960  !   M=0  CONVERT ORIG. NODE TO RENUMBERED NODE
6970  !   M=1  CONVERT RENUMBERED NODE TO ORIG.  NODE
6980  IF M=1 THEN 7040
6990  FOR J=1 TO Node
7000  IF K=Ni(J,2) THEN 7020
7010  NEXT J
7020  Nconv=J
7030  GOTO 7050
7040  Nconv=Ni(K,2)
7050  RETURN Nconv
7060  FNEND
7070  !
7080  SUB Pout
7090  !...SET UP PRINT OR PLOT OUTPUTS
7100  OPTION BASE 1
7120  DIM B$[11]
7130  COM U(*),C(*),Y(*)
7131  COM Ds,Delt,Delta
7140  COM TO,Temp,Dtemp
7150  COM Tm(6),A(8),Csat,Vt,Vct,Ptype
7160  COM Iel,Jj,Kk,L1,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
7170  COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
7180  COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
7190  COM A$[80],Ni(30,2),Ieln(9),Ifrst(9),Ilast(9),Ielm(1000)
7200  COM Iur(30),Iuc(120),Ipos(400),Ncon(30),Iy(*)
7210  B$="RCLQIVMETP "
7220  D$="DB"
7240  Iform=0
7250  IF Iflg<5 THEN 7770
7260  PRINT " OUTPUTS:"
7270  PRINT " VXX PRT/PLO XMIN XMAX VMIN VMAX"
7280  INPUT A$
7290  Iout=FNPval(2)
7300  I=L1+1
7310  L1=8
7320  CALL Rdfld
7330  IF Mm=-1 THEN 8020
7340  IF Iflg=2 THEN Iflg=6
7350  IF A$[I;1]=B$[11;1] THEN I=I+1
7360  IF A$[I;1]<>B$[10;1] THEN 7730
7370  I=I+1
7380  IF A$[I;1]<>B$[3;1] THEN 7770
7390  !.. PLOT DEFAULT: USE LAST PLOT SCALES IF PRT/PLO NOT SPECIFIED
7400  !.. PLOT OUTPUT
7410  Kpos=Mxloc/2
7420  FOR I=1 TO 6
7430  K=Kpos+I
```

```
7440   Ielm(2*K)=A(I)
7460   NEXT I
7480   !.. AC PLOTS
7490   IF Iflg<>9 THEN 7610
7500   Ltype=2
7510   IF A(1)=0 THEN A(1)=Tm(1)
7520   IF A(2)=0 THEN A(2)=Tm(2)
7530   Ielm(2*(Kpos+1))=LOG(A(1))*.434294
7550   Ielm(2*(Kpos+2))=LOG(A(2))*.434294
7560   !...Itt=1 OUTPUT DB GAIN.  Itt=2, OUTPUT PHASE
7570   Itt=Tm(4)
7580   GOTO 7730
7600   !.. TRANSIENT PLOT DEFAULT
7610   IF Iflg<>5 THEN 7680
7620   Ltype=4
7630   IF A(2)=0 THEN Ielm(2*(Kpos+2))=Tm(2)
7640   IF A(4)=0 THEN Ielm(2*(Kpos+4))=20.0
7650   GOTO 7730
7670   !.. SWEPT ELEMENT PLOT DEFAULT
7680   IF A(1)=0 THEN Ielm(2*(Kpos+1))=Tm(1)
7690   IF A(2)=0 THEN Ielm(2*(Kpos+2))=Tm(2)
7700   IF A(4)=0 THEN Ielm(2*(Kpos+4))=20.
7730   Iplt=1
7740   CALL Graph
7750   GOTO 8020
7760   !.. PRINT OUTPUT
7770   Iplt=0
7790   !.. AC PRINT
7800   IF Iflg<>9 THEN 7890
7810   A$[1;1]=B$[7;1]
7820   IF Itt=1 THEN A$[1;1]=D$[1]
7830   PRINT USING 7840;A$[1],Iout,Iout
7840   IMAGE 3X,"FREQUENCY",5X,"V",A,DD,5X,"VP",DD,"(DEG)"
7850   PRINT "---------------------------------------"
7860   GOTO 8020
7880   !.. ALTER PRINT
7890   IF Iflg<>6 THEN 7950
7900   A$[1;1]=B$[Iel;1]
7910   A$[2;1]=CHR$(Ielm(Itt+2))
7920   GOTO 7990
7940   !.. TRANSIENT PRINT
7950   IF Iflg<>5 THEN 8020
7960   A$[1;1]=B$[9;1]
7970   A$[2;1]=B$[11;1]
7990   PRINT USING 8000;A$[1],A$[2],Iout
8000   IMAGE 13X,A,A,15X,"V",DD
8010   PRINT "            +++++++++++++++++++++++++++++"
8020   SUBEND
8030   !
8040   SUB Setup
8050   !...PROCESS CIRCUIT DATA
8051   OPTION BASE 1
8070   DIM Nsorc(80)
8080   COM U(*),C(*),Y(*)
8081   COM Ds,Delt,Delta
8090   COM TO,Temp,Dtemp
```

BIASD.36 LISTING

```
8100  COM Tm(6),A(8),Csat,Vt,Vct,Ptype
8110  COM Iel,Jj,Kk,Ll,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
8120  COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
8130  COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
8140  COM A$[80],Ni(30,2),Ieln(9),Ifrst(9),Ilast(9),Ielm(1000)
8150  COM Iur(30),Iuc(120),Ipos(400),Ncon(30),Iy(30,30)
8160  !
8170  !...INITIALIZE NODE VECTOR
8180  !...REORDER NODE VECTOR
8181  Nsorc$=A$
8190  N=Node-1
8200  FOR I=1 TO N
8210  Ii=I+1
8220  FOR J=Ii TO Node
8230  IF Ni(I,2)<Ni(J,2) THEN 8261
8240  M=Ni(J,2)
8250  Ni(J,2)=Ni(I,2)
8260  Ni(I,2)=M
8261  NEXT J
8270  NEXT I
8280  IF Ieln(6)=0 THEN 8500
8300  !...RENUMBER VOLTAGE SOURCE NODES
8310  Kpos=Ifrst(6)
8320  FOR I=1 TO Ieln(6)
8330  K=Ielm(Kpos+7)
8340  Nsorc(I)=FNNconv(K,0,Ni(*),Node)
8343  Nsorc(I)=FNNconv(K,0,Ni(*),Node)
8350  Kpos=Ielm(Kpos+1)
8360  NEXT I
8380  !.. MOVE SOURCE NODES TO END OF NODE VECTOR
8390  N=Node
8400  Kpos=Ifrst(6)
8410  FOR I=1 TO Ieln(6)
8420  K=Nsorc(I)
8430  IF K=N THEN 8540
8440  IF K>Nnode THEN 8550
8450  Ii=I+1
8460  FOR L=Ii TO Ieln(6)
8470  IF Nsorc(L)<>N THEN 8500
8480  N=N-1
8490  GOTO 8460
8500  NEXT L
8510  M=Ni(K,2)
8520  Ni(K,2)=Ni(N,2)
8530  Ni(N,2)=M
8540  N=N-1
8550  Kpos=Ielm(Kpos+1)
8560  NEXT I
8580  !...RENUMBER ELEMENT NODES
8590  CALL Renum(0)
8610  !...GENERATE INCIDENCE MATRIX FOR NLDE REORDERING
8620  CALL Indmt
8640  !...DETERMINE NEW NODE ORDER AND SET UP SPARSE POINTERS
8650  CALL Nordr
8670  !...REDUCE VOLTAGE SOURCES TO CURRENT EQUIVALENTS
8680  IF Ieln(6)=0 THEN 8710
```

```
8690  CALL Equiv
8700  Mxloc=Lpos
8713  SUBEND
8720  !
8730  SUB Indmt
8740  !.. ROUTINE TO LOAD INCIDENCE MATRIX FOR NODE RECORDING
8750  OPTION BASE 1
8780  COM U(30),C(60),Y(800)
8790  COM Ds,Delt,Delta
8800  COM TO,Temp,Dtemp
8810  COM Tm(6),A(8),Csat,Vt,Vct,Ptype
8820  COM Iel,Jj,Kk,L1,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
8830  COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
8840  COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
8850  COM A$[80],Ni(30,2),Ieln(9),Ifrst(9),Ilast(9),Ielm(1000)
8860  COM Iur(30),Iuc(120),Ipos(400),Ncon(30),Iy(*)
8880  !  EQUIVALENCE: ARRAY(I,J)=LINEAR[(J-1)*DIMENSION+I]: DIMENSION IS THA
8890  !...CLEAR IY MATRIX
8900  FOR I=1 TO Nnode
8910  FOR J=1 TO Nnode
8920  Iy(I,J)=0
8930  NEXT J
8940  NEXT I
8960  !...LOAD INCIDENCE MATRIX
8970  FOR Iel=1 TO 5
8980  K1=Ieln(Iel)
8990  IF K1=0 THEN 9340
9000  Kpos=Ifrst(Iel)
9010  FOR J=1 TO K1
9020  Kk=Ielm(Kpos+7)
9030  L1=Ielm(Kpos+8)
9040  IF Kk=0 THEN 9140
9050  Iy(Kk,Kk)=1
9070  !   IN FORTRAN EQV: Y(1),IY(1,1)
9090  IF L1=0 THEN 9160
9100  Iy(Kk,L1)=1
9120  Iy(L1,Kk)=1
9140  Iy(L1,L1)=1
9160  IF Iel<>4 THEN 9320
9170  !...ADD BJT'S
9180  Mm=Ielm(Kpos+5)
9190  IF Mm=0 THEN 9320
9200  IF Kk=0 THEN 9250
9210  Iy(Kk,Mm)=1
9230  Iy(Mm,Kk)=1
9250  IF L1=0 THEN 9300
9260  Iy(Mm,L1)=1
9280  Iy(L1,Mm)=1
9300  Iy(Mm,Mm)=1
9320  Kpos=Ielm(Kpos+1)
9330  NEXT J
9340  NEXT Iel
9350  SUBEND
9360  !
9370  SUB Nordr
9380  !...ROUTINE TO OPTIMALLY ORDER NODES USING
```

```
9390  !   NON-ZERO OFF-DIAGONAL TERMS (REF.BIAS-N)
9400  !   5-18-77
9402  OPTION BASE 1
9411  DIM Irow(50)
9420  COM U(30),C(60),Y(800)
9430  COM Ds,Delt,Delta
9440  COM TO,Temp,Dtemp
9450  COM Tm(6),A(8),Csat,Vt,Vct,Ptype
9460  COM Iel,Jj,Kk,L1,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
9470  COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
9480  COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
9490  COM A$[80],Ni(30,2),Ieln(9),Ifrst(9),Ilast(9),Ielm(1000)
9500  COM Iur(*),Iuc(*),Ipos(*),Ncon(*),Iy(*)
9530  Nm1=Nnode-1
9540  Nps=0
9550  Nct=0
9560  IF Nnode=1 THEN 9750
9570  FOR I=1 TO Node
9580  !  Iordr(I)=I
9597  Ni(I,1)=I
9600  NEXT I
9601  Knt=0
9610  !...COUNT NUMBER OF OFF-DIAGONAL NON-ZERO ELEMENTS IN ROWS
9620  FOR I=1 TO Nnode
9630  Ncon(I)=I
9640  Irow(I)=0
9660  FOR J=1 TO Nnode
9670  IF I=J THEN 9710
9680  IF Iy(I,J)=0 THEN 9710
9690  Irow(I)=Irow(I)+1
9710  NEXT J
9720  NEXT I
9740  !...  COLUMN AND ROW RENUMBERING AND INDICATOR SETUP
9750  Iu=1
9760  !  Ic=1
9770  Ifill=0
9780  IF Nnode<=1 THEN 10380
9790  FOR I=1 TO Nm1
9800  Iur(I)=Iu
9810  ! ILC(I)=IL
9820  L=Ni(I,1)
9840  !...SEARCH FOR MIN IROW
9850  Nmin=500
9860  FOR J=I TO Nnode
9870  Nr=Ni(J,1)
9880  IF Irow(Nr)>=Nmin THEN 9990
9890  Nmin=Irow(Nr)
9900  !  Iordr(J)=L
9920  Ni(J,1)=L
9930  Ncon(L)=J
9940  !  Iordr(I)=Nr
9960  Ni(I,1)=Nr
9970  Ncon(Nr)=I
9980  L=Nr
9990  NEXT J
10010 !.. ESTABLISH NON-ZERO TERMS IN ROW
```

```
10020 Js=I+1
10030 FOR K=Js TO Nnode
10032 Ic=Ni(K,1)
10040 !  Ic=Iordr(K)
10041 IF Iy(L,Ic)=0 THEN 10080
10060 Iuc(Iu)=Ic
10070 Iu=Iu+1
10080 NEXT K
10100 !.. MOVE DOWN COLUMN AND CHECK FILL-INS
10110 FOR J=Js TO Nnode
10112 Nr=Ni(J,1)
10120 !  Nr=Iordr(J)
10121 Irt=Iy(Nr,L)
10140 IF Irt=0 THEN 10350
10150 ! Ilr(Il)=Nr
10160 Irow(Nr)=Irow(Nr)-1
10180 ! IL=IL+1
10190 Nct=Iur(I)
10200 IF Nct>=Iu THEN 10350
10220 !.. MOVE INDEX ,IC, ACROSS ROW L
10230 Ic=Iuc(Nct)
10240 IF Nr=Ic THEN 10320
10241 IF Iy(Nr,Ic)<>0 THEN 10320
10270 !... OFF DIAGONAL FILL-IN
10280 Irow(Nr)=Irow(Nr)+1
10300 Ifill=Ifill+1
10301 Iy(Nr,Ic)=1
10320 Nct=Nct+1
10330 Nps=Nps+1
10340 GOTO 10200
10350 NEXT J
10360 NEXT I
10370 ! ILC(NNODE)=IL
10380 Iur(Nnode)=Iu
10390 Mloc=Node+Iu
10400 Mxpos=Lpos
10420 !.. PRINT MATRIX STATISTICS
10430 Iut=Iu-1
10440 Nops=Nps+Nnode+3*Iut
10450 I=Nnode*Nnode
10460 J=Node+2*Iu
10470 Nct=100*(I-2*Iut)/I
10480 PRINT "   NNODE =";Nnode;"          IU=";Iut
10490 PRINT "    NOPS =";Nops;"    FILL-INS=";Ifill
10500 PRINT "SPARSITY =";Nct;"%";"    MXPOS=";Mxpos;"    MXYPOS=";J
10501 PRINT
10502 PRINT
10510 J=Node+2*Iu
10520 IF J>300 THEN PRINT "** MARTRIX TOO DENSE **"
10540 !.. ASSIGN OPERATION NUMBERS
10550 IF Nnode=1 THEN 10730
10560 Nps=0
10570 FOR I=1 TO Nm1
10580 Ius=Iur(I)
10590 Iue=Iur(I+1)
10600 Il=Ius
```

```
10610 Ile=Iue
10620 IF Il>=Ile THEN 10720
10630 Nr=Iuc(Il)
10640 Il=Il+1
10650 Iu=Ius
10660 IF Iu>=Iue THEN 10620
10670 Nc=Iuc(Iu)
10680 Nps=Nps+1
10690 Ipos(Nps)=FNIndx(Nr,Nc)
10700 Iu=Iu+1
10710 GOTO 10660
10720 NEXT I
10721 IF Nps>400 THEN PRINT "** MATRIX TOO DENSE **"
10730 SUBEND
10750 SUB Cnv(I,J,K,Row)
10760 SUBEND
10830 !
10840 SUB Renum(M)
10860 !.. RENUMBER ELEMENT NODES
10861 OPTION BASE 1
10870 COM U(30),C(60),Y(800)
10880 COM Ds,Delt,Delta
10890 COM TO,Temp,Dtemp
10900 COM Tm(6),A(8),Csat,Vt,Vct,Ptype
10910 COM Iel,Jj,Kk,Ll,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
10911 COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
10920 COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
10930 COM A$[80],Ni(*),Ieln(*),Ifrst(*),Ilast(*),Ielm(*)
10940 COM Iur(*),Iuc(*),Ipos(*),Ncon(*),Iy(*)
10960 FOR Iel=1 TO 6
10970 K1=Ieln(Iel)
10980 IF K1=0 THEN 11080
10990 Kpos=Ifrst(Iel)
11000 FOR J=1 TO K1
11001 Kk=7
11002 IF Iel=4 THEN Kk=5
11003 FOR I=Kk TO 8
11010 IF I=6 THEN 11060
11020 Ii=Kpos+I
11030 K=Ielm(Ii)
11040 IF K=0 THEN 11060
11050 Ielm(Ii)=FNNconv(K,M,Ni(*),Node)
11060 NEXT I
11070 Kpos=Ielm(Kpos+1)
11071 NEXT J
11080 NEXT Iel
11090 SUBEND
11110 SUB Equiv
11120 !.. STORE LOCATION OF EQUIVALENT SOURCES
11121 OPTION BASE 1
11130 COM U(*),C(*),Y(*)
11140 COM Ds,Delt,Delta
11150 COM TO,Temp,Dtemp
11160 COM Tm(*),A(*),Csat,Vt,Vct,Ptype
11170 COM Iel,Jj,Kk,Ll,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
11180 COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
```

```
11190 COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
11200 COM A$[80],Ni(*),Ieln(*),Ifrst(*),Ilast(*),Ielm(*)
11210 COM Iur(*),Iuc(*),Ipos(*),Ncon(*),Iy(*)
11230 Kpos=Ifrst(6)
11240 Lpos=Mxpos
11250 FOR N=1 TO Ieln(6)
11260 J=Ielm(Kpos+7)
11280 !.. CHECK IF ELEMENT CONNECTED TO VOLTAGE SOURCE
11290 FOR Iel=1 TO 3
11300 K1=Ieln(Iel)
11310 IF K1=0 THEN 11560
11320 Mpos=Ifrst(Iel)
11330 FOR M=1 TO K1
11340 K=Ielm(Mpos+7)
11350 L=Ielm(Mpos+8)
11360 IF J<>K THEN 11390
11370 Nt=L
11380 GOTO 11410
11390 IF L<>J THEN 11560
11400 Nt=K
11410 IF Nt=0 THEN 11560
11420 Ieln(9)=Ieln(9)+1
11430 IF Ieln(9)=1 THEN Ifrst(9)=Mxpos
11440 IF Lpos<=Mxlst THEN 11510
11450 Ierr=-2
11460 !... STORE EQUIVALENT SOURCE FLAGS AND VALUES
11470 !    LPOS+1= ELEMENT LOCATION IN IELN(.)
11480 !    LPOS+2= VOLTAGE SOURCE LOCATON IN IELN(.)
11490 !    LPOS+3= NODE AT WHICH EQUIVALENT CURRENT IS ADDED
11500 !    LPOS+4= ELEMENT TYPE
11510 Ielm(Lpos+4)=Iel
11520 Ielm(Lpos+3)=Nt
11530 Ielm(Lpos+2)=Kpos/2
11540 Ielm(Lpos+1)=Mpos/2
11550 Lpos=Lpos+4
11560 Mpos=Ielm(Mpos+1)
11570 NEXT M
11571 NEXT Iel
11580 Kpos=Ielm(Kpos+1)
11590 NEXT N
11600 SUBEND
11610 !
11620 SUB Graph
11630 !.. INITIALIZE GRAPHICS
11640 !.. DRAW AXIS AND LABEL GRAPH
11650 SUBEND
11660 !
11670 DEF FNIpack(A$,K)
11680 !.. PACK 2A1 FORMAT IN FORTRAN INTO I1 WORD
11684 Q$[1;1]=A$[K;1]
11685 Ii=NUM(Q$[1])
11686 !.. JJ=NUM(A$[K+1])
11687 !.. IPACK=OR(SHFT(II,8,-8),SHFT(JJ,8))
11688 Ipack=Ii
11690 RETURN Ipack
11700 FNEND
```

BIASD.36 LISTING

```
11710 !
11780 SUB Analy
11790 !.. MAIN ANALYSIS ROUTINE
11800 OPTION BASE 1
11810 COM U(*),C(*),Y(*)
11820 COM Ds,Delt,Delta
11830 COM TO,Temp,Dtemp
11840 COM Tm(6),A(8),Csat,Vt,Vct,Ptype
11850 COM Iel,Jj,Kk,Ll,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
11860 COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
11870 COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
11880 COM A$[80],Ni(*),Ieln(*),Ifrst(*),Ilast(*),Ielm(*)
11890 COM Iur(*),Iuc(*),Ipos(*),Ncon(*),Iy(*)
11910 Err=Nnode*Nnode*1.E-10
11920 Rms1=0.
11930 Rms2=0.
11940 IF Iflg<>5 THEN 12000
11950 IF TO<>Delta THEN 11980
11960 Delt=Delta
11970 !.. UPDATE TRANSIENT SOURCES
11980 CALL Updat
11990 !.. UPDATE CAPACITOR CURRENTS IELN(2)=C1
12000 IF Ilen(2)<=0 THEN 12200
12010 Kpos=Ifrst(2)
12020 Tc=1.0
12030 IF Itemp=0 THEN 12050
12040 Tc=1.0+Ielm(2*(Itemp+6))*Dtemp+Ielm(2*(Itemp+7))*Dtemp*Dtemp
12050 FOR I=1 TO Ieln(2)
12060 Mpos=Kpos/2
12070 Ds=0.
12080 IF TO<=0 THEN 12140
12090 Ds=Ielm(2*(Mpos+3))*Tc
12100 Kk=Ielm(Kpos+7)
12110 Ll=Ielm(Kpos+8)
12120 Ds=2.0*Ds*FNDelu(Ll,Kk)/Delt
12130 IF TO>Delta THEN 12143
12131 Ielm(2*(Mpos+5))=0.
12132 Ielm(2*(Mpos+6))=-Ds
12142 GOTO 12160
12143 Ielm(2*(Mpos+5))=Ds+Ielm(2*(Mpos+6))
12150 Ielm(2*(Mpos+6))=-Ds-Ielm(2*(Mpos+5))
12160 Kpos=Ielm(Kpos+1)
12170 NEXT I
12190 !.. UPDATE INDUCTOR CURRENTS
12191 !    L1=IELN(3)
12200 IF Ieln(3)=0 THEN 12380
12210 Tc=1.0
12220 Kpos=Ifrst(3)
12230 FOR I=1 TO Ieln(3)
12240 Mpos=Kpos/2
12250 IF TO<=0 THEN 12290
12260 Ds=Ielm(2*(Mpos+3))*Tc
12270 Ds=.5*Delt*FNDelu(Ll,Kk)/Ds
12280 IF TO>Delta THEN 12320
12290 Ielm(2*(Mpos+5))=0.
12300 Ielm(2*(Mpos+6))=FNDelu(Ll,Kk)*100.00
```

```
12310 GOTO 12350
12320 Ielm(2*(Mpos+5))=Ds+Ielm(2*(Mpos+6))
12330 Ielm(2*(Mpos+6))=Ielm2*((Mpos+5))+Ds
12340 Kpos=Ielm(Kpos+1)
12350 NEXT I
12370 !.. ADD SUPPLIES TO VOLTAGE VECTOR
12371 !   V1=Ieln(6)
12380 IF Ieln(6)=0 THEN 12470
12390 Kpos=Ifrst(6)
12400 FOR I=1 TO Ieln(6)
12410 Mpos=Kpos/2
12420 J=Ielm(Kpos+7)
12430 C(J)=Ielm(2*(Mpos+3))
12440 U(J)=C(J)
12450 Kpos=Ielm(Kpos+1)
12460 NEXT I
12470 Iter=0
12490 !.. ZERO CURRENT MATRIX
12500 FOR I=1 TO Nnode
12510 C(I)=0.
12520 NEXT I
12530 Ii=2*Mloc-Node
12540 FOR J=1 TO Ii
12550 Y(J)=0.
12560 NEXT J
12580 !.. LOAD ELEMENTS INTO Y & C ARRAYS
12597 !   I2=IELN(9)
12650 CALL Eload
12653 IF Ieln(9)=0 THEN 12658
12654 !.. ADD GENERATED CURRENT SOURCES
12656 CALL Gncur
12657 !.. SOLVE NODE EQUATIONS
12658 CALL                                  BIASD.36 LISTING
12670 CALL Solve
12672 IF Ieln(4)=0 THEN 12860
12680 Iter=Iter+1
12690 IF Iter<100 THEN 12760
12710 !.. NO CONVERGENCE -- PRINT LAST NODE VOLTAGES
12720 PRINT "CIRCUIT DOES NOT CONVERGE"
12730 GOTO 13290
12750 !.. COMPUTE MEAN SQUARE EROR OF NODE VOLTAGE CHANCES
12760 Ds=0.
12770 FOR J=1 TO Nnode
12780 Ds=Ds+(C(J)-U(J))^2
12790 NEXT J
12800 S=Ds
12810 IF Iflg=5 OR Iflg=6 THEN 12860
12820 PRINT USING 12830;S
12830 IMAGE "      S=",D.3DE
12850 !.. STORE LAST NODE VOLTAGES
12860 FOR I=1 TO Nnode
12870 U(I)=C(I)
12880 NEXT I
12890 IF Ieln(4)=0 THEN 13030
12910 !.. CHECK FOR CONVERGENCE
12920 IF S<Err AND Rms1<Err AND Rms2<Err AND Iter>2 THEN 13030
```

