เอกสารอ้างอิง

1. เอกชัย ลีลารัศมี. "โครงสร้างของโปรแกรมและข้อมูลในเล็ก." เอกสารประกอบการประชุมวิชาการวิศวกรรมไฟฟ้า 8 สถาบันครั้งที่ 8 หน้า 3.22-3.29 ธันวาคม 2528.
2. Ho, C.W. et al. "The Modified Nodal Approach to Network Analysis." IEEE Trans. on Circuits and Systems, Vol.CAS-22, pp.504-509, June 1975.
3. Chua, L.O. and P.M. Lin "Computer-Aided Analysis of Electronic Circuits: Algorithms and Computational Techniques. "Prentice-Hall Inc., 1975.
4. Vlach, J.R. "Computer Methods for Circuit Analysis and Design." Van-Nostrand Reinhold Electrical/Computer Science and Engineering Series, 1983.
5. Nagel, L.W. "SPICE 2: A Computer Program to Simulate Semiconductor Circuit." University of California, Electronics Research Laboratory. Memorandum ERU-M 520, May 9, 1975.
6. International Business Machines Corp."Advanced Statistical Analysis Program (ASTAP)." Program Reference Manaul, 1973.
7. Thornton, R.D. et al."Characteristics and Limitations of Transistors." John Wiley & Sons, Inc., 1966.
8. Ousterhout, J.K. "CRYSTAL: A timing analyzer for nMOS VLSI Circuits," in 3 rd Cal. Tech. Conf. on VLSI, pp. 57-70, 1983
9. Shanbeck, L.K. and Norin, R.S. "QSPICE: An application of array processor to CAD simulation of IC circuits," in Proc. IC-CAD Conf., Sept. 1983.

ภาคผนวก ก.

คุณสมบัติของโปรแกรมวิเคราะห์วงจรสำเร็จรูป SPEC

SPEC ได้ถูกออกแบบและพัฒนาขึ้นมาครั้งแรก เพื่อให้ใช้ได้กับเครื่องมินิคอมพิวเตอร์ PDP - 11 ซึ่งได้ติดตั้งใช้งานที่ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ดังนั้น คุณสมบัติของโปรแกรมบางส่วน จึงถูก กำหนดจากขีดความสามารถของเครื่อง PDP - 11 อย่างไรก็ตาม เมื่อดัดแปลงโปรแกรมให้ใช้งานกับเครื่องคอมพิวเตอร์ที่มีขีดความสามารถสูงมากขึ้นแล้ว ก็สามารถเพิ่มเติมขีดความสามารถของโปรแกรมให้สูงมากขึ้นอีกได้ เช่น โปรแกรม SPEC ที่ถูกดัด แปลงให้ใช้งานบนเครื่อง VAX - 11 ที่ติดตั้งใช้งาน ณ สำนักบริการคอมพิวเตอร์ มหาวิทยาลัยเชียงใหม่ เป็นโปรแกรมที่มีขีดความสามารถสูงขึ้น

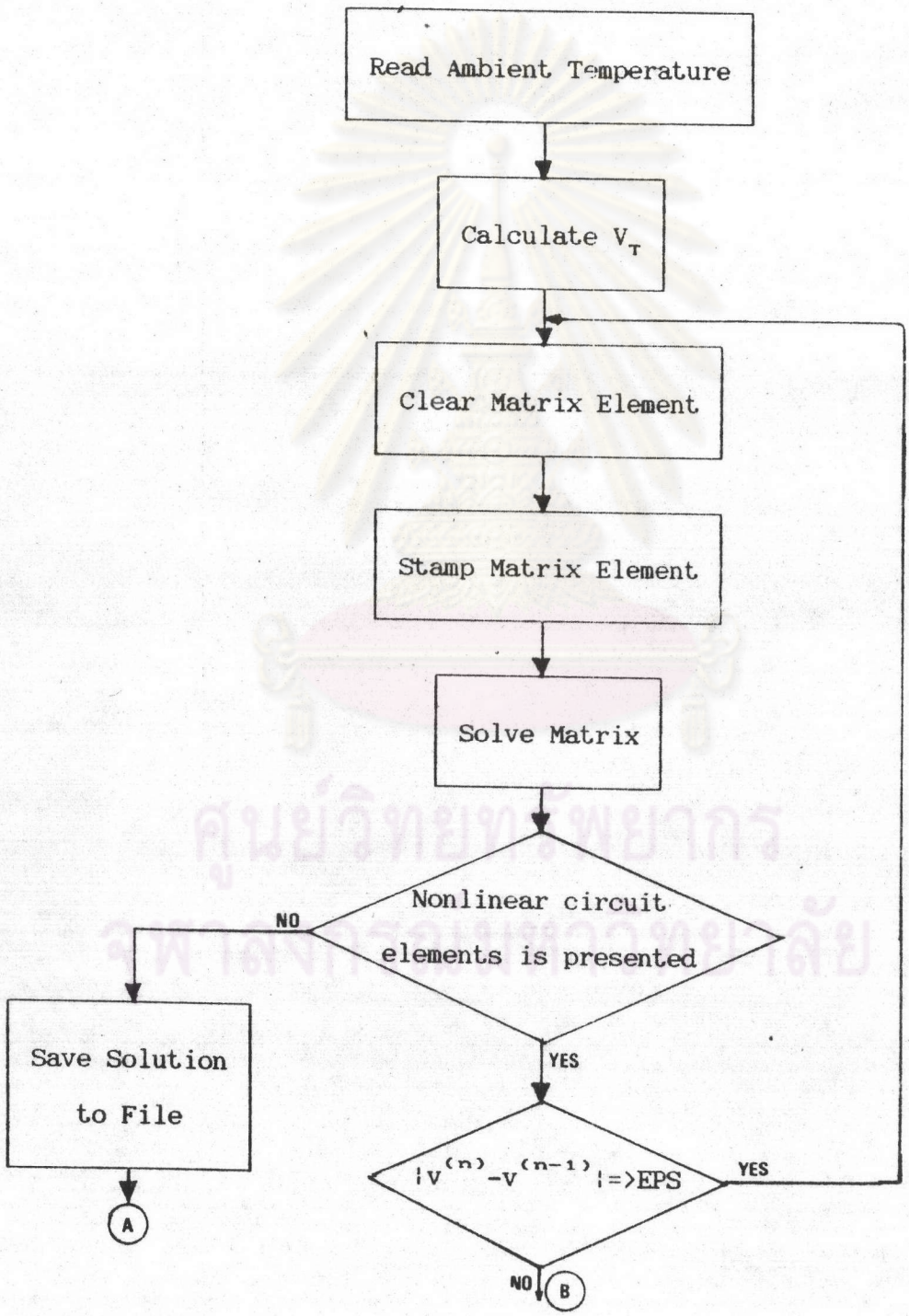
คุณสมบัติของโปรแกรม SPEC ที่ติดตั้งบนเครื่อง PDP - 11

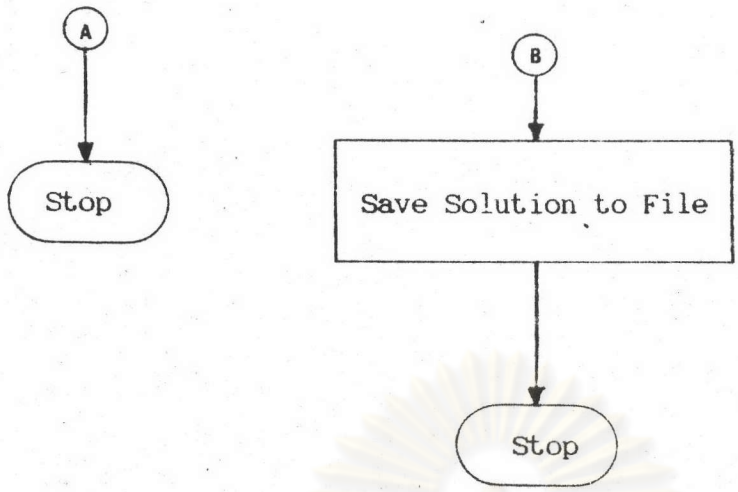
1. โปรแกรม SPEC เขียนขึ้นด้วยภาษา FORTRAN 77 และถูกคอมไพล์โดยคอมพายเลอร์ FORTRAN 77
2. การทำงานของส่วนโปรแกรมต่าง ๆ จะทำงานกันแบบ ใช้เนื้อที่ในหน่วยความจำร่วมกัน (Overlaid Program)
3. มีความสามารถในการคำนวณสมการเมตริกซ์แบบ Full matrix ได้ถึงขนาด 40 x 40 ลำดับ (order)
4. สามารถคำนวณวงจรไฟฟ้าได้โดยมีอุปกรณ์ในวงจรไม่เกิน 60 ตัว

ภาคผนวก ข.

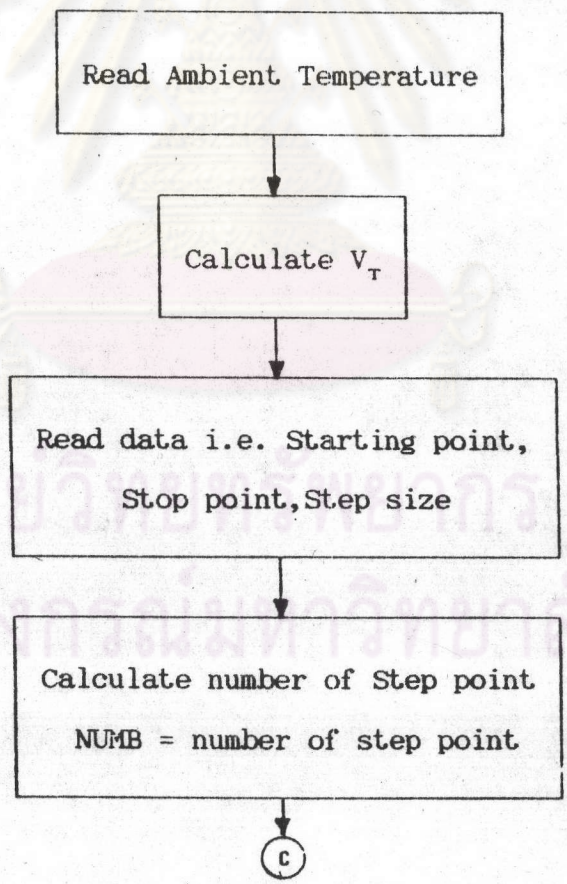
อัลกอริทึมทางคณิตศาสตร์ที่ใช้ในการวิเคราะห์

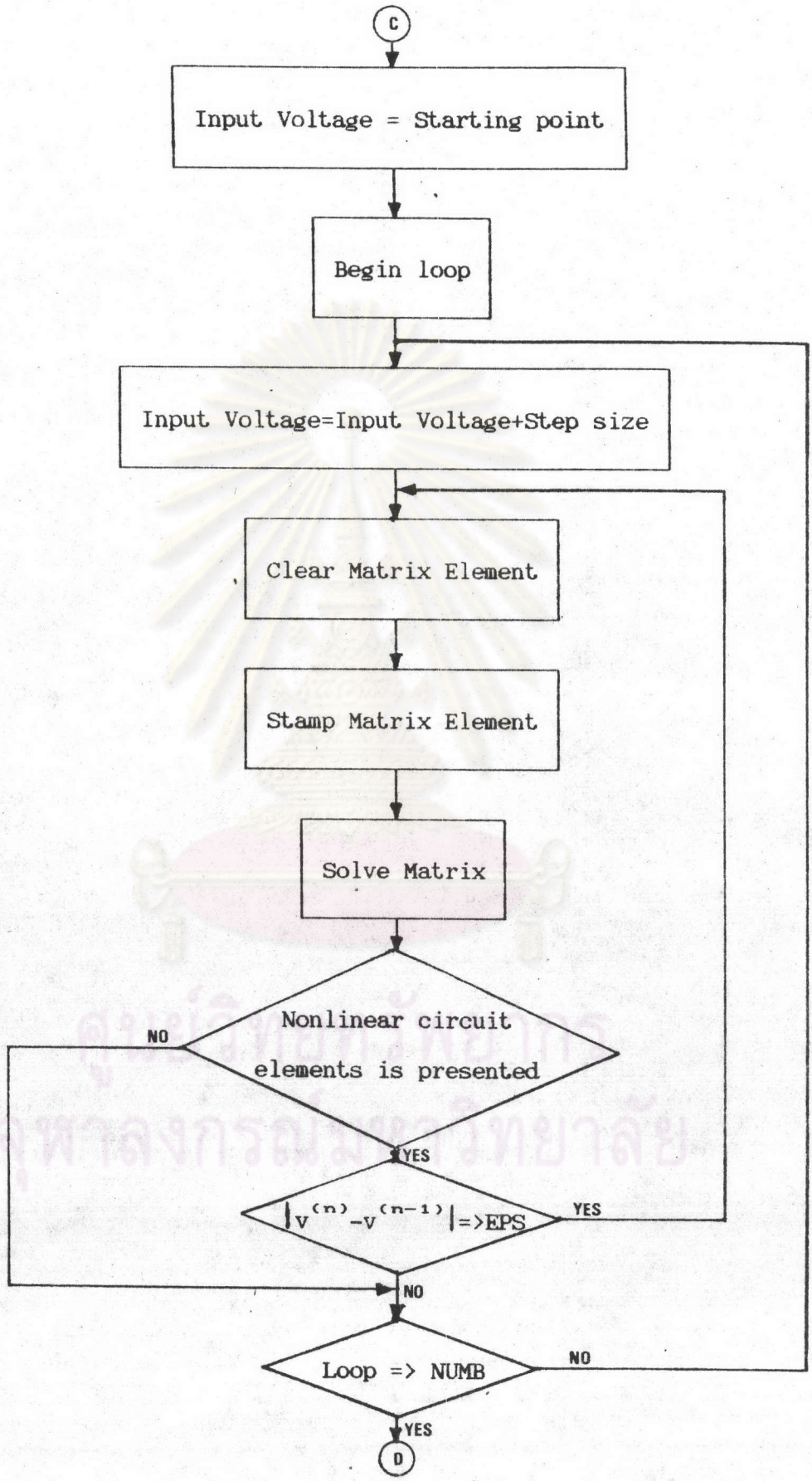
1. การวิเคราะห์จุดทำงานสงบ (DC. Operating Point)

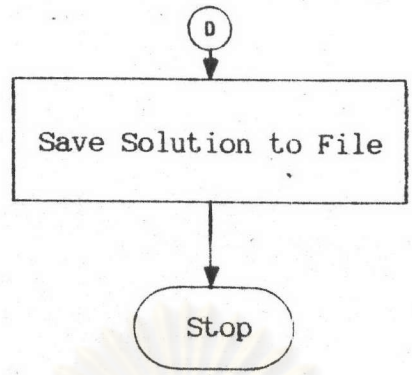




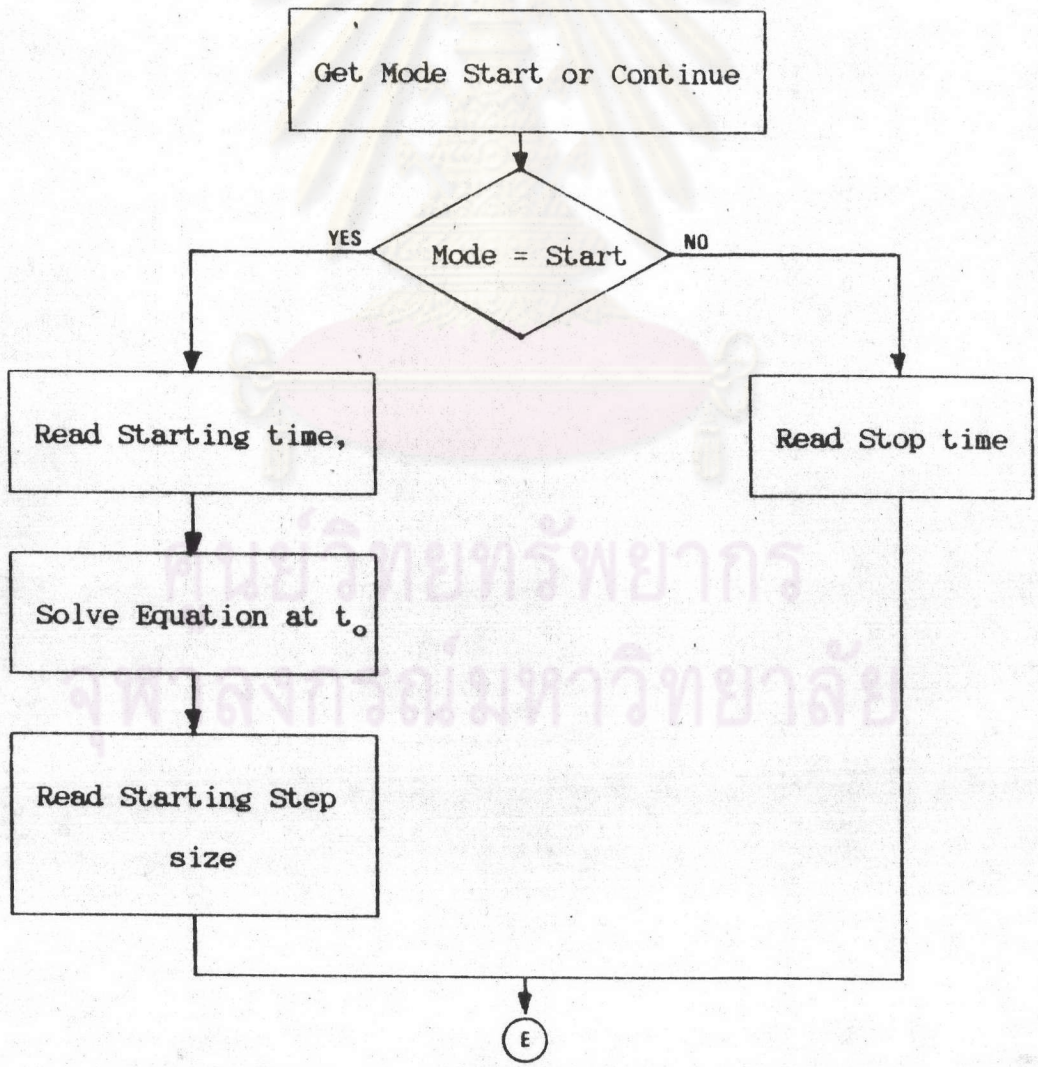
2. การวิเคราะห์คุณสมบัติโอนย้าย (Transfer Characteristics)

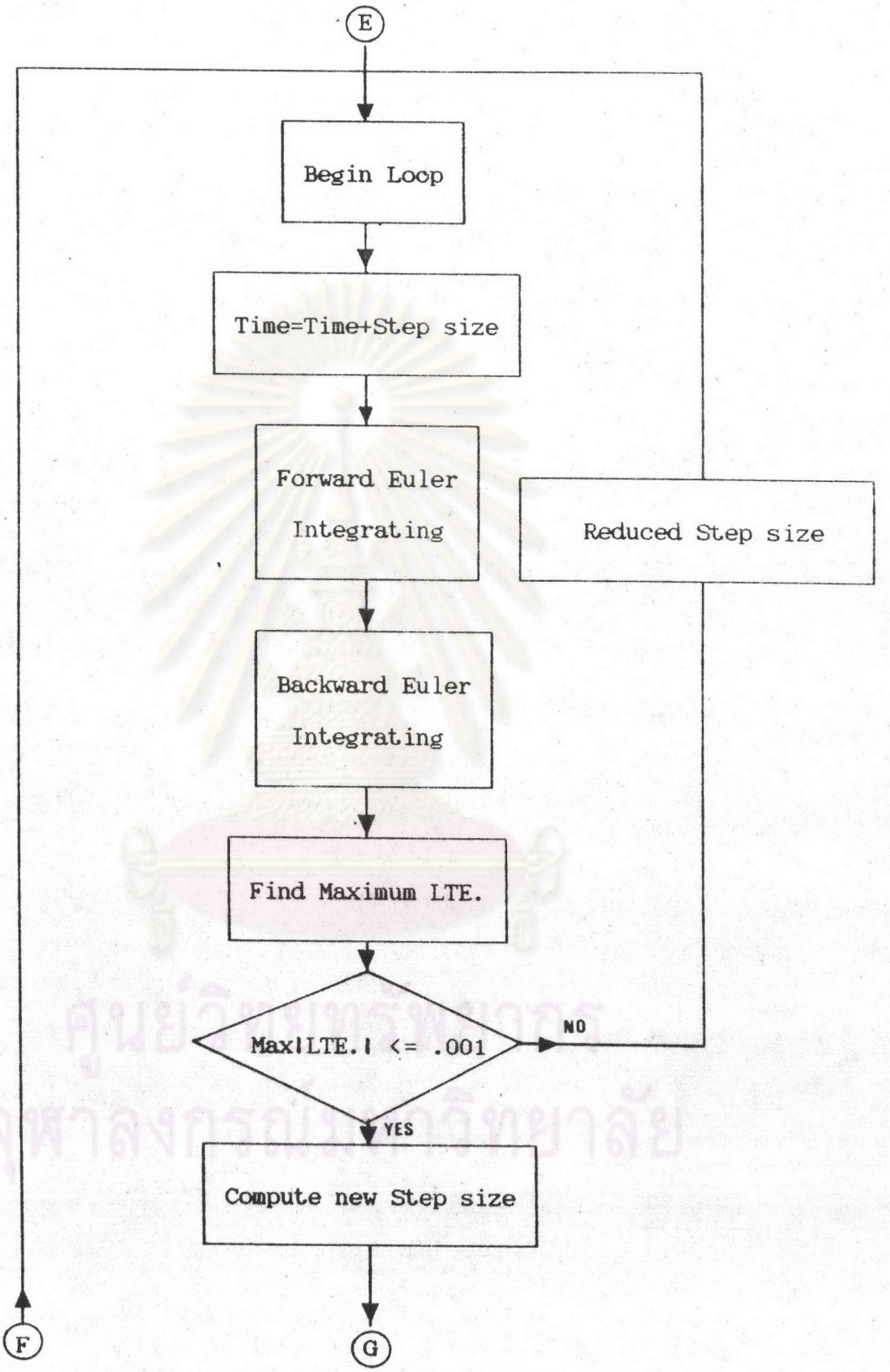


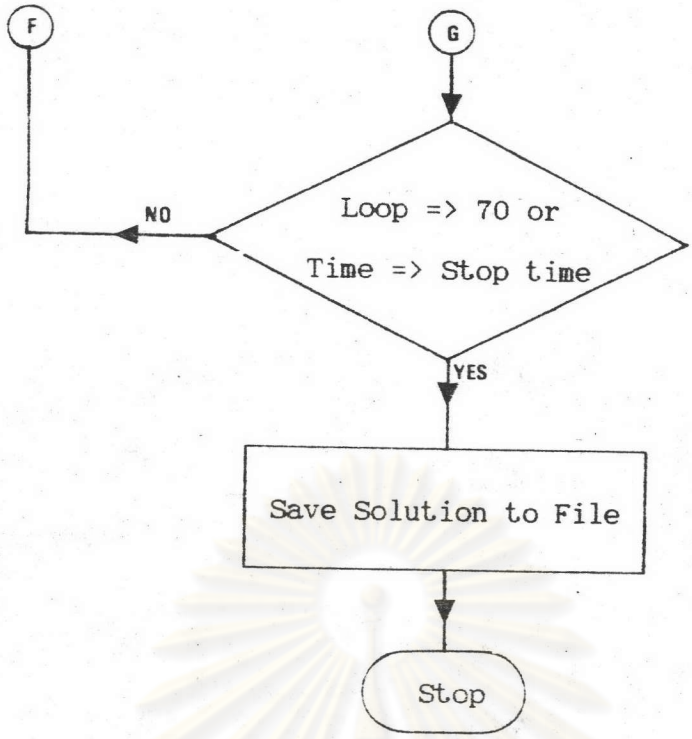

C
Input Voltage = Starting point
Begin loop
Input Voltage=Input Voltage+Step size
Clear Matrix Element
Stamp Matrix Element
Solve Matrix
Nonlinear circuit elements is presented
NO
YES
$|V^{(n)}-V^{(n-1)}|$=>EPS
YES
NO
Loop => NUMB
NO
YES
D



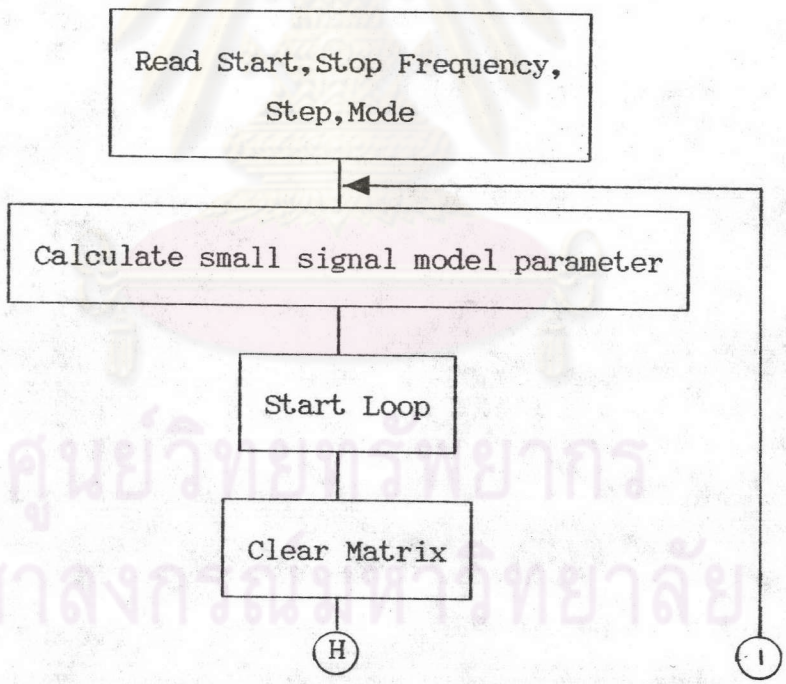
3. การวิเคราะห์คุณสมบัติตอบสนองเชิงเวลา (Transient Analysis)

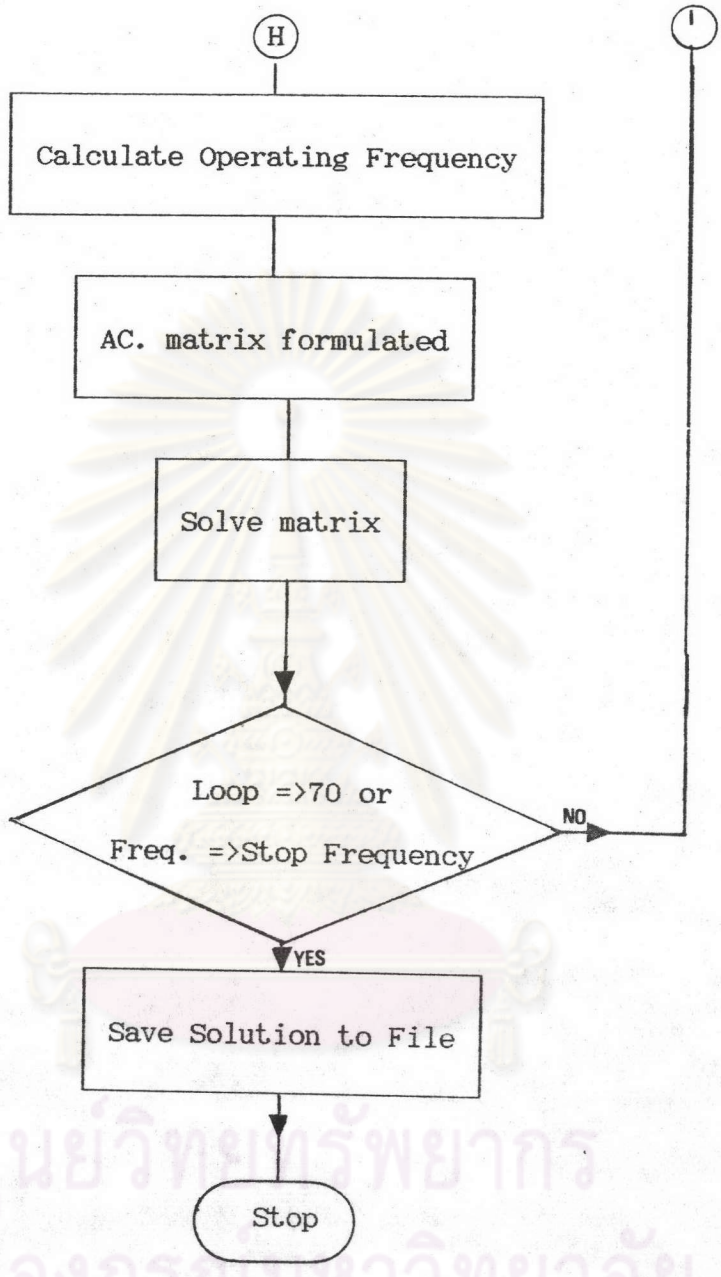






4. การวิเคราะห์คุณสมบัติตอบสนองเชิงความถี่ (Frequency Response Analysis)





ภาคผนวก ค.

