OSA

FET MODELLING USING THE FETSCG AND FETSCG1 SYSTEMS, PART II: LISTING

OSA-87-FC-2-L

February 23, 1987

Optimization Systems Associates

163 Watson's Lane, Dundas, Ontario, Canada

FET MODELLING USING THE FETSCG AND FETSCG1 SYSTEMS, PART II: LISTING

OSA-87-FC-2-L

February 23, 1987

SUBROUTINES OF THE FETSCG AND FETSCG1 SYSTEMS

All subroutines of the FETSCG and FETSCG1 systems are listed in full in this report.

Routines for the FETSCG system

File name	Routine	Index
FETSCG	BRIEF	LISTING-4
	CMPARE	LISTING-5
	FETMDL	LISTING-6
	FETSIM	LISTING-9
	GRAD	LISTING-10
	HEAD	LISTING-11
	HELP	LISTING-12
	OPTIMI	LISTING-13
	PARSER	LISTING-15
	PRESENT	LISTING-18
	SAVCKT	LISTING-18
	WDA	LISTING-19
	WPA	LISTING-20
FETFDFG	CSOLU	LISTING-21
	FDF	LISTING-23
	SIMFET	LISTING-26
SCFILE	BLOCK DATA A	LISTING-30
	BLKBLK	LISTING-31
	CHKTYP	LISTING-33
	DATABLK	LISTING-34
	ERMSG	LISTING-35
	FERMSG	LISTING-36
	FREQBLK	LISTING-37
	GETCHAR	LISTING-39
	GETFET	LISTING-40
	GETFRQ	LISTING-43
	GETINT	LISTING-44
	GETREAL	LISTING-45
	GETSYMBL	LISTING-46
	GETUNIT	LISTING-47
	GETVAR	LISTING-48
	GETWRD	LISTING-50
	NODUMMY	LISTING-51
	NOSYMBL	LISTING-52
	OPTBLK	LISTING-53
	OUTBLK	LISTING-56
	PAD	LISTING-57
	RDCKT	LISTING-58
	RDSCF	LISTING-61
FETPLT	HPPLT	LISTING-63
	FETPLT	LISTING-68