```
12930 IF Iter<6 THEN 12950
12940 IF S>=Rms1 AND Rms1>=Rms2 AND S<.001 THEN 13000
12950 Rms2=Rms1
12960 Rms1=S
12980 !. NO CONVERGENCE: RE-ITERATE
12990 GOTO 12500
13000 IF Iplt>0 THEN 13030
13010 S=SQR(S)
13020 PRINT " MEAN ERROR(VOLTS):";S
13030 IF Iflg<>5 AND Iflg<>6 THEN 13280
13040 Itotl=Itotl+Iter
13050 R=TO
13060 S=C(Jout)
13070 IF Iflg=6 THEN R=Tm(1)
13080 IF Iplt=0 THEN 13120
13090 !.. CALL DRAW(R,S,IPEN,4)
13100 Ipen=1
13110 GOTO 13140
13120 PRINT USING 13130;R,S,Iter
13130 IMAGE 10X,D.3DE,7X,MD.3DE,10X,18D
13140 IF Iflg=6 THEN 13200
13150 TO=TO+Delta
13160 IF TO>Tm(2) THEN 13430
13170 GOTO 11950
13190 !.. INCREMENT SWEPT ALTER VALUE
13200 Tm(1)=Tm(1)+Tm(3)
13210 Mpos=Tm(4)
13220 Ielm(2*(Mpos+3))=Tm(1)
13230 IF Tm(1)<=Tm(2) THEN 11920
13240 Ielm(2*(Mpos+3))=Tm(5)
13250 GOTO 13430
13270 !.. PRINT DC OPERATING POINTS
13271 !   Q1=IELN(4)
13280 IF Ieln(4)=0 THEN 13310
13290 PRINT
13300 PRINT "ITERATIONS:";Iter
13310 Tc=Temp-273
13320 PRINT USING 13330;Tc
13330 IMAGE /,/," T=",8D,"DEG C",/,/
13340 PRINT " NODE VOLTAGES:"
13350 FOR I=1 TO Node
13360 J=Ni(I,2)
13370 PRINT USING 13380;J,C(I)
13380 IMAGE " V",DD,10D.4D
13390 NEXT I
13400 Iflg=4
13410 IF Ieln(4)<>0 THEN CALL Eload
13430 Iflg=4
13440 SUBEND
13450 !
13460 SUB Eload
13480 !.. SUBROUTINE TO LOAD ELEMENTS INTO Y & C ARRAYS
13490 !   FOR AC OR DC ANALYSES
13500 OPTION BASE 1
13510 COM U(*),C(*),Y(*)
13520 COM Ds,Delt,Delta
```

```
13530 COM TO,Temp,Dtemp
13540 COM Tm(*),A(*),Csat,Vt,Vct,Ptype
13550 COM Iel,Jj,Kk,Ll,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
13560 COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
13570 COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mlox,Kpos,Lpos
13580 COM A$[80],Ni(*),Ieln(*),Ifrst(*),Ilast(*),Ielm(*)
13590 COM Iur(*),Iuc(*),Ipos(*),Ncon(*),Iy(*)
13600 !
13610 FOR Iel=1 TO 3
13620 K1=Ieln(Iel)
13630 IF K1=0 THEN 14030
13640 Kpos=Ifrst(Iel)
13650 Tc=1.0
13660 IF Itemp=0 THEN 13690
13670 Nn=Itemp+Iel*2+1
13680 Tc=1.0+Ielm(2*Nn)*Dtemp+Ielm(2*(Nn+1))*Dtemp^2
13690 FOR I=1 TO K1
13700 IF Ielm(Kpos+4)<>0 THEN CALL Elmod(Iel)
13710 Mpos=Kpos/2
13720 Kk=Ielm(Kpos+7)
13730 Ll=Ielm(Kpos+8)
13740 Ds=Ielm(2*(Mpos+3))*Tc
13760 !.. ADD RESISTORS
13770 IF Iel>1 THEN 13830
13780 Ds=1./Ds
13790 CALL Adres
13800 GOTO 14010
13820 !.. ADD CAPACITORS
13830 IF Iel>2 THEN 13920
13840 Ds=Ds/Delt
13850 IF Iflg<>9 THEN 13880
13860 CALL Adcpr
13870 GOTO 14010
13880 Ds=2.0*Ds
13890 GOTO 13980
13910 !.. ADD INDUCTORS
13911 !   I1=Ieln(5)
13920 Ds=Delt/Ds
13930 IF Iflg<>9 THEN 13960
13940 CALL Adcpr
13950 GOTO 14010
13960 Ds=Ds*.5
13970 IF Delt>1.0E+6 THEN Ds=100
13980 CALL Adres
13990 Ds=Ielm(2*(Mpos+6))
14000 CALL Adcur
14010 Kpos=Ielm(Kpos+1)
14020 NEXT I
14030 NEXT Iel
14040 IF Ieln(5)=0 THEN 14190
14060 !.. ADD CURRENT SOURCE FOR DC ANALYSIS
14070 IF Iflg=9 THEN 14190
14080 Kpos=Ifrst(5)
14090 FOR I=1 TO Ieln(5)
14100 Mpos=Kpos/2
14110 Kk=Ielm(Kpos+7)
```

```
14120 L1=Ielm(Kpos+8)
14130 Ds=Ielm(2*(Mpos+3))
14140 CALL Adcur
14150 Kpos=Ielm(Kpos+1)
14160 NEXT I
14180 !.. ADD TRANSISTORS
14181 !   Q1=IELN(4)
14190 IF Ieln(4)=0 THEN 14240
14200 IF Iflg=9 THEN 14230
14210 CALL Bjt
14220 GOTO 14240
14230 CALL Bjtac
14240 SUBEND
14250 !
14260 SUB                                    BIASD.36 LISTING
14280 !.. PERFORMS LU DECOMPOSITION BASED ON RECORDED SPARSITY
14281 !   MODIFIED FROM SINC-S   5-18-77
14290 !
14300 OPTION BASE 1
14310 COM U(*),C(*),Y(*)
14320 COM Ds,Delt,Delta
14330 COM TO,Temp,Dtemp
14340 COM Tm(*),A(*),Csat,Vt,Vct,Ptype
14350 COM Iel,Jj,Kk,L1,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
14360 COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
14370 COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
14380 COM A$[80],Ni(*),Ieln(*),Ifrst(*),Ilast(*),Ielm(*)
14390 COM Iur(*),Iuc(*),Ipos(*),Ncon(*),Iy(*)
14400 !
14410 !.. IUR UPPER TRIANGULAR ROW ELEMENT COUNTER/INDICATOR
14420 !   ILC LOWER TRIANGULAR COLUMN ELEMENT COUNTER/INDICATOR
14480 IF Nnode=1 THEN 14750
14490 Nn=Nnode-1
14500 Knt=0
14510 FOR I=1 TO Nn
14520 !L=IORDR(I)
14540 L=Ni(I,1)
14550 Ius=Iur(I)+Mloc
14560 Iue=Iur(I+1)+Mloc
14561 !IL=ILC(I)+NODE:
14570 Il=Iur(I)+Node
14571 !ILE=ILC(I)+NODE:
14580 Ile=Iur(I+1)+Node
14590 !
14600 !.. DOWN LOWER TRIANGLE COLUMNS
14610 IF Il>=Ile THEN 14740
14620 Ds=Y(Il)/Y(L)
14630 Y(Il)=Ds
14640 Il=Il+1
14650 Iu=Ius
14660 !
14670 !.. ACROSS UPPER TRIANGLE ROWS
14680 IF Iu>=Iue THEN 14610
14690 Knt=Knt+1
14700 K=Ipos(Knt)
14710 Y(K)=Y(K)-Y(Iu)*Ds
```

```
14720 Iu=Iu+1
14730 GOTO 14680
14740 NEXT I
14750 SUBEND
14760 !
14770 SUB Solve
14780 !
14790 !.. PERFORMS FORWARD AND BACKWARD SUBSTITUTION
14800 !   USING SPARSE POINTERS
14810 !   FROM BIAS-N 5-19-77
14820 OPTION BASE 1
14830 COM U(*),C(*),Y(*)
14831 COM Ds,Delt,Delta
14840 COM TO,Temp,Dtemp
14850 COM Tm(*),A(*),Csat,Vt,Vct,Ptype
14860 COM Iel,Jj,Kk,Ll,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
14870 COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
14880 COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
14890 COM A$[80],Ni(*),Ieln(*),Ifrst(*),Ilast(*),Ielm(*)
14900 COM Iur(*),Iuc(*),Ipos(*),Ncon(*),Iy(*)
14910 !
14920 !.. FORWARD SUBSTITUTION
14930 Nn=Nnode-1
14940 IF Nn>0 THEN 14970
14943 !V(1)=C(1)/Y(1)
14950 C(1)=C(1)/Y(1)
14960 GOTO 15380
14970 FOR I=1 TO Nn
14980 !L=IORDR(I)
15000 L=Ni(I,1)
15010 Ds=C(L)
15020 IF Ds=0. THEN 15110
15030 Il=Iur(I)
15040 Ile=Iur(I+1)
15050 Nl=Il+Node
15060 IF Il>=Ile THEN 15110
15070 Nr=Iuc(Il)
15080 C(Nr)=C(Nr)-Y(Nl)*Ds
15090 Il=Il+1
15100 GOTO 15050
15110 NEXT I
15120 !
15130 !. BACK SUBSTITUTION
15140 !L=IORDR(NNODE)
15160 L=Ni(Nnode,1)
15170 C(L)=C(L)/Y(L)
15180 FOR I=1 TO Nn
15190 Nui=Nnode-I
15200 !L=IORDR(NUI)
15220 L=Ni(Nui,1)
15230 Iu=Iur(Nui)
15240 Iue=Iur(Nui+1)
15250 Nl=Iu+Mloc
15260 IF Iu>=Iue THEN 15310
15270 Ic=Iuc(Iu)
15280 C(L)=C(L)-Y(Nl)*C(Ic)
```

```
15290 Iu=Iu+1
15300 GOTO 15250
15310 C(L)=C(L)/Y(L)
15320 NEXT I
15321 FOR Lx=1 TO 3
15324 NEXT Lx
15330 !
15380 SUBEND
15390 !
15400 SUB Elmod(Iel)
15401 !
15410 !.. ROUTINE TO DEFINE ELEMENT MODELS
15420 ON Iel GOTO Res,Cap,Ind
15430 !
15440 !.. RETURN RESISTOR MODEL
15450 Res:SUBEXIT
15460 !
15470 !.. RETURN CAPACITOR MODEL
15480 Cap:SUBEXIT
15490 !
15500 !.. RETURN INDUCTOR MODEL
15510 Ind:SUBEXIT
15520 SUBEND
15530 !
15540 SUB Bjt
15550 !
15560 !.. COMPUTE BJT PARAMETERS
15570 OPTION BASE 1
15580 COM U(*),C(*),Y(*)
15590 COM Ds,Delt,Delta
15600 COM TO,Temp,Dtemp
15610 COM Tm(*),A(*),Csat,Vt,Vct,Ptype
15620 COM Iel,Jj,Kk,Ll,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
15630 COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
15640 COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
15650 COM A$[80],Ni(*),Ieln(*),Ifrst(*),Ilast(*),Ielm(*)
15660 COM Iur(*),Iuc(*),Ipos(*),Ncon(*),Iy(*)
15670 !
15680 IF Iflg<>4 THEN 15710
15681 PRINT
15682 PRINT
15690 PRINT " TRANSISTOR OPERATING POINTS:"
15700 PRINT " NAME     IB          IC          VBE         VBC        BETA
15710 Vt=8.6164E-5*Temp
15720 Vct=Vt*LOG(Vt/1.41459)
15730 COO=1.0
15740 Tc=1.0
15750 Kpos=Ifrst(4)
15760 IF Itemp=0 THEN 15790
15770 COO=(Temp/300)^3*EXP(-13920.0*(1.0/Temp-1.0/300.))
15780 Tc=1.0+Dtemp*Ielm(2*(Itemp+7))+Dtemp^2*Ielm(2*(Itemp+8))
15781 !   Q1=IELN(4)
15790 FOR I=1 TO Ieln(4)
15793 CO=COO
15800 Kk=Ielm(Kpos+7)
15810 Ll=Ielm(Kpos+8)
```

```
15820 Mm=Ielm(Kpos+5)
15830 Mpos=Kpos/2
15840 Itt=Ielm(Kpos+4)
15850 Ptype=Ielm(2*(Itt+3))
15860 Bf0=Ielm(2*(Itt+4))
15870 Br0=Ielm(2*(Itt+5))
15880 Cs0=Ielm(2*(Itt+6))
15900 !.. INITIALIZE PARAMETERS FOR FIRST ITERATION
15910 IF Iter<>0 THEN 16010
15920 IF Iflg<>1 THEN 16010
15930 Vbe=Vct-Vt*LOG(C0*Cs0)
15940 Vbc=-1.0
15950 Ielm(2*(Mpos+5))=.5
15960 Ielm(2*(Mpos+6))=0.
15970 Ielm(2*(Mpos+7))=1.0E-4
15980 Ielm(2*(Mpos+8))=1.0E-12
15990 GOTO 16030
16000 !
16010 Vbe=FNDelu(Mm,L1)*Ptype
16020 Vbc=FNDelu(Kk,L1)*Ptype
16030 Va=Ielm(2*(Itt+7))
16040 Va1=1.0
16050 IF Vbc>=0 THEN 16100
16060 IF Vbc<-Va THEN 16100
16070 Va1=1.0-Vbc/Va
16080 !
16090 !.. PROCESS FORWARD TRANSISTOR
16100 Bf=Bf0*Tc*Va1
16110 C0=C0*Va1
16120 Csat=C0*Cs0*(1.0+1.0/Bf0)/(1.0+1.0/Bf)
16130 CALL Junct(Mpos+5,Vbe,Ccc,Cceq,Gmf)
16140 IF Ielm(2*(Itt+8))<>0. THEN 16210
16150 Grpi=0.
16160 Creq=0.
16170 Crec=0.
16180 GOTO 16270
16200 !.. INCLUDE GENERATION RECOMBINATION CURRENT
16210 Crsat=SQR(Ielm(2*(Itt+8))*Csat)/Bf0
16220 Crec=-Crsat
16230 IF Vbe<-1.2 THEN 16250
16240 Crec=Crsat*EXP(Vbe/Vt/2.)+Crec
16250 Grpi=.5*(Crec+Crsat)/Vt
16260 Creq=Ptype*(Crec-Grpi*Vbe)
16270 Gpif=Gmf/Bf+Grpi
16290 !.. PROCESS REVERSE TRANSISTOR
16300 Br=Br0*Tc*Va1
16310 Csat=C0*Cs0*(1.0+1.0/Br0)/(1.0+1.0/Br)
16320 CALL Junct(Mpos+6,Vbc,Cec,Ceeq,Gmr)
16330 Gpir=Gmr/Br
16340 Gmr=Gmr+(Ccc-Cec)/Va
16350 IF Iflg<>4 THEN 16490
16360 !
16370 !.. PRINT TRANSISTOR OPERATING POINTS
16380 Cb=Ptype*(Ccc/Bf+Cec/Br+Crec)
16390 Cc=Ptype*(Ccc-Cec-Cec/Br)
16400 Vbe=Ptype*Vbe
```

```
16410 Vbc=Ptype*Vbc
16420 Bf=Cc/Cb
16430 Gpif=1.0/Gpif
16440 PRINT USING 16450;CHR$(Ielm(Kpos+2)),Cb,Cc,Vbe,Vbc,Bf,Gmf,Gpif
16450 IMAGE X,"Q",AA,1X,2(X,MD.3DE),2X,3D.4D,2X,3D.4D,3X,3D.2D,X,2(X,MD.3DE
16460 GOTO 16900
16470 !
16480 !.. GND. CONDUCTANCES AND V.D.C.S.CONNECTED TO SUPPLIES
16490 IF Kk<=Nnode THEN 16520
16500 Ceeq=Ceeq-Gmr*U(Kk)
16510 Kk=0
16520 IF L1<=Nnode THEN 16560
16530 Cceq=Cceq+Gmf*U(L1)
16540 Ceeq=Ceeq+Gmr*U(L1)
16550 L1=0
16560 IF Mm<=Nnode THEN 16610
16570 Cceq=Cceq-Gmf*U(Mm)
16580 Mm=0
16590 !
16600 !.. LOAD ADMITTANCE MATRIX
16610 IF Kk=0 THEN 16710
16620 Y(Kk)=Y(Kk)+Gmr+Gpir
16630 IF L1=0 THEN 16670
16640 Jj=FNIndx(Kk,L1)
16650 Y(Jj)=Y(Jj)+Gmf-Gmr-Gpir
16660 Y(Nn)=Y(Nn)-Gpir
16670 IF Mm=0 THEN 16770
16680 Jj=FNIndx(Kk,Mm)
16690 Y(Jj)=Y(Jj)-Gmf
16700 Y(Nn)=Y(Nn)-Gmr
16710 IF Mm=0 THEN 16770
16720 Y(Mm)=Y(Mm)+Gmf+Gpif
16730 IF L1=0 THEN 16810
16740 Jj=FNIndx(Mm,L1)
16750 Y(Jj)=Y(Jj)-Gmf-Gpif+Gmr
16760 Y(Nn)=Y(Nn)-Gpif
16770 IF L1=0 THEN 16810
16780 Y(L1)=Y(L1)+Gpif+Gpir
16790 !
16800 !.. LOAD CURRENT VECTOR
16810 IF Iflg=9 THEN 16900
16820 !
16830 !.. DON'T ADD DC CURRENTS IF AC ANALYSIS
16840 IF Kk=0 THEN 16860
16850 C(Kk)=C(Kk)+(1.0+1.0/Br)*Ceeq-Cceq
16860 IF Mm=0 THEN 16880
16870 C(Mm)=C(Mm)+(1.0+1.0/Bf)*Cceq-Ceeq+Creq
16880 IF L1=0 THEN 16900
16890 C(L1)=C(L1)-Cceq/Bf-Ceeq/Br-Creq
16900 Kpos=Ielm(Kpos+1)
16910 NEXT I
16911 !PRINT "CEC,CEEQ,GMR";Cec,Ceeq,Gmr;"LINE 16911"
16920 SUBEND
16930 !
16940 SUB Junct(J,Vbb,Ccc,Ceq,Gm)
16950 !
```

BIASD.36 LISTING

```
16960 !.. DETERMINE TRANSISTOR IC,GM,GPI
16970 OPTION BASE 1
16980 COM U(*),C(*),Y(*)
16990 COM Ds,Delt,Delta
17000 COM TO,Temp,Dtemp
17010 COM Tm(*),A(*),Csat,Vt,Vct,Ptype
17020 COM Iel,Jj,Kk,L1,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
17030 COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
17040 COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
17050 COM A$[80],Ni(*),Ieln(*),Ifrst(*),Ilast(*),Ielm(*)
17060 COM Iur(*),Iuc(*),Ipos(*),Ncon(*),Iy(*)
17070 !
17080 !.. COLN LIMITING ALGORITHM IMPLEMENTED 8/16/76
17090 Vcrit=Vct-Vt*LOG(Csat)
17091 !PRINT "VCRIT=";Vcrit;"VBB=";Vbb
17093 Dix=1
17094 IF Vcrit<0. AND Vbb<0. THEN Dix=ABS(Vcrit)-ABS(Vbb)
17095 IF Vcrit>0. AND Vbb>0. THEN Dix=Vbb-Vcrit
17096 IF Vcrit>0. AND Vbb<=0. THEN 17210
17100 IF Dix<=1.E-6 THEN 17210
17101 !PRINT "<="
17110 !
17120 !  Vb<Vcrit THEN ITERATE ON VOLTAGE
17130 Ccc=Csat*(EXP(Ielm(2*J)/Vt)-1.0)
17140 Ceq=Ccc+Ielm(2*(J+2))*(Vbb-Ielm(2*J))
17150 IF Ceq<0. THEN 17200
17160 !
17170 !.. ITERATE ON CURRENT
17180 Vbb=Vt*LOG(Ceq/Csat+1.0)
17190 GOTO 17210
17200 Vbb=Vcrit
17210 Ccc=-Csat
17220 IF Vbb<-1.2 THEN 17240
17230 Ccc=Csat*EXP(Vbb/Vt)+Ccc
17240 Gm=(Ccc+Csat)/Vt+1.0E-10
17250 Ceq=Ptype*(Ccc-Gm*Vbb)
17260 Ielm(2*(J+2))=Gm
17270 Ielm(2*J)=Vbb
17284 SUBEND
17290 !
17300 SUB Updat
17310 !
17320 !.. UPDATE TRANSIENT SOURCES
17321 OPTION BASE 1
17330 COM U(*),C(*),Y(*)
17340 COM Ds,Delt,Delta
17350 COM TO,Temp,Dtemp
17360 COM Tm(*),A(*),Csat,Vt,Vct,Ptype
17370 COM Iel,Jj,Kk,L1,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
17380 COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
17390 COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
17400 COM A$[80],Ni(*),Ieln(*),Ifrst(*),Ilast(*),Ielm(*)
17410 COM Iur(*),Iuc(*),Ipos(*),Ncon(*),Iy(*)
17420 !
17430 FOR Iel=5 TO 6
17440 K1=Ieln(Iel)
```

```
17450 IF K1=0 THEN 17970
17460 Kpos=Ifrst(Iel)
17470 FOR J=1 TO K1
17480 Mpos=Kpos/2
17490 Itt=Ielm(Kpos+4)
17500 IF Itt=0 THEN 17950
17510 L=Ielm(2*(Itt+3))-2.0
17520 IF L=2 THEN 17930
17530 B1=Ielm(2*(Itt+4))
17540 B2=Ielm(2*(Itt+5))
17550 B3=Ielm(2*(Itt+6))
17560 B4=Ielm(2*(Itt+7))
17570 IF L=3 THEN 17680
17580 !
17590 !.. SINE
17600 VO=B1
17610 IF B4<>0 THEN 17630
17620 B4=Delta
17630 IF TO<B4 THEN 17650
17640 VO=VO+B2*SIN(6.28319*B3*(TO-B4)+Ielm(2*(Itt+8))/57.296)
17650 GOTO L4600
17660 !
17670 !.. PULSE
17680 Ti=TO
17690 Z=B3
17700 IF Ti>Z THEN 17730
17710 VO=B1
17720 GOTO L4600
17730 Z=Z+B4
17740 IF Ti>=Z THEN 17770
17750 VO=B2-(Ielm(2*(Itt+5))-B1)/B4*(Z-Ti)
17760 GOTO L4600
17770 Z=Z+Ielm(2*(Itt+8))
17780 IF Ti>Z THEN 17810
17790 VO=B2
17800 GOTO L4600
17810 Z=Z+Ielm(2*(Itt+9))
17820 IF Ti>=Z THEN 17850
17830 VO=B1+(B2-B1)/Ielm(2*(Itt+9))*(Z-Ti)
17840 GOTO L4600
17850 S=Ielm(2*(Itt+10))
17860 IF S=0 THEN 17900
17870 Z=Z+S
17880 Ti=Ti-S
17890 GOTO 17690
17900 VO=B1
17910 GOTO L4600
17920 !
17930 CALL Vext(TO,VO,Itt,Ielm(*))
17940 L4600: Ielm(2*(Mpos+3))=VO
17950 Kpos=Ielm(Kpos+1)
17960 NEXT J
17970 NEXT Iel
17980 SUBEND
17990 !
18000 SUB Vext(TO,VO,I,Ielm(*))
```

BIASD.36 LISTING

```
18010 !
18030 !
18040 SUBEND
18050 !
18060 DEF FNDelu(K,L)
18070 !
18080 !.. DETERMINE U(L)-U(K)
18081 OPTION BASE 1
18090 COM U(*),C(*),Y(*)
18100 COM Ds,Delt,Delta
18110 COM TO,Temp,Dtemp
18120 COM Tm(*),A(*),Csat,Vt,Vct,Ptype
18130 COM Iel,Jj,Kk,L1,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
18140 COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
18150 COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
18160 COM A$[80],Ni(*),Ieln(*),Ifrst(*),Ilast(*),Ielm(*)
18170 COM Iur(*),Iuc(*),Ipos(*),Ncon(*),Iy(*)
18180 !
18190 Delu=0.
18200 IF L>0 THEN Delu=U(L)
18210 IF K>0 THEN Delu=Delu-U(K)
18220 RETURN Delu
18230 FNEND
18240 !
18250 SUB Adres
18260 !
18270 !.. ADD RESISTORS TO Y MATRIX
18271 OPTION BASE 1
18280 COM U(*),C(*),Y(*)
18290 COM Ds,Delt,Delta
18300 COM TO,Temp,Dtemp
18310 COM Tm(*),A(*),Csat,Vt,Vct,Ptype
18320 COM Iel,Jj,Kk,L1,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
18330 COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
18340 COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
18350 COM A$[80],Ni(*),Ieln(*),Ifrst(*),Ilast(*),Ielm(*)
18360 COM Iur(*),Iuc(*),Ipos(*),Ncon(*),Iy(*)
18370 !
18380 IF Kk>Nnode THEN Kk=0
18390 IF L1>Nnode THEN L1=0
18400 IF Kk=0 THEN 18460
18410 Y(Kk)=Y(Kk)+Ds
18420 IF L1=0 THEN 18480
18430 Jj=FNIndx(Kk,L1)
18440 Y(Jj)=Y(Jj)-Ds
18450 Y(Nn)=Y(Nn)-Ds
18460 IF L1=0 THEN 18480
18470 Y(L1)=Y(L1)+Ds
18480 SUBEND
18490 !
18500 SUB Adcur
18510 !
18520 !.. ADD CURRENTS TO CURRENT VECTOR
18521 OPTION BASE 1
18530 COM U(*),C(*),Y(*)
18540 COM Ds,Delt,Delta
```