คู่มือการใช้งานโปรแกรม SPEC version 1.0

1. การเริ่มต้นใช้งาน SPEC

เมื่อผู้ใช้อยู่ในสภาวะแวดล้อมของระบบจัดการ (Operating System) RSX-11 บน PDP-11 หรือ VMS บน VAX-11 แล้วสามารถเริ่มต้นใช้โปรแกรม SPEC ได้โดยใช้คำสั่ง

```
>RUN SPEC
```

SPEC จะโต้ตอบผู้ใช้โดยพิมพ์ข้อความในแฟ้มข้อมูล HELLO.MSG ให้ปรากฏบนจอภาพ หลังจากนั้น ผู้ใช้จะอยู่ในสภาวะแวดล้อมของ SPEC โดยอยู่ในโมดคำสั่งของSPEC ซึ่งจะแสดงบนจอภาพ ดังนี้

```
Command:
```

SPEC จะรอรับคำสั่งพื้นฐาน ซึ่งผู้ใช้จะต้องป้อนคำสั่งพื้นฐาน เพื่อให้ SPEC ทำงานต่อไป

2. คำสั่งพื้นฐานใน SPEC

คำสั่ง	การทำงาน
Input	ป้อนข้อมูลวงจรที่ต้องการวิเคราะห์
List	แสดงรายละเอียดของข้อมูลวงจร
Edit	เปลี่ยนแปลงแก้ไขข้อมูลวงจรที่ต้องการวิเคราะห์
Output	แสดงผลลัพธ์ของการวิเคราะห์
Qpoint	ทำการวิเคราะห์หาจุดทำงานสงบของวงจร
Frequency	ทำงานวิเคราะห์หาผลตอบสนองความถี่ของวงจร
TRANSFer	ทำการวิเคราะห์หาคุณสมบัติโอนย้ายของวงจร
TRANSIent	ทำการวิเคราะห์หาผลตอบสนองเชิงเวลาของวงจร
EXit	ออกจากสภาวะแวดล้อมของ SPEC

ตารางที่ 1 แสดงคำสั่งพื้นฐานของ SPEC

การป้อนคำสั่ง ให้พิมพ์คำสั่งที่ต้องการและตามท้ายด้วยปัดแคร่ (Carriage return) คำสั่งที่พิมพ์สามารถใช้คำสั่งย่อ (Abbreviated Command) คือ คำสั่งใน ตารางที่ 1 ที่เป็นตัวอักษรใหญ่ (Capital letter) ก็ได้

3. การป้อนข้อมูลวงจร

ผู้ใช้สามารถเข้าสู่โมด INPUT ได้โดยการพิมพ์คำสั่ง INPUT

Command: INPUT <CR>

หลังจากนั้น SPEC จะโต้ตอบกับผู้ใช้โดยแสดงข้อความข้างล่างนี้ บนจอภาพ

INPUT FROM INPUT FILE (YES:NO):

ผู้ใช้ต้องโต้ตอบกับ SPEC โดยพิมพ์คำว่า YES <CR> หรือ NO <CR>

3.1 การป้อนข้อมูลโดยตรง

เมื่อผู้ใช้เลือกวิธีป้อนข้อมูลโดยตรงด้วยการพิมพ์คำว่า NO <CR> แล้ว SPEC จะโต้ตอบกับผู้ใช้โดยแสดงข้อความข้างล่างนี้ บนจอภาพ

START TO INPUT CIRCUIT:

ผู้ใช้สามารถเริ่มป้อนข้อมูลของวงจรได้ โดยใช้ภาษาข้อมูลของ SPEC เอง หลังจากผู้ใช้ป้อนข้อมูลของวงจรจนจบแล้ว ก็พิมพ์คำว่า END SPEC จะกลับไปอยู่ในโมดคำสั่ง

ตัวอย่าง

```
Command: Input
INPUT FROM AN INPUT FILE (YES:NO): NO
START TO INPUT CIRCUIT:
R1   1    2    10K
C2   2    0    1 U F
VDC  1    0    10V
END
Command:
```

หมายเหตุ ข้อความที่ถูกขีดเส้นใต้เป็นข้อความที่ SPEC โต้ตอบกับผู้ใช้

3.2 การป้อนข้อมูลจากแฟ้มข้อมูล

เมื่อผู้ใช้เลือกวิธีป้อนข้อมูลจากแฟ้มข้อมูล ด้วยการพิมพ์คำว่า YES <CR> แล้ว SPEC จะโต้ตอบกับผู้ใช้ด้วยการแสดงข้อความข้างล่างนี้บนจอภาพ

INPUT FILE NAME (.SPC):

ผู้ใช้จะต้องโต้ตอบกับ SPEC ด้วยการพิมพ์ชื่อแฟ้มข้อมูลวงจรที่ต้องการวิเคราะห์ ซึ่งแฟ้มข้อมูลนี้จะต้องลงท้ายด้วย .SPC

ตัวอย่าง

Command: Input
INPUT FROM AN INPUT FILE (YES:NO) : YES
INPUT FILE NAME (.SPC): TEST .SPC
Command:

3.3 ภาษาอินพุทของ SPEC

ภาษาอินพุทของ SPEC มีลักษณะพื้นฐานประกอบด้วยฟิลด์ต่าง ๆ กัน คือ

NAME n1 n2....nn VALUE OR MODEL NAME

โดย

NAME เป็นฟิลด์ที่บอกถึงชื่อและชนิดของอุปกรณ์
n1, n2....nn เป็นฟิลด์ที่บอกถึงหมายเลขประจำโนดต่าง ๆ ที่อุปกรณ์นั้นต่ออยู่
VALUE เป็นฟิลด์ที่บอกถึงค่าของอุปกรณ์นั้น ๆ
MODEL NAME เป็นฟิลด์ที่บอกถึงชื่อเบอร์อุปกรณ์นั้น ๆ

รายละเอียดของฟิลด์ต่าง ๆ

ฟิลด์	ชนิด	รายละเอียด
NAME	ตัวอักษรผสมตัวเลข ยาวไม่เกิน 8 ตัว	บอกถึงชื่อและชนิดขององค์ประกอบวงจรนั้น ๆ ขึ้นต้นฟิลด์ด้วยตัวอักษรที่บอกถึงชนิดองค์ประกอบ เช่น RB2 ตัวเริ่มต้น คือ R แสดงถึง Resistor VDCIN ตัวเริ่มต้น คือ VDC แสดงถึงแหล่งจ่ายแรงดันแบบกระแสตรง etc.
N_1 N_2..N_n	ตัวเลข Integer แต่ละตัวมีค่าระหว่าง 0-32767	บอกถึงชื่อของโนดต่าง ๆ ชื่อของโนดต่างจะต้องเป็นตัวเลขและไม่จำเป็นต้องเป็นตัวเลขที่เรียงกันก็ได้
VALUE	ตัวเลขผสมตัวอักษร ยาวไม่เกิน 8 ตัว	บอกถึงค่าขององค์ประกอบวงจรนั้น ๆ อาจเป็นตัวเลขล้วน ๆ หรือเป็นตัวเลขผสมตัวอักษรก็ได้ โดยตัวอักษรจะเป็นตัวบอกถึงเทอม Exponential ของค่านั้น ๆ ก. ค่าตัวเลขล้วน อาจเขียนค่าเป็นค่าที่ไม่มีจุดทศนิยม, มีจุดทศนิยม หรือเป็นค่า Exponential ก็ได้ เช่น 1.01E2 มีค่าเท่ากับ 101

ฟิลด์	ชนิด	รายละเอียด
		ข. ค่าตัวเลขผสมตัวอักษร ตัวอักษรที่ใช้แสดงถึงค่า Exponential มี k, K = 10^3 M = 10^6 g, G = 10^9 m = 10^{-3} u, U = 10^{-6} n, N = 10^{-9} p, P = 10^{-12} เช่น 10M = 10,000,000.00
MODEL NAME	ตัวอักษรผสมตัวเลข ยาวไม่เกิน 8 ตัว	บอกถึงเบอร์ของอุปกรณ์ชนิดนั้น ๆ เพื่ออ้างถึงรายละเอียดของอุปกรณ์ในภาษาอินพุท MODEL อีกทีหนึ่ง

3.3.1 ภาษาอินพุทของตัวต้านทาน (Resistor)

รูปแบบ Rxxxxxxx n1 n2 VALUE

ตัวอย่าง R10 5 7 10K

คำอธิบาย n1 และ n2 เป็นเลขหมายประจำตัวโนดที่ตัวต้านทานต่ออยู่

ฟิลด์ VALUE เป็นค่าความต้านทานมีหน่วยเป็น โอห์ม

3.3.2 ภาษาอินพุทของคาปาซิเตอร์ (Capacitor)

รูปแบบ Cxxxxxxx n1 n2 VALUE

ตัวอย่าง C202 6 8 10U

คำอธิบาย n1 และ n2 เป็นเลขหมายประจำโนดที่คาปาซิเตอร์ต่ออยู่ พิลด์ VALUE เป็นค่าความจุของคาปาซิเตอร์มีหน่วยเป็น ฟารัด (Farad)

3.3.3 ภาษาอินพุทของตัวเหนี่ยวนำ (Inductor)

รูปแบบ Lxxxxxxx n1 n2 VALUE

ตัวอย่าง L101 12 1 2m

คำอธิบาย n1 n2 เป็นเลขหมายประจำโนดที่ตัวเหนี่ยวนำต่ออยู่ พิลด์ VALUE เป็นค่าความเหนี่ยวนำ มีหน่วยเป็นเฮนรี่ (Henry)

3.3.4 ภาษาอินพุทของแหล่งจ่ายแรงดันไฟฟ้า ดี.ซี.(D.C. Voltage source)

รูปแบบ VDCxxxxx n1 n2 VALUE

ตัวอย่าง VDC+ 5 0 -12.5

คำอธิบาย n1 และ n2 เป็นเลขหมายประจำโนดที่แหล่งจ่าย แรงดันต่ออยู่ โดย n1 เป็นโนดบวก และ n2 เป็นโนดลบ กระแสไฟฟ้าจะมีทิศทางจากโนด n2 ผ่านแหล่งจ่ายแรงดันมายังโนด n1

3.3.5 ภาษาอินพุทของแหล่งจ่ายแรงดันไฟฟ้า AC (AC. Voltage source)

รูปแบบ a) VACxxxxx n1 n2 MAGNITUDE PHASE

b) VINxxxxx n1 n2 MAGNITUDE PHASE

ตัวอย่าง VACIN 1 0 1.05V 180

คำอธิบาย n1 และ n2 เป็นเลขหมายประจำโนดที่แหล่งจ่ายแรงดันต่ออยู่

MAGNITUDE เป็นค่าของขนาดแรงดันไฟฟ้าที่มีแรงดันออฟเซ็ท (Offset

Voltage) เป็น 0 โวลต์ PHASE เป็นค่ามุมเฟสของแหล่งจ่ายแรงดันไฟฟ้า มี หน่วยเป็นองศา (Degree)

3.3.6 ภาษาอินพุทของแหล่งจ่ายกระแสไฟฟ้า ดี.ซี. (D.C. Current source)

รูปแบบ IDCxxxxx n1 n2 VALUE

ตัวอย่าง IDCBIAS 10 7 0.1 mA

คำอธิบาย n1 และ n2 เป็นเลขหมายประจำโนดที่แหล่งจ่ายกระแสต่ออยู่ ทิศทางของกระแสจะไหลจาก n2 ผ่านแหล่งจ่ายกระแสไปยัง n1

3.3.1 ภาษาอินพุทของแหล่งจ่ายกระแสไฟฟ้า เอ.ซี.(A.C. Current source)

รูปแบบ IACxxxxx n1 n2 MAGNITUDE PHASE

ตัวอย่าง IACINPUT 1 0 0.1 90

คำอธิบาย n1 และ n2 เป็นเลขหมายประจำโนดที่แหล่งจ่ายกระแสไฟฟ้าสลับต่ออยู่ MAGNITUDE เป็นค่าของขนาดกระแสไฟฟ้าสลับต่ออยู่ PHASE เป็นค่ามุมเฟส ของแหล่งจ่ายกระแสไฟฟ้า มีหน่วยเป็นองศา (Degree)

3.3.8 ภาษาอินพุทของแหล่งจ่ายแรงดัน VCVS (Voltage Control Voltage Source)

รูปแบบ VCVSxxxxx n1 n2 n3 n4 GAIN

ตัวอย่าง VCVS 3 2 7 8 100

คำอธิบาย n1 และ n2 เป็นเลขหมายประจำโนดที่แหล่งจ่ายแรงดันควบคุม (Control Voltage) ต่ออยู่ โดย n1 เป็นโนดบวก n2 เป็นโนดลบ กระแสไฟฟ้าจะไหลจากโนด n2 ผ่านแหล่งจ่ายแรงดันควบคุมมายัง n1 n3 และ n4 เป็นเลขหมายประจำโนดของแหล่งจ่ายแรงดันที่ถูกควบคุม (Controlled voltage source) ต่ออยู่ โดย n3 เป็นโนดบวก n4 เป็นโนดลบ กระแสไฟฟ้าจะไหลจากโนด n4 ผ่านแหล่งจ่ายแรงดันถูกควบคุมมายัง n3

GAIN เป็นอัตราขยายของ VCVS โดยมีความสัมพันธ์ว่า

V3 - V4 = GAIN (V1 - V2)

3.3.9 ภาษาอินพุทของแหล่งจ่ายกระแส VCCS (Voltage Control Current Source)

รูปแบบ VCCSxxxxx n1 n2 n3 n4 GMU
ตัวอย่าง VCCS3 2 3 6 7 0.01
คำอธิบาย n1 และ n2 เป็นเลขหมายประจำโนดที่แหล่งจ่ายแรงดันควบคุม (Control Voltage) ต่ออยู่ โดย n1 เป็นโนดบวก n2 เป็นโนดลบ กระแสไฟฟ้าจะไหลจากโนด n2 ผ่านแหล่งจ่ายแรงดันควบคุมมายัง n1 ,n3 และ n4 เป็นเลขหมายประจำโนดของแหล่งจ่ายกระแสที่ถูกควบคุม (Controlled Current Source) โดยกระแสไฟฟ้าจะไหลจาก โนด n4 ผ่านแหล่งจ่ายกระแสมายัง n3

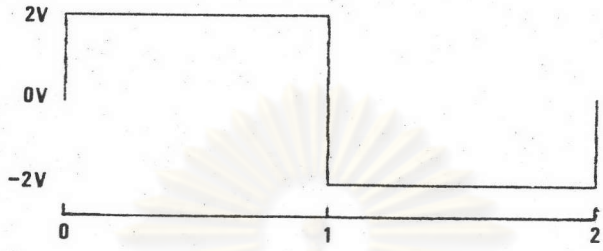
GMU เป็นค่าทรานคอนดักแตนซ์ (Transconductance) โดยมีความสัมพันธ์ดังนี้ คือ

Controlled Current = GMA (V1 - V2)

3.3.10 ภาษาอินพุทของแหล่งจ่ายแรงดันแบบพัลส์ (Pulse Voltage Source)

รูปแบบ VPULSExxx n1 n2 (np t1 v1 t2 v2 ... tn vn)
ตัวอย่าง VPULSE 1 0 (5 0 0 0 2.0 1 2.0 1 -2.0 2 -2.0)
คำอธิบาย n1 และ n2 เป็นเลขหมายประจำโนดที่แหล่งจ่ายแรงดันแบบพัลส์ต่ออยู่ โดย n1 เป็นโนดบวก n2 เป็นโนดลบ กระแสไฟฟ้าจะไหล จากโนด n2 ผ่านแหล่งจ่ายแรงดันพัลส์มายัง n1 np เป็นตัวเลขบอกถึงจำนวนของจุดหักมุมของแรงดันพัลส์ที่กำหนด t1 v1, t2 v2, ..., tn vn เป็นตัวเลขที่เป็นคู่ซึ่งระบุถึงจุดหักมุมของพัลส์ โดยที่จุด t1 มี

ค่า แรงดันเท่ากับ v1 ภาษาอินพุทที่แสดงในตัวอย่างสามารถ แสดงโดยรูปภาพดังรูปข้างล่างนี้

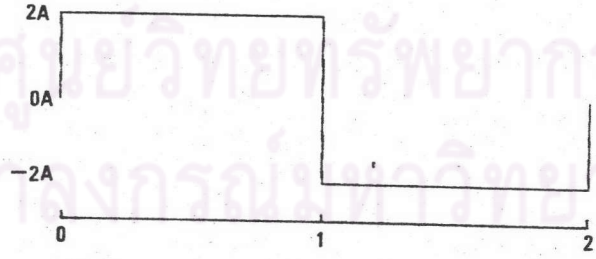


3.3.11 ภาษาอินพุทของแหล่งจ่ายกระแสแบบพัลส์ (Pulse Current Source)

รูปแบบ IPULSExx n1 n2 (np t1 i1 t2 i2 ... tn in)

ตัวอย่าง IPULSE 1 0 (5 0 0 0 2.0 1 2.0 1 -2.0 2 -2.0)

คำอธิบาย n1 และ n2 เป็นเลขหมายประจำโนดที่แหล่งจ่ายกระแสแบบพัลส์ต่ออยู่ ทิศทางของกระแสไฟฟ้าจะไหลจาก n2 ผ่านแหล่งจ่ายกระแสไปยัง n1 np เป็นตัวเลขบอกถึงจำนวนของจุดหักมุมของกระแสพัลส์ที่กำหนด t1 i1 t2 i2 .... tn in เป็นตัวเลขที่เป็นคู่ระบุถึงจุดหักมุมของพัลส์ โดยที่จุด t1 มีค่ากระแสเท่ากับ i1 ภาษาอินพุทที่แสดงในตัวอย่างสามารถแสดงโดยรูปภาพ ได้ดังรูปข้างล่างนี้



3.3.12 ภาษาอินพุทของไดโอด (Diode)

รูปแบบ Dxxxxxxx n1 n2 model name initial condition

ตัวอย่าง D11 1 5 IN40001 VD = 0.4

คำอธิบาย n1 เป็นเลขหมายประจำโนดที่โนดแอโนด (Anode) ต่ออยู่

n2 เป็นเลขหมายประจำโนดที่โนด แคโถด (Cathode) ต่ออยู่ MODEL NAME จะบ่งบอกถึงว่า ไดโอดนี้มีคุณสมบัติตามโมเดลที่กำหนดไว้ในบรรทัดกำหนดคุณสมบัติของโมเดลอีกทีหนึ่ง Initial condition เป็นค่าเริ่มต้นที่กำหนดให้หากผู้ใช้ไม่กำหนดให้SPEC จะกำหนดให้เป็น 0.6 โวลต์

3.3.13 ภาษาอินพุทของทรานซิสเตอร์ (Transistor)

รูปแบบ Qxxxxxxx n1 n2 n3 Model name Initial condition

ตัวอย่าง Q7 3 2 1 25C458 VBE = 0.5 VBC = -1.0

คำอธิบาย n1 เป็นเลขหมายประจำโนดที่โนดเบส (Base) ต่ออยู่ n2 เป็นเลขหมายประจำโนดที่คอลเล็กเตอร์ (Collector) ต่ออยู่ n3 เป็นเลขหมายประจำโนดที่อีมิตเตอร์ (Emitter) ต่ออยู่ MODEL NAME จะบ่งถึงว่า ทรานซิสเตอร์นี้มีคุณสมบัติตามโมเดล ที่กำหนดไว้ในบรรทัดที่ กำหนดคุณสมบัติของโมเดลอีกทีหนึ่ง Initial condition ที่กำหนดให้หากผู้ใช้ไม่กำหนดให้ SPEC จะกำหนดให้ VBE = 0.6 และ VBC = -1.0 โวลต์ ตามลำดับ

3.3.14 ภาษาอินพุทของออปแอมป์ (Operational Amplifier)

รูปแบบ OPAMPxxx n1 n2 n3 model name

ตัวอย่าง OPAMP 2 3 2 4 LM741

คำอธิบาย n1 เป็นเลขหมายประจำโนดที่โนดอินพุทแบบนอนอินเวอติ้ง (Non - Inverting Input) ต่ออยู่ n2 เป็นเลขหมายประจำโนดอินพุทแบบอินเวอติ้ง (Inverting Input) ต่ออยู่ n3 เป็นเลขหมายประจำโนดเอาท์พุท (Output) MODEL NAME จะบ่งถึงว่าออปแอมป์นี้มีคุณสมบัติตามโมเดลที่กำหนดไว้ในบรรทัดที่กำหนดคุณสมบัติของโมเดลอีกทีหนึ่ง

3.3.15 ภาษาอินพุทของโมเดล

รูปแบบ MODEL model name (characteristic specification field)

ตัวอย่าง MODEL LM741 (GAIN =1M POLE1 =10 POLE2 = 1M)

คำอธิบาย ถ้าในวงจรที่กำลังวิเคราะห์เมื่อมีอุปกรณ์ประเภทไดโอด ทรานซิสเตอร์ และออปแอมป์ ซึ่งอ้างถึงชนิดโมเดลของอุปกรณ์แต่ตัวแล้ว ในภาษาอินพุทจะต้องระบุถึงรายละเอียด คุณสมบัติของอุปกรณ์แต่ละตัวด้วย บรรทัด MODEL นี้ ซึ่งคุณสมบัติของอุปกรณ์จะถูกระบุไว้ใน Characteristic specification field หากผู้ใช้ไม่ได้กำหนดคุณสมบัติของอุปกรณ์บางอย่างแล้ว SPEC จะกำหนดค่าทดแทน (Default) ให้กับ Model ชนิดนั้นเอง

3.3.15.1 ภาษาอินพุทของโมเดลไดโอด (Diode Model)

ตัวอย่าง MODEL 1N4001 (IS = 0.1 PA)

NAME	DESCRIPTION	DEFAULT
IS	Saturation current	1 x 10-14 Amp
CJO	Junction Capacitance	0 Farad

3.3.15.2 ภาษาอินพุทของโมเดลทรานซิสเตอร์ (Transistor Model)

ตัวอย่าง MODEL 2SC458 (BF =100 CJC =0.7 PF)

NAME	DESCRIPTION	DEFAULT
CJC	Base - Collector junction capacitor	0 Farad
CJE	Base - Emitter junction capacitor	0 Farad
ALPR		0.5
ALPF		0.99
BR		2
BF		100
IS	Saturation Current	1.0E-14 Amp

3.3.15.3 ภาษาอินพุทของโมเดลออปแอมป์ (Operational Amplifier Model)

ตัวอย่าง MODEL LM741 (RIN = 1M ROUT = 75 GAIN = 1M)

NAME	DESCRIPTION	DEFAULT
GAIN	Voltage Gain	1.0E10
RIN	Input Resistance	1.0E+10 Ohm
CIN	Input Capacitance	0 Farad
RO	Output Resistance	1.0E-10 Ohm
POLE1	Dominant pole	10 HZ
POLE2	Secondary pole	1 MHZ
VS	Supply Voltage	15 Volts

4. การวิเคราะห์จุดทำงานสงบ (DC Operating Point)

ผู้ใช้สามารถเข้าสู่โมดวิเคราะห์หาจุดทำงานสงบได้ด้วยการป้อนคำสั่ง QPOINT

ตัวอย่าง

Command: QPOINT <CR>

หลังจากนั้น SPEC จะโต้ตอบกับผู้ใช้โดยการพิมพ์ข้อความบนจอภาพ

```
+----------------------------+
:  OPERATING POINT ANALYSIS  :
+----------------------------+
input ambient temperature(Celcius)=25

Command:
```

ผู้ใช้จะต้องป้อนข้อมูลอุณหภูมิขณะทำงานของวงจรที่ต้องการวิเคราะห์

เมื่อ SPEC ทำการวิเคราะห์วงจรแล้วเสร็จ ก็จะกลับเข้าสู่โมดคำสั่งต่อไป

5. การวิเคราะห์คุณสมบัติส่งทอด (Transfer Characteristic Analysis)

ผู้ใช้สามารถเข้าสู่โมดวิเคราะห์หาคุณสมบัติส่งทอดได้ด้วยการป้อนคำสั่ง TRANSFER

ตัวอย่าง

Command: TRANSFER <CR>

หลังจากนั้น SPEC จะโต้ตอบกับผู้ใช้โดยการพิมพ์ข้อความบนจอภาพ

```
+-----------------------------------+
:  TRANSFER CHARACTERISTIC RESPONSE  :
+-----------------------------------+
Input ambient temperature(celcious)=25

Sweep:vin 0 5 .1

Command:
```

ผู้ใช้จะต้องป้อนอุณหภูมิขณะทำงานของวงจรที่ต้องการวิเคราะห์ในบรรทัดแรกและป้อนชื่อของแหล่งจ่ายแรงดันขาเข้าที่ต้องการให้มีการกวาด (Sweep) ของแรงดัน จุดเริ่มต้นของแรงดัน จุดสุดท้ายของแรงดันไฟฟ้าที่กวาดไป ขนาดเพิ่มของแรงดันที่กวาด (Step) ตามลำดับ

เมื่อ SPEC ทำการวิเคราะห์วงจรแล้วเสร็จ ก็จะกลับเข้าสู่โมดคำสั่งต่อไป

6. การวิเคราะห์ผลตอบสนองเชิงเวลา

ผู้ใช้สามารถเข้าสู่โมดวิเคราะห์หาผลตอบสนองเชิงเวลาได้ด้วยการป้อนคำสั่ง TRANSIENT

ตัวอย่าง

Command: TRANSIENT <CR>

หลังจากนั้น SPEC จะโต้ตอบกับผู้ใช้โดยการพิมพ์ข้อความบนจอภาพ

```
+----------------------+
!   TRANSIENT ANALYSIS  !
+----------------------+
```