Routines for the FETSCG1 system

Routines common to both the FETSCG and FETSCG1 systems

Assembly language files common to both the FETSCG and FETSCG1 systems

```
c*********************************************************************

c       file name: FETSCG.FOR

c       for program FETSCG (PC/AT version).
c       this file contains the main program itself: FETMDL and other
c       related subroutines

c*********************************************************************

$NOFLOATCALLS



c       prepare interface to C utility.   ref: Microsoft CEXEC.LIB

        interface to integer*2 function spawn
     +          [c,varying,alias:'spawnlp']
     +          (mode)
        integer*2 mode
        end
        interface to integer*2 function system [c]
     +           (string[reference])
        character*1 string
        end



        SUBROUTINE BRIEF(FILNAM,CKTFIL,NCKT)

c       display a summary of the input infomation

        CHARACTER*12 FILNAM,CKTFIL(6)
        IF(NCKT.EQ.0)CALL ERMSG(32,I,J,*20)
        KK=NCKT+3
        WRITE(*,10)FILNAM,NCKT,(CKTFIL(K),K=4,KK)
10      FORMAT(5X,'Working file: ',A12/5X,'Number of circuits: ',I2/5X,
     *  'Measurement files: ',3(A12,3X))
20      RETURN
        END
```

```
      SUBROUTINE CMPARE(JOPT,NCKT,PA,PA0,PDIM,CKTNAME)
      PARAMETER (MP=11,MX=40,MF=20)

c     compare the optimal solution with the starting point
c     (command 'COMPARE')    starting point in PA0, solution in PA

      DIMENSION PA(MP,3),PDIM(MP),PP(6),PA0(MP,3)
      CHARACTER CKTNAME(6)*8,PN(MP)*8,FMT1(3)*15,FMT3(3)*22
      CHARACTER C1*8,C2*8,FMT2(3)*19
      SAVE PN,FMT1,FMT2,FMT3
      DATA PN/'RG  (OH)','RD  (OH)','GDS(/OH)','RI  (OH)','RS  (OH)',
     *'LS  (nH)','CGS (pF)','CDG (pF)','CDS (pF)','G  (/OH)','T   (ps)'/
      DATA FMT1/'(13X,A8)       ','(11X,A8,22X,A8)','(7X,3(A8,14X)) '/
      DATA FMT2,FMT3/'(7X,A8,7X,A8)      ','(2(5X,A8,5X,A8,4X))',
     * '(3(2X,A8,2X,A8,2X))','(2(1X,F14.5),2X,A8)   ',
     *'(2(2(3X,F10.4),4X),A8)','(3(2(2X,F8.4),2X),A8) '/
      IF(NCKT.LE.0)CALL ERMSG(32,I,J,*30)
      IF(JOPT.EQ.0)CALL ERMSG(35,I,J,*30)
      C1='starting'
      C2='solution'
      WRITE(*,*)'          FET MODEL PARAMETERS'
      WRITE(*,FMT1(NCKT))(CKTNAME(K),K=1,NCKT)
      WRITE(*,FMT2(NCKT))(C1,C2,K=1,NCKT)
      KK=2*NCKT
      DO 20 I=1,MP
      DO 10 K=1,NCKT
      K2=2*K
      K1=K2-1
      PP(K1)=PA0(I,K)
10    PP(K2)=PA(I,K)
      WRITE(*,FMT3(NCKT))(PP(K),K=1,KK),PN(I)
20    CONTINUE
      WRITE(*,*)
30    RETURN
      END
```

```
      PROGRAM FETMDL

c     main program of FETSCG
c     A FET modelling program developed for Raytheon Company
c     equivalent circuit has 11 parameters

      PARAMETER (MP=11,MX=40,MF=20)
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     DIMENSIONAL LIMITS:     MP  maximum number of circuit parameters
c                             MX  maximum number of variables
c                             MF  maximum number of frequencies
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,A*80,KYWRD*2,STRG*19
      CHARACTER FILNAM*12,OLDFIL*12,FILEC*27
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3),X(MX)
      DIMENSION PA(MP,3),INPX(MP,3),BND(2,MX),INDX(MX),WGT(8,MF,3)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),FREQ(MF),INPO(MP)
      DIMENSION PDIM(MP),DAT(8,MF,3),FM(8,MF,3),F1(8,MF,3),INPP(MP)
      INTEGER*2 SYSTEM,SPAWN,ICXC