```
18550 COM TO,Temp,Dtemp
18560 COM Tm(*),A(*),Csat,Vt,Vct,Ptype
18570 COM Iel,Jj,Kk,L1,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
18580 COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
18590 COM Mxlst,Mxpos,Mxloc,Ncmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
18600 COM A$[80],Ni(*),Ieln(*),Ifrst(*),Ilast(*),Ielm(*)
18610 COM Iur(*),Iuc(*),Ipos(*),Ncon(*),Iy(*)
18620 !
18630 IF Kk>Nnode THEN Kk=0
18640 IF L1>Nnode THEN L1=0
18650 IF Kk=0 THEN 18670
18660 C(Kk)=C(Kk)-Ds
18670 IF L1=0 THEN 18690
18680 C(L1)=C(L1)+Ds
18690 SUBEND
18700 !
18720 SUB Gncur
18730 !
18740 !.. ROUTINE TO ADD GENERATED CURRENT SOURCES
18750 OPTION BASE 1
18760 COM U(*),C(*),Y(*)
18770 COM Ds,Delt,Delta
18780 COM TO,Temp,Dtemp
18790 COM Tm(*),A(*),Csat,Vt,Vct,Ptype
18800 COM Iel,Jj,Kk,L1,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
18810 COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
18820 COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
18830 COM A$[80],Ni(*),Ieln(*),Ifrst(*),Ilast(*),Ielm(*)
18840 COM Iur(*),Iuc(*),Ipos(*),Ncon(*),Iy(*)
18850 !   I2=IELN(9)
18851 !
18860 L1=0
18870 Kpos=Ifrst(9)
18880 FOR I=1 TO Ieln(9)
18890 Jpos=Ielm(Kpos+1)
18900 Npos=Ielm(Kpos+2)
18910 !
18920 !.. DURING AC ANALYSIS ONLY ADD AC SOURCE CURRENTS
18930 IF Inpt<>Npos AND Iflg=9 THEN 19230
18940 Kk=Ielm(Kpos+3)
18950 M=Ielm(Kpos+4)
18960 Tc=1.0
18970 IF Itemp=0 THEN 19020
18980 Nn=Itemp+2*M+1
18990 Tc=1.0+Ielm(2*Nn)*Dtemp+Ielm(2*(Nn+1))*Dtemp*Dtemp
19000 !
19010 !.. RESISTOR CONNECTED TO VOLTAGE SOURCE
19020 IF M>1 THEN 19070
19030 Ds=-Ielm(2*(Npos+3))/(Ielm(2*(Jpos+3))*Tc)
19040 GOTO 19200
19050 !
19060 !.. CAPACITOR CONNECTED TO VOLTAGE SOURCE
19070 IF M>2 THEN 19150
19080 Ds=Ielm(2*(Npos+3))*Ielm(2*(Jpos+3))*Tc
19090 Ds=-Ds/Delt
19100 IF Iflg=9 THEN 19220
```

BIASD.36 LISTING

```
19110 Ds=Ds*2.0
19120 GOTO 19200
19130 !
19140 !.. INDUCTOR CONNECTED TO VOLTAGE SOURCE
19150 IF M>3 THEN 19240
19160 Ds=Ielm(2*(Npos+3))/Ielm(2*(Jpos+3))
19170 Ds=-Ds*Delt
19180 IF Iflg=9 THEN 19220
19190 Ds=Ds*.5
19200 CALL Adcur
19210 GOTO 19230
19220 CALL Adcpc
19230 Kpos=Kpos+4
19240 NEXT I
19250 SUBEND
19260 !
19270 SUB Acsol
19280 !
19290 !.. MAIN AC ANALYSIS ROUTINE USED WITH BIAST8. 8-12-77
19300 !.. USING STANDARD AC ANALYSIS PROCEDURE WITH AC BJT LOAD
19301 OPTION BASE 1
19310 COM U(*),C(*),Y(*)
19320 COM Ds,Delt,Delta
19330 COM T70,Temp,Dtemp
19340 COM Tm(*),A(*),Csat,Vt,Vct,Ptype
19350 COM Iel,Jj,Kk,Ll,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
19360 COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
19370 COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
19380 COM A$[80],Ni(*),Ieln(*),Ifrst(*),Ilast(*),Ielm(*)
19390 COM Iur(*),Iuc(*),Ipos(*),Ncon(*),Iy(*)
19400 !
19410 Fstop=Tm(2)
19420 Ndec=Tm(3)
19430 Iprt=Tm(4)
19440 Freq=Tm(1)
19450 Fmult=Freq
19460 Ifreq=0
19480 !.. ZERO DC SOURCES
19490 FOR I=Nnode TO Node
19500 C(I)=0.
19510 NEXT I
19520 Vin=Ielm(2*(Inpt+3))
19540 !.. DETERMINE INPUT NODE
19550 In=Inpt*2
19560 In=Ielm(In+7)
19580 !.. STORE UNITY VOLTAGE IN AC INPUT SOURCE
19590 Ielm(2*(Inpt+3))=1.0
19600 IF Iel=6 THEN C(In)=1.0
19610 W=6.2831*Freq
19620 Delt=1.0/W
19640 !.. ZERO COMPLEX CURRENT VECTOR
19650 FOR I=1 TO Node
19660 C(I)=0.
19661 !  EQUIVALENCE C(30),CI(1)
19670 C(29+I)=0.
19680 NEXT I
```

```
19700 !..  ZERO COMPLEX ADMITTANCE MATRIX
19710 Ii=2*Mloc-Node
19720 FOR I=1 TO Ii
19730 Y(I)=0.
19731 !  EQUIVALENCE Y(300),YI(1)
19740 Y(299+I)=0.
19750 NEXT I
19770 !..  LOAD AC MATRIX
19780 CALL Eload
19790 !..  IF CURRENT SOURCE INPUT
19810 IF Iel<>5 THEN 19890
19820 Ii=Inpt*2+7
19830 Kk=Ielm(Ii)
19840 Ll=Ielm(Ii+1)
19850 Ds=1.0
19860 CALL Adcur
19870 !
19880 !..  ADD GENERATED CURRENT SOURCES
19890 IF Ieln(9)=0 THEN 19930
19900 CALL Gncur
19910 !
19920 !..  FORWARD AND BACK SUBSTITUTION TO GET NEW VOLTAGES
19930 CALL                                        BIASD.36 LISTING
19940 CALL Solac
19950 !
19960 U1=C(Jout)
19970 U2=C(29+Jout)
19980 Vmag=U1*U1+U2*U2
19990 Amag=SQR(Vmag)
20000 IF Iplt=0 THEN 20020
20010 IF Iprt<>2 THEN 20060
20020 Phase=ATN(U2/U1)*57.2958
20030 IF U1>0 THEN 20060
20031 S180=180
20033 IF SGN(U2)<0 THEN S180=-180
20040 Phase=Phase+S180
20050 !
20060 IF Iprt=1 THEN Amag=8.68589*LOG(Amag)
20070 IF Iplt=0 THEN 20150
20080 Efreq=LOG(Freq)*.434294
20090 !
20100 IF Iprt>=6 THEN 20170
20110 IF Iprt<>2 THEN 20130
20120 Amag=Phase
20121 !  CALL DRAW(EFREQ,AMAG,IPEN,4)
20130 Ipen=1
20140 GOTO 20170
20150 PRINT USING 20160;Freq,Amag,Phase,U1,U2
20160 IMAGE X,D.3DE,12D,11D,2(D.3DE)
20170 Ifreq=Ifreq+1
20180 IF Ifreq>Ndec THEN 20240
20190 Pwr=Ifreq
20200 Ftem=10.0^(Pwr/Ndec)
20210 Freq=Fmult*Ftem
20220 IF Freq>Fstop THEN 20270
20230 GOTO 19610
```

```
20240 Fmult=10.0*Fmult
20250 Ifreq=0
20260 GOTO 20170
20270 Ielm(2*(Inpt+3))=Vin
20280 SUBEND
20300 SUB Bjtac
20320 !.. LOAD AC BJT MODEL INTO Y & C ARRAYS
20330 !
20340 OPTION BASE 1
20350 COM U(*),C(*),Y(*)
20360 COM Ds,Delt,Delta
20370 COM TO,Temp,Dtemp
20380 COM Tm(*),A(*),Csat,Vt,Vct,Ptype
20390 COM Iel,Jj,Kk,L1,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
20400 COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
20410 COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
20420 COM A$[80],Ni(*),Ieln(*),Ifrst(*),Ilast(*),Ielm(*)
20430 COM Iur(*),Iuc(*),Ipos(*),Ncon(*),Iy(*)
20440 !
20450 Kpos=Ifrst(4)
20460 FOR I=1 TO Ieln(4)
20470 Mpos=Kpos/2
20480 Itt=Ielm(Kpos+4)
20490 Kk=Ielm(Kpos+7)
20500 L1=Ielm(Kpos+8)
20510 Mm=Ielm(Kpos+5)
20520 Ceeq=0.
20530 Cceq=0.
20540 Bf=Ielm(2*(Itt+4))
20550 Gmf=Ielm(2*(Mpos+7))
20560 Gpif=Gmf/Bf
20570 Br=Ielm(2*(Itt+5))
20580 Gmr=Ielm(2*(Mpos+8))
20590 Gpir=Gmr/Br
20610 IF Kk<=Nnode THEN 20640
20620 Ceeq=Ceeq-Gmr*C(Kk)  !..*V(Kk)
20630 Kk=0
20640 IF L1<=Nnode THEN 20680
20650 Cceq=Cceq+Gmf*C(L1)  !C(L1)=V(L1)
20660 Ceeq=Ceeq+Gmr*C(L1)  !     ''
20670 L1=0
20680 IF Mm<=Nnode THEN 20710
20690 Cceq=Cceq-Gmf*C(Mm)
20700 Mm=0
20710 IF Kk=0 THEN 20820
20720 Y(Kk)=Y(Kk)+Gmr+Gpir
20730 C(Kk)=C(Kk)+(1.0+1.0/Br)*Ceeq-Cceq
20740 IF L1=0 THEN 20780
20750 Jj=FNIndx(Kk,L1)
20760 Y(Jj)=Y(Jj)+Gmf-Gmr-Gpir
20770 Y(Nn)=Y(Nn)-Gpir
20780 IF Mm=0 THEN 20890
20790 Jj=FNIndx(Kk,Mm)
20800 Y(Jj)=Y(Jj)-Gmf
20810 Y(Nn)=Y(Nn)-Gmr
20820 IF Mm=0 THEN 20890
```

```
20830 Y(Mm)=Y(Mm)+Gmf+Gpif
20840 C(Mm)=C(Mm)+(1.0+1.0/Bf)*Cceq-Ceeq
20850 IF L1=0 THEN 20920
20860 Jj=FNIndx(Mm,L1)
20870 Y(Jj)=Y(Jj)-Gmf-Gpif+Gmr
20880 Y(Nn)=Y(Nn)-Gpif
20890 IF L1=0 THEN 20920
20900 Y(L1)=Y(L1)+Gpif+Gpir
20910 C(L1)=C(L1)-Cceq/Bf-Ceeq/Br
20920 Kpos=Ielm(Kpos+1)
20921 NEXT I
20930 SUBEND
20940 !
20950 SUB                                    BIASD.36 LISTING
20960 !
20970 !.. PERFORMS LU DECOMPOSITION BASED ON RECORDED SPARSITY
20990 OPTION BASE 1
21000 COM U(*),C(*),Y(*)
21010 COM Ds,Delt,Delta
21020 COM TO,Temp,Dtemp
21030 COM Tm(*),A(*),Csat,Vt,Vct,Ptype
21040 COM Iel,Jj,Kk,L1,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
21050 COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
21060 COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
21070 COM A$[80],Ni(*),Ieln(*),Ifrst(*),Ilast(*),Ielm(*)
21080 COM Iur(*),Iuc(*),Ipos(*),Ncon(*),Iy(*)
21120 IF Nnode=1 THEN 21460
21130 Nn=Nnode-1
21140 Knt=0
21150 FOR I=1 TO Nn
21160 ! L=IORDR(I)
21180 L=Ni(I,1)
21190 Ur=Y(L)
21191 ! Ui=Yi(L)
21200 Ui=Y(299+L)    !EQUVLNCE Y(300),YI(1)
21210 Ds=Ur*Ur+Ui*Ui
21220 Ius=Iur(I)+Mloc
21230 Iue=Iur(I+1)+Mloc
21240 Il=Iur(I)+Node    !Iur=Ilc
21250 Ile=Iur(I+1)+Node
21270 !.. DOWN LOWER TRIANGLE COLUMNS
21280 IF L>=Ile THEN 21450
21290 Dr=(Y(Il)*Ur+Y(299+Il)*Ui)/Ds
21300 Di=(Y(299+Il)*Ur-Y(Il)*Ui)/Ds
21310 Y(Il)=Dr
21320 Y(299+Il)=Di
21330 Il=Il+1
21340 Iu=Ius
21360 !.. ACROSS UPPER TRIANGULAR ROWS
21370 IF Iu>=Iue THEN 21280
21380 Knt=Knt+1
21390 K=Ipos(Knt)
21400 Ar=Y(K)-(Y(Iu)*Dr-Y(299+Iu)*Di)
21410 Y(299+K)=Y(299+K)-(Y(Iu)*Di+Y(299+Iu)*Dr)
21420 Y(K)=Ar
21430 Iu=Iu+1
```

```
21440 GOTO 21370
21450 NEXT I
21460 SUBEND
21470 !
21480 SUB Solac
21500 !.. PERFORMS FORWARD AND BACKWARD SUBSTITUTION
21510 !   USING SPARSE POINTERS
21520 !   MODIFIED FROM BIAS-N 5-9-77
21540 OPTION BASE 1
21550 COM U(*),C(*),Y(*)
21560 COM Ds,Delt,Delta
21570 COM TO,Temp,Dtemp
21580 COM Tm(*),A(*),Csat,Vt,Vct,Ptype
21590 COM Iel,Jj,Kk,Ll,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
21600 COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
21610 COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
21620 COM A$[80],Ni(*),Ieln(*),Ifrst(*),Ilast(*),Ielm(*)
21630 COM Iur(*),Iuc(*),Ipos(*),Ncon(*),Iy(*)
21650 !.. FORWARD SUBSTITUTION
21660 Nn=Nnode-1
21670 IF Nn>0 THEN 21760
21680 Da=Y(1)
21690 Db=Y(300)    !Db=Yi(1)
21700 Ds=Da*Da+Db*Db
21701 ! EQUVLNC C(30),CI(1)
21710 Dr=(C(1)*Da+C(30)*Db)/Ds
21720 ! EQUVLNC C(30),VI(1)
21730 C(30)=(C(30)*Da-C(1)*Db)/Ds
21740 C(1)=Dr      !V(1)=Dr
21750 GOTO 22280
21760 FOR I=1 TO Nn
21770 !  L=IORDR(I)
21790 L=Ni(I,1)
21800 Il=Iur(I)      !ILC(I)
21810 Ile=Iur(I+1)   ! ''
21820 Nl=Il+Node
21830 IF Il>=Ile THEN 21900
21840 Nr=Iuc(Il)     !ILR(Il)
21850 Dr=C(Nr)-(Y(Nl)*C(L)-Y(299+Nl)*C(29+L))
21860 C(29+Nr)=C(29+Nr)-(Y(Nl)*C(29+L)+Y(299+Nl)*C(L))
21870 C(Nr)=Dr
21880 Il=Il+1
21890 GOTO 21820
21900 NEXT I
21920 !.. BACK SUBSTITUTION
21930 !  L=IORDR(NNODE)
21950 L=Ni(Nnode,1)
21960 Da=Y(L)
21970 Db=Y(299+L)
21980 Ds=Da*Da+Db*Db
21990 Dr=(C(L)*Da+C(29+L)*Db)/Ds
22000 C(29+L)=(C(29+L)*Da-C(L)*Db)/Ds
22010 C(L)=Dr
22020 FOR I=1 TO Nn
22030 Nui=Nnode-I
22040 !  L=IORDR(NUI)
```

BIASD.36 LISTING

```
22060 L=Ni(Nui,1)
22070 Iu=Iur(Nui)
22071 Iue=Iur(Nui+1)
22080 N1=Iu+Mloc
22090 IF Iu>=Iue THEN 22160
22100 Ic=Iuc(Iu)
22110 Dr=C(L)-(Y(N1)*C(Ic)-Y(299+N1)*C(29+Ic))
22120 C(29+L)=C(29+L)-(Y(N1)*C(29+Ic)+Y(299+N1)*C(Ic))
22130 C(L)=Dr
22140 Iu=Iu+1
22150 GOTO 22080
22160 Da=Y(L)
22170 Db=Y(299+L)
22180 Ds=Da*Da+Db*Db
22190 Dr=(C(L)*Da+C(299+L)*Db)/Ds
22200 C(299+L)=(C(299+L)*Da-C(L)*Db)*Ds
22210 C(L)=Dr
22220 NEXT I
22230 !.. TRANSFER INTO COMPLEX VOLTAGE VECTOR
22240 !  FOR I=1 TO NNODE
22250 !  V(I)=C(I)
22260 !  VI(I)=CI(I)
22270 !  NEXT I
22280 SUBEND
22290 !
22300 SUB Adcpr
22320 !.. ADD IMAGINARY CONDUCTANCE TO Y MATRIX
22340 OPTION BASE 1
22350 COM U(*),C(*),Y(*)
22360 COM Ds,Delt,Delta
22370 COM TO,Temp,Dtemp
22380 COM Tm(*),A(*),Csat,Vt,Vct,Ptype
22390 COM Iel,Jj,Kk,L1,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
22400 COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
22410 COM Mxlst,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
22420 COM A$[80],Ni(*),Ieln(*),Ifrst(*),Ilast(*),Ielm(*)
22430 COM Iur(*),Iuc(*),Ipos(*),Ncon(*),Iy(*)
22450 IF Kk>Nnode THEN Kk=0
22460 IF L1>Nnode THEN L1=0
22470 IF Kk=0 THEN 22530
22480 Y(299+Kk)=Y(299+Kk)+Ds   !YI(KK)=...
22490 IF L1=0 THEN 22550
22500 Jj=FNIndx(Kk,L1)
22510 Y(299+Jj)=Y(299+Jj)-Ds
22520 Y(299+Nn)=Y(299+Nn)-Ds
22530 IF L1=0 THEN 22550
22540 Y(299+L1)=Y(299+L1)+Ds
22550 SUBEND
22560 !
22570 SUB Adcpc
22590 !.. ADD IMAGINERY CURRENTS TO CURRENT VECTOR
22600 OPTION BASE 1
22610 COM U(*),C(*),Y(*)
22620 COM Ds,Delt,Detla
22630 COM TO,Temp,Dtemp
22640 COM Tm(*),A(*),Csat,Vt,Vct,Ptype
```

BIASD.36 LISTING

```
22650 COM Iel,Jj,Kk,L1,Mm,Nn,Iflg,Jflg,Itt,Iter,Iw,Ir,Iunit
22660 COM Iplt,Ipen,Ltype,Itemp,Itotl,Iout,Jout,Inpt,Iform
22670 COM Mx1st,Mxpos,Mxloc,Ndmax,Node,Nnode,Ierr,Mloc,Kpos,Lpos
22680 COM A$[80],Ni(*),Ieln(*),Ifrst(*),Ilast(*),Ielm(*)
22690 COM Iur(*),Iuc(*),Ipos(*),Ncon(*),Iy(*)
22700 !
22710 IF Kk>Nnode THEN Kk=0
22720 IF L1>Nnode THEN L1=0
22730 IF Kk=0 THEN 22750
22740 C(29+Kk)=C(29+Kk)-Ds  !CI(KK)=...
22750 IF L1=0 THEN 22770
22760 C(29+L1)=C(29+L1)+Ds
22770 SUBEND
```

BIASP.36 LISTING

```
$debug on,partial_eval on$
$ref 150,lines 50$

{ ***********************************************************
  *                                                         *
  *                     BIASP  - 12-9-82                    *
  *                                                         *
  * VERSION 1.0                                             *
  * WRITTEN BY TONY YEE, DECEMBER 1982                      *
  * RUNNING ON HEWLETT-PACKARE 9826/9836 DESKTOP COMPUTERS   *
  * USER INTERACTIVE                                        *
  * SIMULATE DC AND TRANSIENT ANALYSISES OF MOS CIRCUIT ONLY *
  * SPARSE MATRIX ROUTINE - DOUBLE LINKED LIST              *
  * DYNAMIC STORAGE FOR SPARSE MATRIX                       *
  *                                                         *
  *********************************************************** }


program biasp(input,output);
const
   maxequ = 50;          { maximum number of equations simulated }
   maxmos = 50;          { maximum number of capacitors }
   maxres = 10;          { maximum number of resistors }
   maxcap = 25;          { maximum number of capacitors }
   maxvso = 10;          { maximum number of voltage sources }
   maxcso =  4;          { maximum number of current sources }
   maxdio =  5;          { maximum number of diodes }
   maxmosm=  5;          { maximum number of mosfet models }
   maxdiom=  1;          { maximum number of diode model }
   maxsorm=  5;          { maximum number of source models }
   maxcapm=  2;          { maximum number of capacitor models }
type
   entryptr = ^entry;      { a matrix entry }

   entry   = record
              i,j: integer;    { row and column numbers }
              val : real;      { value of the entry }
              nextI: entryptr; { point to next element in the row }
              nextJ: entryptr; { point to next element in the column }
             end;

   headptr = record
               ptr : entryptr;    { point to first entry }
               count:integer;     { count number of elements }
               oldindex:integer;  { row or column index before reordering }
             end;

   headtype = array[0..maxequ] of headptr;

   locptr = array[1..4] of entryptr;  { element pointers to the matrix entr

   val_array  = array[0..maxequ] of real;
   node_array  = array[0..maxequ] of integer;
   name_string  = string[4];
   line_string  = string[80];
```

```
proc_string  = array[1..40] of line_string;
name_array = array[1..10] of name_string;

resistor = record
    name : name_string;        { name of the resistor }
    n1,n2 : integer;           { nodes that connect the resistor }
    value : real;              { value of the resistor }
    tc1,tc2 : real;            { optional temperature coefficients }
    matptr:locptr;             { pointers to matrix locations }
  end;

res_array = array[1..maxres] of resistor;

capacitor = record
    name : name_string;        { name of the capacitor }
    n1,n2 : integer;           { nodes that connect the capacitor }
    value : real;              { value of the capacitor }
    mod_pos : integer;         { where its models is if any }
    initval: real;             { initial voltage value }
    sma_val: array[1..2] of real;  { small signal value }
    matptr:locptr;             { pointers to matrix locations }
  end;

cap_array  = array[1..maxcap] of capacitor;

capmodel = record
    name: name_string;         { name of the capacitor model }
    value: array[1..3] of real; { values of the capacitor model }
  end;

capmod_array  = array[1..maxcapm] of capmodel;

mosfet = record
    name: name_string;         { name of the mosfet }
    node: array[1..4] of integer; { drain,gate,source and bulk nodes}
    mod_pos: integer;          { its model position in model array }
    wc,lc,ad,as: real;         { mosfets' parameters }
    initial: boolean;          { flag to signal initial voltage }
    vgsi,vdsi,vsbi: real;      { intial voltage values }
    satlin,cutlin: integer;    { region indicators }
    sma_val : array[1..13] of real;  { small signal value }
    mggptr,mgstr,mgdptr,mgbptr:locptr; {pointers to matrix locations}
 end;

mos_array  = array[1..maxmos] of mosfet;

mosmodel = record
    name: name_string;         { name of the mosfet model }
    define: integer;           { 0=undefine,1=NMOS,-1=PMOS }
    value: array[1..26] of real; { mosfets' model values }
   end;

mosmod_array  = array[1..maxmosm] of mosmodel;

diode = record
    name : name_string;        { name of the diode }
```

```
        nca,nan:integer;            { positive and negative nodes }
        mod_pos:integer;            { model position in model array }
        scale:real;                 { optional scale factor }
        sma_val: array[1..3] of real; { small signal values }
        matptr:locptr;              { pointers to matrix locations }
     end;
    dio_array  = array[1..maxdio] of diode ;

    diomodel = record
        name :name_string;          { name of diode model }
        satcur: real;               { value of saturation current }
      end;
    diomod_array  = array[1..maxdiom] of diomodel;

    voltage = record
        name: name_string;          { name of the voltage source }
        nodep,noden : integer;      { nodes of the voltage source }
        value : real;               { value of the voltage source }
        mod_pos: integer;           { model position in source model }
        matptr:locaptr;             { pointers to matrix locations }
     end;
    vol_array  = array[1..maxvso] of voltage;

    sormods = record
        name: name_string;          { name of the source model }
        define: integer;            { flag to see it is defined }
        step: array[1..22] of real; { value of the model }
     end;
    sormod_array  = array[1..maxsorm] of sormods;

    current = record
        name:name_string;           { name of current source }
        nodep,noden : integer;      { nodes of the voltage source }
        value:real;                 { value of current source }
        mod_pos: integer;
      end;
    cur_array  = array[1..maxcso] of current;

 var
    row,col:headtype;               { row and column pointers of matrix }
    basr: rhstype;
    tasr,pasr,qasr,nasr:val_array;
    name:name_string;
    inline:line_string;
    command:line_string;
    mos_para:line_string;
    resr:res_array;
    node:node_array;
    capr:cap_array;
    capmod:capmod_array;
    mosfets:mos_array;
    mosmod:mosmod_array;
    diod:dio_array;
    diomod:diomod_array;
    volsor:vol_array;
    sormod:sormod_array;
    nodeval:val_array;
    cursor:cur_array;
    prof:proc_string;
    gout:array[1..12] of integer;

    numres,nummos,numcap,numdio,numcur,numnod:integer;
    numsorm,numvol,numcapm,numdiom,nummosm,totnod: integer;
    nonum,nochar,donemain,error:boolean;
    short,chem,subt,trans,ui,cutlin: boolean;
    cong,satlin,action,preset,exe_proc:boolean;
    i:integer;
    linepos,profpos:integer;
    vt1,vt2,ni1,ni2,eo,esi,eg,qe,conval:real;
    dvdt,didt,T:real;
    maxit,minit,level,iter1,iter2:integer;
    tstep,tstep1,tstep2,tstart,tstart1,tstop:real;
    f9:integer;
    blank:set of char;
    buff,outfile:text;
    numfile:integer;
    filename:name_array;
    ch:char;

procedure initialize;