การวิเคราะห์หาผลตอบสนองเชิงเวลานั้น SPEC จะแบ่งลักษณะการวิเคราะห์ได้เป็น 2 ลักษณะคือ การวิเคราะห์จากจุดเริ่มต้นในครั้งแรก และการวิเคราะห์หาผลตอบสนองในช่วงเวลาถัดไป ผู้ใช้จะต้องตอบ SPEC ว่าต้องการวิเคราะห์จากจุดเริ่มต้น หรือวิเคราะห์ต่อไป

6.1 ผู้ใช้เลือกรูปแบบของการวิเคราะห์จากจุดเริ่มต้น

ผู้ใช้ได้ตอบกับ SPEC ด้วยการป้อนคำว่า START

```
+----------------------+
!   TRANSIENT ANALYSIS  !
+----------------------+
Analyze from starting point OR continue on time(START/CONTINUE):start

Starting time =0
To(time) =1
Input ambient temperature(°C)=25
Starting Step=.01
```

หลังจากนั้น ผู้ใช้ต้องป้อนข้อมูลจุดเวลาเริ่มต้น (Starting time) จุดสุดท้าย อุณหภูมิขณะทำงาน ขนาดความละเอียดของช่วงเวลาในขั้นต้น (Starting step size)

เมื่อ SPEC ทำการวิเคราะห์แล้วเสร็จจะแสดงเวลาที่ SPEC วิเคราะห์ถึง

```
Analyzed to time=     0.2004E+01
*********************************************
*   Caution: output the     RESULTS   before  *
*      continue on transient analysis         *
*********************************************
Command:
```

หาก SPEC ยังวิเคราะห์ไม่ถึงจุดเวลาที่ต้องการ ผู้ใช้จะต้องเข้าโมดแสดงผลลัพธ์เพื่อจัดพิมพ์ผลลัพธ์เสียก่อน หลังจากนั้นจึงเข้าสู่โมด TRANSIENT เพื่อทำการวิเคราะห์ในช่วงเวลาถัดไป ดังหัวข้อ 6.2

6.2 ผู้ใช้เลือกโมดการวิเคราะห์ในช่วงเวลาถัดไป

ผู้ใช้ตอบ SPEC ด้วยการป้อนคำว่า CONTINUE

```
Analyze from starting point OR continue on time     (START/CONTINUE):continue
```

และป้อนข้อมูลเวลา จุดเวลาต่อไปที่ต้องการให้ SPEC วิเคราะห์

```
Continue to time=3
Analyzed to time=     0.30954E+01
*********************************************
*   Caution: output the     RESULTS   before  *
*      continue on transient analysis         *
*********************************************
Command:
```

เมื่อ SPEC ทำการวิเคราะห์แล้วเสร็จ จะแสดงเวลาที่ SPEC วิเคราะห์ถึง ผู้ใช้จะต้องเข้าโมด แสดงผลลัพธ์เพื่อจัดพิมพ์ผลลัพธ์เสียก่อน หลังจากนั้น จึงจะทำการ วิเคราะห์ในช่วงเวลาถัดไปได้

7. การวิเคราะห์ผลตอบสนองความถี่ (Frequency Response Analysis)

ผู้ใช้สามารถเข้าสู่โมดวิเคราะห์หาผลตอบสนองความถี่ได้ ด้วยการป้อนคำสั่ง Frequency

ตัวอย่าง

Command: Frequency <CR>

ในการวิเคราะห์หาผลตอบสนองเชิงความถี่ จะต้องผ่านการวิเคราะห์ หาจุดทำงานสงบเสียก่อน หากผู้ใช้ยังไม่ได้ทำการวิเคราะห์หาจุดทำงานสงบ SPEC จะพิมพ์ ข้อความ

Enter "QPOINT"

เพื่อบังคับให้ผู้ใช้ป้อนคำสั่ง QPOINT ก่อน

เมื่อผู้ใช้ทำการวิเคราะห์จุดทำงานสงบเรียบร้อยแล้ว จึงจะสามารถทำการวิเคราะห์หาผลตอบสนองความถี่ได้ โดย SPEC จะโต้ตอบกับผู้ใช้ ดังนี้

```
+---------------------------+
: FREQUENCY RESPONSE ANALYSIS :
+---------------------------+
Sweep frequency:1 1M 5 dec
```

ผู้ใช้จะต้องป้อนข้อมูล จุดความถี่เริ่มต้น, จุดความถี่ปลาย , จำนวนจุดระหว่างการกวาดความถี่ (Sweep) , วิธีการกวาดความถี่ ตามลำดับ

SPEC แบ่งวิธีการกวาดความถี่ เป็น 2 ลักษณะคือ

ก. Linear Mode ในโมดนี้ ตำแหน่งจุดความถี่ในการกวาด จะอยู่ห่างจากกัน เป็นระยะความถี่เท่ากัน

ผู้ใช้ต้องป้อนข้อมูลของการวิเคราะห์ ดังนี้

Sweep frequency: START STOP STEP MODE

ตัวอย่าง

Sweep frequency: 1K 20K 1K LIN

กวาดความถี่จากจุดความถี่เริ่มต้นที่ 1 KHZ และเพิ่มความถี่จุดละ 1 KHZ ไปจนถึง 20 KHZ

ข. Decade Mode ในโมดนี้ ตำแหน่งจุดความถี่ในการกวาด จะเพิ่มค่าแบบลอการิทึม (Logarithm)

ผู้ใช้ต้องป้อนข้อมูลของการวิเคราะห์ ดังนี้

Sweep frequency: START STOP STEP MODE

ตัวอย่าง

Sweep frequency: 0.1 10 5 DEC

ถ้าวัดความถี่จากจุดความถี่เริ่มต้น ที่ 0.1 HZ ไปจนถึงความถี่ที่จุดสุดท้ายเท่ากับ 10 HZ โดยเพิ่มความถี่ช่วงละ 10 เท่าของความถี่เดิม และในแต่ละช่วงความถี่ มีจำนวนจุดวิเคราะห์เท่ากับ 5 จุด

8. การแสดงผลลัพธ์

ผู้ใช้สามารถแสดงผลลัพธ์ ได้ด้วยการป้อนคำสั่ง

Command: output

หลังจากนั้น SPEC จะโต้ตอบกับผู้ใช้ โดยการพิมพ์ข้อความบนจอภาพ

```
-------------------------
  OUTPUT THE SOLUTION
-------------------------
Printout the solution(YES/NO):no


01 :Operating point solution
02 :Transfer characteristic plot
03 :Transient characteristic plot
04 :Frequency response plot
05 :All

Your selection is:4
```

ผู้ใช้สามารถพิมพ์ผลลัพธ์ทางเครื่องพิมพ์ได้ ด้วยการตอบ YES และสามารถเลือกแสดงผลลัพธ์ ด้วยการป้อน เลข 1 ถึง 5

8.1 ภาคแสดงผลลัพธ์การวิเคราะห์จุดทำงานสงบ

ผู้ใช้เลือกแสดงผลลัพธ์การวิเคราะห์จุดทำงานสงบ ได้โดยการป้อนเลข 1

SPEC จะแสดงผลลัพธ์ ดังภาพข้างล่างนี้

```
:-------------------------------------------------:
:-- SPEC --:      Operating point solution        :
:-------------------------------------------------:
 voltage node        value
 ------------        -----
    1          1.000E+00 Volts
    2          1.000E+00 Volts
    3          1.154E+00 Volts
    4          4.618E-01 Volts
    5          5.428E+00 Volts
    6          1.000E+01 Volts
    7          1.152E+00 Volts
```

Command:

8.2 ภาคแสดงผลลัพธ์การวิเคราะห์คุณสมบัติส่งทอด

ผู้ใช้เลือกแสดงผลลัพธ์การวิเคราะห์คุณสมบัติส่งทอดได้ โดย การป้อน เลข 2

SPEC จะโต้ตอบกับผู้ใช้โดยการพิมพ์ข้อความบนจอภาพ

```
:-------------------------------------------------:
: -- SPEC --    Transfer characteristic response   :
:-------------------------------------------------:
```

Plot Voltage node:5 2

ผู้ใช้สามารถเลือกแสดงผลลัพธ์ของแรงดันที่โนดต่าง ๆ ได้ ครั้งละ 2 โนด ด้วยการป้อนหมายเลขประจำโนด ที่ต้องการแสดงผลลัพธ์ ดังนี้

Plot Voltage Node: 1 3

หรือ

Plot Voltage Node: 1,3

ก็ได้เช่นกัน

SPEC จะแสดงผลลัพธ์ ทั้งค่าตัวเลข และกราฟ ดังรูป

```
:-------------------------------------------------:
: -- SPEC --    Transfer characteristic response   :
:-------------------------------------------------:
```

Plot Voltage node:5 2

Step	Input Voltage	V. Node(5)	V. Node(2)
1	-5.000E-01	-5.000E-01	-1.034E+00
2	-4.500E-01	-4.500E-01	-9.809E-01
3	-4.000E-01	-4.000E-01	-9.274E-01
4	-3.500E-01	-3.500E-01	-8.731E-01

5	-3.000E-01	-3.000E-01	-8.177E-01
6	-2.500E-01	-2.500E-01	-7.604E-01
7	-2.000E-01	-2.000E-01	-6.997E-01
8	-1.500E-01	-1.500E-01	-6.293E-01
9	-1.000E-01	-1.000E-01	1.187E-01
10	-5.000E-02	-5.000E-02	3.260E+00
11	3.725E-08	3.725E-08	6.140E+00
12	5.000E-02	5.000E-02	8.414E+00
13	1.000E-01	1.000E-01	9.603E+00
14	1.500E-01	1.500E-01	9.925E+00
15	2.000E-01	2.000E-01	9.980E+00
16	2.500E-01	2.500E-01	9.988E+00
17	3.000E-01	3.000E-01	9.989E+00
18	3.500E-01	3.500E-01	9.990E+00
19	4.000E-01	4.000E-01	9.990E+00
20	4.500E-01	4.500E-01	9.990E+00
21	5.000E-01	5.000E-01	9.990E+00

V(5)= A V(2)= B

SCALE A= 1.9E-02 SCALE B= 2.2E-01

```
A-->  -6.0E-01  -4.1E-01  -2.2E-01  -2.5E-02   1.7E-01   3.6E-01   5.5E-01
B-->  -2.1E+00   6.9E-02   2.3E+00   4.5E+00   6.7E+00   8.9E+00   1.1E+01
        +.........+.........+.........+.........+.........+.........+
-5.0E-01 :    *    :         :         :         :         :         :
-4.5E-01 :    B A  :         :         :         :         :         :
-4.0E-01 :    B    A         :         :         :         :         :
-3.5E-01 :    B    : A       :         :         :         :         :
-4.5E-01 :    B A  :         :         :         :         :         :
-4.0E-01 :    B    A         :         :         :         :         :
-3.5E-01 :    B    : A       :         :         :         :         :
-3.0E-01 +....B....+....A....+.........+.........+.........+.........+
-2.5E-01 :     B   :       A :         :         :         :         :
-2.0E-01 :     B   :         A         :         :         :         :
-1.5E-01 :     B   :         : A       :         :         :         :
-1.0E-01 :         B         :   A     :         :         :         :
-5.0E-02 +.........+.........+...B...A.+.........+.........+.........+
3.7E-08  :         :         :         :A     B  :         :         :
5.0E-02  :         :         :         : A       :      B  :         :
1.0E-01  :         :         :         :     A   :         :  B      :
1.5E-01  :         :         :         :       A:         :    B    :
2.0E-01 +.........+.........+.........+.........+A........+...B.....+
2.5E-01  :         :         :         :         :  A      :    B    :
3.0E-01  :         :         :         :         :       A :    B    :
3.5E-01  :         :         :         :         :         A:   B    :
4.0E-01  :         :         :         :         :         : A  B    :
4.5E-01 +.........+.........+.........+.........+.........+...BA....+
Command:
```

8.3 ภาคแสดงผลลัพธ์การวิเคราะห์คุณสมบัติตอบสนองเชิงเวลา

ผู้ใช้เลือกแสดงผลลัพธ์การวิเคราะห์คุณสมบัติตอบสนองเชิงเวลาได้ โดยการป้อนเลข 3

SPEC จะโต้ตอบกับผู้ใช้โดยการพิมพ์ข้อความบนจอภาพ

```
:-------------------------------------------------:
: -- SPEC --                 Transient response    :
:-------------------------------------------------:

Plot Voltage node:1 3

Data to be Ploted range from      0.000to      4.988
From Starting time:0

To Stop time:2.0

Step:.1
```

ผู้ใช้สามารถเลือกแสดงผลลัพธ์ของแรงดันที่โนดต่าง ๆ ได้ ครั้งละ 2 โนด ด้วยการป้อนหมายเลขประจำโนด ที่ต้องการแสดงผลลัพธ์ ดังนี้

```
Plot Voltage Node:  1   3
```

หรือ

```
Plot Voltage Node:  1,  3
```

ก็ได้เช่นกัน

หลังจากนั้น SPEC จะแสดงช่วงเวลาที่สามารถแสดงคำตอบได้ ผู้ใช้ต้องป้อนข้อมูล เวลาเริ่มต้น, เวลาปลาย และการเพิ่มเวลา แต่ละช่วง ดังตัวอย่าง

SPEC จะแสดงผลลัพธ์ ทั้งค่าตัวเลขและกราฟ ดังรูป

Time	V.node(1)	V.Node(3)
0.0000000E+00	0.0000000E+00	0.0000000E+00
0.1000000E+00	0.4000000E+01	-0.3999989E+00
0.2000000E+00	0.4000000E+01	-0.7999982E+00
0.3000000E+00	0.4000000E+01	-0.1199998E+01
0.4000000E+00	0.4000000E+01	-0.1599997E+01
0.5000000E+00	0.4000000E+01	-0.1999996E+01
0.6000000E+00	0.4000000E+01	-0.2399996E+01
0.7000000E+00	0.4000000E+01	-0.2799995E+01
0.8000001E+00	0.4000000E+01	-0.3199994E+01
0.9000001E+00	0.4000000E+01	-0.3599993E+01
0.1000000E+01	0.4000000E+01	-0.3999992E+01
0.1100000E+01	0.4000000E+01	-0.4399991E+01
0.1200000E+01	0.4000000E+01	-0.4799990E+01
0.1300000E+01	0.4000000E+01	-0.5199989E+01
0.1400000E+01	0.4000000E+01	-0.5599988E+01
0.1500000E+01	0.4000000E+01	-0.5999987E+01
0.1600000E+01	0.4000000E+01	-0.6399986E+01
0.1700000E+01	0.4000000E+01	-0.6799985E+01
0.1800000E+01	0.4000000E+01	-0.7199984E+01
0.1900000E+01	0.4000000E+01	-0.7599983E+01

V(1)= A V(3)= B

SCALE A= 8.0E-02 SCALE B= 1.5E-01

```
A--> -4.0E-01   4.0E-01   1.2E+00   2.0E+00   2.8E+00   3.6E+00   4.4E+00
B--> -8.4E+00  -6.8E+00  -5.3E+00  -3.8E+00  -2.3E+00  -7.6E-01   7.6E-01
       +.........+.........+.........+.........+.........+.........+
0.0E-01 :    A    :         :         :         :         :    B    :
1.0E-01 :         :         :         :         :         : B  A    :
2.0E-01 :         :         :         :         :        B:    A    :
3.0E-01 :         :         :         :         :      B  :    A    :
4.0E-01 +.........+.........+.........+.........+...B.....+....A....+
5.0E-01 :         :         :         :         :B        :    A    :
6.0E-01 :         :         :         :        B:         :    A    :
7.0E-01 :         :         :         :     B   :         :    A    :
8.0E-01 :         :         :         :  B      :         :    A    :
9.0E-01 +.........+.........+.........+B........+.........+....A....+
1.0E+00 :         :         :       B :         :         :    A    :
1.1E+00 :         :         :     B   :         :         :    A    :
1.2E+00 :         :         :  B      :         :         :    A    :
1.3E+00 :         :         B         :         :         :    A    :
1.4E+00 +.........+.......B.+.........+.........+.........+....A....+
1.5E+00 :         :    B    :         :         :         :    A    :
1.6E+00 :         : B       :         :         :         :    A    :
1.7E+00 :         B         :         :         :         :    A    :
1.8E+00 :      B  :         :         :         :         :    A    :
1.9E+00 +....B....+.........+.........+.........+.........+....A....+
```

Command:

8.4 ภาคแสดงผลลัพธ์การวิเคราะห์คุณสมบัติตอบสนองเชิงความถี่

ผู้ใช้เลือกแสดงผลลัพธ์การวิเคราะห์คุณสมบัติตอบสนองเชิงความถี่ได้ โดยการป้อนเลข 4

SPEC จะโต้ตอบกับผู้ใช้โดยการพิมพ์ข้อความ

```
:-----------------------------------------:
: -- SPEC --        Frequency response    :
:-----------------------------------------:

PRINT V NODE:5
```

ผู้ใช้สามารถเลือกแสดงผลลัพธ์ของแรงดันที่โนดต่าง ๆ ได้ครั้งละ 1 โนด ด้วยการป้อนหมายเลขประจำโนด ที่ต้องการแสดงผลลัพธ์

SPEC จะแสดงผลลัพธ์ทั้งค่าตัวเลข และกราฟ ซึ่งแสดงคุณสมบัติของแรงดันที่จุดนั้น ทั้งขนาดและมุมเฟส ดังภาพ

VOLTAGE NODE 5

FREQUENCY	MAGNITUDE	PHASE
1.000E+00	8.291E-01	-94.668
1.585E+00	1.299E+00	-97.291
2.512E+00	2.006E+00	-101.157
3.981E+00	3.000E+00	-106.312

6.310E+00	4.262E+00	-111.849
1.000E+01	5.755E+00	-115.663
1.585E+01	7.692E+00	-116.320
2.512E+01	1.068E+01	-115.250
3.981E+01	1.548E+01	-115.457
6.310E+01	2.275E+01	-119.091
1.000E+02	3.241E+01	-126.829
6.310E+02	5.886E+01	-166.520
1.000E+03	5.996E+01	-171.394
1.585E+03	6.042E+01	-174.561
2.512E+03	6.060E+01	-176.593
3.981E+03	6.067E+01	-177.899
6.310E+03	6.070E+01	-178.754
1.000E+04	6.071E+01	-179.341
1.585E+04	6.072E+01	-179.786
2.512E+04	6.072E+01	179.815
3.981E+04	6.072E+01	179.377
6.310E+04	6.071E+01	178.804
1.000E+05	6.068E+01	177.973
1.585E+05	6.062E+01	176.706
2.512E+05	6.046E+01	174.737
3.981E+05	6.008E+01	171.666
6.310E+05	5.914E+01	166.907