      LOGICAL THERE
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1     TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/BLK1/PDIM,DAT,INPP,PAO(MP,3)
      DATA INPO/1,3,4,2,8,9,6,5,7,10,11/

c     make sure the auxiliary files are present

      INQUIRE(FILE='FETSCG.MSG',EXIST=THERE)
      IF(.NOT.THERE)CALL FERMSG(4,'FETSCG.MSG  ',I,*2)
      INQUIRE(FILE='CED.EXE',EXIST=THERE)
      IF(.NOT.THERE)CALL FERMSG(4,'CED.EXE     ',I,*2)

      CALL HEAD
      CALL HELP
      OPEN(1,FILE='FETSCG.MSG',STATUS='OLD')
      DO 200 I=1,40
200   READ(1,'(A)')MSG(I)
      CLOSE(1)
      DO 210 I=1,MP
210   INPP(I)=INPO(I)

c     JOPT: have we done optimization?  0/1 = no/yes
c     JERR: have we done simulation?  0/1 = no/yes
c     NCKT: have we read circuit files?  0/(1,2,3)  = no/yes

      JOPT=0
      JERR=0
      NCKT=0

c     loop. read, interprete and act upon a command

100   WRITE(*,'(A$)')' COMD>'
      READ(*,'(A)')A
```

```
      CALL PARSER(A,KYCD,FILNAM)
      GOTO(1,1,2,3,4,5,6,7,7,15,8,9,14,10,11,12,13,100),KYCD

c     commands 'L1' and 'L2'

1     CALL OPTIMI(JOPT,JERR,FM,F1,KYCD)
      IF(JOPT.EQ.0)GOTO 100
      GOTO 6

2     STOP

c     command 'DIR'

3     icxc=system('dir/w'c)
      WRITE(*,*)
      GOTO 100

4     CALL HELP
      GOTO 100

c     command 'PARAMETER'

5     CALL WPA(NCKT,PA,PDIM,OLDFIL,CKTNAME)
      GOTO 100

c     command 'COMPARE'

6     CALL CMPARE(JOPT,NCKT,PA,PAO,PDIM,CKTNAME)
      GOTO 100

c     commands 'SWEEP' and '%ER'

7     CALL FETSIM(FM,F1,JERR)
      CALL WDA(FM,F1,KYCD)
      GOTO 100

c     command 'SENSITIVITY'

8     CALL GRAD(PDIM,CKTNAME)
      GOTO 100

c     command 'PRESENT'

9     CALL PRESENT
      GOTO 100

c     command 'SAVE'

10    CALL SAVCKT(FILNAM,OLDFIL,NX,X)
      GOTO 100

c     command 'READ'

11    CALL RDCKT(FILNAM,OLDFIL,IERROR)
      JOPT=0
```

```
        JERR=0
        GOTO 100

c       command 'EDIT'

12      filec(1:12)=filnam
        filec(13:13)=' '
        filec(14:25)=filnam
        filec(26:26)=char(0)
        filec(27:27)=char(0)
        icxc=spawn(0,loc('ced'c),loc('ced'c),loc(filec),int4(0))
        GOTO 100

c       command 'TYPE'

13      STRG(1:5)='TYPE '
        STRG(6:17)=FILNAM
        STRG(18:18)=CHAR(0)
        STRG(19:19)=CHAR(0)
        ICXC=SYSTEM(STRG)
        GOTO 100

c       command 'BRIEFING'

14      CALL BRIEF(OLDFIL,CKTFIL,NCKT)
        GOTO 100

c       command 'PLOT'

15      CALL FETSIM(FM,F1,JERR)
        CALL FETPLT(FM,1)
        GOTO 100

        END
```

```
      SUBROUTINE FETSIM(FM,F1,JERR)
      PARAMETER (MP=11,MX=40,MF=20)

c     organize model simulation for all frequency points
c     call SIMFET for each frequency

      DIMENSION INXMDA(3),PA(MP,3),INPX(MP,3),BND(2,MX),WGT(8,MF,3)
      DIMENSION INDX(MX),FREQ(MF),F(8),DF(8,MP),INPP(MP),X(MX)
      DIMENSION PDIM(MP),DAT(8,MF,3),FM(8,MF,3),F1(8,MF,3)
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/BLK1/PDIM,DAT,INPP,PA0(MP,3)
      IF(NCKT.LE.0)CALL ERMSG(32,I,J,*100)
      IF(JERR.GT.0)RETURN
      C1=90.0/ASIN(1.0)
      DO 90 I=1,NCKT
      DO 50 J=1,NFR
      CALL SIMFET(FREQ(J),PA(1,I),F,DF,JERR,I)
      DO 40 K=1,4
      K1=2*K-1
      K2=K1+1
      FM(K1,J,I)=SQRT(F(K1)*F(K1)+F(K2)*F(K2))

c     avoid discontinuity in ATAN2

              TEMPA=C1*ATAN2(F(K2),F(K1))
              TEMPB=DAT(K2,J,I)
              IF(ABS(TEMPA-TEMPB).GT.180.0)THEN