{ initialize global constant and variables. }

begin
rewrite(outfile,'console:');
console:=true;
vt1 := 2.585e-2;
vt2 := vt1;
ni1 := 1.45e10;
ni2 := ni1;
eo  := 8.854e-14;
esi := eo*11.7;
eg  := 1.12;
qe  := 1.602e-19;
maxit := 8;
minit := 3;
level :=2;
conval := 0.0003;
dvdt := 0.2;
didt := 0.9;
t := 27;
mos_para:='vtok" gamphilamcgscgdcgbcdbcsbcoxjs pb nsu' +
          'nssuo ecreexetrnfsldixj polbjcend';
numres:=0;nummos:=0;
numcap:=0;numdio:=0;numnod:=0;
numsorm:=0;numcapm:=0;numdiom:=0;nummosm:=0;
end;

procedure pause(var line: integer);
var ch:char;

begin
```

```
 if console then
  begin
  if (line = 22) then
   begin
    writeln;
    write('***** PAUSE *****  hit a key to continue.');
    {beep;}
    read(ch);
    page(outfile);
    line:=1;
   end
  else
   line:=line + 1;
  end;
end;

procedure skipblank(inline:line_string;var linepos:integer);

{ skip over the blank spaces }
var ch:char;
begin
if (linepos <= leng) then
 begin
 linepos:=linepos - 1;
 repeat
  linepos:=linepos + 1;
  ch:=inline[linepos];
 until (linepos = leng) or (not (ch in blank));
if (linepos = leng) and (ch in blank) then
 linepos:=linepos +1;
 end;
end;

procedure skipchar(inline:line_string;var linepos:integer);

{ skip over characters that until a blank }
begin
 if linepos <= leng then
  ch:=inline[linepos];
 while not (ch in blank) and (linepos<=leng) do
  begin
  linepos:=linepos + 1;
  if linepos <= leng then
   ch := inline[linepos];
  end;
 skipblank(inline,linepos);
end;

procedure errr(E: integer;inline:line_string);

{ print out the error messages accordig to the error code E }
begin { errr }
 case E of
  1: writeln(outfile,'UNKNOWN COMMAND');
  2: writeln(outfile,'NEED STOP TIME');
  3: writeln(outfile,'DC or TRANSIENT');
```

```
   4: writeln(outfile,'CANNOT PLOT DC');
   5: writeln(outfile,'NAME IS TOO LONG');
   6: writeln(outfile,'NO SUCH ELEMENT');
   7: writeln(outfile,'NEED NUMBER, #0');
   8: writeln(outfile,'WHAT FILENAME');
   9: writeln(outfile,'UNKNOWN PARAMETER');
   10: writeln(outfile,'UNKNOWN OPTION');
   11: writeln(outfile,'WRONG FILE TYPE');
   12: writeln(outfile,'CANNOT FIND FILE');
   13: writeln(outfile,'ILLEGAL MODEL NAME');
   14: writeln(outfile,'ALREADY EXISTS');
   15: writeln(outfile,'> MAX # OF ELEMENTS');
   16: writeln(outfile,'> MAX # OF MODELS');
  end;
  writeln(outfile,inline);
  writeln(outfile);
 end; { errr }

function checksign(inline:line_string;var linepos:integer):boolean;

{ check if the present character is the sign of the integer
  if it is, advance one character in the input line, and if
  it is a minus sign, return true }
begin
 checksign := false;
 if (linepos <= leng) then
  begin
  ch:=inline[linepos];
  if (ch in ['+', '-']) then
   begin
    if (ch = '-') then
     checksign := true;
    linepos := linepos + 1;
   end;
  end;
end;

procedure gnod(var p1,numnod:integer;var node:node_array);
var i:integer;

{ store the node number of the circuit in array node, and return
  its location on the array node }

function minI(i, j:integer):integer;
{ minimum of two integer numbers }
begin
if (i<j) then minI:=i
else minI:=j;
end;

 begin { gnod }
 if (p1 <> 0) then
  begin
   if p1 < 0 then
    p1 := node[minI(abs(p1),numnod)]
   else
```

```
    begin
     i := 0;
     repeat
      i:=i+1;
     until (i>=numnod) or (p1=node[i]);
     if (p1 <> node[i]) then
      begin
       numnod := numnod + 1;
       node[numnod] := p1;
       p1 := numnod;
      end
     else
      p1:=i;
    end;
  end;
 end; { gnod }

procedure gnam(p1:integer;inline:line_string;var name:name_string);

{ extract the string names from the input line }
var
 i,j,len: integer;
begin
if (leng - p1 >= 3) then
 len:=4
else
 len:=leng-p1+1;
name:=str(inline,p1,len);
i:=0;
repeat
 i:=i+1;
 ch:=name[i];
until (i=len) or (ch=' ');
if (ch=' ') and (i<len) then
 strdelete(name,i+1,len-i);
for j:=i+1 to 4 do
 name:=name+' ';
linepos:=linepos + i - 1;

{ skip non-blank characters }

ch:=inline[linepos];
while (ch <> ' ') and (linepos <= leng) do
 begin
 linepos:=linepos + 1;
 if linepos <= leng then
  ch:=inline[linepos];
 end;
skipblank(inline,linepos);
end;

procedure PrintSelect(inline:line_string;var linepos:integer);
begin
skipchar(inline,linepos);
gnam(linepos,inline,name);
if (name = 'prin') then
```

```
  begin
   close(outfile,'save');
   rewrite(outfile,'printer:');
   console:=false;
  end
 else
  if (name = 'crt ') or (name = '1    ') then
   begin
    close(outfile,'save');
    rewrite(outfile,'console:');
    console:=true;
   end
  else
   writeln(outfile,'ILLEGAL DEVICE NAME');
 end;

 procedure getint(var num: integer;var nonum: boolean);

 { Extract an integer from the inlineut line and return it in num.
   If no integer, it will return 0 and nonum set to true.}

 var minus : boolean;
 begin
 skipblank(inline,linepos);

 { check for sign that precede the integer }
 minus:=checksign(inline,linepos);
 num := 0;nonum:=true;
 if (linepos <= leng) then
  begin
  ch:=inline[linepos];
  if ch in ['0'..'9'] then
   begin
   repeat
    num := num*10 + ord(ch) - ord('0');
    linepos := linepos + 1;
    if (linepos <= leng) then
     ch:=inline[linepos];
   until (linepos>leng) or (not (ch in ['0'..'9']));
   if minus then num := -num;
   nonum := false;
   end;
  end;
 skipblank(inline,linepos);
 end;

 procedure getreal(var num:real;var nonum:boolean);

 { extract a real number from input line }
 var minus,done,done1:boolean;
     divisor:real;i,exp:integer;
 begin
 skipblank(inline,linepos);
 num:=0.0;nonum:=true;exp:=0;
 minus:=checksign(inline,linepos);       { check for sign }
 if (linepos <= leng) then
```

```
begin
ch:=inline[linepos];
if (ch in ['0'..'9','.']) then
 begin
 nonum:=false; done:=false; done1:=false;

 { get decimal part of the real number }
 while (ch in ['0'..'9']) and not done do
  begin
  num:=num*10.0+ord(ch)-ord('0');
  linepos:=linepos+1;
  if (linepos > leng) then
   done:=true
  else
   ch:=inline[linepos];
  end;

 { get the floating part of the real number }
 if not done then
  begin
  if (ch = '.') then
   begin
   linepos := linepos+1;
   divisor:=10.0;
   ch:=inline[linepos];
   while (ch in ['0'..'9']) and not done1 do
    begin
    num:=num+(ord(ch)-ord('0'))/divisor;
    linepos:=linepos+1;
    if (linepos <= leng) then
     ch:=inline[linepos]
    else
     done1 := true;
    divisor := divisor*10.0;
    end;
   end;

  { get the exponent if any }
  if linepos <= leng then
   begin
   if (ch = 'e') then
    begin
    linepos:=linepos + 1;
    getint(exp,nonum);
    end;
   if not nonum and (linepos <= leng) then
    begin
    ch:=inline[linepos];
    if (ch in ['G','M','K','m','u','n','p','f']) then
     begin
     case ch of
     'G': exp := exp+9;
     'M': exp := exp+6;
     'K': exp := exp+3;
     'm': exp := exp-3;
     'u': exp := exp-6;
```

```
        'n': exp := exp-9;
        'p': exp := exp-12;
        'f': exp := exp-15;
        end;
        linepos:=linepos+1;
        end;
       end;
      end;
     end;
    for i:=1 to abs(exp) do
     if (exp < 0) then num :=num/10.0
     else num := num*10;
    if minus then num:=-num;
    end;
   end;
   skipblank(inline,linepos);
  end; {getreal}

  procedure preal(num: real);

  { print out the real number in the format X.XXe+XX }
  var exp,temp,k,digit,bit,m:integer;
  begin
  if (num < 0.0) then
   write(outfile,'-')
  else write(outfile,' ');
  num := abs(num);
  exp := 0;
  if (num < 1.0) then
   begin
   if (num <> 0.0) then
    while (num < 1.0) do
     begin
     num:=num*10.0;
     exp:=exp-1;
     end
   end
  else
   while (num >= 10.0) do
    begin
    num:=num/10.0;
    exp:=exp+1;
    end;
  temp:=round(100.0*num);
  if temp >= 1000 then begin
   temp:=temp div 10;
   exp:=exp+1;
   end;
  for k := 1 to 3 do
   begin
   bit:=1;
   for m:=1 to 3-k do
    bit:=bit*10;
   digit:=temp div bit;
   temp:=temp - digit*bit;
   write(outfile,digit:1);
```

```
 if (k=1) then write(outfile,'.');
 end;
if (exp < 0) then
 begin
 write(outfile,'E-'); exp := -exp
 end
else write(outfile,'E+');
while (exp >= 100) do
 exp:=exp div 10;
temp:=exp div 10;
exp:=exp-temp*10;
write(outfile,temp:1,exp:1);
end;

procedure mosu(var mosmod:mosmod_array);

{ calculation of the mosfets model parameters }
var i:integer;
    p1:real;
begin
for i:=1 to nummosm do
 with mosmod[i] do begin
  if (value[14] = 0.0) then
   value[26]:=define*value[1]-value[3]*sqrt(value[4])
  else
   begin
   value[4]:=2.0*vt2*ln(value[14]/ni2);  {phi}
   p1:=3.2;
   if (value[23] <> 0.0) then
    begin
    p1:=value[23]/abs(value[23]);
    p1:=3.25+eg/2.0-p1*ln(abs(value[23]/ni2))*define*vt2;
    end;
   p1:=p1-(3.25+eg/2.0+define*value[4]/2.0);
   value[26]:=p1-value[15]*qe/value[11]+value[4]*define;
   value[3]:=sqrt(2.0*esi*qe*value[14])/value[11];
   value[1]:=value[26]+value[3]*sqrt(value[4])*define;  {vto}
   value[25]:=sqrt(2.0*esi/(qe*value[14]))*0.01;
   if (value[17] >= 100.0) then
    value[17]:=value[17]*esi/value[11];
   value[2]:=value[11]*value[16]*sqrt(sqr(300/(t+273))*300.0/(t+273.0));
   end;
  if (value[24] = 0.0) then value[24]:=0.5;
  if (value[4] = 0.0) then value[4]:=0.66;
  if (value[13] = 0.0) then value[13]:=0.87;
  if (define > 0) and (value[2] = 0.0) then value[2]:=2.42e-5;
  if (define < 0) and (value[2] = 0.0) then value[2]:=1.02e-5;
 end;
end;

procedure prod(var prof:proc_string;var profpos:integer);

{ retrieve a line from the procedure editor file }
begin
 profpos:=profpos+1;
 if (prof[profpos]='done') then
```

```
  begin
  donemain:=true;
  inline:='end';
  end
 else
  inline:=prof[profpos];
  leng:=strlen(inline);
end;

{ MATH FUNCTIONS }

procedure matza(totnod:integer;var ansr:matype);

{  initialize the matrix arrays }

var i:integer;
begin
for i := 0 to totnod do begin
  row[i].ptr:=nil;
  row[i].count:=0;
  row[i].oldindex:=i;
  col[i].ptr:=nil;
  col[i].count:=0;
  col[i].oldindex:=i;
 end; { for }
end;

procedure matzv(totnod:integer;var nsr:val_array);
var i:integer;
begin
for i:=1 to totnod do
 nsr[i]:=0.0;
end;

procedure mate(var asr1,asr2:val_array;totnod:integer);
var i:integer;
begin
for i:=1 to totnod do
asr1[i]:=asr2[i];
end;

function max6(p1,p2,p3,p4,p5,p6:real):real;

{ returns the maximum of p1-p6 }
var tem:real;
begin
tem:=p1;
if (p2>tem) then tem:=p2;
if (p3>tem) then tem:=p3;
if (p4>tem) then tem:=p4;
if (p5>tem) then tem:=p5;
if (p6>tem) then tem:=p6;
max6:=tem;
end;

function max2(i,j:real):real;
```

```
{ returns the maximum of i and j }

begin
max2:=i;
if (j > i) then max2:=j;
end;

procedure chkf;
var i,mn,tt:integer;
   sat,lin,off:integer;
   tem:real;
begin
for i:=1 to nummos do
 begin
 mn:=mosfets[i].mod_pos;
 tt:=mosmod[mn].define;
 if satlin then
  begin
  sat:=mosfets[i].satlin;
  with mosfets[i] do
  lin:=round(abs(sma_val[2]*tt-sma_val[5])/(sma_val[2]*tt-sma_val[5]));
  lin:=abs(lin+1) div ( lin+1);
  if (sat-lin=0.0) then
   begin
   mosfets[i].satlin:=abs(lin-1);
   if (tstart1-tstep1<>0.0) then
    if (lin<>0.0) then
     begin
     write(outfile,'  ',mosfets[i].name,
                  ' has entered the Saturation region at ');
     tem:=tstart1-tstep1;
     preal(tem);writeln(outfile,' sec');
     end
    else begin
     write(outfile,' ',mosfets[i].name,
                    ' has entered the     Linear  region at ');
     tem:=tstart1-tstep1;
     preal(tem);writeln(outfile,' sec');
     end;
   end;
  end;
 if cutlin then
  begin
  off:=mosfets[i].cutlin;
  with mosfets[i] do
  lin:=round(abs(sma_val[1]-tt*sma_val[4])/(sma_val[1]-tt*sma_val[4]));
  lin:=abs(lin+1) div (lin+1);
  if (off-lin<>0.0) then
   begin
   mosfets[i].cutlin:=abs(lin-1);
   if (tstart1-tstep1<>0.0) then
    if (lin<>0.0) then
     begin
     write(outfile,'  ',mosfets[i].name,
                       ' has entered the        Cutoff   region at ');
```

```
      tem:=tstart1-tstep1;
      preal(tem);writeln(outfile,' sec');
      end
     else
      begin
      write(outfile,'  ',mosfets[i].name,
                        ' has entered the        Linear   region at ');
      tem:=tstart1-tstep1;
      preal(tem);writeln(outfile,' sec');
    end;
    end;
   end;
  if (mosfets[i].sma_val[6] < -0.2) then
   begin
   write(outfile,'  ',mosfets[i].name,'"s Source is forward biased at ');
   tem:=tstart1-tstep1;
   preal(tem);writeln(outfile,' sec');
   end;
  if (mosfets[i].sma_val[6]+mosfets[i].sma_val[5] < -0.2) then
   begin
   write(outfile,'  ',mosfets[i].name,'"s Drain  is forward biased at ');
   tem:=tstart1-tstep1;
   preal(tem);writeln(outfile,' sec');
   end;
  end;
 end;

 procedure int8(var p1,p2,p3,p4,p5,p6:real);
 begin
 if (p1 <= 0.0) then begin
  p5:=0.0;
  p6:=0.0;
  end
 else
  if (tstart1 <> 0.0) then
   begin
   if (iter1 = 0) then p4:=2.0*p1*(p2-p3)/tstep2-p4;
   p5:=2*p1/tstep1;
   p6:=p2*p5+p4;
   end
  else
   begin
    p5:=p1/tstep1;
    p6:=p2*p5;
   end;
 end;

 procedure limt(var p1,p2,p3:real;p4,p5,p6:real);

 { limiting the change in voltage of p1, given the old voltage p2 and
   the change in p3 }
 begin
 if (abs(p3) >=p4) and ((iter1 <> 1) or (tstep1-tstep2=0.0)) then
  p1:=p2+p4*((p1-p2)/abs(p1-p2));
 if (p1<p5) then p1:=p5; {max}
 if (p6<p1) then p1:=p6; {min}
```

```
end;
procedure limy(var vnew:real;vold,delv,vto:real);
var vtsthi,vtstlo,vtox,vtemp:real;
begin
if (iter1 <> 1) or (tstep1-tstep2=0.0) then
 begin
 vtsthi:=abs( 2.0*(vold-vto))+2;
 vtstlo:=vtsthi/2+2;
 vtox:=vto+3.5;
 if (vold<vto) then
  begin
  if (delv>0.0) then
   begin
   vtemp:=vto+0.5;
   if (vnew>vtemp) then vnew:=vtemp
   else
    if (delv>vtstlo) then vnew:=vold+vtstlo;
   end
  else
   if (delv>vtstlo) then vnew:=vold-vtsthi;
  end
 else
  begin
  if (vold<vtox) then
   begin
   if (delv>0.0) then
    begin
    if (vto+4.0<vnew) then vnew:=vto+4.0;
    end
   else
    if (vto-0.5>vnew) then vnew:=vto-0.5;
   end
  else
   begin
   if (delv>0.9) then
  begin
    if (delv>=vtsthi) then vnew:=vold+vtsthi;
    end
   else
    begin
    if (vnew<vtox) then begin
     if ( vto+2>vnew) then vnew:=vto+2; end
    else
     if (-delv>vtstlo) then vnew:=vold-vtstlo;
    end;
   end;
  end;
 end;
end;

{ sparse matrix manipulation routines }

function FindEntry(ptr: entryptr; j:integer):entryptr;

{ find the jth entry from the row list ptr }
var find:boolean;
```

```
begin
find:=false;
FindEntry:=nil;
while (ptr <> nil) and (not find) do begin
 if (ptr^.j >= j) then
  find:=true;
 if (ptr^.j = j) then
  FindEntry:=ptr;
 ptr:=ptr^.nextI;
end; {while}
end; {FindEntry}

function fillin(i,j:integer;val:real):entryptr;

{ add an entry in the matrix }
var newptr:entryptr;
       ptr:entryptr;
   inserted:boolean;

begin
newptr:=FindEntry(row[i].ptr,j);
if newptr <> nil then newptr^.val := newptr^.val + val
else begin
 new(newptr);
 newptr^.i:=i;
 newptr^.j:=j;
 newptr^.val:=val;
 if j <> 0 then row[i].count:=row[i].count+1;
 if i <> 0 then col[j].count:=col[j].count+1;

 { insert entry in the row list }

 ptr:=row[i].ptr;
 if (ptr = nil) then begin
  row[i].ptr:=newptr;
  newptr^.nextI:=nil;
 end {if}
 else begin {2}
  if ( ptr^.j > j) then begin
   newptr^.nextI:=ptr;
   row[i].ptr:=newptr;
  end {if}
  else begin {1}
   inserted:=false;
   while not inserted and (ptr^.nextI <> nil) do
    if (ptr^.nextI^.j >= j) then begin
     newptr^.nextI:=ptr^.nextI;
     ptr^.nextI:=newptr;
     inserted:=true;
    end {if}
   else ptr:=ptr^.nextI;
   if not inserted then begin
    ptr^.nextI:=newptr;
    newptr^.nextI:=nil;
   end; {if not inserted}
  end; {else 1}
 end; {else 2}

 { insert entry in the column list }

 ptr:=col[j].ptr;
 if (ptr = nil) then begin
  col[j].ptr:=newptr;
  newptr^.nextJ:=nil;
 end
 else begin {2}
  if (ptr^.i >= i) then begin
   newptr^.nextJ:=ptr;
   col[j].ptr:=newptr;
  end
  else begin {1}
   inserted:=false;
   while not inserted and (ptr^.nextJ <> nil) do
    if (ptr^.nextJ^.i >= i) then begin
     newptr^.nextJ:=ptr^.nextJ;
     ptr^.nextJ:=newptr;
     inserted:=true;
    end {if}
    else ptr:=ptr^.nextJ;
   if not inserted then begin
    ptr^.nextJ:=newptr;
    newptr^.nextJ:=nil;
   end; {if}
  end; {else 1}
 end; {else 2}
end; { else }
fillin:=newptr;
end;

procedure fillval(var matptr:locptr; i, j:integer; val:real);

{ fill matrix locations according to the pointers stored in matptr }

begin
if (iter1 = 0) and (iter2 = 0) then begin
 matptr[1]:=fillin(i,i,val);
 matptr[2]:=fillin(j,j,val);
 matptr[3]:=fillin(i,j,-val);
 matptr[4]:=fillin(j,i,-val);
end
else begin
 matptr[1]^.val:=matptr[1]^.val+val;
 matptr[2]^.val:=matptr[2]^.val+val;
 matptr[3]^.val:=matptr[3]^.val-val;
 matptr[4]^.val:=matptr[4]^.val-val;
end; {else}
end;

procedure swapptr(var ptr1,ptr2:entryptr);
var ptr:entryptr;
begin
```

```
ptr:=ptr1; ptr1:=ptr2; ptr2:=ptr;
end;

procedure swapint(var i,j:integer);
var k:integer;
begin
k:=i; i:=j; j:=k;
end;

procedure swapval(var val1,val2:real);
var val:real;
begin
val:=val1; val1:=val2; val2:=val;
end;

procedure sortcol(j:integer);

{ sort the column ptr }
var colptr,ptr,temp,prev:entryptr;

  procedure swapcolptr(var ptr,ptr1:entryptr);

  var temp:entryptr;

  begin
  if ptr^.nextJ = ptr1 then begin    { swap adjacent pointers }
   ptr^.nextJ:=ptr1^.nextJ;
   ptr1^.nextJ:=ptr1^.nextJ^.nextJ;
   ptr^.nextJ^.nextJ:=ptr1;
   ptr1:=ptr^.nextJ;
  end
  else begin
   temp:=ptr^.nextJ^.nextJ;
   ptr^.nextJ^.nextJ:=ptr1^.nextJ^.nextJ;
   ptr1^.nextJ^.nextJ:=temp;
   temp:=ptr^.nextJ;
   ptr^.nextJ:=ptr1^.nextJ;
   ptr1^.nextJ:=temp;
  end;
  end; {procedure swapcolptr}

begin {sortcol}
colptr:=col[j].ptr;
while colptr^.nextJ <> nil do begin
 ptr:=colptr^.nextJ;
 while ptr^.nextJ <> nil do begin
  if (ptr^.nextJ^.i  < colptr^.nextJ^.i) then swapcolptr(colptr,ptr);
  ptr:=ptr^.nextJ;
 end; {while}

 if colptr = col[j].ptr then begin {if 2}
  if colptr^.i > colptr^.nextJ^.i then begin {if 1}
   temp:=colptr^.nextJ;
   colptr^.nextJ:=temp^.nextJ;
   temp^.nextJ:=colptr;
   col[j].ptr:=temp;
   end;  {if 1}
   prev:=col[j].ptr;
   colptr:=col[j].ptr;
  end {if 2}
  else begin
   if (prev^.nextJ^.i > colptr^.nextJ^.i) then
    swapcolptr(prev,colptr);
   prev:=prev^.nextJ;
  end;  {else}
  colptr:=colptr^.nextJ;
 end;  {while}
 end;  {procedure sortcol}

 procedure sortrow(j:integer);