A = magnitude B = phase

```
A--> -6.1E+00   6.1E+00   1.8E+01   3.0E+01   4.3E+01   5.5E+01   6.7E+01
B--> -2.2E+02  -1.4E+02  -7.2E+01   1.5E-02   7.2E+01   1.4E+02   2.2E+02
        +.........+.........+.........+.........+.........+.........+
1.0E+00 :    A    :    B    :         :         :         :         :
1.6E+00 :     A   :     B   :         :         :         :         :
2.5E+00 :      A  :      B  :         :         :         :         :
4.0E+00 :       A :       B :         :         :         :         :
6.3E+00 +.......A.+...B.....+.........+.........+.........+.........+
1.0E+01 :        A:  B      :         :         :         :         :
1.6E+01 :         :A B      :         :         :         :         :
2.5E+01 :         :  *      :         :         :         :         :
4.0E+01 :         :  B   A  :         :         :         :         :
6.3E+01 +.........+..B......+..A......+.........+.........+.........+
1.0E+02 :         : B       :         :A        :         :         :
1.6E+02 :         B         :         :         A         :         :
2.5E+02 :         B:        :         :         :         :         :
4.0E+02 :       B  :        :         :         :         A         :
6.3E+02 +.....B...+.........+.........+.........+.........+..A......+
```

```
C*************************************************************
C   SPEC: a Simulation Program for Electronic Circuit
C          written: RAEWAT WANG  28-MAR-84
C*************************************************************
C
      PROGRAM SPEC
C
      INCLUDE    'DECLARE.INC'
      INCLUDE    'PARA.INC'
C
      CALL CLSCREEN
        WRITE(6,1)
1      FORMAT( ' SPEC:simulation program VERSION 1.14 Feb 19,87')
C--:
       OPEN (UNIT=10,FILE='HELLO.MSG',STATUS='OLD',ACCESS='SEQUENTIAL

      CALL COLOR('36','40')
        DO 28 I=1,24
            READ(UNIT=10,END=19,FMT=27)LINE
            WRITE (6,27) LINE
28      CONTINUE
27     FORMAT (A80)
19     CLOSE (UNIT=10)
C----:
      QPID       = .FALSE.
C--:
C
      CALL COLOR('37','40')
10      WRITE (6,11)
11      FORMAT (' Command:',:)
C
      CALL GETL(5)
C
      CALL GETW
C
C----check command word-----
C
      IF ( W(1:1) .EQ. 'I') THEN
      CALL CLSCREEN
              CALL INPUT
      ELSEIF ( W(1:1) .EQ. 'Q' ) THEN
      CALL CLSCREEN
              CALL QPOINT
```

```
      ELSEIF ( W(1:2) .EQ. 'EX' ) THEN
       CALL CLSCREEN
       WRITE (*,*) ' BYE-BYE  have a good day'
              STOP
      ELSEIF (W(1:1) .EQ.  'L' ) THEN
              CALL LIST
      ELSEIF (W(1:6) .EQ.  'TRANSF' ) THEN
      CALL CLSCREEN
              CALL TRANF
      ELSEIF (W(1:1) .EQ. 'O' ) THEN
      CALL CLSCREEN
              CALL OUTPUT
      ELSEIF (W(1:6) .EQ. 'TRANSI' ) THEN
      CALL CLSCREEN
              CALL TRANSI
      ELSEIF (W .EQ. 'PRINT') THEN
      CALL CLSCREEN
              CALL PRINT
      ELSEIF (W(1:1) .EQ. 'F') THEN
             CALL CLSCREEN
             CALL ACAN
      ELSEIF (W(1:2) .EQ. 'ED') THEN
             CALL CLSCREEN
             CALL CHANGE
      ELSEIF (W(1:1) .EQ. 'D') THEN
             CALL DOS
      ELSE
              WRITE (6,211)
211           FORMAT (' ERROR: illegal command')
      ENDIF
C
      GOTO 10
C
      END
C******************
C     GET LINE
C******************
C
      SUBROUTINE GETL(ID)
C
      INCLUDE 'DECLARE.INC'
      INCLUDE 'PARA.INC'
C
      IN     = ID
      LPOINT = 0
      READ (ID,100)LINE
100   FORMAT (A80)
C
      RETURN
      END
C
C*****************
C    GET WORD
C*****************
C
```

```
      SUBROUTINE GETW
C
      INCLUDE 'DECLARE.INC'
      INCLUDE 'PARA.INC'
C
      CALL SKIPSP
      W = '          '
      N = 1
C
10    CALL GETCHR(C)
C
      IF ( C .EQ. '#' ) THEN
        RETURN
      ELSEIF (C .NE. ' ' .AND. C .NE. '=') THEN
        W(N:N) = C
        N = N + 1
        GOTO 10
      ELSE
        RETURN
      ENDIF
C
      END
C
C*************************
C   SKIP SPACE CHARACTER
C*************************
C
      SUBROUTINE SKIPSP
C
      INCLUDE 'DECLARE.INC'
      INCLUDE 'PARA.INC'
C
10    CALL GETCHR(C)
C
      IF (C .EQ. '#') THEN
        GOTO 20
      ELSEIF ( C .NE. ' ' .AND. C .NE. '[' ) THEN
        LPOINT = LPOINT - 1
      ELSE
        GOTO 10
      ENDIF
C
20    CONTINUE
      RETURN
      END
C
C***********************
C    GET CHARACTER
C***********************
C
      SUBROUTINE GETCHR(C)
C
      INCLUDE 'DECLARE.INC'
      INCLUDE 'PARA.INC'
C
```

```
      LPOINT = LPOINT + 1
C
      IF ( LPOINT .GT. 80) THEN
        C = '#'
        RETURN
      ELSE
        C = LINE(LPOINT:LPOINT)
      CALL  UPPERCON(C)
        IF ( C .EQ. '&') CALL GETL(IN)
      ENDIF
C
      RETURN
      END
C*****************************
C  FUNCTION CHARACTER TYPE
C*****************************
      INTEGER FUNCTION CHRTYP(C)
C
      INCLUDE 'DECLARE.INC'
      INCLUDE 'PARA.INC'
C
      IF (C .GE. 'A' .AND. C .LE. 'Z') THEN
        CHRTYP = LETTER
      ELSEIF (C .GE. 'a' .AND. C .LE. 'z') THEN
        CHRTYP = LETTER
      ELSEIF (C .GE. '0' .AND. C .LE. '9') THEN
        CHRTYP = NUMBER
      ELSE
        CHRTYP = OTHER
      ENDIF
C
      RETURN
      END
C*******************************
C   FUNCTION ASCII TO INTEGER
C*******************************
      INTEGER FUNCTION NTA(C)
C
      INCLUDE 'DECLARE.INC'
      INCLUDE 'PARA.INC'
C
      NTA = ICHAR(C) - ICHAR('0')
C
      RETURN
      END
C*****************
C   GET INTEGER
C*****************
      SUBROUTINE GETINT(I)
      INTEGER   NTA,    CHRTYP
C
      INCLUDE 'DECLARE.INC'
      INCLUDE 'PARA.INC'
C
10    CALL GETCHR(C)
```

```
      IF ( C .EQ. '#' ) THEN
            I   = 0
            RETURN
      ENDIF
      IF (CHRTYP(C) .NE. NUMBER) GOTO 10
      I = NTA(C)
C
20    CALL GETCHR(C)
C
      IF (CHRTYP(C) .NE. NUMBER ) THEN
        LPOINT = LPOINT - 1
        RETURN
      ELSE
        I = 10*I + NTA(C)
        GOTO 20
      ENDIF
C
      END
C*************
C   GET REAL
C*************
      SUBROUTINE GETR(R)
      INTEGER   MANTIS, EXP,    SIG,    NTA,    CHRTYP,
     :          SIM,    SIE
      REAL      R
C
      INCLUDE   'DECLARE.INC'
      INCLUDE   'PARA.INC'
C
      MANTIS    = 0
      EXP       = 0
C
5     CALL      GETCHR(C)
      IF (C .EQ. ' ' .OR. C .EQ. '&' .OR. C .EQ. '=') GOTO 5
      SIM = SIG(C)
C
10    CALL      GETCHR(C)
C
      IF (CHRTYP(C) .NE. NUMBER) THEN
        LPOINT = LPOINT - 1
        GOTO 20
      ELSE
        MANTIS = MANTIS*10 + NTA(C)
        GOTO 10
      ENDIF
C
20    CALL      GETCHR(C)
      IF (C .NE. '.') THEN
        LPOINT = LPOINT -1
        GOTO 40
      ENDIF
C
30    CALL      GETCHR(C)
      IF (CHRTYP(C) .NE. NUMBER ) THEN
        LPOINT = LPOINT - 1
```

```
          GOTO 40
        ELSE
          MANTIS = 10*MANTIS + NTA(C)
          EXP = EXP - 1
          GOTO 30
        ENDIF
C
40      CALL GETCHR(C)
        IF (CHRTYP(C) .NE. LETTER) GOTO 50
C
        IF (C .EQ. 'p' .OR. C .EQ. 'P') THEN
          EXP = EXP - 12
        ELSEIF (C .EQ. 'n' .OR. C .EQ. 'N') THEN
          EXP = EXP - 9
        ELSEIF (C .EQ. 'u' .OR. C .EQ. 'U') THEN
          EXP = EXP - 6
        ELSEIF (C .EQ. 'm') THEN
          EXP = EXP - 3
        ELSEIF (C .EQ. 'k' .OR. C .EQ. 'K') THEN
          EXP = EXP + 3
        ELSEIF (C .EQ. 'M') THEN
          EXP = EXP + 6
        ELSEIF (C .EQ. 'g' .OR. C .EQ. 'G') THEN
          EXP = EXP + 9
        ELSEIF (C .EQ. 'e' .OR. C .EQ. 'E') THEN
45        CALL    GETCHR(C)
          IF (C .EQ. ' ') GOTO 45
          SIE = SIG(C)
          CALL    GETINT(I)
          I = I*SIE
          EXP = EXP + I
        ELSE
          GOTO 40
        ENDIF
C
        GOTO 40
50      LPOINT = LPOINT - 1
        R = SIM*MANTIS*(10.**EXP)
        RETURN
        END
C************************
C    SIGN FUNCTION
C************************
        INTEGER FUNCTION SIG(C)
C
        INCLUDE    'DECLARE.INC'
        INCLUDE    'PARA.INC'
C
        IF (C .EQ. '-') THEN
          SIG      = -1
        ELSEIF (C .EQ. '+') THEN
          SIG      = +1
        ELSE
          SIG      = +1
          LPOINT   = LPOINT - 1
```

```
      ENDIF
C
      RETURN
      END
C************************
C   scan mpot array
C************************
      SUBROUTINE SMPOT(J,X)
      INCLUDE   'DECLARE.INC'
      INTEGER   X
      DO 10 J = 1,60
            IF (MPOT(J) .EQ. X) GOTO 20
10    CONTINUE
20    CONTINUE
      RETURN
      END
C*********************
C  SIZE OF MATRIX
C*********************
      SUBROUTINE SIZE(KID)
      INCLUDE   'DECLARE.INC'
      DO 10 J=1,60
            IF (LOC(J) .EQ. 0) THEN
                  KID = J-1
                  RETURN
            ENDIF
10    CONTINUE
      WRITE(6,100)
100   FORMAT( ' ERROR:too much nodes')
      RETURN
      END
C***************************************
C  LOWER CASE CONVERT TO UPPER CASE
C***************************************
      SUBROUTINE  UPPERCON(C)
      CHARACTER*1  C
      INTEGER*2    IC
      IF (C .EQ. 'm' ) THEN
         RETURN
      ELSEIF (C .GE. 'a' .AND. C .LE. 'z') THEN
         IC  = ICHAR(C) - 32
         C   = CHAR(IC)
         RETURN
      ENDIF
      END

C*********************
C   INPUT ROUTINE
C*********************
      SUBROUTINE INPUT
C
      INCLUDE   'DECLARE.INC'
      INCLUDE   'PARA.INC'
      CHARACTER*12  FNAME
      INTEGER*2     LS,LZ
```

```
C
C----type 'END' to terminate INPUT routine
C
      WRITE(6,117)
117   FORMAT (  '------------------')
      WRITE(6,112)
112   FORMAT ('        INPUT       ')
      WRITE (6,117)
C
      WRITE (6,1)
1     FORMAT (' Input from an input file(YES/NO):',:)
      READ (5,3) W
3     FORMAT(A)
      IF ( W(1:1) .EQ. 'Y' .OR. W(1:1) .EQ. 'y') THEN
            IN    = 8
2           WRITE (6,4)
4           FORMAT (' Input file name(.SPC):',:)
            READ (6,3) FNAME
C:
C--: convert file name to upper case :--
C:
            DO 17 I=1,LEN(FNAME)
               CALL UPPERCON(FNAME(I:1))
17          CONTINUE
C:
            LS = INDEX(FNAME,'SPC')
            LZ = INDEX(FNAME,' ')
C
           IF ( LS .EQ. 0) THEN
              FNAME(LZ:LZ+4) = '.SPC'
           ENDIF
C
           OPEN (UNIT=8,STATUS='OLD',FILE=FNAME,ERR=99)
      ELSE
           IN    = 5
           WRITE (6,111)
111        FORMAT (' Start to input circuit:')
           WRITE (6,118)
118        FORMAT (' ',:)
      ENDIF
C
      CALL     NEW
C
10    CALL   GETL(IN)
      CALL   GETW
C
      IF (W .EQ. 'END      ') THEN
        DO 123 JKK = 1,200
           BUF2(JKK) = VAL(JKK)
123     CONTINUE
        CLOSE (UNIT=8)
        RETURN
      ELSEIF (W(1:1) .EQ. 'R') THEN
        CALL EDRI(RH)
      ELSEIF (W(1:1) .EQ. 'V') THEN
```

```
          IF (W(1:3) .EQ. 'VDC') CALL EDLCV(VDCH)
          IF (W(1:3) .EQ. 'VAC') CALL EDVAC(VACH)
          IF (W(1:6) .EQ. 'VPLUSE') CALL EDVP(VPH)
          IF (W(1:4) .EQ. 'VCVS') CALL EDVCVS(VCVH)
          IF (W(1:4) .EQ. 'VCCS') CALL EDVCCS(VCCH)
          IF (W(1:3) .EQ. 'VIN' ) CALL EDVAC(VACH)
        ELSEIF (W(1:1) .EQ. 'I') THEN
          IF (W(1:3) .EQ. 'IDC') CALL EDRI(IDCH)
          IF (W(1:3) .EQ. 'IAC') CALL EDIAC(1ACH)
C         IF (W(1:6) .EQ. 'IPLUSE') CALL EDIP(IPH)
C         IF (W(1:4) .EQ. 'ICVS') CALL EDICVS(ICVH)
C         IF (W(1:4) .EQ. 'ICCS') CALL EDICCS(ICCH)
        ELSEIF (W(1:1) .EQ. 'C') THEN
          CALL EDC(CH)
        ELSEIF (W(1:1) .EQ. 'L') THEN
          CALL EDLCV(LH)
        ELSEIF (W(1:1) .EQ. 'D') THEN
          CALL EDD(DH)
        ELSEIF (W(1:1) .EQ. 'Q') THEN
          CALL EDQ(QH)
        ELSEIF (W(1:5) .EQ. 'OPAMP') THEN
          CALL EDO(OH)
        ELSEIF (W(1:5) .EQ. 'MODEL') THEN
          CALL EDM
        ELSEIF (W(1:1) .EQ. '#') THEN
          CONTINUE
        ELSE
          WRITE (6,108)
 108     FORMAT(' ERROR:element type syntax error')
        ENDIF
C
        GOTO 10
C
 99     WRITE (6,109) FNAME
 109    FORMAT (' Error opening file ',A)
        GOTO 2
        END
C**********
C   NEW
C**********
        SUBROUTINE NEW
C
        INCLUDE   'DECLARE.INC'
        INCLUDE   'PARA.INC'
C
        DO 10 I=1,500
          POIT(I) = 0
 10     CONTINUE
        DO 20 I=1,60
          LOC(I) = 0
          MPOT(I) = 0
          MNAM(I) = '        '
 20     CONTINUE
        DO 25 I = 0,200
          VAL(I) = 0.0
```

```
25     CONTINUE
C
       RH          = 1
       VDCH        = 2
       VACH        = 3
       VPH         = 4
       IDCH        = 5
       IACH        = 6
       IPH         = 7
       CH          = 8
       LH          = 9
       VCVH        = 10
       VCCH        = 11
       ICVH        = 12
       ICCH        = 13
       DH          = 14
       QH          = 15
       OH          = 16
       IP          = 20
       MP          = 1
       IVAL        = 1
       MDEX        = 200
       QPID        = .FALSE.
C
       RETURN
       END


C*************************
C   operating point analysis
C*************************
C
C        ;mode=9          :d.c.operating point analysis
C        ;mode=2          :transfer characteristic analysis
C        ;mode=3          :frequency response analysis
C        ;mode=4          :transient response analysis
C
       SUBROUTINE QPOINT
       INCLUDE   'DECLARE.INC'
       INCLUDE   'DOUBLE.INC'
       INCLUDE   'RES.INC'
       INCLUDE   'PARA.INC'
       LOGICAL*1 FLAG
       INTEGER*2 KK,MODE
C--:
       OPEN (UNIT=1,FILE='C:QDP.DAT',STATUS='UNKNOWN',
      :        FORM='UNFORMATTED',ACCESS='SEQUENTIAL')
c--:
       NOPER = 0
       MODE    = 9
       H       = 1.0E+22
c--:
       WRITE (6,1)
1      FORMAT (' +----------------------------+')
       WRITE (6,2)
```

```
2     FORMAT (' :  OPERATING POINT ANALYSIS  :')
      WRITE (6,1)
      WRITE (6,10)
10    FORMAT (' input ambient temperature(Celcious)=',:)
      CALL      GETL(5)
      CALL      GETR(R)
      TEM1      = R
C-------calculate Vt-------
      VT        = 1.38E-23*(TEM1+273.0)/1.6E-19
      CALL      QPINIT
C--:
      CALL      SIZE(N)
C--:
      DO 20 I = 1,200
         BUF1(I)     = VAL(I)
20    CONTINUE
C--:
      KK  = 0
100   DO 111 I=1,N
            DO 112 J=1,N
                  A(I,J)          = 0.0
112         CONTINUE
                  B(I)            = 0.0
111   CONTINUE


      CALL      LOADDC(H,MODE)
      CALL      LU(N)
      NOPER = NOPER + 1
      IF (POIT(14) .EQ. 0 .AND. POIT(15) .EQ. 0
     :    .AND. POIT(16) .EQ. 0) THEN
            DO 91 I = 1,60
                S(I)    = B(I)
91          CONTINUE
            REWIND (UNIT=1)
            WRITE (UNIT=1) (SNGL(S(I)),I=1,N)
            QPID        = .TRUE.
            CLOSE (UNIT =1)
            RETURN
      ELSE
            CALL STORE(FLAG,H)
            IF (KK .EQ. 0) THEN
                KK    = KK + 1
            ELSEIF (.NOT. FLAG) THEN
                  KK    = KK + 1
                  DO 200 I = 1,N
                        IF(DABS(S(I)-B(I)) .GT. EPS) GOTO 205
200               CONTINUE
                  DO 220 I = 1,60
                       S(I)    = B(I)
220               CONTINUE
                  REWIND(UNIT=1)
                  WRITE (UNIT=1) (SNGL(S(I)),I=1,N)
                  QPID  = .TRUE.
                  WRITE (6,225) KK
```

```
225                FORMAT(' CONVERGE IN',I3)
                   CLOSE (UNIT = 1)
                   RETURN
             ENDIF
205          DO 110 I=1,N
                   S(I) = B(I)
110          CONTINUE
             IF ( KK .GT. 30) THEN
                   WRITE (6,247)
247                FORMAT (' NOT CONVERGE IN 30 ITERATIONS')
                   RETURN
             ENDIF
C--:
C  : check if not converge in 50 calculated loops :--
C--:
      IF ( NOPER .GE. 20) THEN
         WRITE (*,*) ' Circuit is too complicated '
         CALL STEP(0.0,0,'F')
         NOPER          = 0
      ENDIF
C--:
            GOTO 100
      ENDIF
      RETURN
      END
C*****************************************
C   step source operating point analysis
C*****************************************
      SUBROUTINE STEP(STAR,NODE,TM)
      INCLUDE    'DECLARE.INC'
      INCLUDE    'DOUBLE.INC'
      INCLUDE    'RES.INC'
      INCLUDE    'PARA.INC'
      LOGICAL*1 FLAG
      INTEGER*2 KK,MODE,NODE
      CHARACTER*1            TM
      REAL*4     STAR
      KK                  = 0
c--:
C  : scan for supply voltage source :
c--:
      IDX  = POIT(2)
      NUS  = 0
C--:
1     IF (IDX .EQ. 0) GOTO 10
         NUS  = NUS + 1
         INTTEMP(NUS) = IDX + 4
         REALTEMP(NUS)= VAL(POIT(IDX+4))
         IDX          = POIT(IDX)
      GOTO 1
C--:
C10    DO 20 I = 1,200
C         VAL(I) = BUF1(I)
C20    CONTINUE
C--:
```

```
10     CALL RESET
C--:
       CALL SIZE(N)
C--:
       DO 1000 LOOP = 20,1,-1
          INTTEMP(12)       = 0
          KK = 0
C--:
          DO 30 I = 1,NUS
              VAL(POIT(INTTEMP(I))) = REALTEMP(I) / LOOP
30        CONTINUE
C--:
100      DO 111 I=1,N
             DO 112 J=1,N
                   A(I,J)           = 0.0
112          CONTINUE
                   B(I)             = 0.0
111          CONTINUE
          MODE     = 1
          H        = 1.0E+22
          IF (TM .EQ. 'F') THEN
             MODE = 9
          ENDIF
          CALL       LOADDC(H,MODE)
          IF (TM .EQ. 'T') THEN
             B(NODE) = DBLE(STAR)
          ENDIF
          IF (TM .EQ. 'M' ) THEN
             B(NODE) = 100.0*(B(NODE))
          ENDIF
          TM = 'T'
          CALL       LU(N)
          INTTEMP(12) = INTTEMP(12) + 1
C--:
          CALL STORE(FLAG,H)
          IF (KK .EQ. 0) THEN
             KK     = KK + 1
          ELSEIF(.NOT. FLAG) THEN
             KK     = KK + 1
             DO 200 I = 1,N
                IF(DABS(S(I)-B(I)) .GT. EPS) GOTO 205
200          CONTINUE
             DO 110 I = 1,N
                 S(I) = B(I)
110          CONTINUE
             GOTO 900
          ELSE
              GOTO 205
          ENDIF
C--:
C--: check if not converge in 50 calculated loops :--
C--:
205      DO 225 J = 1,N
            S(J)  = B(J)
225      CONTINUE
```

```
          IF ( INTTEMP(12) .EQ. 60 ) THEN
C              TM       = 'M'
               CALL QPINIT
          ENDIF
          IF ( INTTEMP(12) .GE. 100) THEN
               WRITE (*,*) ' Circuit is too complicated '
               CALL CLSCREEN
               WRITE (*,*) ' No Convergence: Modify circuit parameters
               STOP
          ENDIF
          write(*,*) 'inttemp',inttemp(12)
        GOTO 100
C--:
900     CONTINUE
1000    CONTINUE
        write (*,*) (b(i),i=1,n)
        RETURN
        END
C--:
C   : reset junction voltage :
C--:
        SUBROUTINE RESET
        INCLUDE 'DECLARE.INC'
        INCLUDE 'PARA.INC'
        DO 100 I = 1,200
           VAL(I)     = BUF2(I)
100     CONTINUE
        RETURN
        END
C--:
C   : FIND INITIAL VOLTAGE CONDITION :
C--:
        SUBROUTINE QPINIT
        INCLUDE    'DECLARE.INC'
        INCLUDE    'DOUBLE.INC'
        INCLUDE    'RES.INC'
        INCLUDE    'PARA.INC'
        LOGICAL*1 FLAG
        INTEGER*2 KK,MODE
        CALL RESET
        KK                  = 0
c--:
C   : scan for supply voltage source :
c--:
        IDX   = POIT(2)
        NUS   = 0
C--:
1       IF (IDX .EQ. 0) GOTO 10
           NUS   = NUS + 1
           INTTEMP(NUS) = IDX + 4
           REALTEMP(NUS)= VAL(POIT(IDX+4))
           IF (REALTEMP(NUS) .LT. 0.0) SIGN = -1.0
           IF (REALTEMP(NUS) .GT. 0.0) SIGN = +1.0
           VAL(POIT(IDX+4)) = SIGN* 1.50
           IDX              = POIT(IDX)
```

```
      GOTO 1
C--:
10    CALL SIZE(N)
      INTTEMP(12) = 0
      MODE     = 9
      H        = 1.0E+22
C--:
100   DO 111 I=1,N
            DO 112 J=1,N
                  A(I,J)          = 0.0
112         CONTINUE
                  B(I)            = 0.0
111   CONTINUE
      CALL       LOADDC(H,MODE)
      CALL       LU(N)
      INTTEMP(12) = INTTEMP(12) + 1
C--:
      CALL STORE(FLAG,H)
      IF (KK .EQ. 0) THEN
             KK     = KK + 1
      ELSEIF(.NOT. FLAG) THEN
             KK     = KK + 1
             DO 200 I = 1,N
                IF(DABS(S(I)-B(I)) .GT. EPS) GOTO 205
200          CONTINUE
             DO 110 I = 1,N
                 S(I) = B(I)
110          CONTINUE
             GOTO 900
      ELSE
             GOTO 205
      ENDIF
C--:
C--: check if not converge in 60 calculated loops :--
C--:
205   DO 225 J = 1,N
          S(J)  = B(J)
225   CONTINUE
      IF ( INTTEMP(12) .GE. 60) THEN
             WRITE (*,*) ' No Convergence: Modify circuit parameter
             STOP
        ENDIF
      GOTO 100
C--:
900   CONTINUE
      DO 910 I=1,NUS
         VAL(POIT(INTTEMP(I))) = REALTEMP(I)
910   CONTINUE
      RETURN
      END

C*************************************************
C   TRANSFER RESPONSE CHARACTERISTIC ANALYSIS
C*************************************************
      SUBROUTINE TRANF
```

```
      REAL       STAR,    STP,     STS
      INCLUDE    'DECLARE.INC'
      INCLUDE    'DOUBLE.INC'
      INCLUDE    'PARA.INC'
      INCLUDE    'RES.INC'
      LOGICAL*1  FLAG
      INTEGER*2  KKK,      MODE
C--:
      OPEN (UNIT=2,FILE='C:TRANF.DAT',STATUS='UNKNOWN',
     :         FORM='UNFORMATTED',ACCESS='SEQUENTIAL')
C--:
      QPID       = .FALSE.
      MODE       = 1
      H = 1.0E+22
      WRITE (6,*) ' +-------------------------------------+'
      WRITE (6,*) ' :  TRANSFER CHARACTERISTIC RESPONSE   :'
      WRITE (6,*) ' +-------------------------------------+'
      WRITE (6,11)
11    FORMAT (' Input ambient temperature(celcious)=',:)
      CALL  GETL(5)
      CALL  GETR(R)
      TEM1       = R
C-------calculate Vt---------
      VT         = 1.38E-23*(TEM1+273.0)/1.6E-19
1     WRITE (6,10)
10    FORMAT (' Sweep:',:)
C--:!name start stop step
      CALL GETL(5)
      CALL GETW
      CALL GETR(R)
      STAR       = R
      CALL GETR(R)
      STP        = R
      CALL GETR(R)
      STS        = R
      NUM        = NINT(ABS(STAR-STP)/STS)
C--:
      IF (NUM .LT. 0) NUM = -NUM
C--:
      IF (NUM .GT. 70) THEN
            WRITE (6,*)' Sweeping point exceed than 70 points:try ag

            GOTO 1
      ENDIF
      DO 100 I = 1,60
            IF (MNAM(I) .EQ. W) GOTO 110
100   CONTINUE
110   IDX        = MPOT(I)
      CALL SIZE(N)
      IBB         = POIT(IDX+3)
      DO 200 I = 1,NUM+1
            KKK  = 0
            INTTEMP(30) = 0
115         DO 140 II=1,N
                  DO 150 JJ=1,N
                        A(II,JJ)         = 0.0
```

```
150               CONTINUE
                        B(II)           = 0.0
140           CONTINUE
              CALL LOADDC(H,MODE)
              B(IBB)        = DBLE(STAR)