                      IF(TEMPA.GT.TEMPB)THEN
                              TEMPA=TEMPA-360.0
                      ELSE
                              TEMPA=TEMPA+360.0
                      ENDIF
              ENDIF
40    FM(K2,J,I)=TEMPA

      DO 50 K=1,8
      T=DAT(K,J,I)
      IF(ABS(T).GT.0.001)THEN
              T=100.0*ABS((FM(K,J,I)/T)-1.0)
              F1(K,J,I)=AMIN1(999.0,T)
      ELSE
              F1(K,J,I)=0.0
      ENDIF
50    CONTINUE
90    CONTINUE
      JERR=1
100   RETURN
      END
```

```
      SUBROUTINE GRAD(PDIM,CKTNAME)
      PARAMETER (MP=11,MX=40,MF=20)

c     organize model sensitivity evaluation and least-squares objective
c     call SIMFET

      CHARACTER PN(MP)*8,FMT1(3)*11,FMT2(3)*19,C*8,CKTNAME(6)*8
      DIMENSION PA(MP,3),X(MX),INPX(MP,3),BND(2,MX),INDX(MX),G(MP,3)
      DIMENSION INXMDA(3),FREQ(MF),WGT(8,MF,3),PDIM(MP),F(8),DF(8,MP)
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/BLK2/SNR(8,3),KFR(MF,3),DTM(8,MF,3),KOUNT,KODE
      SAVE PN,FMT1,FMT2
      DATA PN/'RG  (OH)','RD  (OH)','GDS(/OH)','RI  (OH)','RS  (OH)',
     *'LS  (nH)','CGS (pF)','CDG (pF)','CDS (pF)','G  (/OH)','T   (ps)'/
      DATA FMT1/'(12X,A8)   ','(2(12X,A8))','(3(12X,A8))'/
      DATA FMT2/'(5X,F15.6,5X,A8)   ','(2(5X,F15.6),5X,A8)',
     *'(3(5X,F15.6),5X,A8)'/
      IF(NCKT.EQ.0)CALL ERMSG(32,I,J,*100)
      M=0
      JSEN=1
      FOBJ=0.0
      DO 10 I=1,NCKT
      DO 10 IP=1,MP
10    G(IP,I)=0.0
      DO 20 I=1,NCKT
      DO 20 J=1,NFR
      CALL SIMFET(FREQ(J),PA(1,I),F,DF,JSEN,I)
      M=M+KFR(J,I)
      DO 20 K=1,8
      WF=SNR(K,I)*WGT(K,J,I)*(F(K)-DTM(K,J,I))
      FOBJ=FOBJ+WF*WF
      WF=2.0*WF
      DO 20 IP=1,MP
20    G(IP,I)=G(IP,I)+WF*DF(K,IP)
      RM=M
      FOBJ=FOBJ*8.0/RM
      C='gradient'
      DO 30 I=1,3
      DO 30 IP=1,MP
30    G(IP,I)=G(IP,I)*PA(IP,I)*8.0/RM
      WRITE(*,90)FOBJ
      WRITE(*,FMT1(NCKT))(CKTNAME(I),I=1,NCKT)
      WRITE(*,FMT1(NCKT))(C,I=1,NCKT)
      DO 40 IP=1,MP
40    WRITE(*,FMT2(NCKT))(G(IP,I),I=1,NCKT),PN(IP)
      WRITE(*,*)
90    FORMAT(10X,'SENSITIVITY OF THE CURRENT MODEL'/10X,
     * 'least-squares error:',F15.8)
100   RETURN
      END
```

```
      SUBROUTINE HEAD
      CALL CLEAR
      WRITE(*,1)
1     FORMAT(33X,'FET MODELLING'//25X,'Interactive Command System'/)
      RETURN
      END
```

```
      SUBROUTINE HELP
      WRITE(*,10)
10    FORMAT(10X,'Commands Available')
      WRITE(*,20)
20    FORMAT(5X,'L1',12X,'start modelling using L1 optimization'/5X,
     *'L2',12X,'start modelling using least-squares optimization'/5X,
     *'STOP          stop this program and return to DOS'/5X,
     *'DIR           list the file directory'/5X,
     *'HELP          display the available commands'/5X,
     *'PARAMETER     display the current model parameters'/5X,
     *'COMPARE       compare starting point with optimization solution'/
     *5X,'%ERROR',8X,'display percentage errors in S-parameter matching'
     */5X,'SWEEP         generate S-parameters of the current model'/
     *5X,'PLOT          plot the match between model and measurement'/
     *5X,'SENSITIVITY   sensitivity analysis for the current model'/
     *5X,'SAVE fname    save current data in a CKT file'/5X,
     *'READ fname    read desired data from a file'/5X,
     *'EDIT fn.ext   edit a disk file using full sreen editor'/5X,
     *'TYPE fn.ext   display the content of a disk file'/5X,