 { sort the jth row pointers }

 var rowptr,ptr,temp,prev:entryptr;

   procedure swaprowptr(var ptr,ptr1:entryptr);

   var temp:entryptr;

   begin
   if ptr^.nextI = ptr1 then begin    { swap adjacent pointers }
    ptr^.nextI:=ptr1^.nextI;
    ptr1^.nextI:=ptr1^.nextI^.nextI;
    ptr^.nextI^.nextI:=ptr1;
    ptr1:=ptr^.nextI;
   end
   else begin
    temp:=ptr^.nextI^.nextI;
    ptr^.nextI^.nextI:=ptr1^.nextI^.nextI;
    ptr1^.nextI^.nextI:=temp;
    temp:=ptr^.nextI;
    ptr^.nextI:=ptr1^.nextI;
    ptr1^.nextI:=temp;
   end;
   end;

 begin  { sortrow }
 rowptr:=row[j].ptr;
 while rowptr^.nextI <> nil do begin
  ptr:=rowptr^.nextI;
  while ptr^.nextI <> nil do begin
   if (ptr^.nextI^.j  < rowptr^.nextI^.j) then
    swaprowptr(rowptr,ptr);
   ptr:=ptr^.nextI;
  end;  {while}

  if rowptr = row[j].ptr then begin {2}

   if rowptr^.j > rowptr^.nextI^.j then begin {1}
    temp:=rowptr^.nextI;
    rowptr^.nextI:=temp^.nextI;
    temp^.nextI:=rowptr;
   row[j].ptr:=temp;
  end; {if 1}

  prev:=row[j].ptr;
  rowptr:=row[j].ptr;
 end {if 2}
 else begin
  if (prev^.nextI^.j > rowptr^.nextI^.j) then
   swaprowptr(prev,rowptr);
  prev:=prev^.nextI;
 end; {else}
 rowptr:=rowptr^.nextI;
 end; {while}
end;

procedure swap(var ptr1,ptr2:headentry; i,j,code:integer);

var ptr:entryptr;

begin
swapptr(ptr1.ptr,ptr2.ptr);
swapint(ptr1.count,ptr2.count);
swapint(ptr1.oldindex,ptr2.oldindex);

case code of
1:  begin
    ptr:=ptr1.ptr;
    while ptr <> nil do begin
     ptr^.i:=i;
     ptr:=ptr^.nextI;
    end;

    ptr:=ptr2.ptr;
    while ptr <> nil do begin
     ptr^.i :=j;
     ptr:=ptr^.nextI;
    end;

    ptr:=ptr1.ptr;                    { sort column pointers }
    while ptr <> nil do begin
     sortcol(ptr^.j);
     ptr:=ptr^.nextI;
    end;

    ptr:=ptr2.ptr;                    { sort column pointers }
    while ptr <> nil do begin
     sortcol(ptr^.j);
     ptr:=ptr^.nextI;
    end;
    end;

2:  begin
    ptr:=ptr1.ptr;
    while ptr <> nil do begin
     ptr^.j := i;
     ptr:=ptr^.nextJ;
```

```
      end;

      ptr:=ptr2.ptr;
      while ptr <> nil do begin
       ptr^.j := j;
       ptr:=ptr^.nextJ;
      end;

      ptr:=ptr1.ptr;                         { sort column pointers }
      while ptr <> nil do begin
       sortrow(ptr^.i);
       ptr:=ptr^.nextJ;
      end;

      ptr:=ptr2.ptr;                         { sort column pointers }
      while ptr <> nil do begin
       sortrow(ptr^.i);
       ptr:=ptr^.nextJ;
      end;
     end; { 2: }
   end; {case}
  end;

  procedure CheckDiaZero(var row,col:headtype);

  { check for diagonal zeroes that may exist }
  var   i:integer;
      ptr:entryptr;
    pivot,backpivot:entryptr;
     find:boolean;

  begin
  for i:=1 to totnod do begin
   ptr:=FindEntry(row[i].ptr,i);
   if ptr <> nil then
    if ptr^.val = 0.0 then
     ptr:=nil;
   if ptr=nil  then begin
    pivot:=col[i].ptr;
    find:=false;
    backpivot:=nil;
    while (pivot <> nil) and (not find) do
     if (pivot^.i > i) and (pivot^.val <> 0.0) then
      find:=true
     else begin
      if (pivot^.i < i) and (pivot^.val <> 0.0) then
       backpivot:=pivot;
      pivot:=pivot^.nextJ;
     end; {else}
    if find then begin
     swapval(basr[i],basr[pivot^.i]);
     swap(row[pivot^.i],row[i],pivot^.i,i,1);
    end {if}
    else
     if (backpivot <> nil) then begin
      swapval(basr[i],basr[backpivot^.i]);
    swap(row[backpivot^.i],row[i],backpivot^.i,i,1);
   end {if}
   else
    writeln('Matrix is Singular. CheckDiaZero');
  end; {if}
 end; {for}
end;

procedure reorder(var row,col:headtype);

var minrow,i,j:integer;
    minprod,prod:integer;
    tempptr,ptr,pivot,pivotptr,rowptr:entryptr;
    greater:boolean;

begin
for i:=1 to totnod - 1 do begin
 minprod:=(row[i].count-1)*(col[i].count-1);
 minrow:=i;

 { find the pivot element using Markowitz algorithm }

 for j:=i + 1 to totnod do begin
  prod:=(row[j].count-1)*(col[j].count-1);
  if prod<minprod then begin
   minprod:=prod;
   minrow:=j;
  end
  else
   if (prod=minprod) and (col[j].count < col[minrow].count) then
       minrow:=j;
 end; {for}

 if (minrow<>i) then begin
  swap(row[i],row[minrow],i,minrow,1);
  swap(col[i],col[minrow],i,minrow,2);
  swapval(basr[i],basr[minrow]);
 end;

 { generate fillins from the pivot }

 pivot:=FindEntry(row[i].ptr,i);
 col[i].ptr:=pivot;
 ptr:=pivot^.nextI;
 while ptr <> nil do begin
  col[ptr^.j].count:=col[ptr^.j].count -1;
  ptr:=ptr^.nextI;
 end; {while}
 ptr:=pivot^.nextJ;
 while ptr<>nil do begin
  row[ptr^.i].count := row[ptr^.i].count - 1;
  pivotptr:=pivot^.nextI;
  rowptr:=ptr^.nextI;
  while pivotptr <> nil do begin
   greater:=false;
   while not greater and (rowptr <> nil) do
```

```
     if rowptr^.j < pivotptr^.j then
      rowptr:=rowptr^.nextI
     else greater := true;
    if rowptr = nil then
     tempptr:=fillin(ptr^.i,pivotptr^.j,0.0)
    else
     if pivotptr^.j <> rowptr^.j then
      tempptr:=fillin(ptr^.i,pivotptr^.j,0.0);
    pivotptr:=pivotptr^.nextI;
   end; {while}
   ptr:=ptr^.nextJ;
  end; {while}
 end; {for}
 col[totnod].ptr:=FindEntry(row[totnod].ptr,totnod);
 end; {procedure reorder}

 procedure lufactorize(var col:headtype);

 var
  pivot,ptr,pivotptr,rowptr:entryptr;
  i:integer;

 begin
 for i:=1 to totnod - 1 do begin
  pivot:=col[i].ptr;
  ptr:=pivot^.nextJ;
  while ptr <> nil do begin
   ptr^.val:=ptr^.val/pivot^.val;
   pivotptr:=pivot^.nextI;
   rowptr:=ptr^.nextI;
   while pivotptr <> nil do begin
    while rowptr^.j <> pivotptr^.j do
     rowptr:=rowptr^.nextI;
    rowptr^.val:=rowptr^.val - pivotptr^.val*ptr^.val;
    pivotptr:=pivotptr^.nextI;
   end; {while}
   ptr:=ptr^.nextJ;
  end; {while}
 end; { for }
 end; { procedure lufactorize }

 procedure solve(var row,col:headtype;var rhs,tasr:b1);

 var i:integer;
   pivot,ptr:entryptr;
   sum:real;

 begin
 { forward substitution }

 for i:=2 to totnod do begin
  ptr:=row[i].ptr;
  sum:=0.0;
  while (ptr^.j < i) do begin
   sum:=sum+ptr^.val*rhs[ptr^.j];
   ptr^.val:=0.0;
   ptr:=ptr^.nextI;
  end; {while}
  rhs[i]:=rhs[i]-sum;
 end; {for}

  { backward substitution }

 for i:= totnod downto 1 do begin
  pivot:=col[i].ptr;
  sum:=0.0;
  ptr:=pivot^.nextI;
  while ptr <> nil do begin
   sum:=sum + ptr^.val*rhs[ptr^.j];
   ptr^.val:=0.0;
   ptr:=ptr^.nextI;
  end;
  tasr[col[i].oldindex]:=(rhs[i] - sum) / pivot^.val;
  rhs[i]:=tasr[col[i].oldindex];
  pivot^.val:=0.0;
 end; {for}
 end; {solve}


 { ANALYSIS ROUTINES }

 procedure rstr(resr:res_array);
 var i:integer;
 begin
 for i:=1 to numres do
  with resr[i] do begin

   { calculate the first and second order temperature effect }
   if (tc1 <> 0.0) or (tc2<>0.0) then
    value:=value/(1.0+(t-300.0)*(tc1+(t-300)*tc2));

   { add value to the matrix location }
   fillval(matptr,n1,n2,val);
  end; { with }
 end;

 procedure caprs(var capr:cap_array;var capmod:capmod_array);
 var i,cm:integer;
  icn,gc,v1,v2,va,vb,vc,ovc,ic: real;
  cex,cpa,phi,val:real;
 begin
 for i:=1 to numcap do
  with capr[i] do begin

  { get previous iteration results }
  icn:=sma_val[1];
  gc:=sma_val[2];
  cm:=mod_pos;
  v1:=qasr[n1];
  va:=nasr[n1];
  v2:=qasr[n2];
  vc:=v1-v2;       { previous iteration vc }
  ovc:=va-vb;      { -2 iteration vc }

  { use initial condition if any }
  if ui and (initval<>0.0) then begin
   vc:=1;
   ovc:=vc;
  end;

  val:=value;

  { process with the capacitor if modeled }
  if (cm<>0) then begin
   cpa:=capmod[cm].value[1];
   cex:=capmod[cm].value[2];
   phi:=capmod[cm].value[3];
   val:=val*cpa/exp(cex*ln(1-vc/phi));
  end;

  { calculate the equivalent conductance and current
    and add to the matrix }
  int8(val,vc,ovc,icn,gc,ic);
  fillval(matptr,n1,n2,gc);
  basr[row[n1].oldindex].val:=basr[row[n1].oldindex].val+ic;
  basr[row[n2].oldindex].val:=basr[row[n2].oldindex].val+ic;
  sma_val[1]:=icn;
  sma_val[2]:=gc;
  end; {with}
end; {procedure caprs}

procedure sup8(vm:integer;var vo:real);
var i:integer;
 ti,th,tn:real;
begin
ti:=0.0;th:=0.0;tn:=0.0;
i:=3;
while (ti+tn<tstart1+tstep1) do
 begin
 th:=ti;
 ti:=sormod[vm].step[i];
 if (ti=0.0) then
  begin
  tn:=tn+th;  i:=3;
  end
 else begin
  if (ti+tn<tstart1+tstep1) then i:=i+2;
  if (i>22) then i:=3
  end;
 end;
if (ti+tn = tstart1+tstep1) then
 vo:=sormod[vm].step[i+1]
else with sormod[vm] do begin
 vo:=step[i+1]-(step[i+1]-step[i-1])*(ti+tn-tstart1-tstep1)/(ti-step[i-2]);
 end;
end;
```

```
procedure vsrc(var volsor:vol_array);
var i:integer;
    vo:real;
begin
for i:=1 to numvol do
 with volsor[i] do begin
 vo:=value;
 if trans and (mod_pos<>0) then sup8(mod_pos,vo);
 basr[row[numnod+i].oldindex].val:=vo;
 if (iter1 = 0) and (iter2 = 0) then begin
  matptr[1]:=fillin(nodep,numnod+i,1);
  matptr[2]:=fillin(nomnod+i,nodep,1);
  matptr[3]:=fillin(noden,numnod+i,-1);
  matptr[4]:=fillin(numnod+i,noden,-1);
  end
 else fillval(matptr,nodep,noden,1);
 end;
end;

procedure supi(im:integer;var io:real);
var i:integer;
  ti,th,tn:real;
begin
ti:=0.0;th:=0.0;tn:=0.0;
i:=3;
while (ti+tn<tstart1+tstep1) do
 begin
 th:=ti;
 ti:=sormod[im].step[i];
 if (ti=0.0) then begin
  tn:=tn+th;
  i:=3;
  end
 else
  begin
  if (ti+tn<tstart1+tstep1) then i:=i+2;
  if (i>11) then i:=3;
  end;
 end;
if (ti+tn<tstart1+tstep1) then
 io:=sormod[im].step[i+1]
else with sormod[im] do begin
 io:=step[i+1]-(step[i+1]-step[i-1])*(ti+tn-tstart1-tstep1)/(ti-step[i-2]);
 end;
end;

procedure isrc(var cursor:cur_array);
var i:integer;
  io:real;
begin
for i:=1 to numcur do
 with cursor[i] do begin
 io:=value;
 if trans and (mod_pos<>0) then supi(mod_pos,io);
 basr[row[nodep].oldindex].val:=basr[row[nodep].oldindex].val-io;
 basr[row[noden].oldindex].val:=basr[row[noden].oldindex].val+io;
```

```
  end;
 end;

 procedure diods(var diod:dio_array;var diomod:diomod_array);
 var i:integer;
  ido,gm,ad,is,vd,ovd,p7,id,gd,diff:real;
 begin
 for i:=1 to numdio do
  with diod[i] do begin
  ido:=sma_val[2];
  gm:=sma_val[3];
  ad:=scale;
  is:=ad*diomod[i].satcur;
  vd:=tasr[nca]-tasr[nan];
  ovd:=sma_val[1];
  if (ovd<0.0) and (vd<0.0) then begin
   diff:=vd-ovd;
   limt(vd,ovd,diff,5.0,0.0,50.0)
   end
  else begin
   id:=ido+is*gm;
   p7:=didt*id/(is*exp(ovd/(1.02*vt2))/(1.02*vt2));
   if (p7<0.1) then p7:=0.1;
   diff:=vd-ovd;
   limt(vd,ovd,diff,p7,-0.1,50.0);
   end; {else}
  sma_val[1]:=vd;
  id:=is*(exp(vd/(1.02*vt2))-1);
  gd:=id/(1.02*vt2);
  ido:=id-vd*sma_val[3];
  sma_val[2]:=ido;
  sma_val[3]:=gd;
  basr[ansr.row[nca].oldindex].val:=basr[ansr.row[nca].oldindex].val-ido;
  basr[ansr.row[nan].oldindex].val:=basr[ansr.row[nan].oldindex].val+ido;
  fillval(matptr,nca,nan,gd);
  end; {with}
 end;

 procedure mosf(var mosfets:mos_array;var mosmod:mosmod_array);

 var nd,ng,ns,nb,tt,i:integer;
     vgs,vds,vsb,lc1,ovgs,ovds,ovsb,dvgs,dvds,dvsb:real;
     ovt,ovsa,oids,gds,gm,gmb,vto,kp,gam,phi,lam,is,cox:real;
     ldi,bjc,vbi,xd,va,p37,p40,p54,p59,p60,p61,vdb:real;
     cgs,cgd,cgb,cdb,csb:real;
     dvgsq,dvgdq,dvgbq,dvdbq,dvsbq,dvgsn,dvgdn,dvgbn,dvdbn,dvsbn,oi:real;
     p45,p47,pb,gcgs,icgs,gcgd,icgd,gcgb,icgb,gcdb,icdb,gcsb,icsb:real;
     vcr,ttr,gsb,gdb,p39,xj,eex:real;

 procedure idsg1(i:integer);

 { level 1 mosfet current calculations }

 var k,fac,vgst:real;

 begin
```

```
with mosfet[i] do begin
 if vsb > 0 then
  begin
  fac:=sqrt(phi+vsb);
  ovt:=vto+gam*(fac-sqrt(phi));
  fac:=gam/(fac+fac);
  end
 else begin
  fac:=0;
  ovt:=vto;
 end;
 vgst:=vgs-ovt
 sma_val[1]:=ovt;
 ovt:=tt*ovt;
 ovsa:=max2(0.0,vgst);
 sma_val[2]:=ovsa;
 if (vgst < 0) then begin
  oids:=0.0; gm:=0.0; gds:=0.0; gmb:=0.0
  end
 else begin
  k:=kp*wc/lc1*(1+lam*vds);
  if (vgst <= vds) then   { saturation region }
   begin
    oids:=k*vgst*vgst*0.5*tt;
    gm:=k*vgst;
    gds:=lam*kp*wc/lc1*vgst*vgst*0.5;
    gmb:=gm*fac;
   end
  else begin               { linear region }
   oids:=k*vds*(vgst-0.5*vds);
   gm:=k*vds;
   gds:=k*(vgst-vds)+lam*kp*wc/lc1*vds*(vgs-0.5*vds);
   gmb:=bm*fac;
   end;
  end;
  gsb:=0.0; gdb:=0.0;
  oids:=oids*tt;
  sma_val[3]:=oids;
  gds:=gds+1e-8;
  sma_val[12]:=gds;
  sma_val[13]:=gm;
  oids:=(oids*tt-gm*vgs-gds*vds-gmb*vsb)*tt;
 end; {with}
end;

procedure idsg(i:integer);
var p41,p42,p43,p44,p45,p47,p48,p49:real;
begin
p41:=gam*(1-p37);
p44:=sqr(p41);
if (p44=0.0) then p42:=1e-20
else begin
 if(1.0+4.0*(vgs-vbi+phi+vsb)/p44<0.0) then p42:=1e-20
 else p42:=sqrt(1.0+4.0*(vgs-vbi+phi+vsb)/p44);
 end;
p43:=p44*(1.0-p42)/2.0;
```

```
p45:=vgs-vbi;
ovsa:=p45+p43;
ovsa:=max2(ovsa,0.0);
mosfets[i].sma_val[2]:=ovsa;
p47:=vsb+phi;
p48:=vds+p47;
if (vds>ovsa) then
 begin
 if (p47<0.0) or (ovsa+p47<0.0) then
  begin
  p47:=0.0; ovsa:=0.0;
  end;
 p49:=2*p41*(exp(1.5*ln(ovsa+p47))-exp(1.5*ln(p47)))/3.0;
 oids:=(p45-ovsa/2.0)*ovsa-p49;
 gds:=0;
 p44:=p45-ovsa-p41*sqrt(ovsa+p47);
 gm:=ovsa-(1-p42)*p44/p42;
 gmb:=p41*(sqrt(ovsa+p47)-sqrt(p47))-p44/p42;
 end
else
 begin
 if (p47<0.0) then p47:=0.0;
 if (p48<0.0) then p48:=0.0;
 p49:=2*p41*(sqrt(p48*p48*p48)-sqrt(p47*p47*p47))/3.0;
 oids:=vds*(p45-vds/2)-p49;
 gds:=p45-vds-p41*sqrt(p48);
 gm:=vds;
 gmb:=-p41*(sqrt(p48)-sqrt(p47));
 end;
end;

begin {mosf}
for i:=1 to nummos do
 with mosfets[i] do begin
 nd:=node[1];
 ng:=node[2];
 ns:=node[3];
 nb:=node[4];
 tt:=mosmod[mod_pos].define;

 vgs:=tt*(tasr[ng]-tasr[ns]);  {VGS}
 vds:=tt*(tasr[nd]-tasr[ns]);  {VDS}
 vsb:=tt*(tasr[ns]-tasr[nb]);   {VSB}

 if ui and initial then begin
  ovgs:=vgsi; vgs:=vgsi;
  ovds:=vdsi; vds:=vdsi;
  ovsb:=vsbi; vsb:=vsbi;
  dvgs:=0.0; dvds:=0.0; dvsb:=0.0;
  end
 else begin
  ovgs:=sma_val[4];
  ovds:=sma_val[5];
  ovsb:=sma_val[6];
  dvgs:=vgs-ovgs;
  dvds:=vds-ovds;
 dvsb:=vsb-ovsb;
 ovt:=tt*sma_val[1];
 ovsa:=sma_val[2];
 oids:=tt*sma_val[3];
 gds:=sma_val[12];
 gm:=sma_val[13];
 f9:=0;
 end;

vto:=mosmod[mod_pos].value[1];
kp:=mosmod[mod_pos].value[2];
gam:=mosmod[mod_pos].value[3];
phi:=mosmod[mod_pos].value[4];
lam:=mosmod[mod_pos].value[5];
is:=mosmod[mod_pos].value[12];
cox:=mosmod[mod_pos].value[11];
ldi:=mosmod[mod_pos].value[21];
bjc:=mosmod[mod_pos].value[24];
vbi:=mosmod[mod_pos].value[26];
xd:=mosmod[mod_pos].value[25];
lc1:=lc-2.0*ldi;

if (ovds<=0.0) and ((ovds<>0.0) or (vds<=0.0)) then begin
 f9:=1;
 vgs:=vgs-vds;
 ovgs:=ovgs-ovds;
 dvgs:=dvgs-dvds;
 vsb:=vsb+vds;
 ovsb:=ovsb+ovds;
 dvsb:=dvsb+dvds;
 vds:=-vds;
 ovds:=-ovds;
 dvds:=-dvds;
end;

limy(vgs,ovgs,dvgs,ovt);
limt(vsb,ovsb,dvsb,5.0,0.0,50.0);
if (nd-ng=0) or (ng-ns=0) and (f9<>0) then
 vds:=vgs
else
 begin
 if (ovgs>ovt) then
  begin
  if (vds>1.001*ovsa) then
   begin
   va:=didt*oids/(kp*(vgs-ovt));
   if (dvdt>va) then va:=dvdt;
   va:=ovsa-va-1.0e-3;
   limt(vds,ovds,dvds,5.0,va,50.0);
   end
  else
   begin
   va:=0.0;
   if (ovgs-ovt-ovds<>0.0) then
    va:=didt*oids/(kp*(vgs-ovt-ovds));
   if (dvdt>va) then va:=dvdt;
```

```
      limt(vds,ovds,dvds,va,0.0,50.0);
      end;
     end
    else
     limt(vds,ovds,dvds,5.0,0.0,50.0);
    end;

  if (level=1) then
    idsg1(i)
  else begin
   p37:=0.0;
   if short then begin
    xj:=mosmod[mod_pos].value[22];
    p37:=xd*sqrt(vsb+phi);
    p37:=xj*(sqrt(1.0+2.0*p37/xj)-1.0)/lc1;
    end;
   ovt:=vto+gam*(1-p37)*(sqrt(vsb+phi)-sqrt(phi));
   sma_val[1]:=ovt;
   ovt:=tt*ovt;
   p39:=0.0;
   if subt and (cox<>0.0) then
    begin
    p39:=qe*(mosmod[mod_pos].value[20]);
    p40:=gam/sqrt(4.0*(vsb+phi));
    p39:=vt2*(cox*p40+p39+cox)/cox;
    end;
   p40:=ovt+p39;  {VT+DEL}

   if (vgs<p40) then begin
    if subt then begin
     p54:=vgs;
     vgs:=p40;
     idsg(i);
     vgs:=p54;
     p54:=exp((vgs-p40)/p39);
     gds:=gds*p54;
     gm:=p54*(gm+oids/p39);
     gmb:=gmb*p54;
     oids:=oids*p54;
     end
    else begin
     oids:=0.0;gds:=0.0;gm:=0.0;gmb:=0.0;
    end;
   end
   else
    idsg(i);

   if (f9<>0) then begin
    gsb:=is*ad/vt2;
    gdb:=is*as/vt2;
   end
   else begin
    gsb:=is*as/vt2;
    gdb:=is*ad/vt2;
   end;
```

```
eex:=mosmod[mod_pos].value[18];
p40:=1;
if (eex<>0.0) then begin
 vcr:=mosmod[mod_pos].value[17];
 ttr:=mosmod[mod_pos].value[19];
 if (vcr<=vgs-ovt-ttr*vds) then
  p40:=exp(eex*ln(vcr/(vgs-ovt-ttr*vds)));
 end;

p60:=0;
if (lam=0.0) and chem then begin
 p59:=(vds-ovsa)/4.0;
 p60:=sqrt(p59*p59+1);
 p61:=sqrt(p59+p60);
 if (vds<1e-5) then kp:=0.0
 else begin
  lam:=xd*p61/(lc1*vds);
  p60:=xd*(1.0+p59/p60)/(8.0*lc1*vds*p61);
 end,
end;

gds:=gds+p60*vds*oids/(1.0-lam*vds);
kp:=kp*p40*wc/(lc1*(1-lam*vds));
oids:=oids*kp*tt;
sma_val[3]:=oids;
gds:=gds*kp+1e-8;
sma_val[12]:=gds;
gm:=gm*kp;
sma_val[13]:=gm;
gmb:=gmb*kp;
oids:=(oids*tt-gm*vgs-gds*vds-gmb*vsb)*tt;
end;

if (iter1 = 0) and (iter2 = 0) then begin
 mgbptr[4]:=fillin(nb,ns,-gsb);
 mgdptr[4]:=fillin(nb,nd,-gdb);
 mggptr[4]:=fillin(nb,nb,-gdb);
 if (f9<>0) then begin
  mgdptr[2]:=fillin(nd,ng,-gm);
  mgsptr[2]:=fillin(ns,ng,gm);
  mgsptr[3]:=fillin(nd,ns,-gds);
  mgsptr[4]:=fillin(ns,nd,-gm+gmb-gds);
  mgbptr[3]:=fillin(ns,nb,-gmb-gsb);
  mgdptr[3]:=fillin(nd,nb,gmb-gdb);
  mggptr[2]:=fillin(nd,nd,gm-gmb+gds+gdb);
  mggptr[3]:=fillin(ns,ns,gds+gsb);
  basr[row[nd].oldindex]:=basr[row[nd].oldindex]+oids;
  basr[row[ns].oldindex]:=basr[row[ns].oldindex]-oids;
 end {if}
 else begin
  mgdptr[2]:=fillin(nd,ng,gm);
  mgsptr[2]:=fillin(ns,ng,-gm);
  mgsptr[3]:=fillin(nd,ns,-gm+gmb-gds);
  mgsptr[4]:=fillin(ns,nd,-gds);
  mgbptr[3]:=fillin(ns,nb,gmb-gsb);
  mgdptr[3]:=fillin(nd,nb,-gmb-gdb);
```