              CALL LU(N)
              INTTEMP(30) = INTTEMP(30) + 1
              IF (POIT(14) .EQ. 0 .AND. POIT(15) .EQ. 0
     :        .AND. POIT(16) .EQ. 0) THEN
                    DO 91 JJ=1,60
                          SOL(JJ,I)       = SNGL(B(JJ))
91                  CONTINUE
              ELSE
                    CALL STORE(FLAG,H)
                    IF (KKK .EQ. 0) THEN
                          KKK       = KKK + 1
                    ELSEIF (.NOT. FLAG) THEN
                          KKK       = KKK + 1
                          DO 120 JJ = 1,N
                                IF(DABS(S(JJ)-B(JJ)) .GT. EPS) GOTO 20
120                       CONTINUE
                          DO 220 JJ = 1,N
                                SOL(JJ,I) = SNGL(B(JJ))
                                S(JJ)     = B(JJ)
220                       CONTINUE
                          GOTO 178
                    ENDIF
205                 DO 130 J = 1,N
                       S(J) = B(J)
130                 CONTINUE
                    IF (INTTEMP(30) .GE. 20) THEN
                        CALL STEP(STAR,IBB,'T')
                        INTTEMP(30) = 0
                        KKK         = 0
                    ENDIF
                    IF ( KKK .GT. 25) THEN
                        WRITE (6,*) 'NOT CONVERGE IN 25 ITERATIONS'
                        RETURN
                    ENDIF
                    GOTO 115
              ENDIF
178           ST(I,1)       = STAR
              STAR          = STAR + STS
200     CONTINUE
        REWIND (UNIT=2)
        WRITE (UNIT=2) N,NUM,STP,(ST(I,1),I=1,NUM)
          DO 202 I=1,N
            DO 201 J=1,70
                WRITE (UNIT=2) SOL(I,J)
201         CONTINUE
202       CONTINUE
        CLOSE (UNIT = 2)
        RETURN
        END
C*********************
```

```
C   TRANSIENT ANALYSIS
C*************************
      SUBROUTINE TRANSI
      INCLUDE   'DECLARE.INC'
      INCLUDE   'DOUBLE.INC'
      INCLUDE   'PARA.INC'
      INCLUDE   'RES.INC'
      REAL*4    TIS,TIF,LTE
      LOGICAL*1 FLAG
      DOUBLE PRECISION  VD
      INTEGER   K,KK
      QPID      = .FALSE.
C--:
      OPEN (UNIT=3,FILE='C:SIENT.DAT',STATUS='UNKNOWN',
     :        FORM='UNFORMATTED',ACCESS='SEQUENTIAL')
C--:
      WRITE (6,*)   ' +----------------------+'
      WRITE (6,*)   ' !   TRANSIENT ANALYSIS  !'
      WRITE (6,*)   ' +----------------------+'
      CALL SIZE(N)
      WRITE (6,6)
6     FORMAT (     '  Analyze from starting point OR continue on time
     :(START/CONTINUE):',:)
      CALL GETL(5)
      CALL GETW(W)
      IF ( W(1:1) .EQ. 'S'.OR. W(1:1) .EQ. 's') THEN
            WRITE (6,1)
1           FORMAT (' Starting time =',:)
            CALL        GETL(5)
            CALL        GETR(R)
            TIS = R
            WRITE (6,2)
2           FORMAT ('   To(time) =',:)
            CALL        GETL(5)
            CALL        GETR(R)
            TIF = R
            TIME        = TIS
            NSTP        = 1
            WRITE (6,10)
10          FORMAT (' Input ambient temperature(~C)=':)
            CALL GETL(5)
            CALL GETR(R)
            TEM1        = R
C----calculate Vt-----
            VT  = 1.38E-23 * (TEM1 + 273.0)/1.6E-19
            CALL SIZE(N)
            MODE        = 1
            GOTO 209
C--:solve eqaution at t0
11          CALL STORLC
            BTIM(1)     = TIME
            DO 5 I = 1,N
                  SOL(I,1)      = SNGL(B(I))
5           CONTINUE
            TD  = TIME
```

```
              NSTP          = 2
              MODE          = 4
C----input Hstart------
              WRITE (6,19)
19            FORMAT (' Starting Step=',:)
              CALL GETL(5)
              CALL GETR(R)
              H             = R
              HSTA          = R
      ELSEIF(W(1:1) .EQ. 'C' .OR. W(1:1) .EQ. 'c') THEN
      BTIM(1) = BTIM(NSTP-1)
              NSTP          = 2
      MODE = 4
              DO 22 I=1,N
                    SOL(I,1)      = SNGL(B(I))
22            CONTINUE
              WRITE (6,20)
20            FORMAT (' Continue to time=',:)
              CALL GETL(5)
              CALL GETR(R)
              TIF = R
      ENDIF
C
      IDX           = POIT(4)
      IVAL          = POIT(IDX+4)
      PERD          = VAL(IVAL)
      KK            = IVAL
C---process forward euler---
90    IF ( H .EQ. 0.0) H=HSTA
      TIME          = TD
      TIME          = TIME - (IFIX(TIME/VAL(KK))*VAL(KK))
      CALL          CHSTP(TIME,H)
      TIME          = TD + H
      TIME          = TIME - (IFIX(TIME/VAL(KK))*VAL(KK))
c--------capacitor integrated--
      IDX           = POIT(8)
100   IF (IDX .EQ. 0) GOTO 200
      IVAL          = POIT(IDX+3)
      XN            = VAL(IVAL+1)
      DX            = VAL(IVAL+2)
      VAL(IVAL+3)= XN + H*DX
C--:save F.E. result
      IDX           = POIT(IDX)
      GOTO 100
C----inductor integrated---
200   IDX           = POIT(9)
201   IF (IDX .EQ. 0) GOTO 210
      IB            = POIT(IDX+3)
      IVAL          = POIT(IDX+4)
      XN            = VAL(IVAL+2)
      DX            = VAL(IVAL+1)/VAL(IVAL)
      VAL(IVAL+3) = XN + H*DX
C--:save F.E. result
      IDX           = POIT(IDX)
      GOTO 201
```

```
C----opamp  integrated----
210   IDX       = POIT(16)
211   IF (IDX .EQ. 0) GOTO 220
      IVAL      = POIT(IDX+6)
      MDEX      = MPOT(POIT(IDX+5)) - 1000
      XN        = VAL(IVAL+1)
      DX        = VAL(IVAL+2)/VAL(MDEX-2)
      VAL(IVAL) = XN + H*DX
      IDX       = POIT(IDX)
      GOTO 211
C----diode integrated-----
220   IDX       = POIT(14)
221   IF ( IDX .EQ. 0) GOTO 230
      IVAL      = POIT(IDX + 4)
      XN        = VAL(IVAL+1)
      DX        = VAL(IVAL+2)
      VAL(IVAL+3)       = XN + H*DX
      IDX       = POIT(IDX)
      GOTO 221
C---transistor integrated----
230   IDX       = POIT(15)
231   IF (IDX .EQ. 0) GOTO 209
      IVAL      = POIT(IDX + 5)
      XN        = VAL(IVAL+2)
      DX        = VAL(IVAL+4)
      VAL(IVAL+6)       = XN + H*DX
      XN        = VAL(IVAL+3)
      DX        = VAL(IVAL+5)
      VAL(IVAL+7)       = XN + H*DX
      IDX       = POIT(IDX)
      GOTO 231
C----process BACKWARD EULER-------
209   K         = 0
      INTTEMP(31) = 0
C--:
300   DO 311 I=1,N
            DO 312 J=1,N
               A(I,J)       = 0.0
312         CONTINUE
               B(I)  =0.0
311   CONTINUE
      IF (MODE .EQ. 1) H=1.0E+22
      IF (H .EQ. 0.0) H=1.0E-25
      CALL LOADDC(H,MODE)
      CALL LU(N)
      INTTEMP(31) = INTTEMP(31) + 1
C----check nonlinear device----
      IF (POIT(14) .EQ. 0 .AND. POIT(15) .EQ. 0
     :    .AND. POIT(16) .EQ. 0 ) THEN
            GOTO 500
      ELSE
C-----------check convergence------
            CALL STORE(FLAG,H)
            IF ( K .EQ. 0) THEN
               K     = K+1
```

```
            ELSEIF (.NOT. FLAG) THEN
                  K      = K+1
                  DO 400 I=1,N
                        IF (DABS(S(I) - B(I)) .GT. EPS) GOTO 405
400               CONTINUE
                  GOTO 500
            ENDIF
405         DO 410 I=1,N
                  S(I)   = B(I)
410         CONTINUE
            IF (K .GT. 50) THEN
                WRITE (6,*) ' not converge in 50 iterations'
                RETURN
            ENDIF
            IF (INTTEMP(31) .GE. 20) THEN
               CALL STEP(0.0,0,'F')
               INTTEMP(31) = 0
               K           = 0
            ENDIF
C--:
            GOTO 300
C--:
      ENDIF
500   IF (NSTP .EQ. 1) GOTO 11
C-----find maximum LTE local truncation error----
      RMAX       = 0.0
      IDX        = POIT(8)
600   IF (IDX .EQ. 0) GOTO 610
      IROW       = POIT(IDX+1)
      JCOL       = POIT(IDX+2)
      IVAL       = POIT(IDX+3)
      IF ( IROW .EQ. 0 ) THEN
            TEM1         = -B(JCOL)
      ELSEIF (JCOL .EQ. 0 ) THEN
            TEM1         = B(IROW)
      ELSE
            TEM1         = B(IROW) - B(JCOL)
      ENDIF
      LTE        = 0.5 * (ABS(VAL(IVAL+3) - TEM1))
      IF (LTE .GT. RMAX) RMAX = LTE
      IDX        = POIT(IDX)
      GOTO 600
610   IDX        = POIT(9)
620   IF (IDX .EQ. 0) GOTO 630
      IB         = POIT(IDX+3)
      IVAL       = POIT(IDX+4)
      LTE        = 0.5*(ABS(VAL(IVAL+3) - B(IB)))
      IF (LTE .GT. RMAX) RMAX = LTE
      IDX        = POIT(IDX)
      GOTO 620
630   IDX        = POIT(16)
640   IF (IDX .EQ. 0) GOTO 650
      IROW       = POIT(IDX+1)
      JCOL       = POIT(IDX+2)
      IVAL       = POIT(IDX+6)
```

```
      MDEX       = MPOT(POIT(IDX+5)) -1000
      IF ( IROW .EQ. 0 ) THEN
            TEM1         = -B(JCOL)
      ELSEIF (JCOL .EQ. 0 ) THEN
            TEM1         = B(IROW)
      ELSE
            TEM1         = B(IROW) - B(JCOL)
      ENDIF
      LTE        = 0.5*(ABS(VAL(IVAL)-TEM1))
      IF (VAL(MDEX-2) .EQ. 0.0) LTE=0.0
      IF (LTE .GT. RMAX) RMAX = LTE
      IDX        = POIT(IDX)
      GOTO 640
650   IDX        = POIT(14)
651   IF ( IDX .EQ. 0 ) GOTO 660
      IROW       = POIT(IDX+1)
      JCOL       = POIT(IDX+2)
      IVAL       = POIT(IDX+4)
      MDEX       = MPOT(POIT(IDX+3)) - 2000
      IF (JCOL .EQ. 0) THEN
            TEM1         = B(IROW)
      ELSEIF (IROW .EQ. 0) THEN
            TEM1         = -B(JCOL)
      ELSE
            TEM1         = B(IROW) - B(JCOL)
      ENDIF
      LTE        = 0.5*(ABS(VAL(IVAL+3) - TEM1))
      IF (VAL(MDEX-2) .EQ. 0.0) LTE=0.0
      IF (LTE .GT. RMAX) RMAX=LTE
      IDX        = POIT(IDX)
      GOTO 651
660   IDX        = POIT(15)
661   IF (IDX .EQ. 0) GOTO 700
      IROW       = POIT(IDX+1)
      JCOL       = POIT(IDX+2)
      IB         = POIT(IDX+3)
      MDEX       = MPOT(POIT(IDX+4)) - 3000
      IVAL       = POIT(IDX+5)
      TYPE       = VAL(MDEX-6)
C--:lte from BE junction capacitance:--
      IF (IROW .EQ. 0) THEN
            TEM1         = -B(IB)
      ELSEIF (IB .EQ. 0) THEN
            TEM1         = B(IROW)
      ELSE
            TEM1         = B(IROW) - B(IB)
      ENDIF
c      IF (TYPE .EQ. -2.0) TEM1 = -TEM1
      LTE        = 0.5*(ABS(VAL(IVAL+6) - TEM1))
      IF (VAL(MDEX-1) .EQ. 0.0) LTE = 0.0
      IF (LTE .GT. RMAX) RMAX = LTE
C--:lte from BC junction capacitance:--
      IF (IROW .EQ. 0) THEN
            TEM1         = -B(JCOL)
      ELSEIF (JCOL .EQ. 0) THEN
```

```
            TEM1          = B(IROW)
      ELSE
            TEM1          = B(IROW) - B(JCOL)
      ENDIF
c      IF (TYPE .EQ. -2.0) TEM1 = -TEM1
      LTE         = 0.5*(ABS(VAL(IVAL+7) - TEM1))
      IF (VAL(MDEX) .EQ. 0.0) LTE = 0.0
      IF (LTE .GT. RMAX) RMAX = LTE
C--:
      IDX         = POIT(IDX)
      GOTO 661
C----compute new step size-----
700   LTE         = RMAX
      IF (LTE .EQ. 0.0) LTE=1.0E-10
      RMAX        = LTE/(H*H)
      IF (LTE .LE. 0.001) THEN
            TIME          = TD + H
            TD            = TIME
            BTIM(NSTP)    = TIME
            DO 710 I = 1,N
                  SOL(I,NSTP)    = SNGL(B(I))
710         CONTINUE
            CALL STORLC
            IF ( LTE .EQ. 0.0) THEN
                  H       = H * 2.0
            ELSE
                  H       = 0.9*SQRT(0.001/RMAX)
            ENDIF
            IF ( H .GT. PERD/5.) H = PERD/15.
      ELSE
            H     = 0.9*SQRT(0.001/RMAX)
            GOTO 90
      ENDIF
      NSTP        = NSTP + 1
D      write(6,*)nstp-1,time,h
D      write (6,*)(sol(mm,nstp-1),mm=1,6)
      IF (TD.GT. TIF) THEN
            CONTINUE
      ELSEIF (NSTP .EQ. 71) THEN
           WRITE (6,*)  ' Exceed then 71 points'
      ELSE
           GOTO 90
      ENDIF
      WRITE (6,800) TD
800   FORMAT('  Analyzed to time=',E15.5)
C      WRITE (6,*) CHAR(7)
      WRITE (6,*) ' ************************************************'
      WRITE (6,*) ' *   Caution: output the    RESULTS   before   *'
      WRITE (6,*) ' *     continue on transient analysis           *'
      WRITE (6,*) ' ************************************************'
      REWIND (UNIT=3)
      WRITE (UNIT=3) N,(BTIM(I),I=1,70)
      DO 999 I=1,N
      DO 998 J=1,70
         WRITE (UNIT=3) SOL(I,J)
```

```
998   CONTINUE
999   CONTINUE
      DATATEMP(1) = BTIM(1)
      DATATEMP(2) = BTIM(NSTP-1)
C--:
      CLOSE (UNIT = 3)
C--:
      RETURN
      END
C************************************
C  CHECK STEP SIZE TO CORNER POINT *
C************************************
      SUBROUTINE CHSTP(TID,STEP)
      INCLUDE   'DECLARE.INC'
      INCLUDE   'DOUBLE.INC'
      REAL*4 TF,TID,STEP
      IDX       = POIT(4)
      IB        = POIT(IDX+3)
      IVAL      = POIT(IDX+4) + 1
100   IVAL      = IVAL+2
      IF (TID .GT. VAL(IVAL)) GOTO 100
      TF        = VAL(IVAL)
      IF ( TID .EQ. TF) THEN
            STEP        = STEP/2.0
      ELSEIF ((TID+STEP) .GT. TF) THEN
            STEP        = TF - TID
      ENDIF
C      IF ((TID+STEP) .GT. TF) STEP=TF-TID
C      IF (TF .EQ. TID) STEP=1.E-5
      RETURN
      END
C*********************************
C   FIND VOLTAGE FOR INPUT VPLUSE
C*********************************
      SUBROUTINE FINDV
      INCLUDE   'DECLARE.INC'
      INCLUDE   'DOUBLE.INC'
      INCLUDE   'PARA.INC'
      REAL*4    TS,TF,VS,VG
      IDX       = POIT(4)
      IB        = POIT(IDX+3)
      IVAL      = POIT(IDX+4) + 1
100   IVAL      = IVAL+2
      IF(TIME .GT. VAL(IVAL)) GOTO 100
      TS        = VAL(IVAL-2)
      TF        = VAL(IVAL)
      VS        = VAL(IVAL-1)
      VF        = VAL(IVAL+1)
      IF (VS .EQ. VF) THEN
            IF(TIME .EQ. TF) THEN
C                 IF(TF .EQ. VAL(IVAL+2)) THEN
C                       B(IB)   = VAL(IVAL+3)
C                       RETURN
C                 ELSE
                        B(IB)   = VF
```

```
C                       RETURN
C                 ENDIF
            ELSE
                  B(IB) = VS
                  RETURN
            ENDIF
      ELSE
            IF (TS .EQ. TF) THEN
                  B(IB) = VF
                  RETURN
            ELSE
                  IF (VS .GT. VF) THEN
                        B(IB)=DBLE((TF-TIME)*(VS-VF)/(TF-TS)+VF)
                        RETURN
                  ELSE
                        B(IB)=DBLE((TIME-TS)*(VF-VS)/(TF-TS)+VS)
                        RETURN
                  ENDIF
            ENDIF
      ENDIF
      END
C**************************
C   stamp element
C**************************
      SUBROUTINE LOADDC(STEP,MODE)
      INCLUDE   'DECLARE.INC'
      INCLUDE   'DOUBLE.INC'
      INCLUDE   'PARA.INC'
      DOUBLE PRECISION  AA,     BB,     CC,     P1,     P2,     AV,
     :                  VD,     STEP,   GTO,    RS,     CDNO,   V1,
     :                  V2,     CJC,    CJE,    GB,     GC,     GE
C-----
C----load resistor---
      IDX       = POIT(1)
110   IF (IDX .EQ. 0) GOTO 120
      IROW      = POIT(IDX+1)
      JCOL      = POIT(IDX+2)
      G         = DBLE(1/VAL(POIT(IDX+3)))
C
      A(IROW,IROW)      = A(IROW,IROW) + G
      A(IROW,JCOL)      = A(IROW,JCOL) - G
      A(JCOL,IROW)      = A(JCOL,IROW) - G
      A(JCOL,JCOL)      = A(JCOL,JCOL) + G
C
      IDX       = POIT(IDX)
      GOTO 110
C---load d.c. voltage source---
120   IDX       = POIT(2)
121   IF (IDX .EQ. 0) GOTO 130
      IROW      = POIT(IDX+1)
      JCOL      = POIT(IDX+2)
      IB        = POIT(IDX+3)
      A(IROW,IB)        = 1.
      A(JCOL,IB)        = -1.
      A(IB,IROW)        =  1.
```

```
        A(IB,JCOL)          = -1.
        B(IB)       = VAL(POIT(IDX+4))
        IDX         = POIT(IDX)
        GOTO 121
C---stamp d.c. current source---
130     IDX         = POIT(5)
131     IF ( IDX .EQ. 0) GOTO 140
        IROW        = POIT(IDX + 1)
        JCOL        = POIT(IDX + 2)
        B(IROW)     = B(IROW) + VAL(POIT(IDX+3))
        B(JCOL)     = B(JCOL) - VAL(POIT(IDX+3))
        IDX         = POIT(IDX)
        GOTO 131
C---stamp diode---
140     IDX         = POIT(14)
141     IF (IDX .EQ. 0) GOTO 150
        IROW        = POIT(IDX+1)
        JCOL        = POIT(IDX+2)
        MDEX        = MPOT(POIT(IDX+3)) - 2000
        IVAL        = POIT(IDX+4)
C----------set parameter----------
        VD          = VAL(IVAL)
        IS          = VAL(MDEX)
        CC          = VAL(MDEX-1)
        V1          = VAL(IVAL+1)
        TEMP        = DEXP(VD/VT)
        GD          = IS*TEMP/VT
        CD          = IS*(TEMP-1)
        CK          = CD - (GD*VD)
C
        IF ( MODE .EQ. 1 .OR. MODE .EQ. 9) THEN
              GTO = GD
              CTNO= CK
        ELSEIF (MODE .EQ. 4) THEN
              GTO = GD     + CC/STEP
              CTNO= CK - CC/STEP*V1
        ENDIF
C
        A(IROW,IROW) = A(IROW,IROW) + GTO
        A(IROW,JCOL) = A(IROW,JCOL) - GTO
        A(JCOL,IROW) = A(JCOL,IROW) - GTO
        A(JCOL,JCOL) = A(JCOL,JCOL) + GTO
        B(IROW)    = B(IROW) - CTNO
        B(JCOL)    = B(JCOL) + CTNO
        IDX         = POIT(IDX)
        GOTO 141
C---stamp opamp---
150     IDX         = POIT(16)
151     IF (IDX .EQ. 0) GOTO 160
        IROW        = POIT(IDX+1)
        JCOL        = POIT(IDX+2)
        IB          = POIT(IDX+3)
        JB          = POIT(IDX+4)
        MDEX        = MPOT(POIT(IDX+5)) - 1000
        IVAL        = POIT(IDX+6)
```

```
C--:set parameter:--
      AV          = DBLE(VAL(MDEX))
      G           = DBLE(1.0/VAL(MDEX-1))
      RS          = DBLE(VAL(MDEX-3))
      CC          = DBLE(VAL(MDEX-2))
      P1          = DBLE(VAL(MDEX-4))
      P2          = DBLE(VAL(MDEX-5))
      GB          = 1.0E+20
      V1          = DBLE(VAL(MDEX-6))
C--: V1 = supply voltage
      V2          = DBLE(VAL(IVAL+5))
C--: V2 = output voltage of opamp
      V1          = V1 - 0.1
C--: set Vout saturated = Vsupply-0.1
C--:  generate unit ramp factor  :--
      IF (V2 .GT. V1) THEN
            GC  = +1.0
C--:  unit ramp = +1
            V1  = V1
C--:!Vout saturated
      ELSEIF (V2 .LT. -V1) THEN
            GC  = -1.0
C--:  unit ramp = -1
            V1  = -(V1)
C--:  Vout saturated
            A(IB,JB) = -2.0
C--: !set quadant 4
      ELSE
            GC  = 0.0
      ENDIF
C--:
      IF (MODE .EQ. 1 .OR. MODE .EQ. 9) THEN
            AA  = 0.0
            BB  = 0.0
      ELSEIF(MODE .EQ. 4) THEN
            IF (P1 .EQ. 0.0 .OR. P2 .EQ. 0.0) THEN
                  AA    = 0.0
                  BB    = 0.0
            ELSE
                  AA    = (P1+P2)/(P1*P2)
                  BB    = 1.0/(P1*P2)
            ENDIF
      ENDIF
C--:stamp opamp:--
      A(IROW,IROW) = A(IROW,IROW) + G + CC/STEP
      A(IROW,JCOL) = A(IROW,JCOL) - G - CC/STEP
      A(JCOL,IROW) = A(JCOL,IROW) - G - CC/STEP
      A(JCOL,JCOL) = A(JCOL,JCOL) + G + CC/STEP
      A(IB,JB)     = A(IB,JB) + 1.0
      A(IB,IB)     = A(IB,IB) + GC*GB
      A(JB,IROW)         = +AV
      A(JB,JCOL)         = -AV
      A(JB,IB)           = A(JB,IB) - 1.0
      A(JB,JB)           = +RS
      B(IB)              = B(IB) + GC*GB*V1
```

```
C--:
      IF (MODE .EQ. 4) THEN
            A(JB,IB)      = -(1.+AA/STEP+BB/(STEP*STEP))
            A(JB,JB)      = +RS * (1.+AA/STEP+BB/(STEP*STEP))
            B(IROW)       = B(IROW) + CC*VAL(IVAL+1)/STEP
            B(JCOL)       = B(JCOL) - CC*VAL(IVAL+1)/STEP
            B(JB)         = -(BB/STEP*VAL(IVAL+4))-(AA+BB/STEP)*
     :                                                VAL(IVAL+3)/STE
      ENDIF
      IDX         = POIT(IDX)
      GOTO 151
C---stamp transistor---
160   IDX         = POIT(15)
161   IF (IDX .EQ. 0) GOTO 170
      IROW        = POIT(IDX+1)
      JCOL        = POIT(IDX+2)
      IB          = POIT(IDX+3)
      MDEX        = MPOT(POIT(IDX+4))-3000
      IVAL        = POIT(IDX+5)
      CJC         = DBLE(VAL(MDEX))
      CJE         = DBLE(VAL(MDEX-1))
      ALPR        = DBLE(VAL(MDEX-2))
      ALPF        = DBLE(VAL(MDEX-3))
      CES         = VAL(MDEX-4)/ALPF
      CCS         = VAL(MDEX-4)/ALPR
C      GB         = DBLE(1.0/VAL(MDEX-5))
      TYPE        = VAL(MDEX-6)
      VBE         = VAL(IVAL)
      VBC         = VAL(IVAL+1)
C----------set parameter------
      TEM1        = DEXP(VBE/VT)
C--:  defalut set NPN transistor
      TEM2        = DEXP(VBC/VT)
      TEM3        = TEM1 - 1.0 - (TEM1*VBE/VT)
      TEM4        = TEM2 - 1.0 - (TEM2*VBC/VT)
      YEEX        = +CES*TEM1/VT
      YCCX        = +CCS*TEM2/VT
      YECX        = ALPR*CCS*TEM2/VT
      YCEX        = ALPF*CES*TEM1/VT
      CETX        = ALPR*CCS*TEM4-CES*TEM3
      CCTX        = ALPF*CES*TEM3-CCS*TEM4
      IF ( TYPE .EQ. -2.0) THEN
C--: set PNP parameter
            CETX          = -CETX
            CCTX          = -CCTX
      ENDIF
C----------start EBER MOLL model-----
      A(IROW,IROW)        = A(IROW,IROW) + YEEX + YCCX - YECX - YCEX
      A(IROW,JCOL)        = A(IROW,JCOL) - YCCX + YECX
      A(IROW,IB)          = A(IROW,IB)   - YEEX + YCEX
      A(JCOL,IROW)        = A(JCOL,IROW) - YCCX + YCEX
      A(JCOL,JCOL)        = A(JCOL,JCOL) + YCCX
      A(JCOL,IB)          = A(JCOL,IB)   - YCEX
      A(IB,IROW)          = A(IB,IROW)   - YEEX + YECX
      A(IB,JCOL)          = A(IB,JCOL)   - YECX
```

```
      A(IB,IB)            = A(IB,IB)      + YEEX
      B(IROW)             = B(IROW) + CETX + CCTX
      B(JCOL)             = B(JCOL) - CCTX
      B(IB)               = B(IB)   - CETX
C----stamp for transient anslysis----
      IF ( MODE .EQ. 4 ) THEN
           CJE = CJE/STEP
           CJC = CJC/STEP
C------------------------
           A(IROW,IROW)          = A(IROW,IROW) + CJE + CJC
           A(JCOL,JCOL)          = A(JCOL,JCOL) + CJC
           A(IB,IB)              = A(IB,IB)     + CJE
           A(IROW,JCOL)          = A(IROW,JCOL) - CJC
           A(IROW,IB)            = A(IROW,IB)   - CJE
           A(JCOL,IROW)          = A(JCOL,IROW) - CJC
           A(IB,IROW)            = A(IB,IROW)   - CJE
C-----------
           CES = CJE * VAL(IVAL+2)
           CCS = CJC * VAL(IVAL+3)
C-----------
           B(IROW)      = B(IROW) + CES + CCS
           B(JCOL)      = B(JCOL) - CCS
           B(IB)        = B(IB)   - CES
      ENDIF
      IDX         = POIT(IDX)
      GOTO 161
C--------stamp Vin or Vac------
170   IDX         = POIT(3)
171   IF (IDX .EQ. 0) GOTO 180
      IROW        = POIT(IDX+1)
      JCOL        = POIT(IDX+2)
      IB          = POIT(IDX+3)
      A(IROW,IB)            =  1.
      A(JCOL,IB)            = -1.
      A(IB,IROW)            =  1.
      A(IB,JCOL)            = -1.
      B(IB)                 = VAL(POIT(IDX+4))
      IF ( MODE .EQ. 9) B(IB) = 0.0
      IDX         = POIT(IDX)
      GOTO 171
C-------stamp capacitor---------
180   IDX         = POIT(8)
181   IF ( IDX .EQ. 0 ) GOTO 190
      IROW        = POIT(IDX+1)
      JCOL        = POIT(IDX+2)
      IVAL        = POIT(IDX+3)