     *'PRESENTATION  call the user-defined routine PRESENT'/5X,
     *'BRIEFING      display the current status'/)
      RETURN
      END
```

```
      SUBROUTINE OPTIMI(JOPT,JERR,FM,F1,KYCD)
      PARAMETER (MP=11,MX=40,MF=20)

c     prepare for optimization. call L1C or QUASI2

      DIMENSION PA(MP,3),X(MX),INPX(MP,3),BND(2,MX),INDX(MX),PAO(MP,3)
      DIMENSION INXMDA(3),FREQ(MF),WGT(8,MF,3),XX(MX),INPP(MP)
      DIMENSION PDIM(MP),DAT(8,MF,3),FM(8,MF,3),F1(8,MF,3)
      INTEGER*2 MODE1
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/BLK1/PDIM,DAT,INPP,PAO
      COMMON/BLK2/SNR(8,3),KFR(MF,3),DTM(8,MF,3),KOUNT,KODE
      COMMON/L11/MARK,FF1(250),DFF1(250,MX),FF(250),DFF(250,MX)
      COMMON/L12/C(MX),DC(MX,MX),CLOC(MX)

      IF(NCKT.LE.0)CALL ERMSG(32,I,J,*100)

      N=NX
      IF(NX.LE.0)CALL ERMSG(33,I,J,*100)
      JOPT=0
      L=0

c     exponential transformation on the variables
c     lower bounds on the original variables are conveniently handled

      DO 20 I=1,N
      TEMP=X(I)-BND(1,I)
      IF(TEMP.GT.4.5E-5)THEN
              XX(I)=ALOG(TEMP)
      ELSE
              XX(I)=-10.0
      ENDIF
      IF(INDX(I).EQ.2)THEN

c     transform the upper bounds on the original variables

              L=L+1
              DO 10 J=1,N
10            DC(L,J)=0.0
              DC(L,I)=-1.0
              C(L)=ALOG(BND(2,I)-BND(1,I))
      ENDIF
20    CONTINUE

      M=0
      DO 60 I=1,NCKT
      DO 60 J=1,NFR
60    M=M+KFR(J,I)

c     save the starting point

      DO 70 I=1,NCKT
      DO 70 J=1,MP
70    PAO(J,I)=PA(J,I)
```

```
      IF(M.EQ.0)CALL ERMSG(34,I,J,*100)
      JOPT=1
      KOUNT=0
      KODE=KYCD
      DX=0.2
      MAXF=1000
      EPS=0.01
      IF(ETERM.GE.1.E-8)EPS=ETERM
      LEQ=0
      CALL FETSIM(FM,F1,JERR)
      CALL FETPLT(FM,2)
      JERR=0

c     start optimization

      IF(KODE.EQ.1)CALL L1C(N,M,L,LEQ,XX,DX,EPS,MAXF)
      IF(KODE.EQ.2)CALL QUASI2(N,M,L,LEQ,XX,DX,EPS,MAXF)
      CALL RSTXOR
      CALL CLRGRF
      CALL MODE(MODE1,3)
      WRITE(*,*)
      IF(MARK.GT.0)THEN
              WRITE(*,*)'      Optimization Completed'
      ELSE
              WRITE(*,*)'      Optimization Interrupted'
      ENDIF
      WRITE(*,'(6X,''after '',I4,'' iterations'')')MAXF

c     reverse the exponential transformation on the variables

      DO 80 I=1,NX
80    X(I)=EXP(XX(I))+BND(1,I)
      DO 90 K=1,NCKT
      DO 90 I=1,MP
      IF(INPX(I,K).GT.0)PA(I,K)=X(INPX(I,K))
90    CONTINUE

100   RETURN
      END
```

```
      SUBROUTINE PARSER(A,KYCD,FILNAM)

c     interprete a input command contained in the ASCII string A
c     return a integer code KYCD

      CHARACTER A*80,WRD*80,COMD(17)*2,KYWRD*2,FILNAM*12
      LOGICAL THERE
      SAVE COMD
      DATA COMD/'L1','L2','ST','DI','HE','PA','CO','%E','SW','PL',
     *          'SE','PR','BR','SA','RE','ED','TY'/
      KYCD=18
      J0=1
      CALL GETWRD(A,J0,J1,WRD,LENG)
      IF(LENG.EQ.0)RETURN
      IF(LENG.EQ.1)GOTO 20
      KYWRD=WRD(1:2)
      DO 10 I=1,17
      IF(KYWRD.EQ.COMD(I))GOTO 30
10    CONTINUE
20    CALL ERMSG(31,I,J,*100)

c     is it a file related command?

30    IF(I.GE.14)THEN

c     yes, search the input string for a file name

40          CALL GETWRD(A,J0,J1,WRD,LENG)
            IF(LENG.EQ.0)THEN

c           file name missing, ask for it

                WRITE(*,'(A$)')'     File name: '
                READ(*,'(A)')A