```
    mggptr[2]:=fillin(nd,nd,gds+gdb);
    mggptr[3]:=fillin(ns,ns,gm-gmb+gds+gsb);
    basr[row[nd].oldindex]:=basr[row[nd].oldindex]-oids;
    basr[row[ns].oldindex]:=basr[row[ns].oldindex]+oids;
   end; {else}
  end {if}
  else begin
   mgbptr[4]^.val:=mgbptr[4]^.val - gsb;
   mgdptr[4]^.val:=mgdptr[4]^.val - gdb;
   mggptr[4]^.val:=mggptr[4]^.val - gdb;
   if f9 <> 0 then begin
    mgdptr[2]^.val:=mgdptr[2]^.val - gm;
    mgsptr[2]^.val:=mgsptr[2]^.val + gm;
    mgsptr[3]^.val:=mgsptr[3]^.val - gds;
    mgsptr[4]^.val:=mgsptr[4]^.val - gm+gmb-gds;
    mgbptr[3]^.val:=mgbptr[3]^.val - gmb-gsb;
    mgdptr[3]^.val:=mgdptr[3]^.val + gmb-gdb;
    mggptr[2]^.val:=mggptr[2]^.val + gm-gmb+gds+gdb;
    mggptr[3]^.val:=mggptr[3]^.val + gds+gsb;
    basr[row[nd].oldindex]:=basr[row[nd].oldindex]+oids;
    basr[row[ns].oldindex]:=basr[row[ns].oldindex]-oids;
   end
   else begin
    mgdptr[2]^.val:=mgdptr[2]^.val + gm;
    mgsptr[2]^.val:=mgsptr[2]^.val - gm;
    mgsptr[3]^.val:=mgsptr[3]^.val - gm+gmb-gds;
    mgsptr[4]^.val:=mgdptr[4]^.val - gds;
    mgbptr[3]^.val:=mgbptr[3]^.val + gmb-gsb;
    mgdptr[3]^.val:=mgdptr[3]^.val - gmb-gdb;
    mggptr[2]^.val:=mggptr[2]^.val + gds+gdb;
    mggptr[3]^.val:=mggptr[3]^.val + gm-gmb+gds+gsb;
    basr[row[nd].oldindex]:=basr[row[nd].oldindex]-oids;
    basr[row[ns].oldindex]:=basr[row[ns].oldindex]+oids;
   end; {else}
  end; {else of if iter1 = 0}

  if trans then
   begin
   cgs:=mosmod[mod_pos].value[6];  {CGSOVL}
   cgd:=mosmod[mod_pos].value[7];  {CGDOVL}
   cgb:=mosmod[mod_pos].value[8];  {CGB}
   cdb:=mosmod[mod_pos].value[9];  {CDB}
   csb:=mosmod[mod_pos].value[10]; {CSB}
   pb:=mosmod[mod_pos].value[13];  {PB}
   if (pb=0.0) then pb:=0.87;
   p40:=vgs-vds;  {VGD}
   vdb:=vds+vsb;  {VDB}
   {vgb:=p40+vdb;} {VGB}

   if (max6(cgs,cgd,cgb,cdb,csb,cox) > 0.0) then
    begin
    cgs:=wc*cgs;  {COVL*W}
    cgd:=cgd*wc;  {COVL*W}
    cgb:=cgb*wc*lc1;
    cdb:=cdb*ad;
    csb:=csb*as;
  if (f9<>0) then
   begin
   p40:=cgs;
   cgs:=cgd;
   cgd:=p40;
   end;

{ dsitribute cox over gate, drain, source, and bulk junctions }

  if (cox<>0.0) then
   begin
   cox:=cox*wc*lc1*1e4;
   if (vgs<ovt-phi) then cgb:=cgb+cox
   else
    if (vgs<=ovt-phi/2.0) then cgb:=cgb+cox*(ovt-vgs)/phi
    else
     if (vgs<ovt) then begin
      cgb:=cgb+cox*(ovt-vgs)/phi;
      cgs:=cgs+2.0*cox*(vgs-ovt+phi/2.0)/(1.5*phi);
      end
     else
      if (vgs<ovt+vds) then cgs:=cgs+2.0*cox/3.0
      else
       begin
       p45:=ovsa-vgs;
       p47:=2.0*ovsa-vgs;
       cgs:=cgs+2.0*cox*(1-p45*p45/(p47*p47))/3.0;
       cgd:=cgd+2.0*cox*(1-ovsa*ovsa/(p47*p47))/3.0;
       end
   end; { if cox <> 0.0 }

  if level <> 1 then
  if (vdb/pb>-1.0) and (vsb/pb>-1.0) then
   begin
   if (f9<>0) then begin
  cdb:=cdb/exp(bjc*ln(1+vsb/pb));
    csb:=csb/exp(bjc*ln(1+vdb/pb));
    end
   else begin
    cdb:=cdb/exp(bjc*ln(1+vdb/pb));
    csb:=csb/exp(bjc*ln(1+vsb/pb));
    end;
   end;

  dvgsq:=tt*(qasr[ng]-qasr[ns]);
  dvgsn:=tt*(nasr[ng]-nasr[ns]);
  dvgdq:=tt*(qasr[ng]-qasr[nd]);
  dvgdn:=tt*(nasr[ng]-nasr[nd]);
  dvgbq:=tt*(qasr[ng]-qasr[nb]);
  dvgbn:=tt*(nasr[ng]-nasr[nb]);
  dvdbq:=tt*(qasr[nd]-qasr[nb]);
  dvdbn:=tt*(nasr[nd]-nasr[nb]);
  dvsbq:=tt*(qasr[ns]-qasr[nb]);
  dvsbn:=tt*(nasr[ns]-nasr[nb]);

  oi:=sma_val[7];
  int8(cgs,dvgsq,dvgsn,oi,gcgs,icgs);
  sma_val[7]:=oi;
  oi:=sma_val[8];
  int8(cgd,dvgdq,dvgdn,oi,gcgd,icgd);
  sma_val[8]:=oi;
  oi:=sma_val[11];
  int8(cgb,dvgbq,dvgbn,oi,gcgb,icgb);
  sma_val[11]:=oi;
 if (f9<>0) then
   begin
   oi:=sma_val[10];
   int8(cdb,dvdbq,dvdbn,oi,gcdb,icdb);
   sma_val[10]:=oi;
   oi:=sma_val[9];
   int8(csb,dvsbq,dvsbn,oi,gcsb,icsb);
   end
  else
   begin
   oi:=sma_val[10];
   int8(cdb,dvdbq,dvdbn,oi,gcdb,icdb);
   sma_val[10]:=oi;
   oi:=sma_val[9];
   int8(csb,dvsbq,dvsbn,oi,gcsb,icsb);
   end;

  if (iter1 = 0) and (iter2 = 0) then begin
   mgbptr[2]:=fillin(ng,nb,-gcgb);
   mgbptr[1]:=fillin(nb,ng,-gcgb);
   mgbptr[3]:=fillin(ns,nb,-gcsb);
   mgbptr[4]:=fillin(nb,nd,-gcsb);
   mgdptr[3]:=fillin(nd,nb,-gcdb);
   mgdptr[4]:=fillin(nb,nd,-gcdb);
   mggptr[1]:=fillin(ng,ng,gcgd+gcgb+gcgs);
   mggptr[4]:=fillin(nb,nb,gcgb+gcdb+gcsb);
   basr[row[ng].oldindex]:=basr[row[ng].oldindex]+tt*(icgs+icgd+icgb);
   basr[row[nb].oldindex]:=basr[row[nb].oldindex]-tt*(icgb+icdb+icsb);
   if (f9<>0) then begin
    mgdptr[1]:=fillin(ng,nd,-gcgs);
    mgdptr[2]:=fillin(nd,ng,-gcgs);
    mgsptr[1]:=fillin(ng,ns,-gcgd);
    mgsptr[2]:=fillin(ns,ng,-gcgd);
    mggptr[2]:=fillin(nd,nd,gcgs+gcdb);
    mggptr[3]:=fillin(ns,ns,gcgd+gcsb);
    basr[row[nd].oldindex]:=basr[row[nd].oldindex]-tt*(icgs-icdb);
    basr[row[ns].oldindex]:=basr[row[ns].oldindex]-tt*(icgd-icsb);
   end
   else begin
    mgdptr[1]:=fillin(ng,nd,-gcgd);
    mgdptr[2]:=fillin(nd,ng,-gcgd);
    mgsptr[1]:=fillin(ng,ns,-gcgs);
    mgsptr[2]:=fillin(ns,ng,-gcgs);
    mggptr[2]:=fillin(nd,nd,gcgd+gcdb);
    mggptr[3]:=fillin(ns,ns,gcgs+gcsb);
    basr[row[nd].oldindex]:=basr[row[nd].oldindex]-tt*(icgd-icdb);
    basr[row[ns].oldindex]:=basr[row[ns].oldindex]-tt*(icgs-icsb);
   end;
```

```
    end
    else begin {2}
     mgbptr[2]^.val:=mgbptr[2]^.val - gcgb;
     mgbptr[1]^.val:=mgbptr[1]^.val - gcgb;
     mgbptr[3]^.val:=mgbptr[3]^.val - gcsb;
     mgbptr[4]^.val:=mgbptr[4]^.val - gcsb;
     mgdptr[3]^.val:=mgdptr[3]^.val - gcdb;
     mgdptr[4]^.val:=mgdptr[4]^.val - gcdb;
     mggptr[1]^.val:=mggptr[1]^.val + gcgd+gcgb+gcgs;
     mggptr[4]^.val:=mggptr[4]^.val + gcgb+gcdb+gcsb;
     basr[row[ng].oldindex]:=basr[row[ng].oldindex]+tt*(icgs+icgd+icgb);
     basr[row[nb].oldindex]:=basr[row[nb].oldindex]-tt*(icgb+icdb+icsb);
     if (f9<>0) then begin
      mgdptr[1]^.val:=mgdptr[1]^.val - gcgs;
      mgdptr[2]^.val:=mgdptr[2]^.val - gcgs;
      mgsptr[1]^.val:=mgsptr[1]^.val - gcgd;
      mgsptr[2]^.val:=mgsptr[2]^.val - gcgd;
      mggptr[2]^.val:=mggptr[2]^.val + gcgs+gcdb;
      mggptr[3]^.val:=mggptr[3]^.val + gcgd+gcsb;
      basr[row[nd].oldindex]:=basr[row[nd].oldindex]-tt*(icgs-icdb);
      basr[row[ns].oldindex]:=basr[row[ns].oldindex]-tt*(icgd-icsb);
     end
     else begin {1}
      mgdptr[1]^.val:=mgdptr[1]^.val - gcgd;
      mgdptr[2]^.val:=mgdptr[2]^.val - gcgd;
      mgsptr[1]^.val:=mgsptr[2]^.val - gcgs;
      mgsptr[2]^.val:=mgsptr[2]^.val - gcgs;
      mggptr[2]^.val:=mggptr[2]^.val + gcgd+gcdb;
      mggptr[3]^.val:=mggptr[3]^.val + gcgs+gcsb;
      basr[row[nd].oldindex]:=basr[row[nd].oldindex]-tt*(icgd-icdb);
      basr[row[ns].oldindex]:=basr[row[ns].oldindex]-tt*(icgs-icsb);
      end; {else 1}
     end; {esle 2}
    end;
   end;

 if (f9<>0) then
  begin
  vgs:=vgs-vds;
  vsb:=vsb+vds;
  vds:=-vds;
  end;
 sma_val[4]:=vgs;
 sma_val[5]:=vds;
 sma_val[6]:=vsb;
 f9:=0;
 end;
end;

{ CIRCUIT EDITOR ROUTINES }

procedure resrc(var resr:res_array);

{ retrieve and store the data for a resistor }

var n1,n2,i: integer;
```

```
    val,tc1,tc2: real;
begin
gnam(linepos,inline,name);        { get the name of the resistor }
getint(n1,nonum);                 { get node 1 of the resistor }
getint(n2,nonum);                 { get node 2 of the resistor }
getreal(val,nonum);               { get value of the resistor }
if nonum then errr(7,inline)
else
 begin
 getreal(tc1,nonum);              { get optional temperature coeficient }
 getreal(tc2,nonum);              { get optional temperature coeficient }

 { check if the resistor is already in before }
 i := 1;
 while (i <= numres) and (resr[i].name <> name) do
  i:=i+1;

 if (i > maxres) then
  errr(15,inline)
 else
  begin
   if (i > numres) then begin   { new resistor }
     numres:=numres + 1;
     resr[i].name := name;
    end;
   gnod(n1,numnod,node);          { store data in resistor record }
   gnod(n2,numnod,node);
   resr[i].n1:=n1;
   resr[i].n2:=n2;
   resr[i].value:=1/val;
   resr[i].tc1:=tc1;
   resr[i].tc2:=tc2;
  end;
 end
end;

procedure caprc(var capr:cap_array);

{ retrieve data for a capacitor }
var n1,n2,i,j,k,initcond: integer;
    mod_name:name_string;
    val,initval: real;
begin
gnam(linepos,inline,name);        { get name of the capacitor }
initcond:=0;                      { no initial condition }
getint(n1,nonum);
getint(n2,nonum);
getreal(val,nonum);
if nonum then errr(7,inline)
else begin
 i := 1;
 while (i <= numcap) and (capr[i].name <> name) do
  i:=i+1;
 if ( i > maxcap) then errr(15,inline)
 else
  begin
```

```
   getreal(initval,nonum);      { get initial value of the capacitor }
   if (not nonum) then
    initcond:=1;
   j:=0;
   if (linepos <= leng) and (inline[linepos] = 'c') then
    begin
     gnam(linepos,inline,mod_name);         { get model name if any}
     j:=1;
     while (j <= numcapm) and (capmod[i].name <> mod_name) do
      j:=j+1;
     if j > maxcapm then errr(16,inline)
     else
      if (j > numcapm) then begin
        numcapm := numcapm+1;
        capmod[j].name:=mod_name;
        for k:=1 to 3 do
         capmod[j].value[k]:=0.0;
       end;
    end;
   if (j <= maxcapm) then
    begin
     if i > numcap then begin
       numcap := numcap + 1;
       capr[i].name := name;
      end;
     gnod(n1,numnod,node);      { node transformation }
     gnod(n2,numnod,node);      { node transformation }
     capr[i].n1:=n1;            { store data in capacitor record }
     capr[i].n2:=n2;
     capr[i].mod_pos:=j;
     capr[i].value := val;
     capr[i].sma_val[1] :=0.0;
     capr[i].sma_val[2] :=0.0;
     if (initcond <> 0) then
      capr[i].initval := initval
     else
      capr[i].initval := 0;
    end;
  end;
 end;
end;

procedure mosfc(var mosfets:mos_array);

{ retrieve and store data for a mosfet }
var nodes:array[1..4] of integer;   { store node number }
    para:array[1..4] of real;       { store initial junction voltage }
    mod_name:name_string;
    i,j,k:integer;
begin
gnam(linepos,inline,name);      {"get name of the mos }
for i := 1 to 4 do              { get node number of the mosfet }
 getint(nodes[i],nonum);
if nonum then errr( 7,inline)
else
 begin
```

```
i:=1;
while(i<=nummos) and (mosfets[i].name <> name) do
 i:=i+1;
if (i > maxmos) then errr(15,inline)
else
 if (linepos <= leng) and (inline[linepos] = 'm') then
  begin
   gnam(linepos,inline,mod_name);
   j:=1;
   while (j<=nummosm) and (mosmod[j].name <> mod_name) do
    j:=j+1;
   if j > maxmosm then errr(16,inline)
   else
    if (j > nummosm) then
     begin
      nummosm:=nummosm+1;
      mosmod[j].name:=mod_name;
      mosmod[j].define:=0;
      for k:=1 to 26 do
       mosmod[j].value[k]:=0;
     end;
   if j <= maxmosm then
    begin              { store data in mosfet record }
     if (i > nummos) then begin
       nummos:=nummos+1;
       mosfets[i].name:=name;
      end;
     mosfets[i].mod_pos:=j;
     for j:=1 to 4 do begin
       gnod(nodes[j],numnod,node);
       mosfets[i].node[j]:=nodes[j];
      end;
     for k:=1 to 4 do
      getreal(para[k],nonum);
     if (para[1] = 0.0) then para[1]:=1.0e-5;
     if (para[2] = 0.0) then para[2]:=1.0e-5;
     mosfets[i].satlin:=0;
     mosfets[i].cutlin:=0;
     mosfets[i].wc:=para[1];
     mosfets[i].lc:=para[2];
     mosfets[i].ad:=para[3];
     mosfets[i].as:=para[4];
     getreal(mosfets[i].vgsi,nonum);
     if nonum then
      mosfets[i].initial:=false
     else
      mosfets[i].initial:=true;
     getreal(mosfets[i].vdsi,nonum);
     getreal(mosfets[i].vsbi,nonum);
     for k:=1 to 13 do
      mosfets[i].sma_val[k]:=0.0;
    end;
  end
 else
  errr(13,inline);
end;
```

```
end;

procedure diodc(var diod:dio_array);

{ retrive and store the data for a diode }
var i,j,k,n1,n2:integer;
    mod_name:name_string;
    scale:real;
begin
gnam(linepos,inline,name);      { get name of the diode }
getint(n1,nonum);
getint(n2,nonum);
if nonum then errr(7,inline)
else
 begin
  i:=1;
  while(i<=numdio) and (diod[i].name <> name) do
   i:=i+1;
  if (i > maxdio) then errr(15,inline)
  else
   if (linepos <= leng) and (inline[linepos] = 'd') then
    begin
     gnam(linepos,inline,mod_name);
     j:=1;
     while (j<=numdiom) and (diomod[j].name <> mod_name) do
      j:=j+1;
     if j > maxdiom then errr(16,inline)
     else
      if (j > numdiom) then             { store data in diode record }
       begin
        numdiom:=numdiom+1;
        diomod[j].name:=mod_name;
        diomod[j].satcur:=0.0;
       end;
     if j <= maxdiom then
      begin
       if i > numdio then
        begin
         numdio:=numdio+1;
         diod[i].name:=name;
        end;
       gnod(n1,numnod,node);
       gnod(n2,numnod,node);
       diod[i].nca:=n1;
       diod[i].nan:=n2;
       getreal(diod[j].scale,nonum);
       if (diod[j].scale=0.0) then diod[j].scale:=1.0;
       diod[i].mod_pos:=j;
       for k:=1 to 3 do
        diod[i].sma_val[k]:=0;
      end;
    end
   else
    errr(13,inline);
 end;
end;
```

```
procedure voltc(var volsor:vol_array);

{ retrieve and store the data for a voltage source }
var i,j,k,n1,n2:integer;
    mod_name:name_string;
    val:real;
begin
gnam(linepos,inline,name);
getint(n1,nonum);
getint(n2,nonum);
getreal(val,nonum);
if nonum then errr(7,inline)
else
 begin
 i:=1;
 while(i<=numvol) and (volsor[i].name <> name) do
  i:=i+1;
 if i > maxvso then errr(15,inline)
 else
  begin
   j:=0;
   if (linepos < leng) and (inline[linepos] = 's') then
    begin
     gnam(linepos,inline,mod_name);
     j:=1;
     while (j<=numsorm) and (sormod[j].name <> mod_name) do
      j:=j+1;
     if j > maxsorm then errr(16,inline)
     else
      if j > numsorm then
      begin
       numsorm:=numsorm+1;
       sormod[j].name:=mod_name;
       sormod[j].define:=0;
       for k:=1 to 22 do
        sormod[j].step[k]:=0;
      end;
    end;
   if (j <= maxsorm) then
    begin
     if i > numvol then begin
       numvol:=numvol+1;
       volsor[i].name:=name;
      end;
     gnod(n1,numnod,node);
     gnod(n2,numnod,node);
     volsor[i].nodep:=n1;
     volsor[i].noden:=n2;
     volsor[i].mod_pos:=j;
     volsor[i].value:=val;
     if (n2=0) then nodeval[n1]:=val;
     if (n1=0) then nodeval[n2]:=-val;
    end;
  end;
 end;
```

```
end;

procedure curnc(var cursol:cur_array);

{ retrieve data for a current source }
var i,j,k,n1,n2:integer;
    mod_name:name_string;
    val:real;
begin
gnam(linepos,inline,name);
getint(n1,nonum);
getint(n2,nonum);
getreal(val,nonum);
if nonum then errr(7,inline)
else
 begin
 i:=1;
 while(i<=numcur) and (cursor[i].name <> name) do
  i:=i+1;
 if i > maxcso then errr(15,inline)
 else
  begin
   j:=0;
   if (linepos < leng) and (inline[linepos] = 's') then
    begin
     gnam(linepos,inline,mod_name);
     j:=1;
     while (j<=numsorm) and (sormod[j].name <> mod_name) do
      j:=j+1;
     if j > maxsorm then errr(16,inline)
     else
      if j > numsorm then
       begin
       numsorm:=numsorm+1;
       sormod[j].name:=mod_name;
       sormod[j].define:=0;
       for k:=1 to 22 do
        sormod[j].step[k]:=0;
      end;
    end;
   if j <= maxsorm then
    begin
     if i > numcur then begin   { store data in current source record }
       numcur:=numcur+1;
       cursor[i].name:=name;
      end;
     gnod(n1,numnod,node);
     gnod(n2,numnod,node);
     cursor[i].nodep:=n1;
     cursor[i].noden:=n2;
     cursor[i].mod_pos:=j;
     cursor[i].value:=val;
    end;
  end;
 end;
end;
```

```
procedure smodc(var sormod:sormod_array);

{ get and process the source model data }
var i,j,k:integer;
    finish:boolean;
    p3,p5,p6,p7,p8,p9,p10,pa:  real;
begin
gnam(linepos,inline,name);
i:=1;
while (i<=numsorm) and (sormod[i].name <> name) do
 i:=i+1;
if i > maxsorm then errr(15,inline)
else
 begin
 if i > numsorm then
  begin
   numsorm:=numsorm + 1;
   sormod[i].name :=name;
   sormod[i].define :=-1;
   for k:=1 to 22 do
    sormod[i].step[k]:=0.0;
  end;
 getreal(p3,nonum);
 if nonum then
  begin
  k:=2;sormod[i].define:=1;
  write(' INITIAL VALUE ');
  readln(inline);  linepos:=1;
  leng:=strlen(inline);
  getreal(p3,nonum);
  if not nonum then
   begin
   sormod[i].step[1]:=0;
   sormod[i].step[2]:=p3;
   finish:=false;
   while (k<22) and not finish do
    begin
    write(' TIME ');
    readln(inline);linepos:=1;
    leng:=strlen(inline);
    getreal(p3,nonum);
    if not nonum then
     begin
     if (p3 <> 0.0) then
      begin
      sormod[i].step[k+1]:=p3;
      write(' VALUE:  ');
      readln(inline);linepos:=1;
      leng:=strlen(inline);
      getreal(p3,nonum);
      if not nonum then
       begin
       sormod[i].step[k+2] :=p3;
       k:=k+2;
       end
     else finish:=true;
     end
    else finish:=true
    end;
   end;
  end
 end
else
 begin
 getreal(p5,nonum);
 if nonum then errr(7,inline)
 else
  begin
  getreal(p6,nonum);
  if nonum then errr(7,inline)
  else
   begin
   getreal(p7,nonum);
   if nonum then errr(7,inline)
   else
    begin
    getreal(p10,nonum);
    if nonum then
     begin
     p10:=2.0*p7;
     p8:=p7/10.0;
     p9:=p7/10.0;
     end
    else
     begin
     getreal(pa,nonum);
     if not nonum then p8:=pa
     else p8:=0;
     getreal(pa,nonum);
     if not nonum then p9:=pa
     else p9:=0;
     end;
    sormod[i].define:=1;
    sormod[i].step[1]:=0;
    sormod[i].step[2]:=p3;  {initial value}
    sormod[i].step[3]:=p6;  {td}
    sormod[i].step[4]:=p3;
    sormod[i].step[5]:=p6+p8;
    sormod[i].step[6]:=p5;  {final value}
    sormod[i].step[7]:=p6+p7+p8;
    sormod[i].step[8]:=p5;
    sormod[i].step[9]:=p6+p7+p8+p9;
    sormod[i].step[10]:=p3;
    sormod[i].step[11]:=p10;  {period}
    sormod[i].step[12]:=p3;
    sormod[i].step[13]:=p10+p6;
    sormod[i].step[14]:=p3;
    sormod[i].step[15]:=p10+p6+p8;
    sormod[i].step[16]:=p5;
    sormod[i].step[17]:=p10+p6+p7+p8;
    sormod[i].step[18]:=p5;
```

```
        sormod[i].step[19]:=p10+p6+p7+p8+p9;
        sormod[i].step[20]:=p3;
        sormod[i].step[21]:=2.0*p10;
        sormod[i].step[22]:=p3;
        end;
       end;
      end;
     end;
    end;
   end;

   procedure veryc;

   { Print the connectivity of each node. If a node has less than two connecti
     it will print warning messages for DC and transient analysises. }

   var i,j,k,count,line: integer;
   begin
   page(outfile);
   line:=1;
   writeln(outfile,' ':20,'*NODE TABLE*');pause(line);
   for i := 1 to numnod do
    begin
    count := 0;        { count number of connection in a node }