      AA          = VAL(IVAL)/STEP
      BB          = VAL(IVAL+1)
      A(IROW,IROW)          = A(IROW,IROW)  + AA
      A(IROW,JCOL)          = A(IROW,JCOL)  - AA
      A(JCOL,IROW)          = A(JCOL,IROW)  - AA
      A(JCOL,JCOL)          = A(JCOL,JCOL)  + AA
      B(IROW)               = B(IROW)       + AA*BB
      B(JCOL)               = B(JCOL)       - AA*BB
      IDX         = POIT(IDX)
```

```
      GOTO 181
C-------stamp inductor----------
190   IDX        = POIT(9)
191   IF (IDX .EQ. 0) GOTO 200
      IROW       = POIT(IDX+1)
      JCOL       = POIT(IDX+2)
      IB         = POIT(IDX+3)
      IVAL       = POIT(IDX+4)
      A(IROW,IB)          =  1.
      A(JCOL,IB)          = -1.
      A(IB,IROW)          =  1.
      A(IB,JCOL)          = -1.
      A(IB,IB)            = A(IB,IB)-VAL(IVAL)/STEP
      B(IB)               = B(IB) - VAL(IVAL)/STEP*VAL(IVAL+2)
      IDX        = POIT(IDX)
      GOTO 191
C---stamp VPULSE----
200   IDX        = POIT(4)
201   IF ( IDX .EQ. 0 ) GOTO 210
      IROW       = POIT(IDX+1)
      JCOL       = POIT(IDX+2)
      IB         = POIT(IDX+3)
      IVAL       = POIT(IDX+4)
      A(IROW,IB)          = 1.
      A(JCOL,IB)          = -1.
      A(IB,IROW)          = 1.
      A(IB,JCOL)          = -1.
      IF (MODE .EQ. 1 .OR. MODE .EQ. 9) THEN
C--:!qpoint analysis mode
            B(IB)         = VAL(IVAL+2)
      ELSEIF (MODE .EQ. 4) THEN
C--:!transient analysis
            CALL FINDV
      ENDIF
      IDX                 = POIT(IDX)
      GOTO 201
C--: stamp VCVS : --
C
210   IDX = POIT(10)
211   IF (IDX .EQ. 0 ) GOTO 220
      IROW    = POIT(IDX+1)
      JCOL    = POIT(IDX+2)
      IB      = POIT(IDX+3)
      JB      = POIT(IDX+4)
      MDEX    = POIT(IDX+5)
      IVAL    = POIT(IDX+6)
      A(IB,MDEX)  =  1.0
      A(JB,MDEX)  = -1.0
      A(MDEX,IROW)= -VAL(IVAL)
      A(MDEX,JCOL)=  VAL(IVAL)
      A(MDEX,IB)  =  1.0
      A(MDEX,JB)  = -1.0
      IDX         = POIT(IDX)
      GOTO 211
C:
```

```
C--: STAMP VCCS : --
C:
220    IDX = POIT(11)
221    IF (IDX .EQ. 0) GOTO 230
       IROW = POIT(IDX+1)
       JCOL = POIT(IDX+2)
       IB   = POIT(IDX+3)
       JB   = POIT(IDX+4)
       IVAL = POIT(IDX+5)
       A(IB,IROW) = A(IB,IROW) + VAL(IVAL)
       A(IB,JCOL) = A(IB,JCOL) - VAL(IVAL)
       A(JB,IROW) = A(JB,IROW) - VAL(IVAL)
       A(JB,JCOL) = A(JB,JCOL) + VAL(IVAL)
       IDX = POIT(IDX)
       GOTO 221
230    CONTINUE
       RETURN
       END
C*****************************************
c   store voltage for nonlinear analysis
C*****************************************
       SUBROUTINE STORE(FLAG,STEP)
       INCLUDE    'DECLARE.INC'
       INCLUDE    'DOUBLE.INC'
       INCLUDE    'PARA.INC'
       DOUBLE PRECISION  V0,     V1,     VD,     VCRI,    STEP,    BUF
       LOGICAL*1 FLAG
       FLAG       = .FALSE.
C       B(0)      = 0.0
C---------diode voltage node--------
       IDX        = POIT(14)
110    IF (IDX .EQ. 0) GOTO 200
       IROW       = POIT(IDX+1)
       JCOL       = POIT(IDX+2)
       IB         = POIT(IDX+3)
C---------find VCRI-----------------
       VCRI       = 0.78
C-----------------------------------
       V0         = VAL(POIT(IDX+4))
       IF(JCOL .EQ. 0) THEN
            V1   = B(IROW)
       ELSEIF (IROW .EQ. 0) THEN
            V1   = -B(JCOL)
       ELSE
            V1   = B(IROW) -  B(JCOL)
       ENDIF
       IF (V1 .LT. VCRI) THEN
            IF (V1 .LT. -100.0) THEN
                VD = V1/2.0
C               FLAG = .TRUE.
            ELSE
                    VD  = V1
            ENDIF
       ELSE
                BUFFER        = V1 - V0
```

```
                    VD     = V0 + (VT*DLOG((BUFFER/VT) + 1))
                    FLAG   = .TRUE.
      ENDIF
      VAL(POIT(IDX+4))   = SNGL(VD)
      IDX          = POIT(IDX)
      GOTO 110
C---------transistor voltage node----
200   IDX          = POIT(15)
210   IF (IDX .EQ. 0) GOTO 300
C-----------find VCRI----------
     .VCRI         = 0.78
C------------------------------
      IROW         = POIT(IDX+1)
      JCOL         = POIT(IDX+2)
      IB           = POIT(IDX+3)
      IVAL         = POIT(IDX+5)
      MDEX         = MPOT(POIT(IDX+4)) - 3000
      TYPE         = VAL(MDEX-6)
      V0           = VAL(IVAL)
C--: NPN transistor :--
      IF (IROW .EQ. 0) THEN
          V1       = -B(IB)
      ELSEIF (IB .EQ. 0) THEN
          V1       = B(IROW)
      ELSE
          V1       = B(IROW) - B(IB)
      ENDIF
C--: PNP transistor :--
      IF ( TYPE .EQ. -2.0 ) THEN
            V1   = -V1
      ENDIF
C--------store Vbe------
      IF (V1 .LT. VCRI) THEN
            IF (V1 .LT. -100.0) THEN
                VD = V1/2.0
C               FLAG = .TRUE.
            ELSE
                VD  = V1
            ENDIF
       ELSE
                BUFFER          = V1 - V0
                VD     = V0 + (VT*DLOG((BUFFER/VT) + 1))
                FLAG   = .TRUE.
       ENDIF
       VAL(IVAL)          = VD
C--------store Vbc------
      V0           = VAL(IVAL+1)
C--: NPN transistor  :--
      IF (IROW .EQ. 0) THEN
          V1       = -B(JCOL)
      ELSEIF ( JCOL .EQ. 0) THEN
          V1       = + B(IROW)
      ELSE
          V1       = B(IROW) - B(JCOL)
      ENDIF
```

```
C--: PNP transistor :==
      IF (TYPE .EQ. -2.0) THEN
            V1  = -V1
      ENDIF
      IF (V1 .LT. VCRI) THEN
            IF (V1 .LT. -100.0) THEN
                VD = V1/2.0
C               FLAG = .TRUE.
            ELSE
                    VD  = V1
            ENDIF
       ELSE
                  BUFFER          = V1 - V0
                  VD     = V0 + (VT*DLOG((BUFFER/VT) + 1))
                  FLAG   = .TRUE.
       ENDIF
      VAL(IVAL+1)          = VD
      IDX                  = POIT(IDX)
      GOTO 210
C--:store opamp:--
300   IDX        = POIT(16)
310   IF (IDX .EQ. 0) GOTO 400
      IB         = POIT(IDX+3)
      IVAL       = POIT(IDX+6)
      VAL(IVAL+5)         = B(IB)
      IDX        = POIT(IDX)
      GOTO 310
400   CONTINUE
      RETURN
      END
C*************************************************
C   STORE VOLTAGE&CURRENT OF REACTIVE ELEMENT
C*************************************************
      SUBROUTINE STORLC
      DOUBLE PRECISION  VD
      INCLUDE    'DECLARE.INC'
      INCLUDE    'DOUBLE.INC'
      INCLUDE    'PARA.INC'
C---------store capacitor------------
      IDX        = POIT(8)
100   IF (IDX .EQ. 0) GOTO 110
      IROW       = POIT(IDX+1)
      JCOL       = POIT(IDX+2)
      IVAL       = POIT(IDX+3)
      VC         = VAL(IVAL+1)
      IF (IROW .EQ. 0 ) THEN
            VAL(IVAL+1) = -B(JCOL)
      ELSEIF(JCOL .EQ. 0) THEN
            VAL(IVAL+1) = B(IROW)
      ELSE
            VAL(IVAL+1) = B(IROW) - B(JCOL)
      ENDIF
      VAL(IVAL+2)         = (VAL(IVAL+1) - VC)/ H
      IDX        = POIT(IDX)
      GOTO 100
```

```
C--------store inductor--------------
 110  IDX        = POIT(9)
 111  IF (IDX .EQ. 0) GOTO 120
      IROW       = POIT(IDX+1)
      JCOL       = POIT(IDX+2)
      IB         = POIT(IDX+3)
      IVAL       = POIT(IDX+4)
      IF (IROW .EQ. 0 ) THEN
            VAL(IVAL+1) = -B(JCOL)
      ELSEIF(JCOL .EQ. 0) THEN
            VAL(IVAL+1) = B(IROW)
      ELSE
            VAL(IVAL+1) = B(IROW) - B(JCOL)
      ENDIF
      VAL(IVAL+2) = B(IB)
      IDX        = POIT(IDX)
      GOTO 111
C---------store opamp------------------
120   IDX        =   POIT(16)
121   IF (IDX .EQ. 0) GOTO 130
      IROW       = POIT(IDX+1)
      JCOL       = POIT(IDX+2)
      IB         = POIT(IDX+3)
      JB         = POIT(IDX+4)
      IVAL       = POIT(IDX+6)
      MDEX       = MPOT(POIT(IDX+5)) - 1000
      VC         = VAL(IVAL+1)
      IF (IROW .EQ. 0) THEN
            VAL(IVAL+1) = -B(JCOL)
      ELSEIF(JCOL .EQ. 0) THEN
            VAL(IVAL+1) = B(IROW)
      ELSE
            VAL(IVAL+1) = B(IROW) - B(JCOL)
      ENDIF
      VAL(IVAL+2) = (VAL(IVAL+1)-VC)*VAL(MDEX-2)/SNGL(H)
      VC         = VAL(IVAL+3)
      VAL(IVAL+3) = B(JB)*VAL(MDEX-3)+B(IB)
      VAL(IVAL+4) = (VAL(IVAL+3)-VC)/SNGL(H)
      IDX        = POIT(IDX)
      GOTO 121
C-----store diode----
130   IDX        = POIT(14)
131   IF (IDX .EQ. 0) GOTO 140
      IROW       = POIT(IDX+1)
      JCOL       = POIT(IDX+2)
      IVAL       = POIT(IDX+4)
      VC         = VAL(IVAL+1)
      IF (JCOL .EQ. 0) THEN
            VAL(IVAL+1) = B(IROW)
      ELSEIF (IROW .EQ. 0) THEN
            VAL(IVAL+1) = -B(JCOL)
      ELSE
            VAL(IVAL+1) = B(IROW) - B(JCOL)
      ENDIF
      VAL(IVAL+2)        = (VAL(IVAL+1) - VC)/SNGL(H)
```

```
      IDX       = POIT(IDX)
      GOTO 131
C-----store transistor-----
140   IDX       = POIT(15)
141   IF(IDX .EQ. 0) GOTO 150
      IROW      = POIT(IDX + 1)
      JCOL      = POIT(IDX + 2)
      IB        = POIT(IDX + 3)
      IVAL      = POIT(IDX + 5)
      MDEX      = MPOT(POIT(IDX+4)) - 3000
      TYPE      = VAL(MDEX-6)
      VC        = VAL(IVAL+2)
      IF (IROW .EQ. 0) THEN
            VAL(IVAL+2) = -B(IB)
      ELSEIF(IB .EQ. 0) THEN
            VAL(IVAL+2) = B(IROW)
      ELSE
            VAL(IVAL+2) = B(IROW) - B(IB)
      ENDIF
c      IF (TYPE .EQ. -2.0) VAL(IVAL+2) = -VAL(IVAL+2)
      VAL(IVAL+4)       = (VAL(IVAL+2)-VC)/SNGL(H)
      VC        = VAL(IVAL+3)
      IF (IROW .EQ. 0) THEN
            VAL(IVAL+3) = -B(JCOL)
      ELSEIF (JCOL .EQ. 0) THEN
            VAL(IVAL+3) = B(IROW)
      ELSE
            VAL(IVAL+3) = B(IROW) - B(JCOL)
      ENDIF
c      IF (TYPE .EQ. -2.0) VAL(IVAL+3)  = -VAL(IVAL+3)
      VAL(IVAL+5)       = (VAL(IVAL+3)-VC)/SNGL(H)
      IDX       = POIT(IDX)
      GOTO      141
150   CONTINUE
      RETURN
      END

C*************************
C  EDIT AC CURRENT SOURCE
C*************************
      SUBROUTINE EDIAC(KID)
      INCLUDE 'DECLARE.INC'
      POIT(KID)       = IP
      MNAM(MP)        = W
      MPOT(MP)        = IP
      CALL SLOC
      CALL GETR(R)
      POIT(IP+3)      = IVAL
      VAL(IVAL)       = R
      CALL GETR(R)
      VAL(IVAL+1)     = R
C!store phase angle
      KID     = IP
      IVAL    = IVAL + 2
      MP      = MP + 1
```

```
        IP      = IP + 4
        RETURN
        END
C************************************
C    LOAD RESISTOR&CURRENT SOURCE
C************************************
      SUBROUTINE EDRI(KID)
C
      INCLUDE   'DECLARE.INC'
      INCLUDE   'PARA.INC'
C
      POIT(KID) = IP
C!store header
C
      MNAM(MP)  = W
C!store name & mpot array
      MPOT(MP)  = IP
C
      CALL      SLOC
C!store n1 & n2
C
      CALL      GETR(R)
C!store value
      POIT(IP+3)= IVAL
      VAL(IVAL) = R
C
      KID       = IP
C!advance array pointer
      IVAL      = IVAL + 1
      MP        = MP + 1
      IP        = IP + 4
C
      RETURN
      END
C**********************
C   SCAN LOC ARRAY
C**********************
      SUBROUTINE SLOC
C
      INCLUDE   'DECLARE.INC'
      INCLUDE   'PARA.INC'
C
C---store N1 & N2---
      CALL      GETINT(I)
C!store n1
      IF (I .EQ. 0) THEN
                POIT(IP+1) = 0
                GOTO 20
      ENDIF
      DO 10 J = 1,60
        IF (I .EQ. LOC(J)) THEN
                POIT(IP+1) = J
                GOTO 20
        ELSEIF (LOC(J) .EQ. 0) THEN
                LOC(J) = I
```

```
                POIT(IP+1) = J
                GOTO 20
          ELSE
                GOTO 10
          ENDIF
10        CONTINUE
C
20        CALL      GETINT(I)
C!store n2
          IF ( I .EQ. 0) THEN
                POIT(IP+2) = 0
                GOTO 40
          ENDIF
          DO 30 J = 1,60
            IF (I .EQ. LOC(J)) THEN
                POIT(IP+2) = J
                GOTO 40
            ELSEIF (LOC(J) .EQ. 0) THEN
                LOC(J) = I
                POIT(IP+2) = J
                GOTO 40
            ELSE
                GOTO 30
            ENDIF
30        CONTINUE
40        IF (W(1:1) .EQ. 'Q' .OR. W(1:1) .EQ. 'O' .OR. W(1:4) .EQ. 'VC
     :       .OR. W(1:4) .EQ. 'VCCS' .OR. W(1:4) .EQ. 'ICVS' .OR. W(1:
     :       .EQ. 'ICCS' ) THEN
C!store n3
              CALL          GETINT(I)
              IF ( I .EQ. 0) THEN
                   POIT(IP+3)     = 0
                   GOTO 60
              ENDIF
              DO 50 J = 1,60
                   IF ( I .EQ.LOC(J)) THEN
                        POIT(IP+3)       = J
                        GOTO 60
                   ELSEIF (LOC(J) .EQ. 0) THEN
                        LOC(J)           = I
                        POIT(IP+3)       = J
                        GOTO 60
                   ENDIF
50            CONTINUE
60            CONTINUE
          ELSE  RETURN
          ENDIF
          IF (W(1:4) .EQ. 'VCVS' .OR. W(1:4) .EQ. 'VCCS' .OR. W(1:4) .E
         :    'ICVS' .OR. W(1:4) .EQ. 'ICCS') THEN
              CALL          GETINT(I)
              IF ( I .EQ. 0 ) THEN
                   POIT(IP+4)     = 0
                   GOTO 80
              ENDIF
              DO 70 J = 1,60
```

```
                    IF (I .EQ. LOC(J)) THEN
                          POIT(IP+4)      = J
                          GOTO 80
                    ELSEIF (LOC(J) .EQ. 0) THEN
                          LOC(J)  = I
                          POIT(IP+4)      = J
                          GOTO 80
                    ENDIF
70          CONTINUE
80    CONTINUE
      ELSE
            RETURN
      ENDIF
      RETURN
      END
C**********************8
C   print array
C**********************
      SUBROUTINE PRINT
      INCLUDE   'DECLARE.INC'
      INCLUDE   'PARA.INC'
      WRITE (6,*)' I   POIT  LOC      VAL       MPOT                MNAM
      DO 100 I = 1,40
          WRITE(6,99)I,POIT(I),LOC(I),VAL(I),MPOT(I),MNAM(I)
99    FORMAT (I3,2I6,E12.5,I6,A20)
100   CONTINUE
        DO 200 I = 180,200
        WRITE(6,*)I,VAL(I)
200     CONTINUE
      RETURN
      END
C*****************
C EDIT CAPACITOR
C*****************
      SUBROUTINE EDC(KID)
      INCLUDE   'DECLARE.INC'
      POIT(KID) = IP
C!store header
      MNAM(MP)  = W
      MPOT(MP)  = IP
      CALL SLOC
C!store n1 & n2
      CALL GETR(R)
      VAL(IVAL)          = R
      POIT(IP+3)         = IVAL
      KID                = IP
      MP                 = MP + 1
      IP                 = IP + 4
      IVAL               = IVAL + 4
      RETURN
      END
C*************************************************
C  EDIT  INDUCTOR & D.C.VOLTAGE SOURCE
C*************************************************
      SUBROUTINE EDLCV(KID)
```

```
      INCLUDE   'DECLARE.INC'
      INCLUDE   'PARA.INC'
      POIT(KID) = IP
C!store header
C
      MNAM(MP)  = W
C!store name & mpot array
      MPOT(MP)  = IP
C
      CALL      SLOC
C!store n1 & n2
C
      CALL      STBCR(IDX)
C!store BCR
      POIT(IP+3)= IDX
C
      CALL      GETR(R)
C!store value
      POIT(IP+4)= IVAL
      VAL(IVAL) = R
      VAL(IVAL+1)=0.
      VAL(IVAL+2)=0.
      KID       = IP
      MP        = MP + 1
      IP        = IP + 5
C
      IF (W(1:3) .EQ. 'VDC') THEN
C!check if voltage source
            IVAL= IVAL+1
      ELSE
            IVAL= IVAL+4
C!else  = L & C
      ENDIF
C
      RETURN
      END
C***************
C   STORE BCR
C***************
      SUBROUTINE STBCR(IDX)
C
      INCLUDE   'DECLARE.INC'
      INCLUDE   'PARA.INC'
C
      DO 10 J = 1,60
            IF (LOC(J) .EQ. 0) GOTO 20
10    CONTINUE
C
20    LOC(J)    = -1
      IDX       =  J
      RETURN
      END
C****************
C   edit diode
C****************
```

```
      SUBROUTINE EDD(KID)
      INCLUDE   'DECLARE.INC'
      INCLUDE   'PARA.INC'
      POIT(KID) = IP
C!store header
      MNAM(MP)  = W
C!store name & mpot array
      MPOT(MP)  = IP
      CALL      SLOC
C!store n1&n2
      CALL      GETW
C!getmodel name
      DO 10 I = 1,60
C!check model name
            IF (W .EQ. MNAM(I)) THEN
                  POIT(IP+3)    = I
                  GOTO 20
            ELSEIF(MNAM(I) .EQ. '        ') THEN
                  POIT(IP+3)    = I
                  MNAM(I)       = W
                  MPOT(I)       = 2000 + MDEX
                  MP            = MP + 1
                  MDEX          = MDEX - 2
                  GOTO 20
            ENDIF
10    CONTINUE
      WRITE (6,111)
111   FORMAT(' ERROR:device name full')
      STOP
20    POIT(IP+4)        = IVAL
C!store initial Vd
        CALL GETW
        IF (W(1:2) .EQ. 'VD') THEN
                CALL GETR(R)
                VAL(IVAL) = R
        ELSE
                VAL(IVAL) = 0.3
        ENDIF
      KID       = IP
C!advance array pointer
      IVAL      = IVAL + 4
      MP        = MP + 1
      IP        = IP + 5
      RETURN
      END
C*********************
C   edit transistor
C*********************
      SUBROUTINE EDQ(KID)
      INCLUDE   'DECLARE.INC'
      INCLUDE   'PARA.INC'
      POIT(KID) = IP
      MNAM(MP)  = W
      MPOT(MP)  = IP
      CALL      SLOC
```

```
      CALL      GETW
      DO 10 I =1,60
            IF (W .EQ. MNAM(I)) THEN
                  POIT(IP+4)     = I
                  GOTO 20
            ELSEIF (MNAM(I) .EQ. '          ') THEN
                  POIT(IP+4)     = I
                  MNAM(I)        = W
                  MPOT(I)        = 3000 + MDEX
                  MP             = MP + 1
                  MDEX           = MDEX - 7
                  GOTO 20
            ENDIF
10    CONTINUE
      WRITE (6,111)
111   FORMAT (' ERROR:device name full')
      STOP
20    POIT(IP+5)          = IVAL
      VAL(IVAL)           = 0.3
      VAL(IVAL+1)         = -1.0
30      CALL GETW
        CALL GETR(R)
        IF (W(1:1) .EQ. ' ' .OR. W(1:1) .EQ. ']') GOTO 40
        IF ( W(1:3) .EQ. 'VBE' ) THEN
                VAL(IVAL) = R
        ELSEIF (W(1:3) .EQ. 'VBC') THEN
                VAL(IVAL+1) = R
        ENDIF
        GOTO 30
40      CONTINUE
      KID                 = IP
      IVAL                = IVAL + 8
      MP                  = MP + 1
      IP                  = IP + 6
      RETURN
      END
C*********************
C   edit opamp
C*********************
      SUBROUTINE        EDO(KID)
      INCLUDE   'DECLARE.INC'
      INCLUDE   'PARA.INC'
      POIT(KID) = IP
      MNAM(MP)  = W
      MPOT(MP)  = IP
      CALL SLOC
      CALL STBCR(IDX)
      POIT(IP+4)= IDX
      CALL GETW
      DO 10 I = 1,60
            IF ( W .EQ. MNAM(I)) THEN
                  POIT(IP+5) = I
                  GOTO 20
            ELSEIF (MNAM(I) .EQ. '          ') THEN
                  POIT(IP+5)     = I
```

```
                    MNAM(I)         = W
                    MPOT(I)         = 1000+MDEX
                    MP              = MP + 1
                    MDEX            = MDEX - 7
                    GOTO 20
                ENDIF
10      CONTINUE
        WRITE (6,111)
111     FORMAT(' ERROR:device name full')
        STOP
20      KID         = IP
        MP          = MP+1
        POIT(IP+6)= IVAL
        IVAL        = IVAL + 6
        IP          = IP + 7
        RETURN
        END
C**********************
C   edit vcvs
C**********************
        SUBROUTINE EDVCVS(KID)
        INCLUDE    'DECLARE.INC'
        POIT(KID) = IP
        MNAM(MP)   = W
        MPOT(MP)   = IP
        CALL SLOC
        CALL STBCR(IDX)
        POIT(IP+5)          = IDX
        CALL GETR(R)
        POIT(IP+6)          = IVAL
        VAL(IVAL)           = R
        KID                 = IP
        IVAL                = IVAL+ 1
        MP                  = MP+1
        IP                  = IP+7
        RETURN
        END
C***************
C   edit vccs
C***************
        SUBROUTINE EDVCCS(KID)
        INCLUDE    'DECLARE.INC'
        POIT(KID) = IP
        MNAM(MP)   = W
        MPOT(MP)   = IP
        CALL SLOC
        CALL GETR(R)
        POIT(IP+5)          = IVAL
        VAL(IVAL)           = R
        KID                 = IP
        IVAL                = IVAL+1
        MP                  = MP+1
        IP                  = IP+6
        RETURN
        END
```

```
C*******************
C   edit model
c*******************
      SUBROUTINE EDM
      INCLUDE   'DECLARE.INC'
      CALL GETW
      DO 10 J = 1,60
            IF (MNAM(J) .EQ. W) THEN
                  IF(MPOT(J) .GE. 1000 .AND. MPOT(J) .LT. 2000)
     :                  CALL EDMO(MPOT(J))
                  IF (MPOT(J) .GE. 2000 .AND. MPOT(J) .LT. 3000)
     :                  CALL EDMD(MPOT(J))
                  IF (MPOT(J) .GE. 3000 .AND. MPOT(J) .LT. 4000)
     :                  CALL EDMQ(MPOT(J))
                  RETURN
            ENDIF
10    CONTINUE
      WRITE (6, 11) W
11    FORMAT(' ERROR:mismatch model name>',:,a9)
      RETURN
      END
C************************
C   EDIT OPAMP MODEL
C************************
      SUBROUTINE EDMO(KID)
      INCLUDE   'DECLARE.INC'
      MD        = KID - 1000
      CALL SKIPSP
10    CALL GETW
      IF (W .EQ. '          ' .OR. W .EQ. ']          ') GOTO 20
            IF(W(1:2) .EQ. 'GA' ) NBAS = 0
            IF(W(1:2) .EQ. 'RI' ) NBAS = 1
            IF(W(1:2) .EQ. 'CI' ) NBAS = 2
            IF(W(1:2) .EQ. 'RO' ) NBAS = 3
            IF(W .EQ. 'POLE1    ') NBAS = 4
            IF(W .EQ. 'POLE2    ') NBAS = 5
            IF(W(1:2) .EQ. 'VS')  NBAS = 6
            CALL GETR(R)
            VAL(MD-NBAS)          = R
            GOTO 10
C---fill default value----
20    IF (VAL(MD)   .EQ. 0.) VAL(MD) = 1.0E10
C!gain
      IF (VAL(MD-1) .EQ. 0.) VAL(MD-1) = 1.0E10
C!input resistance
      IF (VAL(MD-3) .EQ. 0.) VAL(MD-3) = 1.0E-10
C!output resistance
      IF (VAL(MD-6) .EQ. 0.) VAL(MD-6) = 15.0
C!vsupply=15.0
      IF (VAL(MD-4) .EQ. 0.) VAL(MD-4) = 10.0
      IF (VAL(MD-5) .EQ. 0.) VAL(MD-5) = 1.0E6
C
      RETURN
      END
C************************
```

```
C     edit diode model
C*************************
      SUBROUTINE EDMD(KID)
      INCLUDE   'DECLARE.INC'
      MD        = KID-2000
      CALL      SKIPSP
10    CALL GETW
      IF ( W .EQ. '         ' .OR. W .EQ. ']         ') GOTO 20
            IF (W .EQ. 'IS        ') NBAS = 0
            IF (W .EQ. 'CJO       ') NBAS = 1
            CALL GETR(R)
            VAL(MD-NBAS)          = R
            GOTO 10
20    IF (VAL(MD)   .EQ. 0.) VAL(MD)   = 1.E-14
      RETURN
      END
C*************************
C   edit transistor model
C*************************
      SUBROUTINE EDMQ(KID)
      INCLUDE   'DECLARE.INC'
      MD        = KID - 3000
      CALL SKIPSP
10    CALL GETW
      IF (W .EQ. '          ' .OR. W .EQ. ']         ') GOTO 20
            IF (W .EQ. 'CJC       ') NBAS = 0
            IF (W .EQ. 'CJE       ') NBAS = 1
            IF (W .EQ. 'ALPR      ') NBAS = 2
            IF (W .EQ. 'ALPF      ') NBAS = 3
            IF (W .EQ. 'IS        ') NBAS = 4