                J0=1
                GOTO 40
            ENDIF
            FILNAM='              '
            FILNAM=WRD(1:LENG)
            IF(I.GE.16)GOTO 60

c           command 'save' or 'read'. put in the omitted extension '.DAT'

            DO 50 K=1,LENG
            IF(WRD(K:K).EQ.'.')GOTO 60
50          CONTINUE
            L1=LENG+1
            IF(L1.GT.9)L1=9
            L2=L1+3
            FILNAM(L1:L2)='.DAT'

60          IF(I.EQ.17.OR.I.EQ.15)THEN

c           command 'read' or 'type'. make sure the file exits
```

```
                INQUIRE(FILE=FILNAM,EXIST=THERE)
                IF(.NOT.THERE)CALL FERMSG(1,FILNAM,I,*100)
            ENDIF
        ENDIF

        KYCD=I
100     RETURN
        END
```

```
      SUBROUTINE PRESENT
      PARAMETER (MP=11,MX=40,MF=20)

c     gateway for user implemented operations. See manual
c     basic information:
c             NCKT:   number of circuits
c             PA:     current values of circuit parameters

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3),X(MX)
      DIMENSION PA(MP,3),INPX(MP,3),BND(2,MX),INDX(MX),WGT(8,MF,3)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),FREQ(MF),PDIM(MP)
      DIMENSION DAT(8,MF,3),INPP(MP),PAO(MP,3)
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1     TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/BLK1/PDIM,DAT,INPP,PAO

      RETURN
      END
```

```
      SUBROUTINE SAVCKT(FILNAM,OLDFIL,NX,X)
      PARAMETER (MP=11,MX=40,MF=20)

c     save the current infomation to a Super-Compact file

      CHARACTER TT(100)*80,VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,FILNAM*12,OLDFIL*12,AX*8,AT*80
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),X(MX)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8)
      INTEGER*2 LENG
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1     TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/SC1/TD(6),IPT(18),XDIM(MX),VDIM(MX),IVTYP(MX)

       IF(N.LE.0)GOTO 20
       DO 10 I=1,NX
       TEMP=X(I)/XDIM(I)
       IF(TEMP.GT.99.9999.OR.TEMP.LT.0.001)THEN
               WRITE(AX,'(E8.3)')TEMP
       ELSE
               WRITE(AX,'(F8.5)')TEMP
       ENDIF
       J1=LOCX(2,I)
       J2=J1+7
       TT(LOCX(1,I))(J1:J2)=AX
10     CONTINUE
20     OPEN(2,FILE=FILNAM,STATUS='NEW')
       OLDFIL=FILNAM
       DO 30 I=1,LINET
       AT=TT(I)(1:80)
       LENG=80
       CALL STRIM(AT,LENG)
       WRITE(2,'(A)')AT(1:LENG)
30     CONTINUE
       CLOSE(2)
       RETURN
       END
```

```
      SUBROUTINE WDA(FM,F1,KYCD)

c     display simulation results (commands 'SWEEP' and '%ER')

      PARAMETER (MP=11,MX=40,MF=20)
      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,TN(8)*4,CHR
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3),X(MX)
      DIMENSION PA(MP,3),WGT(8,MF,3),INPX(MP,3),BND(2,MX),INDX(MX)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),FREQ(MF)
      DIMENSION PDIM(MP),DAT(8,MF,3),FM(8,MF,3),F1(8,MF,3),INPP(MP)
      INTEGER*2 KYCODE,IKY
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1     TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/BLK1/PDIM,DAT,INPP,PA0(MP,3)
      SAVE TN
      DATA TN/'MS11','AS11','MS21','AS21','MS12','AS12','MS22','AS22'/
      IF(NCKT.LE.0)RETURN
      I=1
10    IF(KYCD.EQ.8)THEN
              WRITE(*,110)CKTNAME(I),CKTFIL(INXMDA(I))
              WRITE(*,'(6X,''(GHZ)'',8(4X,A4))')(TN(K),K=1,8)