    { print out the node number }
    writeln(outfile);pause(line);
    writeln(outfile,' ':15,'NODE',node[i]:4);pause(line);
    writeln(outfile);pause(line);

    for j := 1 to nummos do          { print the mos }
     for k:=1 to 4 do
      if (i = mosfets[j].node[k]) then
       begin
        write(outfile,' ':20,mosfets[j].name);

        case k of
        1: writeln(outfile,'  Drain');
        2: writeln(outfile,'  Gate');
        3: writeln(outfile,'  Source');
        4: writeln(outfile,'  Bulk');
        end;

        count := count+1;
        pause(line);
       end;

    for j:=1 to numvol do           { print voltage source }
     begin
     if (i = volsor[j].nodep) then begin
      count:= count+1;
      writeln(outfile,' ':20,volsor[j].name,'  +Node');pause(line);
      end;
     if (i = volsor[j].noden) then begin
      count := count+1;
      writeln(outfile,' ':20,volsor[j].name,'  -Node');pause(line);
```

```
  end;
 end;

for j := 1 to numcur do         { print current source }
 begin
 if (i = cursor[j].nodep) then begin
  count:= count+1;
  writeln(outfile,' ':20,cursor[j].name,'  From'); pause(line);
  end;
 if (i = cursor[j].noden) then begin
  count:= count+1;
  writeln(outfile,' ':20,cursor[j].name,'  To'); pause(line);
  end;
 end;

for j:= 1 to numdio do          { print diode }
 begin
 if (i = diod[j].nan) then begin
  count := count+1;
  writeln(outfile,' ':20,diod[j].name,'  Anode'); pause(line);
  end;
 if (i = diod[j].nca) then begin
  count := count+1;
  writeln(outfile,' ':20,diod[j].name,'  Cathod'); pause(line);
  end;
 end;

for j:= 1 to numres do          { print resistor }
 begin
 if (i = resr[j].n1) then begin
  count :=count+1;
  writeln(outfile,' ':20,resr[j].name); pause(line);
  end;
 if (i = resr[j].n2) then begin
  count := count + 1;
  writeln(outfile,' ':20,resr[j].name); pause(line);
  end;
 end;

if (count <2) then begin
 writeln(outfile,'  WARNING:    DC'); pause(line);
 end;

for j:= 1 to numcap do          { print capacitor }
 begin
 if (i = capr[j].n1) then begin
  count := count+1;
  writeln(outfile,' ':20,capr[j].name); pause(line);
  end;
 if (i = capr[j].n2) then begin
  count := count+1;
  writeln(outfile,' ':20,capr[j].name); pause(line);
  end;
 end;

if (count < 2) then begin
```

```
    writeln(outfile,'     WARNING:  TR'); pause(line);
    end;
  end;
end;

procedure newcc;

{ clear arrays for new circuit }
var i: integer;
begin
 for i:= 1 to maxequ do
  begin
  node[i] := 0;
  nodeval[i] := 0.0;
  end;
 numres := 0;
 numcap := 0;
 nummos := 0;
 numcur := 0;
 numdio := 0;
 numnod := 0;
 numsorm := 0;
 numvol := 0;
end;

procedure popr(var resr:res_array;name:name_string);

{ delete a resistor from resistor array }
var i,j:integer;
begin
i:=1;
while (i<=numres) and (resr[i].name <> name) do
 i:=i+1;
if (i>numres) then errr(6,inline)
else begin
  numres:=numres-1;
  for j:=i to numres do
   resr[j]:=resr[j+1];
 end;
end;

procedure popc(var capr:cap_array;name:name_string);

{ delete a capacitor from capacitor array }
var i,j:integer;
begin
i:=1;
while (i<=numcap) and (capr[i].name <> name) do
 i:=i+1;
if (i>numcap) then errr(6,inline)
else begin
  numcap:=numcap-1;
  for j:=i to numcap do
   capr[j]:=capr[j+1];
 end;
end;
```

```
procedure popm(var mosfets:mos_array;name:name_string);

{ delete a mosfet from mos array }
var i,j:integer;
begin
i:=1;
while (i<=nummos) and (mosfets[i].name <> name) do
 i:=i+1;
if (i>nummos) then errr(6,inline)
else begin
  nummos:=nummos-1;
  for j:=i to nummos do
   mosfets[j]:=mosfets[j+1];
 end
end;

procedure popd(var diod:dio_array;name:name_string);

{ delete a diode from mos array}
var i,j:integer;
begin
i:=1;
while ( i<=numdio) and (diomod[i].name <> name) do
 i:=i+1;
if (i>numdio) then errr(6,inline)
else begin
  numdio:=numdio-1;
  for j:=i to numdio do
   diomod[j]:=diomod[j+1];
 end;
end;

procedure popv(var volsor:vol_array;name:name_string);

{ delete a voltage source from voltage source array }
var i,j:integer;
begin
i:=1;
while (i<=numvol) and  (volsor[i].name <> name) do
 i:=i+1;
if (i>numvol) then errr(6,inline)
else begin
  numvol:=numvol-1;
  for j:=i to numvol do
   volsor[j]:=volsor[j+1];
  end;
end;

procedure popi(var cursor:cur_array;name:name_string);

{ delete a current source from current source array }
var i ,j:integer;
begin
i:=1;
while (i<=numcur) and (cursor[i].name <> name) do
```

```
 i:=i+1;
if (i>numcur) then errr(6,inline)
else begin
  numcur:=numcur-1;
  for j:=i to numcur do
   cursor[j]:=cursor[j+1];
  end
end;

procedure kilec;

{ delete a element from its corresponding element array }
begin
skipchar(inline,linepos);
if (not (inline[linepos] in ['a'..'z'])) then errr(6,inline)
else begin
 gnam(linepos,inline,name);
 case name[1] of
 'r':popr(resr,name);
 'c':popc(capr,name);
 'm':popm(mosfets,name);
 'd':popd(diod,name);
 'v':popv(volsor,name);
 'i':popi(cursor,name);
 otherwise
  errr(6,inline);
 end;
 end;
end;

procedure presc(var nodeval:val_array);

{ preset node voltage }
var nonum:boolean;
    p1:integer; p2:real;
begin
skipchar(inline,linepos);
nonum:=false;
while not nonum do
 begin
 getint(p1,nonum);                { get node number }
 getreal(p2,nonum);               { get value of the node }
 if not nonum and (p1<>0) then
  begin
   gnod(p1,numnod,node);
   nodeval[p1]:=p2;
  end;
 end;
end;

procedure ListCirc;

{ list all the elements in the circuit }
var init:boolean;
    i,j,line:integer;
begin
```

```
page(outfile);
line:=1;
writeln(outfile,' ':20,'CIRCUIT ELEMENT');pause(line);
writeln(outfile);pause(line);

if (numres>0) then                 { print resistor elements }
 begin
 writeln(outfile,' ':20,'*RESISTOR*');pause(line);
 writeln(outfile,' ':5,'NAME  N1 N2    VALUE TC1','TC2':12);pause(line);
 for i:=1 to numres do
  begin
  write(outfile,' ':5,resr[i].name,node[resr[i].n1]:4,node[resr[i].n2]:3);
  write(outfile,'  ');preal(1/resr[i].value);write(outfile,' ':2);
  if (resr[i].tc1 <> 0.0) then preal(resr[i].tc1)
  else write(outfile,' ':11);write(outfile,' ':2);
  if (resr[i].tc2 <> 0.0) then preal(resr[i].tc2);
  writeln(outfile);pause(line);
  end;
 end;
writeln(outfile);pause(line);

if (numcap>0) then                 { print capacitor elements }
 begin
 writeln(outfile,' ':20,'*CAPACITOR*');pause(line);
 writeln(outfile,' ':5,'NAME  N1 N2    VALUE      I. C.      MODEL');
 pause(line);
 for i:=1 to numcap do
  begin
  write(outfile,' ':5,capr[i].name,node[capr[i].n1]:4,node[capr[i].n2]:3);
  write(outfile,'  ');preal(capr[i].value); write(outfile,'  ');
  if (capr[i].initval <> 0.0) then preal(capr[i].initval)
  else write(outfile,' ':11);
  if (capr[i].mod_pos <> 0) then
    write(outfile,' ':5,capmod[capr[i].mod_pos].name);
  writeln(outfile);pause(line);
  end;
 end;
writeln(outfile);pause(line);

if (numvol>0) then                 { print voltage source }
 begin
 writeln(outfile,' ':20,'*VOLTAGES SOURCES*');pause(line);
 writeln(outfile,' ':5,'NAME  N+ N-    VALUE      MODEL');pause(line);
 for i:=1 to numvol do
  begin
  write(outfile,' ':5,volsor[i].name,node[volsor[i].nodep]:4,
                    node[volsor[i].noden]:3);
  write(outfile,'  ');preal(volsor[i].value);
  if (volsor[i].mod_pos <> 0 ) then
        write(outfile,' ':5,sormod[volsor[i].mod_pos].name);
  writeln(outfile);pause(line);
  end;
 end;
writeln(outfile);pause(line);

if (numcur>0) then
```

```
   begin
   writeln(outfile,' ':20,'*CURRENT SOURCES*');pause(line);
   writeln(outfile,' ':5,'NAME   NF NT    VALUE       MODEL');pause(line);
   for i:=1 to numcur do
    begin
    write(outfile,' ':5,cursor[i].name,node[cursor[i].nodep]:4,
                        node[cursor[i].noden]:3);
    write(outfile,'   ');preal(cursor[i].value);
    if (cursor[i].mod_pos <> 0) then
      write(outfile,' ':5,sormod[cursor[i].mod_pos].name);
    writeln(outfile);pause(line);
    end;
   end;
  writeln(outfile);pause(line);

  if (numdio>0) then                  { print diode elements }
   begin
   writeln(outfile,' ':20,'*DIODES*');pause(line);
   writeln(outfile,' ':5,'NAME N+ N-    AREA      MODEL');pause(line);
   for i:=1 to numdio do
    begin
    write(outfile,' ':5,diod[i].name,node[diod[i].nca]:4,node[diod[i].nan]:3)
    write(outfile,'    ',diod[i].scale:3:2);
    if (diod[i].mod_pos <> 0) then
     writeln(outfile,' ':5,diomod[diod[i].mod_pos].name)
    else writeln(outfile);
    pause(line);
    end;
   end;
  writeln(outfile);pause(line);

  init:=false;
  if (nummos > 0) then
   begin
   writeln(outfile,' ':20,'*MOSFETS*');pause(line);
   writeln(outfile,' ':5,'NAME  ND NG NS NB    MODEL',
                         'WC':8,'LC':11,'AD':11,'AS':11);pause(line);
   for i := 1 to nummos do
    begin
    write(outfile,' ':5,mosfets[i].name,' ');
    for j:=1 to 4 do
     write(outfile,node[mosfets[i].node[j]]:3);
    if mosfets[i].initial then init:=true;
    if (mosfets[i].mod_pos <> 0) then
     write(outfile,' ':5,mosmod[mosfets[i].mod_pos].name,' ')
    else write(outfile,' ':9);
    write(outfile,'  ');preal(mosfets[i].wc);
    write(outfile,'  ');preal(mosfets[i].lc);
    write(outfile,'  ');preal(mosfets[i].ad);
    write(outfile,'  ');preal(mosfets[i].as);
    writeln(outfile);pause(line);
    end;
   if init then
    begin
    writeln(outfile);pause(line);
    writeln(outfile,' ':20,'*MOSFET INITIAL CONDITION*');pause(line);
    writeln(outfile,' ':5,'NAME      Vgsi        Vdsi        Vsbi');pause(line);
    for j:= 1 to nummos do
     begin
     if mosfets[i].initial then
      begin
      write(outfile,' ':5,mosfets[i].name,'       ');
      write(outfile,'  ');preal(mosfets[i].vgsi);
      write(outfile,'  ');preal(mosfets[i].vdsi);
      write(outfile,'  ');preal(mosfets[i].vsbi);
      writeln(outfile);pause(line);
      end;
     end;
    end;
   end;
  writeln(outfile);pause(line);

  { print preset node values }
  init:=false;i:=1;
  while (i<=numnod) and not init do
   begin
    if (nodeval[i] <> 0) then init:=true;
    i:=i+1;
   end;
  if init then
   begin
   writeln(outfile,' ':20,'*PRESET TRANSIENT VOLTAGES*');pause(line);
   writeln(outfile,' ':5,'NODE    VALUE    NODE    VALUE    ',
                       ' NODE    VALUE    NODE    VALUE');pause(line);
   write(outfile,' ':4);j:=0;
   for i:=1 to numnod do
    if (nodeval[i] <> 0) then
     begin
     write(outfile,node[i]:4,' ':2);
     preal(nodeval[i]);write(outfile,' ');
     j:=j+1;
     if ((j div 4)*4=j) then begin
      writeln(outfile);pause(line);
      write(outfile,' ':4);end;
     end;
   end;
  writeln(outfile);pause(line);

  if (numsorm > 0) then
   begin
   writeln(outfile);pause(line);
   writeln(outfile,' ':20,'*SOURCE MODEL SPECIFICATIONS*');pause(line);
   for i:=1 to numsorm do
    begin
    writeln(outfile);pause(line);
    writeln(outfile,' ':5,'MODEL:',' ':3,sormod[i].name);pause(line);
    if (sormod[i].define <> 0) then
     begin
     if (sormod[i].define < 0) then
      begin
      write(outfile,'INITIAL VALUE >':33,' ');
      preal(sormod[i].step[2]);writeln(outfile);pause(line);
    write(outfile,'PULSED  VALUE >':33,' ');
    preal(sormod[i].step[6]);writeln(outfile);pause(line);
    write(outfile,'PULSE  DELAY  >':33,' ');
    preal(sormod[i].step[3]);writeln(outfile);pause(line);
    write(outfile,'PULSE  WIDTH  >':33,' ');
    preal(sormod[i].step[7]-sormod[i].step[5]);writeln(outfile);pause(line)
    write(outfile,'RISE   TIME   >':33,' ');
    preal(sormod[i].step[5]-sormod[i].step[3]);writeln(outfile);pause(line)
    write(outfile,'FALL   TIMW   >':33,' ');
    preal(sormod[i].step[9]-sormod[i].step[7]);writeln(outfile);pause(line)
    write(outfile,'PERIOD        >':33,' ');
    preal(sormod[i].step[20]/2.0);writeln(outfile);pause(line);
    end
   else
    begin
    writeln(outfile,' ':20,'  TIME      VALUE');pause(line);
    j:=1;init:=false;
    while (j<=6) and not init do
     if (sormod[i].step[2*j-1] <> 0.0) or (j=1) then
      begin
      write(outfile,' ':18);preal(sormod[i].step[2*j-1]);
      write(outfile,'  ');preal(sormod[i].step[2*j]);
      writeln(outfile);pause(line);
      j:=j+1;
      end
     else init:=true;
    end;
   end
  else begin
   writeln(outfile,' ':5,'         **MODEL UNDEFINED**');pause(line);
   end;
  end;
 end;
end;



{ MODEL ROUTINES }
procedure ListModel;

{ print all models in the circuit }
var i,j,k,line:integer;
begin
page(outfile);
line:=1;
writeln(outfile);pause(line);writeln(outfile);pause(line);
writeln(outfile,' ':23,'*DEVICE MODELS*');pause(line);
writeln(outfile);pause(line);writeln(outfile);pause(line);

for i:=1 to nummosm do              { print mosfet models }
 begin
 write(outfile,'MOS MODEL':28,'  "',mosmod[i].name,'"');
 if (mosmod[i].define <0) then writeln(outfile,'       P-CHANNEL')
  else writeln(outfile,'  N-CHANNEL');
 pause(line);
 for j:=1 to 24 do
```

```
    begin
    write(outfile,' ':5);
    for k:=1 to 3 do write(outfile,mos_para[3*(j-1)+k]);
    write(outfile,' > ');
    preal(mosmod[i].value[j]);
    if (j=(j div 3)*3) then begin
     writeln(outfile);pause(line);
     end; { if }
    end; { for }
   writeln(outfile);pause(line);
   end;
  writeln(outfile);pause(line);

  for i:=1 to numdiom do           { print diode models }
   begin
   writeln(outfile,'DIO MODEL  "':21,diomod[i].name,'"');pause(line);
   write(outfile,' IS > ':11);
   preal(diomod[i].satcur);writeln(outfile);pause(line);
   end;
  writeln(outfile);pause(line);

  for i:= 1 to numcapm do          { print capacitor models }
   begin
   writeln(outfile,'CAP MODEL  "':21,capmod[i].name,'"');pause(line);
   write(outfile,'C/A > ':11);  preal(capmod[i].value[1]);
   write(outfile,'Exp > ':11);  preal(capmod[i].value[2]);
   write(outfile,'Phi > ':11);  preal(capmod[i].value[3]);
   writeln(outfile);pause(line);
   end;
  writeln(outfile);pause(line);
  end;

  procedure mosmods(var mosmod:mosmod_array);

  { get mosfet model parameters }
  var done:boolean;
      p1,i,j,len:integer;
      p2:real;
      nam:string[3];
  begin
  gnam(linepos,inline,name);
  i:=1;  done:=false;
  while (i<=nummosm) and (mosmod[i].name <> name) do
   i:=i+1;
  if (i > maxmosm) then errr(16,inline)
  else
   begin
    if (i <= nummosm) and (mosmod[i].define <> 0) then
     done:=true;
    if (i > nummosm) then
     begin
      nummosm:=nummosm + 1;
      mosmod[i].name:=name;
     end;
    if not done then
     begin
     p1:=1;
     if (linepos <= leng) and (inline[linepos] = 'p') then
      p1:=-1;
     mosmod[i].define:=p1;
     for j:=1 to 26 do
      mosmod[i].value[j]:=0;
    end;
  end;
 done:=false;
 while not done do
  begin
  linepos:=1;
  write(outfile,'Parameter ');
  readln(inline);
  leng:=strlen(inline);
  if (leng > 3) then len:=3
  else len:=leng;
  nam:=str(inline,1,len);
  p1:=strpos(nam,mos_para);
  if (p1=0) then errr(9,inline)
  else
   if (p1=73) then begin
    mosu(mosmod);
    done:=true;
    end
   else begin
    linepos:=linepos + 3;
    skipblank(inline,linepos);
    getreal(p2,nonum);
    if (p2 <> 0.0) then
     begin
     mosmod[i].value[(p1 div 3) +1]:=p2;
     write(outfile,nam);  preal(p2);writeln(outfile);
     end;
    end;
  end;
 end;

procedure diomods(var diomod:diomod_array);

{ retrive and store diode model parameter }
var i:integer;
    p1:real;
begin
gnam(linepos,inline,name);
i:=1;
while (i<=numdiom) and (diomod[i].name <> name) do
 i:=i+1;
if i > maxdiom then errr(16,inline)
else
 begin
  if (i > numdiom) then
   begin
    numdiom:=numdiom+1;
    diomod[i].name:=name;
   end;
```

```
    getreal(p1,nonum);
    if nonum or (p1=0.0) then errr(7,inline)
    else
     diomod[i].satcur:=p1*sqr((t+273)/300)*exp(7.54e3*(1/300-1/(t+273)));
  end;
end;

procedure cmod(var capmod:capmod_array);

{ retrieve and store the capacitor model parameters }
var i,j:integer;
begin
gnam(linepos,inline,name);
i:=1;
while (i<=numcapm) and (capmod[i].name <> name) do
 i:=i+1;
if (i > maxcapm) then errr(16,inline)
else
 begin
  if (i > numcapm) then
   begin
    numcapm:=numcapm+1;
    capmod[i].name:=name;
   end;
  for j:=1 to 3 do
   getreal(capmod[i].value[j],nonum);
 end;
end;

{ OPTION EDITOR ROUTINES }

procedure liop;

{ print out all the options that are set and if not set, the default }
begin
 page(outfile);
 writeln(outfile);writeln(outfile);
 writeln(outfile,' ':30,'*ANALYSIS OPTIONS*');
 writeln(outfile);
 write(outfile,' ':31,'TSTEP > ');
 preal(tstep);writeln(outfile);
 writeln(outfile,' ':31 ,'MINIT > ',minit:4);
 writeln(outfile,' ':31,'MAXIT > ',maxit:4);
 write(outfile,' ':31,'CONVG > ');
 preal(conval);writeln(outfile);
 write(outfile,' ':31,'DV/DT > ');
 preal(dvdt);writeln(outfile);
 write(outfile,' ':31,'DI/DT > ');
 preal(didt);writeln(outfile);
 writeln(outfile);
 write(outfile,' ':31,'SHORTCH ');
 if short then writeln(outfile,' ON')
  else writeln(outfile,' OFF');
 write(outfile,' ':31,'CHANLMD ');
 if chem then writeln(outfile,' ON')
  else writeln(outfile,' OFF');
  write(outfile,' ':31,'SUBTCON ');
  if subt then writeln(outfile,' ON')
   else writeln(outfile,' OFF');
  write(outfile,' ':31,'SAT/LIN ');
  if satlin then writeln(outfile,' ON')
   else writeln(outfile,' OFF');
  write(outfile,' ':31,'CUT/LIN ');
  if cutlin then writeln(outfile,' ON')
   else writeln(outfile,' OFF');
  write(outfile,' ':31,'USE I C ');
  if ui then writeln(outfile,' ON')
   else writeln(outfile,' OFF');
  write(outfile,' ':31,'PRESET  ');
  if preset then writeln(outfile,' ON')
   else writeln(outfile,' OFF');
end;

{ PROCEDURE ROUTINES }

procedure killp(var prof:proc_string);

{ kill the procedure array }
var i:  integer;
begin
  for i:=1 to 40 do prof[i]:='';
end; {killp}

procedure listp(prof:proc_string);

{ list the procedure arrays }
var i,line:integer;
begin
 writeln(outfile);writeln(outfile);writeln(outfile);
 writeln(outfile,' ':7,'*PROCEDURE FILE*');writeln(outfile);
 writeln(outfile,' ':2,'LINE #       Command');
 line:=6;
 i:=1;
 while (i<=40) and (prof[i] <> 'done') do
  begin
   writeln(outfile,' ':4,i:2,' ':4,prof[i]);
   pause(line);
   i:=i+1;
  end;
end; {listp}

procedure writep(var prof:proc_string);

{ enter commands and datas in procedure array }
var i:integer;

begin
 i:=0;
 repeat
  i:=i+1;
  write('Command ',i:2,':  ');
  readln(prof[i]);
 until (prof[i] = 'done') or (i = 40);
end; {writep}

procedure insert(var prof:proc_string);

{ insert a line in the procedure array }
var i, j:integer;

begin
 skipchar(inline,linepos);
 getint(i,nonum);
 if nonum then
  writeln(outfile,'ERROR--NUMBER EXPECTED AFTER COMMAND')
 else
  begin
   if (prof[40] = 'done') then prof[39] := 'done';
   for j:=39 downto i do
    prof[j+1]:=prof[j];
   write(outfile,'Command ');
   readln(prof[i]);
  end;
end; {insert}

procedure delete(var prof:proc_string);

{ delete a line from the procedure array }
var i, j:integer;

begin
 skipchar(inline,linepos);
 getint(i,nonum);
 if nonum then
  writeln(outfile,'ERROR--NUMBER EXPECTED AFTER COMMAND')
 else
  for j:= i to 39 do
   prof[j]:=prof[j+1];
end;{delete}

procedure savep(var prof:proc_string);

{ save the procedure arrays in the file 'procf' }
var i:integer;
begin
 rewrite(buff,'procf');
 for i:=1 to 40 do
  writeln(buff,prof[i]);
 close(buff,'save');
end;

procedure getp(var prof:proc_string);

{ get the procedure arrays from the file 'procf' }
var i:integer;
begin
 reset(buff,'procf');
 for i:=1 to 40 do
```

```
   readln(buff,prof[i]);
  end;

  { Saving and Retrieving circuit description files }

  procedure getfile(var flist:boolean;var i:integer);

  { check the the file name is already in the name list in 'files' }
  var j:integer;

  begin {getfile}
   reset(buff,'files');
   readln(buff,numfile);
   for j:=1 to numfile do
    readln(buff,filename[j]);
   gnam(linepos,inline,name);
   i:=0;
   repeat
    i:=i + 1;
   until (name=filename[i]) or (i>=numfile);
   flist:=false;
   if (i<numfile) or (name=filename[i]) then
    flist:=true;
  end; {getfile}

  procedure CreateFile(var filename:name_array);

  { create the name for the circuit description file }
  var i:integer;
     flist:boolean;

  begin {create}
   skipchar(inline,linepos);
   if linepos < leng then
    begin
     getfile(flist,i);
     if not flist then
      begin

       { create file with name }
       numfile:=numfile+1;
       filename[numfile]:=name;

       { save the file list }
       rewrite(buff,'files');
       writeln(buff,numfile);
       for i:=1 to numfile do
        writeln(buff,filename[i]);
       close(buff,'save');
      end
     else
      errr(14,inline);
    end
   else
    errr(8,inline);
  end; { create }
```

```
procedure kill(var filename:name_array);

{ remove circuit description file from medium storage }
var i, j:integer;
  flist:boolean;

begin { kill }
 skipchar(inline,linepos);
 if linepos < leng then
  begin
   getfile(flist,i);
   if flist then
    begin

     { purge file }
     rewrite(buff,name);
     close(buff,'purge');

     { update file list }
     numfile:=numfile-1;
     for j:=i to numfile do
      filename[j]:=filename[j+1];

     { save file list }
     rewrite(buff,'files');
     writeln(buff,numfile);
     for j:=1 to numfile do
      writeln(buff,filename[j]);
     close(buff,'save');
    end
  end
 else
  errr(8,inline);
end; { kill }

procedure load(var filename:name_array);