            IF (W .EQ. 'RO        ') NBAS = 5
            IF (W .EQ. 'NPN       ') THEN
                VAL(MD-6) = -1.
                GOTO 10
            ENDIF
            IF (W .EQ. 'PNP       ') THEN
                VAL(MD-6) = -2.
                GOTO 10
            ENDIF
            CALL GETR(R)
            IF (W(1:2) .EQ. 'BF' ) THEN
                  NBAS  = 3
                  R     = R/(R+1.0)
            ELSEIF(W(1:2) .EQ. 'BR') THEN
                  NBAS  = 2
                  R     = R/(R+1.0)
            ENDIF
            VAL(MD-NBAS)          = R
            GOTO 10
20    IF(VAL(MD-2) .EQ. 0.) VAL(MD-2) = 0.3
C!ALPR
      IF(VAL(MD-3) .EQ. 0.) VAL(MD-3) = 0.99
C!ALPF
      IF(VAL(MD-4) .EQ. 0.) VAL(MD-4) = 1.0E-14
C!IS
```

```
      IF(VAL(MD-5) .EQ. 0.) VAL(MD-5) = 1.0E7
C!RO default=10M
      IF(VAL(MD-6) .EQ. 0.) VAL(MD-6) = -1.0
C!TYPE npn
      RETURN
      END
C*******************************
C   EDIT A.C. VOLTAGE SOURCE
C*******************************
      SUBROUTINE EDVAC(KID)
      INCLUDE   'DECLARE.INC'
      POIT(KID) = IP
      MNAM(MP)  = W
      MPOT(MP)  = IP
      CALL SLOC
      CALL STBCR(IDX)
      POIT(IP+3) = IDX
      IF ( W(1:3) .EQ. 'VIN') THEN
            CALL GETR(R)
            IF ( R .NE. 0.0 ) THEN
               VAL(IVAL) = R
            ELSE
               VAL(IVAL)   = 0.0
            ENDIF
            VAL(IVAL+1) = 0.0
      ELSE
            CALL GETR(R)
            VAL(IVAL)   = R
C!store magnitude
            CALL GETR(R)
            VAL(IVAL+1) = R
C!store phase angle
      ENDIF
      POIT(IP+4)= IVAL
      KID       = IP
      IVAL      = IVAL+2
      MP        = MP+1
      IP        = IP+5
      RETURN
      END
C************************
C   EDIT VOLTAGE PLUSE
C************************
      SUBROUTINE EDVP(KID)
      INCLUDE   'DECLARE.INC'
      POIT(KID) = IP
      MNAM(MP)  = W
      MPOT(MP)  = IP
      CALL      SLOC
      CALL STBCR(IDX)
      POIT(IP+3)        = IDX
      POIT(IP+4)        = IVAL
      CALL SKIPSP
      CALL      GETINT(I)
C!GET NUMBER OF PWL POINT
```

```
      NPW         = I*2
      KID         = IP
      MP          = MP+ 1
      IP          = IP+ 5
      ISTA        = IVAL
C------save PWL point-------
100   IVAL        = IVAL+1
      CALL        GETR(R)
      VAL(IVAL)           = R
      NPW         = NPW - 1
      IF (NPW .EQ. 0) THEN
            VAL(ISTA) = VAL(IVAL-1)-VAL(ISTA+1)
C!calculate period
            IVAL          = IVAL+1
            RETURN
      ELSE
            GOTO 100
      ENDIF
       END

C**************************************
C   Crout Doolittle with pivoting
C**************************************
      SUBROUTINE LU(N)
      DOUBLE PRECISION  SUM
D     LOGICAL*1 T,F
C
      INCLUDE    'DECLARE.INC'
      INCLUDE    'DOUBLE.INC'
      INCLUDE    'PARA.INC'
C
C
C
      I = 1
      CALL       PIVOT(N,I)
C
      DO 100 J = 2,N
            A(1,J) = A(1,J)/A(1,1)
100   CONTINUE
      DO 900 I =2,N
            DO 800 IR = I,N
                  DO 700 K = 1,I-1
                        A(IR,I) = A(IR,I) - A(IR,K)*A(K,I)
700               CONTINUE
800         CONTINUE
            CALL PIVOT(N,I)
            DO 600 J = I+1,N
                  SUM = A(I,J)
                  DO 500 K = 1,I-1
                        SUM = SUM - A(I,K)*A(K,J)
500               CONTINUE
                  A(I,J) = SUM/A(I,I)
600         CONTINUE
900   CONTINUE
C--------FORWARD SUBSTITUTION---------
```

```
      B(1) = B(1)/A(1,1)
      DO 910 I = 2,N
            SUM = B(I)
            DO 810 K = 1,I-1
                  SUM = SUM - A(I,K)*B(K)
810         CONTINUE
            B(I) = SUM/A(I,I)
910   CONTINUE
C-------BACKWARD SUBSTITUTION---------
      DO 920 I = N-1,1,-1
            DO 820 K = I+1,N
                  B(I) = B(I) - A(I,K)*B(K)
820         CONTINUE
920   CONTINUE
C
      RETURN
      END
C*******************
C   row pivoting
c*******************
      SUBROUTINE PIVOT(N,I)
      DOUBLE PRECISION  PE
C
      INCLUDE    'DECLARE.INC'
      INCLUDE    'DOUBLE.INC'
      INCLUDE    'PARA.INC'
C
      PE         = A(I,I)
      IPR        = I
      DO 100 JR = I+1,N
            IF (DABS(A(JR,I)) .LT. DABS(PE)) GOTO 100
            PE  = A(JR,I)
            IPR = JR
100   CONTINUE
      IF ( IPR .EQ. I ) GOTO 300
      DO 200 K = 1,N
            TEMP      = A(IPR,K)
            A(IPR,K)  = A(I,K)
            A(I,K)    = TEMP
200   CONTINUE
      TEMP      = B(IPR)
      B(IPR)    = B(I)
      B(I)      = TEMP
300   CONTINUE
C
      RETURN
      END
C************************
C   listing of circuit
C************************
      SUBROUTINE LIST
      INCLUDE    'DECLARE.INC'
      INCLUDE    'PARA.INC'
C--:
1200  CALL CLSCREEN
```

```
      WRITE (6,1000)
1000  FORMAT (///,'     Statistical Report of Circuit Elements:',//)
C--:
      LINEC  = 0
C--:
1     FORMAT ('     ',A9,I6,I9,1P,E12.2)
2     FORMAT ('     NAME   FROM NODE  TO NODE     VALUE',/)
3     FORMAT ('     ',A9,I6,I9,1P,E11.2,1X,1P,E11.2)
4     FORMAT ('     NAME   FROM NODE  TO NODE    VALUE    PHASE ANGLE

5     FORMAT (4X,A9,I3,7X,I3,9X,A9)
6     FORMAT (4X, 'NAME   ANODE      CATHODE     MODEL ',/)
7     FORMAT (4X,A9,I3,7X,I3,7X,I3,9X,A9)
8     FORMAT (4X, 'NAME   COLLECTOR   BASE   EMITTER          MODEL',
9     FORMAT (4X, 'NAME   NON-INVERT   INVERT   OUTPUT        MODEL',
      NUMB       = 0
      IDX        = POIT(1)
      IF (IDX .EQ. 0) GOTO 17
C--:list resistor
      LINEC = LINADD(LINEC)
      NUM        = 0
      WRITE (6,11)
11    FORMAT (' Resistor  Reports:',/)
      WRITE (6,2)
10    INTTEMP(1)          = LOC(POIT(IDX+1))
      INTTEMP(2)          = LOC(POIT(IDX+2))
      REALTEMP(1)         = VAL(POIT(IDX+3))
      CALL       SMPOT(NID,IDX)
      CHARTEMP(1)         = MNAM(NID)
      WRITE (*,1) CHARTEMP(1),INTTEMP(1),INTTEMP(2),REALTEMP(1)
      NUM        = NUM+1
      NUMB       = NUMB+1
      IF (POIT(IDX) .EQ. 0) THEN
            GOTO 15
      ELSE
      CALL LINCOUNT(LINEC)
            IDX = POIT(IDX)
            GOTO 10
      ENDIF
C15    WRITE (*,16) NUM
C16    FORMAT (/,' Total number of Resistors  =',i6,/)
15       WRITE (*,*) ' Total number of Resistors =',NUM
      NUMB = NUMB + NUM
17    CONTINUE
c-------listing of capacitor------
      LINEC = LINADD(LINEC)
      NUM        = 0
      IDX        = POIT(8)
      IF (IDX .EQ. 0) GOTO 27
      WRITE (6,21)
21    FORMAT ( ' Capacitor Report:',/)
      WRITE (6,2)
20    INTTEMP(1)          = LOC(POIT(IDX+1))
      INTTEMP(2)          = LOC(POIT(IDX+2))
      REALTEMP(1)         = VAL(POIT(IDX+3))
      CALL SMPOT(NID,IDX)
```

```
      CHARTEMP(1)       = MNAM(NID)
      WRITE (6,1) CHARTEMP(1),INTTEMP(1),INTTEMP(2),REALTEMP(1)
      NUM       = NUM + 1
      IF (POIT(IDX) .EQ. 0) THEN
            GOTO 25
      ELSE
      CALL LINCOUNT(LINEC)
            IDX = POIT(IDX)
            GOTO 20
      ENDIF
25    WRITE (6,26) NUM
26    FORMAT (/,' Total number of Capacitor =',i3,/)
      NUMB      = NUMB + NUM
27    CONTINUE
C-------inductor element-------
      LINEC = LINADD(LINEC)
      NUM       = 0
      IDX       = POIT(9)
      IF (IDX .EQ. 0) GOTO 37
      WRITE (6,31)
31    FORMAT (' Inductor Report:',/)
      WRITE (6,2)
30    INTTEMP(1)        = LOC(POIT(IDX+1))
      INTTEMP(2)        = LOC(POIT(IDX+2))
      REALTEMP(1)       = VAL(POIT(IDX+4))
      CALL SMPOT(NID,IDX)
      CHARTEMP(1)       = MNAM(NID)
      WRITE (6,1) CHARTEMP(1),INTTEMP(1),INTTEMP(2),REALTEMP(1)
      NUM       = NUM + 1
      IF (POIT(IDX) .EQ. 0) THEN
            GOTO 35
      ELSE
      CALL LINCOUNT(LINEC)
            IDX = POIT(IDX)
            GOTO 30
      ENDIF
35    WRITE (6,36) NUM
36    FORMAT (/,' Total number of Inductor =',I3,/)
      NUMB      = NUMB + 1
37    CONTINUE
c--------listing of d.c. voltage source--------
      LINEC = LINADD(LINEC)
      NUM       = 0
      IDX       = POIT(2)
      IF ( IDX .EQ. 0) GOTO 47
      WRITE (6,41)
41    FORMAT (' D.C. Voltage Source Report:',/)
      WRITE (6,2)
40    INTTEMP(1)        = LOC(POIT(IDX+1))
      INTTEMP(2)        = LOC(POIT(IDX+2))
      REALTEMP(1)       = VAL(POIT(IDX+4))
      CALL SMPOT(NID,IDX)
      CHARTEMP(1)       = MNAM(NID)
      WRITE (6, 1) CHARTEMP(1),INTTEMP(1),INTTEMP(2),REALTEMP(1)
      NUM       = NUM + 1
```

```
      IF (POIT(IDX) .EQ. 0) THEN
            GOTO 45
      ELSE
      CALL LINCOUNT(LINEC)
            IDX = POIT(IDX)
            GOTO 40
      ENDIF
45    WRITE (6,46) NUM
46    FORMAT(/,' Total number of D.C. Voltage Source =',I3,/)
      NUMB      = NUMB + NUM
47    CONTINUE
c--------listing of d.c. current source------
      LINEC = LINADD(LINEC)
      NUM       = 0
      IDX       = POIT(5)
      IF (IDX .EQ. 0) GOTO 57
      WRITE (6,51)
51    FORMAT (' D.C. Current  Source Report:',/)
      WRITE (6, 2)
50    INTTEMP(1)        = LOC(POIT(IDX+1))
      INTTEMP(2)        = LOC(POIT(IDX+2))
      REALTEMP(1)       = VAL(POIT(IDX+3))
      CALL SMPOT(NID,IDX)
      CHARTEMP(1)       = MNAM(NID)
      WRITE (6, 1) CHARTEMP(1),INTTEMP(1),INTTEMP(2),REALTEMP(1)
      NUM       = NUM + 1
      IF (POIT(IDX) .EQ. 0) THEN
            GOTO 55
      ELSE
      CALL LINCOUNT(LINEC)
            IDX = POIT(IDX)
            GOTO 50
      ENDIF
55    WRITE (6, 56) NUM
56    FORMAT (/,' Total number of D.C. Current Source =',I3,/)
      NUMB      = NUMB + NUM
57    CONTINUE
C--:
C--------listing of a.c. current source------
      LINEC = LINADD(LINEC)
      NUM       = 0
      IDX       = POIT(6)
      IF (IDX .EQ. 0) GOTO 67
      WRITE (6,61)
61    FORMAT (' A.C. Current  Source Report:',/)
      WRITE (6, 4)
60    INTTEMP(1)        = LOC(POIT(IDX+1))
      INTTEMP(2)        = LOC(POIT(IDX+2))
      REALTEMP(1)       = VAL(POIT(IDX+3))
      REALTEMP(2) = VAL(POIT(IDX+3)+ 1)
      CALL SMPOT(NID,IDX)
      CHARTEMP(1)       = MNAM(NID)
      WRITE (6, 3) CHARTEMP(1),INTTEMP(1),INTTEMP(2),REALTEMP(1),REA
MP(2)
      NUM       = NUM + 1
      IF (POIT(IDX) .EQ. 0) THEN
```

```
            GOTO 65
      ELSE
      CALL LINCOUNT(LINEC)
            IDX = POIT(IDX)
            GOTO 60
      ENDIF
65    WRITE (6, 66) NUM
66    FORMAT (/,' Total number of A.C. Current Source =',I3,/)
      NUMB      = NUMB + NUM
67    CONTINUE
c--------listing of a.c. voltage source--------
      LINEC = LINADD(LINEC)
      NUM       = 0
      IDX       = POIT(3)
      IF ( IDX .EQ. 0) GOTO 77
      WRITE (6,71)
71    FORMAT (' A.C. Voltage Source Report:',/)
      WRITE (6,4)
70    INTTEMP(1)          = LOC(POIT(IDX+1))
      INTTEMP(2)          = LOC(POIT(IDX+2))
      REALTEMP(1)         = VAL(POIT(IDX+4))
      REALTEMP(2) = VAL(POIT(IDX+4) + 1)
      CALL SMPOT(NID,IDX)
      CHARTEMP(1)         = MNAM(NID)
      WRITE (6, 3) CHARTEMP(1),INTTEMP(1),INTTEMP(2),REALTEMP(1),REA
4P(2)
      NUM       = NUM + 1
      IF (POIT(IDX) .EQ. 0) THEN
            GOTO 75
      ELSE
      CALL LINCOUNT(LINEC)
            IDX = POIT(IDX)
            GOTO 70
      ENDIF
75    WRITE (6,76) NUM
76    FORMAT(/,' Total number of A.C. Voltage Source =',I3,/)
      NUMB      = NUMB + NUM
77    CONTINUE
C--:
C  :------listing of Diode --------
C--:
      LINEC = LINADD(LINEC)
      NUM       = 0
      IDX       = POIT(14)
      IF ( IDX .EQ. 0) GOTO 87
      WRITE (6,81)
81    FORMAT (' Diode Report:',/)
      WRITE (6,6)
80    INTTEMP(1)          = LOC(POIT(IDX+1))
      INTTEMP(2)          = LOC(POIT(IDX+2))
      CALL SMPOT(NID,IDX)
      CHARTEMP(1)         = MNAM(NID)
      CHARTEMP(2) = MNAM(POIT(IDX+3))
      WRITE (6, 5) CHARTEMP(1),INTTEMP(1),INTTEMP(2),CHARTEMP(2)
      NUM       = NUM + 1
      IF (POIT(IDX) .EQ. 0) THEN
```

```
            GOTO 85
      ELSE
      CALL LINCOUNT(LINEC)
            IDX = POIT(IDX)
            GOTO 80
      ENDIF
85    WRITE (6,86) NUM
86    FORMAT(/,' Total number of Diode =',I3,/)
      NUMB      = NUMB + NUM
C--:
87    CONTINUE
C--:
C  :------listing of Transistor --------
C--:
      LINEC = LINADD(LINEC)
      NUM       = 0
      IDX       = POIT(15)
      IF ( IDX .EQ. 0) GOTO 97
      WRITE (6,91)
91    FORMAT (' Transistor Report:',/)
      WRITE (6,8)
90    INTTEMP(1)          = LOC(POIT(IDX+1))
      INTTEMP(2)          = LOC(POIT(IDX+2))
      INTTEMP(3) = LOC(POIT(IDX+3))
      CALL SMPOT(NID,IDX)
      CHARTEMP(1)         = MNAM(NID)
      CHARTEMP(2) = MNAM(POIT(IDX+4))
      WRITE (6, 7) CHARTEMP(1),INTTEMP(2),INTTEMP(1),INTTEMP(3),
     :             CHARTEMP(2)
      NUM       = NUM + 1
      IF (POIT(IDX) .EQ. 0) THEN
            GOTO 95
      ELSE
      CALL LINCOUNT(LINEC)
            IDX = POIT(IDX)
            GOTO 90
      ENDIF
95    WRITE (6,96) NUM
96    FORMAT(/,' Total number of Transistor =',I3,/)
      NUMB      = NUMB + NUM
97    CONTINUE
C--:
C  :------listing of Operational Amplifier --------
C--:
      LINEC = LINADD(LINEC)
      NUM       = 0
      IDX       = POIT(16)
      IF ( IDX .EQ. 0) GOTO 107
      WRITE (6,101)
101   FORMAT (' Operational Amplifiler Report:',/)
      WRITE (6,9)
100   INTTEMP(1)          = LOC(POIT(IDX+1))
      INTTEMP(2)          = LOC(POIT(IDX+2))
      INTTEMP(3) = LOC(POIT(IDX+3))
      CALL SMPOT(NID,IDX)
```

```
      CHARTEMP(1)        = MNAM(NID)
      CHARTEMP(2) = MNAM(POIT(IDX+5))
      WRITE (6, 7) CHARTEMP(1),INTTEMP(1),INTTEMP(2),INTTEMP(3),
     :             CHARTEMP(2)
      NUM        = NUM + 1
      IF (POIT(IDX) .EQ. 0) THEN
            GOTO 105
      ELSE
      CALL LINCOUNT(LINEC)
            IDX = POIT(IDX)
            GOTO 100
      ENDIF
105   WRITE (6,106) NUM
106   FORMAT(/,' Total number of Operational Amplifier =',I3,/)
      NUMB       = NUMB + NUM
107   CONTINUE
      RETURN
      END
C--:
C  : Line number Count :
C--:
      SUBROUTINE LINCOUNT(LINEC)
      LINEC = LINEC + 1
      IF (LINEC .GE. 13 ) THEN
         PAUSE 'Press <RETURN> to Continue'
         LINEC = 0
      ENDIF
      RETURN
      END
C--:
C  : Add Linenuber counter plus 7 for each element type :
c--:
      INTEGER*2 FUNCTION LINADD(LI)
      LINADD = LI + 7
      RETURN
      END
C-----------------------
C  :
C--: UTILITY ROUTINE :--
C  :
C-----------------------
C--:
C  :Clear Screen:
C--:
      SUBROUTINE CLSCREEN
      WRITE (*,*) CHAR(27),'[2J'
      RETURN
      END
C--:
C  : Dos Function Command call :
C--:
      SUBROUTINE DOS
      CALL CLSCREEN
      PAUSE 'Enter DOS Command >'
      RETURN
```

```
      END
C--:
C   : step input voltage devide by 20
C--:
      SUBROUTINE STEPIN(STAR,STS,NODE)
      INCLUDE   'DECLARE.INC'
      INCLUDE   'DOUBLE.INC'
      INCLUDE   'RES.INC'
      INCLUDE   'PARA.INC'
      LOGICAL*1 FLAG
      INTEGER*2 KK,MODE,NODE
C--:
      write (*,*) star,sts,node
      DO 10 I=1,200
         VAL(I)     = BUF1(I)
10    CONTINUE
      STP      = STS/10.
C--:
      MODE     = 1
      H        = 1.0E+22
      CALL SIZE(N)
C--:
      DO 1000 JK = 1,11
         KK                  = 0
100      DO 111 I=1,N
            DO 112 J=1,N
                  A(I,J)          = 0.0
112         CONTINUE
                  B(I)            = 0.0
111      CONTINUE
         CALL       LOADDC(H,MODE)
         write(*,*) 'starin',star,node
         B(NODE)    = DBLE(STAR)
         CALL       LU(N)
C--:
         write (*,*) 'kk=',kk
         CALL STORE(FLAG,H)
         IF (KK .EQ. 0) THEN
            KK      = KK + 1
         ELSEIF (.NOT. FLAG) THEN
            KK      = KK + 1
            write(*,*) (b(iik),iik=1,n)
            DO 200 I = 1,N
                IF(DABS(S(I)-B(I)) .GT. EPS) GOTO 205
200         CONTINUE
            GOTO 206
         ENDIF
205      DO 110 I = 1,N
            S(I) = B(I)
110      CONTINUE
C--:
         GOTO 100
C--:
206      DO 130 I = 1,N
            S(I)   = B(I)
```

```
130       CONTINUE
C--:
          STAR    = STAR + STP
1000   CONTINUE
       STAR = STAR - STP
       RETURN
       END
C--:
C  :Set Color:
C--:
       SUBROUTINE COLOR(FORG,BACK)
       CHARACTER*2 FORG,BACK
       WRITE (*,*) CHAR(27),'[',FORG,';',BACK,'m'
       RETURN
       END

C**********************
C  A.C. ANALYSIS
C**********************
       SUBROUTINE ACAN
       INCLUDE    'DECLARE.INC'
       INCLUDE    'COMPLEX.INC'
       REAL*4     FSTA,FEND,FTEP
       INTEGER*2 NUMB,K,KDEX
C--:                                          !check if qpoint had been do
       IF (.NOT. QPID) THEN
             WRITE (6,*)' Enter "QPOINT"'
             RETURN
       ENDIF
C--:
       OPEN (UNIT=4,FILE='C:AC.DAT',STATUS='UNKNOWN',
      :         FORM='UNFORMATTED',ACCESS='SEQUENTIAL')
       WRITE (6,*) CHAR(27),'[1;34;47m','+--------------------------

      :,CHAR(27),'[0m'
       WRITE (6,*) CHAR(27),'[1;34;47m',': FREQUENCY RESPONSE ANALYSI

      :,CHAR(27),'[0m'
       WRITE (6,*) CHAR(27),'[1;34;47m','+--------------------------

      :,CHAR(27),'[0m',
C--:
1      WRITE (6,10)
10     FORMAT (' Sweep frequency:',:)
       CALL GETL(5)
       CALL GETR(R)
C--:get start frequency
       FSTA       = R
       CALL GETR(R)
C--:get stop frequency
       FEND       = R
       CALL       GETR(R)
       FTEP       = R
       CALL GETW
C--: check solution point :--
       IF ( W(1:3) .EQ. 'LIN' ) THEN
             NUMB         = IFIX((FEND-FSTA)/FTEP) + 1
             KDEX         = 11
```

```
            KDEX        = 11
C--:set linear mode
      ELSE
            NUMB        = NINT(ALOG10(FEND/FSTA))
            KDEX        = 22
C--:set decade mode
            FTEP        = 10.0 ** (1.0/FTEP)
      ENDIF
      IF (NUMB .GT. 70 ) THEN
            WRITE (6,*)' Solution exceed than 70 points'
            GOTO 1
      ENDIF
C--: simulate :--
      K = 1
      CALL SIZE(N)
C-- :
      WRITE (*,*) CHAR(27),'[30m'
C--: calculate frequency :--
100   DO 101 I=1,N
            DO 102 J=1,N
                  CA(I,J)       = CMPLX(0.0,0.0)
102         CONTINUE
            CB(I)               = CMPLX(0.0,0.0)
101   CONTINUE
      IF ( K .EQ. 1) THEN
            FREQ        = FSTA
C--:starting frequency analysis
      ELSEIF(KDEX .EQ. 11) THEN
C--:linear mode
            FREQ        = FREQ + FTEP
      ELSEIF(KDEX .EQ. 22) THEN
C--:decade mode
            FREQ        = FREQ * FTEP
      ENDIF
      CALL LOADAC
      CALL ACLU(N)
C--: store solution :--
      DO 200 I = 1,N
            CSOL(I,K)   = CB(I)
            BTIM(K)     = FREQ
200   CONTINUE
      K = K + 1
      IF (FREQ .GE. FEND .OR. K .GT. 70 ) THEN
      REWIND (UNIT =4)
            WRITE(UNIT = 4) N,K-1,(BTIM(I),I=1,K-1)
      DO 998 I = 1,N
        DO 999 J=1,K-1
            WRITE(UNIT = 4) CSOL(I,J)
999     CONTINUE
998   CONTINUE
      CLOSE (UNIT = 4)
      ELSE
            GOTO 100
      ENDIF
      WRITE (*,*) CHAR(27),'[0m'
```

```
      RETURN
      END
C**********************
C  STAMP AC. MODEL
C**********************
      SUBROUTINE LOADAC
      INCLUDE 'DECLARE.INC'
      INCLUDE 'COMPLEX.INC'
      REAL*4  CJC,    CJE,    ALPR,   ALPF,   CES,    CCS,    GB,
     :        TYPE,   VBE,    VBC,    TEM1,   TEM2,   TEM3,   TEM4
     :        PASE,   AV,     G,      P1,     P2,     X,      Y,
     :        VD,     IS,     CC,     V1,     GD,     CD,     CK,
     :        GPI,    GMU,    GO,     GM,     CPI,    CMU
      COMPLEX*8       CCMU,   CCPI,   CGPI,   CGMU,   CGO,    CGM
C--:
      OMEG    = 2.0 * PI * FREQ
C--:
C   stamp resistor
C--:
      IDX     = POIT(1)
101   IF (IDX .EQ. 0) GOTO 110
      IROW    = POIT(IDX+1)
      JCOL    = POIT(IDX+2)
      IVAL    = POIT(IDX+3)
      R       = 1.0/VAL(IVAL)
      CR      = CMPLX(R,0.0)
      CA(IROW,IROW)   = CA(IROW,IROW) + CR
      CA(IROW,JCOL)   = CA(IROW,JCOL) - CR
      CA(JCOL,IROW)   = CA(JCOL,IROW) - CR
      CA(JCOL,JCOL)   = CA(JCOL,JCOL) + CR
      IDX     = POIT(IDX)
      GOTO 101
C--:
C  stamp capacitor
C--:
110   IDX     = POIT(8)
111   IF (IDX .EQ. 0) GOTO 120
      IROW    = POIT(IDX+1)
      JCOL    = POIT(IDX+2)
      IVAL    = POIT(IDX+3)
      R       = VAL(IVAL) * OMEG
      CR      = CMPLX(0.0,R)
      CA(IROW,IROW)   = CA(IROW,IROW) + CR
      CA(IROW,JCOL)   = CA(IROW,JCOL) - CR
      CA(JCOL,IROW)   = CA(JCOL,IROW) - CR
      CA(JCOL,JCOL)   = CA(JCOL,JCOL) + CR
      IDX     = POIT(IDX)
      GOTO 111
C--:
C  stamp inductor
C--:
120   IDX     = POIT(9)
121   IF (IDX .EQ. 0) GOTO 130
      IROW    = POIT(IDX+1)
      JCOL    = POIT(IDX+2)
```

```
          IB        = POIT(IDX+3)
          IVAL      = POIT(IDX+4)
          R         = 1.0/(VAL(IVAL) * OMEG)
          CR        = CMPLX(0.0,R)
          CA(IROW,IROW)     = CA(IROW,IROW) - CR
          CA(IROW,JCOL)     = CA(IROW,JCOL) + CR
          CA(JCOL,IROW)     = CA(JCOL,IROW) + CR
          CA(JCOL,JCOL)     = CA(JCOL,JCOL) - CR
          CA(IB,IB)         = CMPLX(1.0,0.0)
          IDX       = POIT(IDX)
          GOTO 121
C--: stamp DC voltage source :--
130       IDX       = POIT(2)
131       IF (IDX .EQ. 0 ) GOTO 140
          IROW      = POIT(IDX+1)
          JCOL      = POIT(IDX+2)
          IB        = POIT(IDX+3)
          IVAL      = POIT(IDX+4)
          R         = 0.0
C--:SET VIRTUAL GROUNDVAL(IVAL)
          CR        = CMPLX(R,0.0)
          CA(IROW,IB)       = CMPLX(+1.0,0.0)
          CA(JCOL,IB)       = CMPLX(-1.0,0.0)
          CA(IB,IROW)       = CMPLX(+1.0,0.0)
          CA(IB,JCOL)       = CMPLX(-1.0,0.0)
          CB(IB)            = CR
          IDX       = POIT(IDX)
          GOTO 131
C--: stamp AC voltage source :--
140       IDX       = POIT(3)
141       IF ( IDX .EQ. 0) GOTO 150
          IROW      = POIT(IDX+1)
          JCOL      = POIT(IDX+2)
          IB        = POIT(IDX+3)
          IVAL      = POIT(IDX+4)
          R         = VAL(IVAL)
          PASE      = VAL(IVAL+1)
          CR        = CMPLX(R*COS(PASE),R*SIN(PASE))
          CA(IROW,IB)       = CMPLX(+1.0,0.0)
          CA(JCOL,IB)       = CMPLX(-1.0,0.0)
          CA(IB,IROW)       = CMPLX(+1.0,0.0)
          CA(IB,JCOL)       = CMPLX(-1.0,0.0)
          CB(IB)            = CR
          IDX       = POIT(IDX)
          GOTO 141
C--: stamp DC current source :--
150       IDX       = POIT(5)
151       IF (IDX .EQ. 0) GOTO 160
          IROW      = POIT(IDX+1)
          JCOL      = POIT(IDX+2)
          IVAL      = POIT(IDX+3)
          R         = VAL(IVAL)
          CR        = CMPLX(R,0.0)
          CB(IROW)          = CB(IROW) - CR
          CB(JCOL)          = CB(JCOL) + CR
```

```
          IDX      = POIT(IDX)
          GOTO 151
C--: stamp AC current source :--
160       IDX      = POIT(6)
161       IF (IDX .EQ. 0) GOTO 170
          IROW     = POIT(IDX+1)
          JCOL     = POIT(IDX+2)
          IVAL     = POIT(IDX+3)
          R        = VAL(IVAL)
          PASE     = VAL(IVAL+1)
          CR       = CMPLX(R*COS(PASE),R*SIN(PASE))
          CB(IROW)          = CB(IROW) - CR
          CB(JCOL)          = CB(JCOL) + CR
          IDX      = POIT(IDX)
          GOTO 161
C--: stamp opamp :--
170       IDX      = POIT(16)
171       IF (IDX .EQ. 0) GOTO 180
          IROW     = POIT(IDX+1)
          JCOL     = POIT(IDX+2)
          IB       = POIT(IDX+3)
          JB       = POIT(IDX+4)
          MDEX     = MPOT(POIT(IDX+5)) - 1000
          IVAL     = POIT(IDX+6)
          AV       = VAL(MDEX)
          G        = 1.0/VAL(MDEX-1)
          CC       = VAL(MDEX-2) * OMEG
          R        = VAL(MDEX-3)
          P1       = VAL(MDEX-4)
          P2       = VAL(MDEX-5)
          IF (P1 .EQ. 0.0 .OR. P2 .EQ. 0.0) THEN
                   CAV      = CMPLX(AV,0.0)
          ELSE
                   X        = 1.0 - (FREQ*FREQ/(P1*P2))
                   Y        = FREQ*(P1+P2)/(P1*P2)
                   CAV      = CMPLX(AV,0.0)/CMPLX(X,Y)
          ENDIF
          CR       = CMPLX(G,CC)
          CAV      = -1.0/CAV
C--:
          CA(IROW,IROW)     = CA(IROW,IROW) + CR
          CA(IROW,JCOL)     = CA(IROW,JCOL) - CR
          CA(JCOL,IROW)     = CA(JCOL,IROW) - CR
          CA(JCOL,JCOL)     = CA(JCOL,JCOL) + CR
          CA(IB,JB)         = CA(IB,JB)     + CMPLX(+1.0,0.0)
C         CA(JB,IROW)       = CA(JB,IROW)   + CAV
C         CA(JB,JCOL)       = CA(JB,JCOL)   - CAV
C         CA(JB,IB)         = CA(JB,IB)     + CMPLX(-1.0,0.0)
C         CA(JB,JB)         = CA(JB,JB)     + CMPLX(+R,0.0)
          CA(JB,IROW)       = CA(JB,IROW)   + CMPLX(+1.0,0.0)
          CA(JB,JCOL)       = CA(JB,JCOL)   - CMPLX(+1.0,0.0)
          CA(JB,IB)         = CA(JB,IB)     - CAV
          CA(JB,JB)         = CA(JB,JB)     - (R*CAV)
          IDX      = POIT(IDX)
          GOTO 171
```

```
C--:stamp transistor:--
180     IDX     = POIT(15)
181     IF (IDX .EQ. 0) GOTO 190
        IROW    = POIT(IDX+1)
        JCOL    = POIT(IDX+2)
        IB      = POIT(IDX+3)
        MDEX    = MPOT(POIT(IDX+4))-3000
        IVAL    = POIT(IDX+5)
        CJC     = VAL(MDEX)
        CJE     = VAL(MDEX-1)
        ALPR    = VAL(MDEX-2)
        ALPF    = VAL(MDEX-3)
        CES     = VAL(MDEX-4)/ALPF
        CCS     = VAL(MDEX-4)/ALPR
        GB      = 1.0/VAL(MDEX-5)
        TYPE    = VAL(MDEX-6)
        VBE     = VAL(IVAL)
        VBC     = VAL(IVAL+1)
C-----------set parameter------
        TEM1    = EXP(VBE/SNGL(VT))
D       WRITE (*,1111) VBE,SNGL(VT),TEM1,VBC
D1111   FORMAT (' VBE,VT,TEM1,VBC',E10.3,D10.3,2E10.3)
C--:defalut set NPN transistor
        TEM2    = EXP(VBC/SNGL(VT))
        GPI     = (1.0 - ALPF) * CES/SNGL(VT)*TEM1
        GMU     = (1.0 - ALPR) * CCS/SNGL(VT)*TEM2
        GO      = CCS/SNGL(VT) * TEM2 * ALPR
        GM      = ALPF*CES/SNGL(VT)*TEM1 - CCS/SNGL(VT)*TEM2*ALPR
        CMU     = CJC * ((1.0 - VBC)**(-0.5))
        CPI     = CJE * ((1.0 - VBE)**(-0.5))
      IF ( TYPE .EQ. -2.0) THEN
                GM      = - GM
        ENDIF
        CPI     = CPI*OMEG
C--:CB junction reactance
        CMU     = CMU*OMEG
C--:CE junction reactance
        CCMU    = CMPLX(0.0,CMU)
        CCPI    = CMPLX(0.0,CPI)
        CGPI    = CMPLX(GPI,0.0)
        CGMU    = CMPLX(GMU,0.0)
        CGO     = CMPLX(GO,0.0)
        CGM     = CMPLX(GM,0.0)
C----------start EBER MOLL model-----
        CA(IROW,IROW)   = CA(IROW,IROW) + CGPI + CCPI + CGMU + CCMU
        CA(IROW,JCOL)   = CA(IROW,JCOL) - CGMU - CCMU
        CA(IROW,IB)     = CA(IROW,IB)   - CGPI - CCPI
        CA(JCOL,IROW)   = CA(JCOL,IROW) + CGM  - CGMU - CCMU
        CA(JCOL,JCOL)   = CA(JCOL,JCOL) + CGO  + CGMU + CCMU
        CA(JCOL,IB)     = CA(JCOL,IB)   - CGM  - CGO
        CA(IB,IROW)     = CA(IB,IROW)   - CGPI - CCPI - CGM
        CA(IB,JCOL)     = CA(IB,JCOL)   - CGO
        CA(IB,IB)       = CA(IB,IB)     + CGPI + CCPI + CGM + CGO
        IDX     = POIT(IDX)
        GOTO 181
```

```
C---stamp diode---
190     IDX      = POIT(14)
191     IF (IDX .EQ. 0) GOTO 200
        IROW     = POIT(IDX+1)
        JCOL     = POIT(IDX+2)
        MDEX     = MPOT(POIT(IDX+3)) - 2000
        IVAL     = POIT(IDX+4)
C----------set parameter----------
        VD       = VAL(IVAL)
        IS       = VAL(MDEX)
        CMU      = VAL(MDEX-1)
        TEM1     = EXP(VD/SNGL(VT))
        GD       = IS*TEM1/SNGL(VT)
        CD       = IS*(TEM1-1)
        CK       = CD - (GD*VD)
        CMU      = CMU * ((1.0- VD)**0.5)
        CMU      = CMU * OMEG
        CR       = CMPLX(GD,0.0)
        CAV      = CMPLX(CK,0.0)
C
        CA(IROW,IROW) = CA(IROW,IROW) + CR + CMPLX(0.0,CMU)
        CA(IROW,JCOL) = CA(IROW,JCOL) - CR + CMPLX(0.0,CMU)
        CA(JCOL,IROW) = CA(JCOL,IROW) - CR + CMPLX(0.0,CMU)
        CA(JCOL,JCOL) = CA(JCOL,JCOL) + CR + CMPLX(0.0,CMU)
        CB(IROW)      = CB(IROW) - CAV
        CB(JCOL)      = CB(JCOL) + CAV
        IDX      = POIT(IDX)
        GOTO 191
200     CONTINUE
        RETURN
        END
C************************************
C   Crout Doolittle with pivoting
C************************************
      SUBROUTINE ACLU(N)
      COMPLEX*8          SUM
D     LOGICAL*1 T,F
C
      INCLUDE   'DECLARE.INC'
      INCLUDE   'COMPLEX.INC'
      INCLUDE   'PARA.INC'
C
C
C
      I = 1
C      CALL      CPIVOT(N,I)
C
      DO 100 J = 2,N
           CA(1,J) = CA(1,J)/CA(1,1)
100   CONTINUE
      DO 900 I =2,N
           DO 800 IR = I,N
                DO 700 K = 1,I-1
                     CA(IR,I) = CA(IR,I) - CA(IR,K)*CA(K,I)
700             CONTINUE
```

```
800         CONTINUE
C           CALL CPIVOT(N,I)
            DO 600 J = I+1,N
                  SUM = CA(I,J)
                  DO 500 K = 1,I-1
                        SUM = SUM - CA(I,K)*CA(K,J)
500               CONTINUE
                  CA(I,J) = SUM/CA(I,I)
600         CONTINUE
900   CONTINUE
C---------FORWARD SUBSTITUTION---------
      CB(1) = CB(1)/CA(1,1)
      DO 910 I = 2,N
            SUM = CB(I)
            DO 810 K = 1,I-1
                  SUM = SUM - CA(I,K)*CB(K)
810         CONTINUE
            CB(I) = SUM/CA(I,I)
910   CONTINUE
C--------BACKWARD SUBSTITUTION---------
      DO 920 I = N-1,1,-1
            DO 820 K = I+1,N
                  CB(I) = CB(I) - CA(I,K)*CB(K)
820         CONTINUE
920   CONTINUE
C
      RETURN
      END
C********************
C   row pivoting
c********************
      SUBROUTINE CPIVOT(N,I)
      COMPLEX*8          PE
C
      INCLUDE   'DECLARE.INC'
      INCLUDE   'COMPLEX.INC'
      INCLUDE   'PARA.INC'
C
      PE      = CA(I,I)
      IPR     = I
      DO 100 JR = I+1,N
            IF (CABS(CA(JR,I)) .LT. CABS(PE)) GOTO 100
            PE  = CA(JR,I)
            IPR = JR
100   CONTINUE
      IF ( IPR .EQ. I ) GOTO 300
      DO 200 K = 1,N
            TEMP        = CA(IPR,K)
            CA(IPR,K)   = CA(I,K)
            CA(I,K)     = TEMP
200   CONTINUE
      TEMP      = CB(IPR)
      CB(IPR)   = CB(I)
      CB(I)     = TEMP
300   CONTINUE
```

```
C
      RETURN
      END

C**********************
C   OUTPUT SOLUTION
C**********************
      SUBROUTINE OUTPUT
      INCLUDE 'DECLARE.INC'
C      INCLUDE  'DOUBLE.INC'
      INCLUDE 'PARA.INC'
      INCLUDE 'RES.INC'
      WRITE (*, 1)
      WRITE (*, 1)
1     FORMAT (1H1)
      WRITE (*,*)'-------------------------'
      WRITE (*,*) '      OUTPUT THE SOLUTION   '
      WRITE (*,*) '-------------------------'
      WRITE (*, 11)
11    FORMAT('  Output to Filename <RETURN for NONE> :',:)
      CALL       GETL(5)
      CALL       GETW
      IF ( W(1:8) .EQ. '        ' ) THEN
         IN    =    6
      ELSE
         IN    =    8
         OPEN (UNIT=8,FILE=W,ACCESS='SEQUENTIAL',STATUS='UNKNOWN'
     &    ,FORM='FORMATTED',ERR=99)
      ENDIF
      WRITE (*,*) ' '
      WRITE (*,*) ' '
      WRITE (*,*) ' 01   :Operating point solution'
      WRITE (*,*) ' 02   :Transfer characteristic plot'
      WRITE (*,*) ' 03   :Transient characteristic plot'
      WRITE (*,*) ' 04   :Frequency response plot'
      WRITE (*,*) ' 05   :All'
      WRITE (*, 12)
12    FORMAT(//'                Your selection is:',:)
      CALL       GETL(5)
      CALL       GETINT(I)
      IF ( I .EQ. 1) THEN
            CALL PROP(IN)
      ELSEIF (I .EQ. 2) THEN
            CALL PRTF(IN)
      ELSEIF (I .EQ. 3) THEN
            CALL PRTS(IN)
      ELSEIF ( I .EQ. 4) THEN
            CALL PRAC(IN)
      ELSEIF (I .EQ. 5) THEN
            CALL PROP(IN)
            CALL PRTF(IN)
            CALL PRTS(IN)
            CALL PRAC(IN)
      ENDIF
      IF (IN .EQ. 8 ) THEN
```

```
         CLOSE(UNIT=8)
       ENDIF
       RETURN
99     WRITE (*,*) ' ERROR: no workspace for file ',w
       RETURN
       END
C*****************************
C   OPERATING POINT OUTPUT
C*****************************
       SUBROUTINE PROP(IN)
       INTEGER   TOM,    PEE
       INCLUDE 'DECLARE.INC'
       COMMON /DATE/     ID,JD,KD
       INCLUDE   'PARA.INC'
       INCLUDE   'RES.INC'
       DIMENSION TOM(40,2)
       CALL SIZE(N)
C--:
       OPEN (UNIT=1,FILE='C:QDP.DAT',STATUS='UNKNOWN',
     :         FORM='UNFORMATTED',ACCESS='SEQUENTIAL')
C--:
       REWIND (UNIT = 1)
       READ (UNIT=1,ERR=10,END=15) (ST(I,1),I= 1,N)
15     CLOSE (UNIT = 1)
C--:
       WRITE (IN,25)
       WRITE (IN,26)
       WRITE (IN,27)
       WRITE (IN,30)
       WRITE (IN,31)
25     FORMAT (' :-------------------------------------------------:')
26     FORMAT (' :-- SPEC --:      Operating point solution         :')
27     FORMAT (' :-------------------------------------------------:')
30     FORMAT ('    voltage node         value')
31     FORMAT ('    ------------         -----')
       DO 100 I = 1,N
            TOM(I,1) = LOC(I)
            TOM(I,2) = I
100    CONTINUE
       DO 200 I = 1,N
            PEE = TOM(I,1)
            ND  = TOM(I,2)
D        WRITE (*,*) PEE,ND
            DO 300 J = I,N
                  IF ( PEE .GT. TOM(J,1)) THEN
                        PEE = TOM(J,1)
                        ND  = TOM(J,2)
                        NK  = J
                  ENDIF
300         CONTINUE
            IF ( PEE .NE. TOM(I,1)) THEN
                  TOM(NK,1) = TOM(I,1)
                  TOM(NK,2) = TOM(I,2)
            ENDIF
            IF ( PEE .NE. -1) THEN
```

```
                  WRITE (IN,350)PEE,ST(ND,1)
350               FORMAT('      ',I6,'         ',1P,E10.3,' Volts')
          ENDIF
200   CONTINUE
      RETURN
C--:
10    WRITE (*,*) ' ERROR: Open file error'
      RETURN
C--:
      END

C*********************************
C   OUTPUT TRANSFER RESPONSE
C*********************************
      SUBROUTINE PRTF(IN)
      INCLUDE 'DECLARE.INC'
      COMMON /DATE/ ID,JD,KD
      INCLUDE 'PARA.INC'
      INCLUDE   'RES.INC'
      WRITE (IN,1)
1     FORMAT (1H1)
      WRITE (IN,10)
      WRITE (IN,11)
      WRITE (IN,10)
10    FORMAT (' :-------------------------------------------------------

11    FORMAT (' : -- SPEC --    Transfer characteristic response

      OPEN (UNIT=2,FILE='C:TRANF.DAT',STATUS='UNKNOWN',
     :        FORM='UNFORMATTED',ACCESS='SEQUENTIAL')
      REWIND (UNIT =2)
      READ (UNIT=2,END=20) N,NUM,STP,(ST(I,1),I=1,NUM)
      DO 2302 I=1,N
         DO 2301 J=1,70
            READ(UNIT=2) SOL(I,J)
2301  CONTINUE
2302   CONTINUE
C
20    CLOSE (UNIT = 2)
      WRITE (*, 21)
21    FORMAT(' Plot Voltage node:',:)
      CALL       GETL(5)
      CALL       GETINT(I)
      N1         = I
      CALL       CLOC(N1,I)
      NN1        = I
      CALL       GETINT(I)
      N2         = I
      CALL       CLOC(N2,I)
      NN2        = I
      IF (N2 .EQ. 0) THEN
          MD = 1
      ELSE
          MD = 2
      ENDIF
      DO 100 I=1,NUM+1
            ST(I,2)      = SOL(NN1,I)
```

```
            ST(I,2)      = SOL(NN1,I)
100    CONTINUE
       IF ( MD .EQ. 2) THEN
            DO 200 I = 1,NUM+1
                  ST(I,3)       = SOL(NN2,I)
200         CONTINUE
       ENDIF
       WRITE (IN,201) N1,N2
       DO 300 I=1,NUM+1
            WRITE(IN,205)I,ST(I,1),ST(I,2),ST(I,3)
300    CONTINUE
201    FORMAT(//,' Step  Input Voltage   V. Node(',I3,')   V. Node(',
      &I3,')')
205    FORMAT (I4,1P,3E15.3)
       CALL PLOT(ST,MD,NUM,IN,N1,N2)
       RETURN
       END
C************
C   PLOT
C************
C       MD=NUMBER OF VARIABLE TO BE PLOTTED
C       NLL =NUMBER OF DATA STEP
C       IN=DEVICE NUMBER
C       AND=DATA ARRAY TO BE PLOTTED
C-----------------------------------------------
       SUBROUTINE PLOT(ANS,MD,NLL,IN,N1,N2)
       INCLUDE 'DECLARE.INC'
       INCLUDE 'PARA.INC'
       INCLUDE   'RES.INC'
       REAL      ANS,    HEAD
       CHARACTER*62      MAR1,   MAR2,   MAR3
       DIMENSION ANS(70,3),      HEAD(7)
C       EQUIVALENCE      (ANS(1,1),ST(1,1))
       MAR1='+.........+.........+.........+.........+.........+.....
      &+'
       MAR2=':         :         :         :         :         :
      &:'
C--FIND YMAX,YMIN--
       DO 200 I = 2,MD+1
            YMIN = ANS(1,I)
            YMAX = YMIN
            DO 100 J = 1,NLL
                 IF (ANS(J,I) .LT. YMIN) THEN
                       YMIN = ANS(J,I)
                 ELSEIF (ANS(J,I) .GT. YMAX) THEN
                       YMAX = ANS(J,I)
                 ENDIF
100         CONTINUE
            IF ( I .EQ. 2) THEN
                  YMIA = YMIN
                  YMAA = YMAX
            ELSEIF (I .EQ. 3) THEN
                  YMIB = YMIN
                  YMAB = YMAX
            ENDIF
```

```
200   CONTINUE
      YSCA =(YMAA - YMIA)/50.
C--FIND YBAS--
      YBAA = YMIA - YSCA*5.
      IF ( MD .EQ. 2) THEN
            YSCB = (YMAB - YMIB)/50.
            YBAB = YMIB - YSCB*5.
      ENDIF
      IF (N1 .EQ. 9999) THEN
            WRITE(IN,*)' A = magnitude      B = phase'
      ELSE
            WRITE (IN,201) N1,N2
            WRITE (IN,202) YSCA,YSCB
201         FORMAT (//,' V(',I2,' )= A    V(',I2,' )= B')
202         FORMAT (//,' SCALE A=',1PE10.1,'     SCALE B=',1PE10.1,//)
      ENDIF
C--PRINT HEADING--
      DO 400 I =1,7
            HEAD(I)      = YBAA
            YBAA         = YBAA + YSCA*10.0
400   CONTINUE
      WRITE(IN,500) (HEAD(I),I=1,7)
500   FORMAT(' A-->',1P,8E10.1)
      IF (MD .EQ. 2) THEN
            DO 600 I = 1,7
                  HEAD(I)        = YBAB
600               YBAB           = YBAB + YSCB*10.0
            CONTINUE
            WRITE(IN,501) (HEAD(I),I=1,7)
501         FORMAT (' B-->',1P,8E10.1)
      ENDIF
      WRITE(IN,10)MAR1
10    FORMAT(10X,A61)
C---FIND BASE VARIABLE PRINT POSITION---
      XBA        = ANS(1,1)
      K          = 1
      L          = 1
      DO 1000 I=1,NLL
            IF ( K .EQ. 5) THEN
                  MAR3   = MAR1
            ELSE
                  MAR3   = MAR2
            ENDIF
            XPR = ST(I,1)
                  JPA    = ((ANS(L,2)-YMIA)/YSCA)+6.0
                  IF ( MD .EQ. 2) JPB =((ANS(L,3)-YMIB)/YSCB)+6.0
                  MAR3(JPA:JPA) = 'A'
                  IF (MD .EQ. 2) MAR3(JPB:JPB) = 'B'
                  IF ( JPA .EQ. JPB ) MAR3(JPB:JPB) = '*'
                  WRITE(IN,20) XPR,MAR3
20                FORMAT (1PE9.1,1X,A61)
                  IF (K .EQ. 5) K = 0
                  K = K+1
                  L = L+1
1000  CONTINUE
```

```
      RETURN
      END
C**********************
C  FIND NODE LOCATION
C**********************
C       ;INPUT = KIN ->NODE NUMBER TO BE CHECKED
C       ;OUTPUT= KID ->MATRIX POSITION
C
      SUBROUTINE CLOC(KID,KIN)
      INCLUDE   'DECLARE.INC'
      DO 100 I=1,60
            IF ( KID .EQ. LOC(I) ) THEN
                  KIN   = I
                  RETURN
            ENDIF
100   CONTINUE
      END
C****************************
C  PLOT TRANSIENT RESPONSE
C****************************
      SUBROUTINE PRTS(IN)
      INCLUDE 'DECLARE.INC'
C      INCLUDE 'DOUBLE.INC'
      INCLUDE 'PARA.INC'
      INCLUDE   'RES.INC'
      COMMON /DATE/     ID,JD,KD
      WRITE (IN,10)
      WRITE (IN,11)
      WRITE (IN,10)
10    FORMAT (' :-------------------------------------------------:')
11    FORMAT (' : -- SPEC --                    Transient response    :')
C--:
      OPEN (UNIT=3,FILE='C:SIENT.DAT',STATUS='UNKNOWN',
     :        FORM='UNFORMATTED',ACCESS='SEQUENTIAL')
C--:
      REWIND (UNIT = 3)
      READ (UNIT=3,END=20) N,(BTIM(I),I=1,70)
      DO 999 I = 1,N
         DO 998 J = 1,70
            READ (UNIT=3,END=20) SOL(I,J)
998      CONTINUE
999   CONTINUE
20    CLOSE (UNIT = 3)
      WRITE (*, 21)
21    FORMAT (' Plot Voltage node:',:)
      CALL GETL(5)
      CALL GETINT(I)
      N1          = I
      CALL CLOC(N1,I)
      NN1         = I
      CALL GETINT(I)
      N2          = I
      CALL CLOC(N2,I)
      NN2         = I
      IF (N2 .EQ. 0) THEN
```

```
            MD   = 1
      ELSE
            MD   = 2
      ENDIF
      WRITE (*,400) DATATEMP(1),DATATEMP(2)
400   FORMAT (' Data to be Ploted range from ',f10.3,'to ',f10.3)
25    WRITE (*, 30)
30    FORMAT(' From Starting time:',:)
      CALL GETL(5)
      CALL GETR(R)
      TIS         = R
      WRITE (*, 31)
31    FORMAT(' To Stop time:',:)
      CALL GETL(5)
      CALL GETR(R)
      TIF         = R
32    FORMAT(' Step:',:)
      CALL GETL(5)
      CALL GETR(R)
      HH          = R
      NUM         = IFIX(ABS(TIF-TIS)/HH)
      IF (NUM .GT. 50) THEN
            WRITE (*,*) ' Point times exceed than 50 points'
            GOTO 25
      ENDIF
      TIME        = TIS
      NSTP        = 1
      I = 1
100   IF (TIME .LT. BTIM(NSTP)) THEN
            NSTP          = NSTP+1
            GOTO 100
      ENDIF
200   T1          = BTIM(NSTP)
      T2          = BTIM(NSTP+1)
      IF (TIME .GE. T2) THEN
            NSTP          = NSTP+1
            GOTO 200
      ENDIF
      ST(I,1)     = TIME
      V1          = SOL(NN1,NSTP)
      V2          = SOL(NN1,NSTP+1)

      ST(I,2)     = V1+(TIME-T1)/(T2-T1)*(V2-V1)
      V1          = SOL(NN2,NSTP)
      V2          = SOL(NN2,NSTP+1)
      ST(I,3)     = V1+ (TIME-T1)/(T2-T1)*(V2-V1)
      IF (TIME .GT. TIF) THEN
            NUM = I - 1
            GOTO 222
      ELSEIF (I .GE. 50) THEN
            NUM = I - 1
            GOTO 222
      ELSE
            I             = I + 1
            TIME          = TIME + HH
```

```
            GOTO 100
         ENDIF
 222     WRITE (IN,201) N1,N2
         DO 300 I = 1,NUM
            WRITE (IN,205) ST(I,1),ST(I,2),ST(I,3)
 300     CONTINUE
 201     FORMAT (//,' Time          V.node(',I3,')','    V.Node(',I3,
        & ')')
 205     FORMAT (3E15.7)
         CALL PLOT(ST,MD,NUM,IN,N1,N2)
         RETURN
         END
C*****************************
C  AC FREQUENCY RESPONSE PLOT
C*****************************
         SUBROUTINE PRAC(IN)
         INCLUDE 'DECLARE.INC'
         INCLUDE 'COMPLEX.INC'
         INCLUDE 'RES.INC'
         COMMON /DATE/    ID,JD,KD
         INTEGER*2        K,N1
         REAL*4           MAG,PH
         REAL*4  X,Y
         WRITE (*,*) CHAR(27),'[44;36;1m',':----------------------------
--------
        :-----:',CHAR(27),'[0m'
         WRITE (*,*) CHAR(27),'[44;36;1m',': -- SPEC --          Frequenc
response
        :      :',CHAR(27),'[0m'
         WRITE (*,*) CHAR(27),'[44;36;1m',':----------------------------
--------
        :-----:',CHAR(27),'[0m'
         WRITE (*,*) CHAR(27),'[36m'
         WRITE (*,10)
 10      FORMAT (' PRINT V NODE:',:)
         READ (5,*) N1
         CALL CLOC(N1,I)
         NN1      = I
         OPEN (UNIT=4,FILE='C:AC.DAT',STATUS='UNKNOWN',
        :        FORM='UNFORMATTED',ACCESS='SEQUENTIAL')
         REWIND (UNIT=4)
         READ (UNIT =4,END=11) N,K,(BTIM(I),I=1,K)
         DO 998 I = 1,N
            DO 999 J = 1,K
               READ (UNIT=4,END=11) CSOL(I,J)
 999        CONTINUE
 998     CONTINUE
 11      CLOSE (UNIT = 4)
 12        FORMAT('    ')
           WRITE (IN,12)
           WRITE (IN,12)
           WRITE (IN,104) N1
 104       FORMAT(' VOLTAGE NODE',I3)
           WRITE(IN,12)
           WRITE (IN,106)
           WRITE (IN,105)
           WRITE (IN,106)
 105       FORMAT('     FREQUENCY       MAGNITUDE       PHASE')
```

```
106     FORMAT('     ---------       ---------       -----')
        DO 100 I = 1,K-1
                CR      = CSOL(NN1,I)
                X       = REAL(CSOL(NN1,I))
                Y       = AIMAG(CSOL(NN1,I))
                MAG     = CABS(CR)
                PH      = ATAN2(Y,X)*180.0/PI
                WRITE (IN,107) BTIM(I),MAG,PH
107             FORMAT(1P,E13.3,E15.3,0P,F12.3)
                ST(I,1) = BTIM(I)
                ST(I,2) = MAG
                ST(I,3) = PH
100     CONTINUE
        MD      = 2
        N1      = 9999
        N2      = 0
        WRITE (IN,111)
111     FORMAT (1H1,'   ')
        CALL PLOT(ST,MD,K,IN,N1,N2)
      WRITE (*,*) CHAR(27),'[0m'
        RETURN
        END
C*******************************
C  CHANGE VALUE OF THE ELEMENT
C*******************************
        SUBROUTINE      CHANGE
        INCLUDE 'DECLARE.INC'
        INCLUDE 'PARA.INC'
        WRITE (6,10)
10      FORMAT (' Name:',:)
        READ (5,11) W
      DO 17 I = 1,LEN(W)
         CALL UPPERCON(W(I:1))
17    CONTINUE
        DO 20 I=1,60
                IF (W .EQ. MNAM(I) ) GOTO 21
20      CONTINUE
C--:
      WRITE (*,14) W
14    FORMAT ( ' No Circuit elements Named ',A12,/)
      RETURN
C--:
11      FORMAT (A)
21      MP      = I
        WRITE (6,22) MNAM(MP)
22      FORMAT (' Change ',A8,' From ',:)
C--:change resistance:--
        IF (W(1:1) .EQ. 'R' .OR. W(1:3) .EQ. 'IDC') THEN
                IDX     = MPOT(MP)
                IVAL    = POIT(IDX + 3)
                WRITE (6,25) VAL(IVAL)
                CALL    GETL(5)
                CALL    GETR(R)
                VAL(IVAL)       = R
                WRITE(6,26) MNAM(MP),R
```

```
25              FORMAT (1P,E9.2,' To:',:)
26              FORMAT (' Change  ',A8,'To ',1P,E9.2,' Completed')
        ENDIF
C--:
C   : change voltage source :
c--:
      IF (W(1:3) .EQ. 'VDC') THEN
         IDX      = MPOT(MP)
         IVAL     = POIT(IDX + 4)
         WRITE (6,25) VAL(IVAL)
         CALL  GETL(5)
         CALL  GETR(R)
         VAL(IVAL) = R
         WRITE (6,26) MNAM(MP) ,R
      ENDIF
        RETURN
        END
```

ประวัติ

นายเรวัตร หวังปรีดาเลิศกุล สำเร็จการศึกษาวิศวกรรมศาสตรบัณฑิต จาก สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง ในปี พ.ศ. 2523 ปัจจุบันรับราชการ ในตำแหน่งอาจารย์ สังกัดภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่