{ load the circuit description from file that already saved }
var flist:boolean;
      i, j:integer;

begin {load}
 skipchar(inline,linepos);
 if linepos < leng then
  begin
  getfile(flist,i);
  if flist then
   begin
    reset(buff,name);
    readln(buff,numres,numcap,nummos,numdio,numcur,numvol);
    readln(buff,nummosm,numsorm,numcapm,numdiom,numnod);
    for i:=1 to numvol do
     with volsor[i] do
      readln(buff,value,name,nodep,noden,mod_pos);
    for i:=1 to numcur do
```

```
      with cursor[i] do
       readln(buff,value,name,nodep,noden,mod_pos);
     for i:=1 to numres do
      with resr[i] do
       readln(buff,name,n1,n2,value,tc1,tc2);
     for i:=1 to numcap do
      with capr[i] do
       readln(buff,name,n1,n2,mod_pos,value,initval);
     for i:=1 to numcapm do
      with capmod[i] do
       readln(buff,name,value[1],value[2],value[3]);
     for i:=1 to numdio do
      with diod[i] do begin
       readln(buff,name,nca,nan,mod_pos);
       readln(buff,scale,sma_val[1],sma_val[2],sma_val[3]);
     end;
     for i:=1 to numdiom do
      readln(buff,diomod[i].name,diomod[i].satcur);
     for i:=1 to nummos do
      with mosfets[i] do begin
       readln(buff,name,node[1],node[2],node[3],node[4]);
       readln(buff,mod_pos,satlin,cutlin,initial);
       readln(buff,wc,lc,ad,as,vgsi,vdsi,vsbi);
       for j:=1 to 13 do
        readln(buff,sma_val[j]);
      end;
     for i:=1 to nummosm do
      with mosmod[i] do begin
      readln(buff,name,define);
      for j:=1 to 26 do
       readln(buff,value[j]);
     end;
     for i:=1 to numsorm do
      with sormod[i] do begin
       readln(buff,name,define);
       for j:=1 to 22 do
       readln(buff,step[j]);
     end;
     for i:=1 to 12 do
      readln(buff,gout[i]);
     for i:=0 to numnod do
      readln(buff,node[i]);
     for i:=1 to numnod do
      readln(buff,nodeval[i]);
     close(buff);
    end
   else
    errr(12,inline)
   end
 else
  errr(8,inline);
 end; { load }

 procedure CatFile(filename:name_array);

 { list the names of all circuit description file on line }
```

```
var i:integer;

begin {catf}
 reset(buff,'files');
 readln(buff,numfile);
 for i:=1 to numfile do
  readln(buff,filename[i]);
 page(outfile);
 writeln(outfile);writeln(outfile);
 writeln(outfile,'*FILE NAMES*':30);
 for i:=1 to numfile do
  writeln(outfile,' ':16,i:2,' ':4,filename[i]);
 writeln(outfile);
end;  {catf}

procedure save(filename:name_array);

{ save the current circuit description on disc }
var flist:boolean;
    i,j:integer;

begin { save }
skipchar(inline,linepos);
if linepos < leng then
 begin
  getfile(flist,i);
  if flist then
   begin
    rewrite(buff,name);
    writeln(buff,numres:3,numcap:3,nummos:3,numdio:3,numcur:3,numvol:3);
    writeln(buff,nummosm:3,numsorm:3,numcapm:3,numdiom:3,numnod:3);
    for i:=1 to numvol do
     with volsor[i] do
      writeln(buff,value,name,nodep:3,noden:3,mod_pos:3);
    for i:=1 to numcur do
     with cursor[i] do
      writeln(buff,value,name,nodep:3,noden:3,mod_pos:3);
    for i:=1 to numres do
     with resr[i] do
      writeln(buff,name,n1:3,n2:3,' ',value,' ',tc1,' ',tc2);
    for i:=1 to numcap do
     with capr[i] do
      writeln(buff,name,n1:3,n2:3,mod_pos:3,' ',value,' ',initval);
    for i:=1 to numcapm do
     with capmod[i] do
      writeln(buff,name,value[1],' ',value[2],' ',value[3]);
    for i:=1 to numdio do
     with diod[i] do begin
      writeln(buff,name,nca:3,nan:3,mod_pos:3);
      writeln(buff,scale:3,' ',sma_val[1],' ',sma_val[2],' ',sma_val[3]);
     end;
    for i:=1 to numdiom do
     writeln(buff,diomod[i].name,diomod[i].satcur);
    for i:=1 to nummos do
     with mosfets[i] do begin
      writeln(buff,name,node[1]:3,node[2]:3,node[3]:3,node[4]:3);
      writeln(buff,mod_pos:3,satlin:3,cutlin:3,initial);
      writeln(buff,wc,' ',lc,' ',ad,' ',as,' ',vgsi,' ',vdsi,' ',vsbi);
      for j:=1 to 13 do
       writeln(buff,sma_val[j]);
     end;
    for i:=1 to nummosm do
     with mosmod[i] do begin
      writeln(buff,name,define:3);
      for j:=1 to 26 do
       writeln(buff,value[j]);
     end;
    for i:=1 to numsorm do
     with sormod[i] do begin
      writeln(buff,name,define:3);
      for j:=1 to 22 do
       writeln(buff,step[j]);
     end;
    for i:=1 to 12 do
     writeln(buff,gout[i]);
    for i:=0 to numnod do
     writeln(buff,node[i]);
    for i:=1 to numnod do
     writeln(buff,nodeval[i]);
    close(buff,'save');
   end
  else
   errr(12,inline);
  end
 else
  errr(8,inline);
end;  {save}

procedure solv;
var i:integer;
 temp:array[1..maxequ] of integer;

begin
if (iter1 = 0) and (iter2 = 0) then
 begin
  for i:=1 to totnod do begin
   if (row[i].ptr^.j = 0) then
    row[i].ptr:=row[i].ptr^.nextI;
   if (col[i].ptr^.i = 0) then
    col[i].ptr:=col[i].ptr^.nextJ;
  end;
  CheckDiaZero(row,col);
  reorder(row,col);
  for i:=1 to totnod do
   temp[i]:=row[i].oldindex;
  for i:=1 to totnod do
   row[temp[i]].oldindex:=i;
 end;
lufactorize(col);
solve(row,col,basr,tasr);
matzv(totnod,basr);
end;
```

```
procedure conv(var tasr,pasr:val_array);
var j:integer;
    i:real;
begin
i:=0.0;
for j:=1 to totnod do
 begin
 if i<abs(tasr[j]-pasr[j]) then
  i:=abs(tasr[j]-pasr[j]);
 end;
if i<conval then cong:=true;
end;

procedure outa;
var i:integer;
begin
if gout[1]<>0 then
 begin
 write(outfile,' ':2);
 preal(tstart1);
 end;
i:=1;
while (i<=6) and (gout[i]<>0) do
 begin
 if (gout[i] <= totnod) then
  begin
  write(outfile,' ':2);
  preal(tasr[gout[i] ]);
  end
 else begin
  write(outfile,' ':2);
  {preal(mosfets[gout[i]-totnod]XXXXXXXX);}
  end;
 i:=i+1;
 end;
writeln(outfile);
end;

procedure work(var trans:boolean);
label 10;
var i:integer;
begin
if not trans then
 begin
 totnod := numnod + numvol;
 iter1 := 0; iter2 := 0;
 cong := false;
 while not cong or (iter1 <= 2) do
  begin
  rstr(resr);
  vsrc(volsor);
  isrc(cursor);
  diods(diod,diomod);
  mosf(mosfets,mosmod);
  mate(pasr,tasr,totnod);
```

```
    solv;
    cong := false;
    conv(tasr,pasr);
    if not cong or (iter1 <= 2) then
     begin
     iter1 := iter1 + 1;
     iter2 := iter2 + 1;
     end
    end;
   writeln(outfile,' ':10,iter2:4 ,'  iterations');writeln(outfile);
   action:=false;
   end
  else
   begin
   if (tstart1 = 0) then
    begin
    iter2 := 0;
    tstart1 := tstart;
    end;
   tstep1 := tstep;
   iter1 := 0;
   cong := false;
  10: while (not cong or (iter1 <= 1)) and (iter1 <=maxit) do
    begin
    rstr(resr);
    vsrc(volsor);
    isrc(cursor);
    caprs(capr,capmod);
    diods(diod,diomod);
    mosf(mosfets,mosmod);
    mate(pasr,tasr,totnod);
    solv;
    cong := false;
    conv(tasr,pasr);
    if (not cong or (iter1 <= 1)) and (iter1 <= maxit) then
     iter1 := iter1 + 1;
    end;
   preset := false;
   ui := false;
   if (iter1 > maxit) then
    begin
    if (tstep/80.0 > tstep1/8.0) then
     tstep1 := tstep/80.0
    else tstep1:=tstep1/8.0;
    iter2 := iter1 + iter2;
    iter1 := 0;
    if (iter2 < 1500) then
     goto 10;
    writeln(outfile,'THAT IS ALL SHE WROTE');
    donemain:=true;
    end
   else
    begin
    tstart1 := tstart1 + tstep1;
    tstep2 := tstep1;
    mate(nasr,qasr,totnod);
  mate(qasr,pasr,totnod);
  if (iter1 < minit) and (tstep < 2*tstep1) then
   tstep1 := tstep;
  if (iter1 < minit) and (2*tstep1 < tstep) then
   tstep1 := 2*tstep1;
  iter2 := iter2 + iter1;
  if cutlin or satlin then
   chkf;
  outa;
  if (tstart1 <= tstop) then
   begin
   iter1 := 0;
   goto 10;
   end;
  writeln(outfile,' ':10,iter2:4,'  iterations');writeln(outfile);
  action:=false;
  end;
 end;
end;

procedure CircuitEditor(var inline:line_string;var linepos:integer);

{ Enterin the Circuit Editor }
var done: boolean;
begin
 done := false;
 while not done do
 begin
  if exe_proc then prod(prof,profpos)
  else begin
   writeln(outfile,'CIRCUIT');
   readln(inline);
   leng:=strlen(inline);
   end;
 linepos := 1;
 inline:=strltrim(inline);                    { strip  the leading blank }
 if (leng >= 1) then
  case inline[1] of
  'r','R': resrc(resr);
  'c','C': caprc(capr);
  'm','M': mosfc(mosfets);
  'd','D': diodc(diod);
  'v','V': voltc(volsor);
  'i','I': curnc(cursor);
  's','S': smodc(sormod);
  'e','E': begin
           veryc;
           done:=true;
           action:=false;
           end;
  'l','L': ListCirc;
  'n','N': newcc;
  'k','K': kilec;
  'p','P': presc(nodeval)
  otherwise
   errr(6,inline);
  end;
 end
end;

procedure ModelEditor(var inline:line_string;var linepos:integer);

{ Entering model editor }
var done : boolean;
begin
 done := false;
 while not done do
  begin
  if exe_proc then prod(prof,profpos)
  else begin
   writeln(outfile,'MODELS');
   readln(inline);
   leng:=strlen(inline);
   end;
  linepos := 1;
  inline:=strltrim(inline);
  if (leng >= 1) then
   case inline[1] of
   'm','M':  mosmods(mosmod);
   'd','D':  diomods(diomod);
   'c','C':  cmod(capmod);
   'l','L':  ListModel;
   'e','E':  begin
              done := true;
              action:=false;
             end;
   'n','N':  begin
              nummosm := 0;
              numcapm := 0;
              numdiom := 0;
             end;
   otherwise
    errr(13,inline);
   end; { case }
  end;
end;

procedure OptionEditor(var inline:line_string;var linepos:integer);

{ Entering the Option Editor }
var   done: boolean;
         i: integer;
   options: line_string;
begin
 done := false;
 options:='tsTSmiMImaMAcoCOchCHcuCUdvDVdiDI'
        + 'shSHsuSUsaSAuiUIliLIenENprPRmoMO';

 while not done do
  begin
  if exe_proc then prod(prof,profpos)
  else begin
```

```
   writeln(outfile,'OPTIONS');
   readln(inline);
   leng:=strlen(inline);
  end;
 linepos := 1;
 inline:=strltrim(inline);
 if (1 <= leng) then
  begin
   name:=str(inline,1,2);
   i:=strpos(name,options);
   i:=(i + 3) div 4;
   case i of
   1:  begin
        skipchar(inline,linepos);
        getreal(tstep,nonum);
       end;
   2:  begin
        skipchar(inline,linepos);
        getint(minit,nonum);
       end;
   3:  begin
        skipchar(inline,linepos);
        getint(maxit,nonum);
       end;
   4:  begin
        skipchar(inline,linepos);
        getreal(conval,nonum);
       end;
   5:  if not chem then chem:=true
       else chem:=false;
   6:  if not cutlin then cutlin:=true
       else cutlin:=false;
   7:  begin
        skipchar(inline,linepos);
        getreal(dvdt,nonum);
       end;
   8:  begin
        skipchar(inline,linepos);
        getreal(didt,nonum);
       end;
   9:  if not short then short:=true
       else short:=false;
   10: if not subt then subt:=true
       else subt:=false;
   11: if not satlin then satlin:=true
       else satlin :=false;
   12: if not ui then ui:=true
       else ui:=false;
   13: liop;
   14: begin
        done := true;
        action:=false;
       end;
   15: if not preset then preset:=true
       else preset:=false;
   16: begin
```

```
        skipchar(inline,linepos);
        getint(level,nonum);
        end;
     otherwise
      errr(10,inline);
     end; { end case }
    end; { end if }
   end; { end while }
 end; { option }

 procedure ProcEditor(var prof:proc_string);

 { Entering the procedure editor }
 var done: boolean;

 begin {proc}
  done:=false;
  while not done do
   begin
    writeln(outfile,'PROCEDURE');
    linepos:=1;
    readln(inline);
    leng:=strlen(inline);
    inline:=strltrim(inline);                         { skip over the leading blan
    if (1 <= leng) then
    case inline[1] of
    'i','I':  insert(prof);
    'd','D':  delete(prof);
    'l','L':  listp(prof);
    'k','K':  killp(prof);
    'w','W':  writep(prof);
    'u','U':  exe_proc:=true;
    'e','E':  begin
               action:=false;
               done:=true;
              end;
    's','S':  savep(prof);
    'g','G':  getp(prof);
    otherwise
     errr(1,inline);
    end;
   end;
 end; {proc}

 procedure prdc(var node:node_array);

 { print the dc analysis operating point results }
 { it first print the voltages of all the nodes, then the mosfet current
   from drain to source and finally, the voltage source current }

 var i:integer;
 begin
 page(outfile);writeln(outfile);writeln(outfile);
 writeln(outfile,' ':18,'*D.C. OPERATION POINT*');
 writeln(outfile);writeln(outfile);
 writeln(outfile,' ':20,'NODE VOLTAGES');
```

```
writeln(outfile);
for i:=1 to numnod do
 begin
 write(outfile,'  (',node[i]:3,') >  ');
 preal(tasr[i]);
 if (i=(i div 4)*4) then writeln(outfile);
 end;
writeln(outfile);
if (nummos > 0) then
 begin
 writeln(outfile);writeln(outfile);
 writeln(outfile,' ':20,'MOSFET CURRENTS');
 writeln(outfile);
 for i:=1 to nummos do
  begin
  write(outfile,'  ',mosfets[i].name,' ');
  preal(mosfets[i].sma_val[3]);
  if (i = (i div 4) * 4) then writeln(outfile);
  end;
 writeln(outfile);
 end;
if (numvol > 0) then
 begin
 writeln(outfile); writeln(outfile);
 writeln(outfile,' ':20,'VOLTAGE SOURCE CURRENT');
 writeln(outfile);
 for i:=1 to numvol do
  begin
  write(outfile,' ':2,volsor[i].name,' ');
  preal(tasr[numnod+i]);
  if (i = (i div 4)*4) then writeln(outfile);
  end;
 writeln(outfile);writeln(outfile);writeln(outfile);
 end;
end;

procedure prtran(var node:node_array);
var i,p1,p5:integer;
    find,done:boolean;
begin
for i:=1 to 6 do
 gout[i]:=0;
p5:=1;done:=false;skipchar(inline,linepos);
while (p5<=6) and not done and (linepos<=leng) do
 begin
 skipblank(inline,linepos);
 case inline[linepos] of
 'v': begin                     { get the node number }
      linepos:=linepos+2;
      getint(p1,nonum) ;
      if nonum then begin
       errr(6,inline);done:=true;
       end
      else begin
       i:=0;
       repeat
```

```
         i:=i+1;
         until (node[i]=p1) or (i=numnod);
         if (i<=numnod) then
          begin
          gout[p5]:=i;
          p5:=p5+1;
          linepos:=linepos+2;
          end
         else begin
          errr(6,inline);done:=true;
          end;
         end;
        end;
   'i': begin              { retrive name the current to be printed }
        i:=linepos+1;
        linepos:=linepos+2;
        repeat
         i:=i+1;
        until (i>leng) or (inline[i]=')') or (i=(linepos+5));
        if (i <= leng) and (inline[i] = ')') then
         begin
         gnam(linepos,inline,name);
         if (name[1] = 'v') then    { retrive voltage name }
          begin
          i:=1;
          while (i <=numvol) and (volsor[i].name<>name) do
           i:=i+1;
          if (volsor[i].name = name) then
           begin
           gout[p5]:=i+numnod;
           p5:=p5+1;
           skipchar(inline,linepos);
           end
          else
           begin
           errr(6,inline);done:=true;
           end;
          end
         else                        { mosfet current }
          begin
          i:=1;
          while (i<=nummos) and (mosfets[i].name <> name) do
           i:=i+1;
          if mosfets[i].name = name then
           begin
           gout[p5]:=i+numnod+nummos;
           p5:=p5+1;
           skipchar(inline,linepos);
           end
          else
           begin
           errr(6,inline);done:=true;
           end;
          end;
         end
        else
        begin
        errr(6,inline);  done:=true;
        end;
       end;
 otherwise
       done:=true;
 end;
 end;
end;

procedure print(var inline:line_string;var linepos:integer);

{ decode the print statement to if to print dc or to print transient }
var opt:line_string;
begin
skipchar(inline,linepos);
if (linepos <= leng) then
 begin
  opt:='dcDCtrTR';
  name:=str(inline,linepos,2);
  i:=strpos(name,opt);
  i:= (i + 3) div 4;
  case i of
  1:  if (totnod<>0) then prdc(node);
  2:  prtran(node);
  otherwise
   errr(3,inline);
  end;
 end;
end;

procedure newc;

{ clear the old circuit elements }
var i:integer;

begin
  totnod:=0;numcap:=0;numdio:=0;numdiom:=0;
  nummosm:=0;numcapm:=0;numnod:=0;numres:=0;
  numsorm:=0;numvol:=0;numcur:=0;nummos:=0;
  matzv(maxequ,tasr);
  matzv(maxequ,pasr);
  matzv(maxequ,qasr);
  matzv(maxequ,nasr);
  matzv(maxequ,nodeval);
  for i:=1 to maxequ do
   node[i]:=0;
end;

procedure DCInitial;

{ initialize variables for dc operating point calculation }
var i,j:integer;
begin
trans:=false;tstart1:=0.0;tstep1:=0.0;
tstop:=0.0;tstart:=0.0;tstep:=0.0;tstep2:=0.0;
```

```
matzv(maxequ,pasr);
matzv(maxequ,tasr);
matzv(maxequ,qasr);
matzv(maxequ,nasr);
for i:=1 to maxmos do
 for j:=1 to 13 do
  mosfets[i].sma_val[j]:=0.0;
for i:=1 to maxcap do
 for j:=1 to 2 do
  capr[i].sma_val[j]:=0.0;
for i:=1 to maxdio do
 for j:=1 to 3 do
  diod[i].sma_val[j]:=0.0;
end;

procedure tran(inline:line_string;var linepos:integer);

{ input the transient time and the step }
begin
skipchar(inline,linepos);
getreal(tstop,nonum);            { get stop time }
if nonum then errr(2,inline)
else
 begin
 getreal(tstep,nonum);
 if nonum and (tstep=0.0) then  {if not given time step,set to default }
  tstep:=tstop/50;
 if not nonum then
  begin
  getreal(tstart,nonum);         { get start time,optional }
  if nonum then tstart:=0.0;     { default is 0)
  if (tstep=0.0) then
   tstep:=(tstop-tstart)/50;

  { set initial voltage to preset voltage if the preset option is set }
  if preset then mate(tasr,nodeval,maxequ);
  trans:=true;
  end;
 end;
end;

procedure temperature(inline:line_string;var linepos:integer);

{ get the temperature under which the simulation is done
  and recalculate all the parameter that affetct by the temperature }

var p1:real;
begin
skipchar(inline,linepos);
getreal(T,nonum);
if nonum then T:=27.0;
write(outfile,'TEMP:  ');        { print out the present temperature }
preal(T);writeln(outfile);
p1:=T+273.0;
eg:=1.16-7.02e-4*p1*p1/(p1+1108);
vt2:=vt1*p1/300;
```

```
ni2:=ni1*sqrt(sqr(p1/300)*p1/300)*exp(-0.5*eg*qe/1.38e-23*(1/p1-1/300));
mosu(mosmod);              { recalculate the mosfet parameters }
end;

procedure runc;

{ start either dc or transient analysis depending on which is set in
  the previous commands. it will initialize all necessary variables
  and matrixes before the actual analysis }
var i:integer;
 done:boolean;
begin
totnod:=numnod+numvol;     { calculate the size of the matrix }
matza(totnod,ansr);        { initialize the matrix }
matzv(totnod,basr);        { initialize the right side of the matrix }
action:=true;              { signal to start the analysis }
if not trans then          { start time is 0 for dc analysis }
 tstart1:=0.0
else                       { initialize for transient analysis }
 begin
 if not preset then mate(tasr,pasr,totnod);
 mate(qasr,tasr,totnod);
 mate(nasr,qasr,totnod);
 iter1:=0;
 tstart1:=0.0;iter1:=0;

 { print the heading for transient analysis }
 page(outfile);writeln(outfile);writeln(outfile);
 writeln(outfile,'*TRANSIENT ANALYSIS*':38);
 writeln(outfile);writeln(outfile);
 write(outfile,'  TIME(SEC)   ');
 i:=1;done:=false;
 while (i<=6) and not done do
  if (gout[i] <> 0.0) then
   begin
   if (gout[i]<=numnod) then
    write(outfile,'  V(',node[gout[ i]]:2,')     ');
   if (gout[i]>numnod) and (gout[i]<=totnod) then
    write(outfile,'  I(',volsor[gout[i]-numnod].name,')    ');
   if (gout[i] > totnod) then
    write(outfile,'  I(',mosfets[gout[i]-numnod].name,')          ');
   i:=i+1;
   end
  else done:=true;
 writeln(outfile);
 end;
end;

{ main program }

begin
 initialize;
 donemain:=false;
 blank:=['=',' ','('];
 while not donemain do
  begin
```

```
   if action then work(trans);
   command:='ciCIcrCRcaCAmoMOopOPouOUpcPCprPRneNEdcDCtrTR'+
            'teTEkiKIloLOgoGOsaSAenEN';
   if not donemain then
    begin
    linepos := 1;
    if exe_proc then prod(prof,profpos)
    else
     begin
      writeln('BIASP.45');
      readln(inline);
      leng:=strlen(inline);
     end;
    inline:=strltrim(inline);
    if linepos < leng then
    begin
    name:=str(inline,1,2);
    i:=strpos(name,command);
    i:=(i + 3) div 4;
    case i of
    1:  CircuitEditor(inline,linepos);   {enter Circuit Editor}
    2:  CreateFile(filename);            {create data file and file name}
    3:  CatFile(filename);               {list data file on line}
    4:  ModelEditor(inline,linepos);     {enter Model Editor}
    5:  OptionEditor(inline,linepos);    {enter Option Editor}
    6:  PrintSelect(inline,linepos);     {select output device}
    7:  ProcEditor(prof);                {enter Procedure Editor}
    8:  print(inline,linepos);           {print either DC or TRAN}
    9:  newc;
    10:  DCInitial;
    11:  tran(inline,linepos);
    12:  temperature(inline,linepos);
    13:  kill(filename);
    14:  load(filename);
    15:  runc;
    16:  save(filename);
    17:  donemain := true;
    otherwise
     errr(1,inline);
    end;
    end
    else
     writeln(outfile,'ERROR: COMMAND MUST BE AT LEAST TWO CHARACTERS LONG');
    end;
   end;
   close(outfile,'save');
 end. {main}
```

REFERENCES:

[1] D. O. Pederson, private communication.

[2] Brian L. Biehl, "Interactive Electronic Circuit Simulation on Small Computer Systems," U.S. Army Electronics Research and Development Command Harry Diamond Laboratories, Report HDL-TM-79-30, November 1979.

[3] G. Taylor and A. R. Newton, "BIASL report" ,1976 Asilomar Conference.

[4] R. S. Gyurcsik, "BIASB.45: An I.C. Simulator For Desktop Computers," University of California, Berkeley, Master Report, November 1981.

[5] W. McCalla, "Computer-Aided Circuit Simulation Techniques",Pre-Publication Manuscript,1979

[6] A. R. Newton, private communication.

[7] H. Shichman and D. A.Hodges, "Modeling and Simulation of Insulated-Gate Field-Effect Transistor Switching Circuits", IEEE J Solid-State Circuits, pp. 285-289.

[8] A. Vladimirescu and S. Liu, "The Simulation Of MOS Intregrated Circuits Using SPICE2", Electronics Research Laboratory, university of California, Berkekey, feb. 1980.

[9] R. S. Muller and T. I. Kamins, Device Electronics for Integrated Circuits, New York: Wiley, 1977.