              DO 20 J=1,NFR
20            WRITE(*,120)FREQ(J),(F1(K,J,I),K=1,8)
      ELSE
              JO=0
30            WRITE(JO,130)CKTNAME(I)
              WRITE(JO,'(1X,''*    (GHZ)'',8(4X,A4))')(TN(K),K=1,8)
              DO 40 J=1,NFR
40            WRITE(JO,140)FREQ(J),(FM(K,J,I),K=1,8)
              IF(CKTFIL(I)(1:1).NE.' ')THEN
                      IF(JO.EQ.2)THEN
                              CLOSE(2)
                      ELSE
                              OPEN(2,FILE=CKTFIL(I),STATUS='NEW')
                              JO=2
                              GOTO 30
                      ENDIF
              ENDIF
      ENDIF
      IF(I.GE.NCKT)GOTO 100
      WRITE(*,'(/5X,''Press any key to continue ...'')')
      IKY=KYCODE(IKY)
      I=I+1
      WRITE(*,*)
      GOTO 10
100   RETURN
110   FORMAT(/19X,'PERCENTAGE ERRORS IN S-PARAMETER MATCHING'/20X,
     *'ABS[ (Model-Measurement)/Measurement ] %'/3X,'Frequency',7X,
     *'model: ',A8,6X,'measurement: ',A12)
120   FORMAT(1X,F9.1,2X,8(F7.2,'%'))
130   FORMAT(1X,'*',15X,'S-PARAMETERS OF MODEL ',A8/1X,'*  Frequency')
140   FORMAT(1X,F9.1,2X,4(F8.4,F8.2))
      END
```

```
      SUBROUTINE WPA(NCKT,PA,PDIM,FILNAM,CKTNAME)
      PARAMETER (MP=11,MX=40,MF=20)

c     display the current values of the parameters (command 'PA')

      DIMENSION PA(MP,3),PDIM(MP),PP(3)
      CHARACTER CKTNAME(6)*8,PN(MP)*8,FMT1(3)*15,FMT2(3)*19
      CHARACTER FILNAM*12
      SAVE PN,FMT1,FMT2
      DATA PN/'RG  (OH)','RD  (OH)','GDS(/OH)','RI  (OH)','RS  (OH)',
     *'LS  (nH)','CGS (pF)','CDG (pF)','CDS (pF)','G  (/OH)','T   (ps)'/
      DATA FMT1/'(16X,A8)       ','(12X,A8,10X,A8)','(5X,3(A8,10X)) '/
      DATA FMT2/'(5X,F18.5,5X,A8)   ','(1X,2F18.5,5X,A8)  ',
     *'(1X,3(F12.5,6X),A8)'/
      IF(NCKT.LE.0)CALL ERMSG(32,I,J,*30)
      WRITE(*,'(6X,''FET MODEL PARAMETERS    file: '',A12)')FILNAM
      WRITE(*,FMT1(NCKT))(CKTNAME(K),K=1,NCKT)
      DO 20 I=1,MP
      DO 10 K=1,NCKT
10    PP(K)=PA(I,K)*PDIM(I)
      WRITE(*,FMT2(NCKT))(PP(K),K=1,NCKT),PN(I)
20    CONTINUE
      WRITE(*,*)
30    RETURN
      END
```

```
c*********************************************************************

c       file name: FETFDFG.FOR

c       for program FETSCG (PC/AT version)·
c       this group of files are responsible for model simulation

c*********************************************************************

$NOFLOATCALLS



        SUBROUTINE CSOLU(Y,A,B,C,D)

c       solve a complex system of linear equations using LU factorization

        COMPLEX*8 Y(6,6),A(6),B(6),C(6),D(6),Z,T1,T2,T3,T4
        DO 20 I=1,5
        Z=Y(I,I)
        IF(Z.EQ.(0.0,0.0))GO TO 110
        M=I+1
        DO 20 J=M,6
        Y(I,J)=Y(I,J)/Z
        DO 10 K=M,6
10      Y(K,J)=Y(K,J)-Y(K,I)*Y(I,J)
20      CONTINUE

        A(1)=1.0/Y(1,1)
        C(1)=1.0
        DO 40 I=2,6
        T1=0.0
        T2=0.0
        DO 30 K=1,I-1
        T1=T1-Y(I,K)*A(K)
30      T2=T2-Y(K,I)*C(K)
        A(I)=T1/Y(I,I)
40      C(I)=T2

        B(6)=1.0/Y(6,6)
        C(6)=C(6)/Y(6,6)
        D(6)=1.0/Y(6,6)
        DO 60 L=2,6
        I=7-L
        T1=A(I)
        T2=0.0
        T3=C(I)
        T4=0.0
        DO 50 K=I+1,6
        T1=T1-Y(I,K)*A(K)
        T2=T2-Y(I,K)*B(K)
        T3=T3-Y(K,I)*C(K)
50      T4=T4-Y(K,I)*D(K)
        A(I)=T1
        B(I)=T2
```

```
      C(I)=T3/Y(I,I)
60    D(I)=T4/Y(I,I)

      RETURN

110   WRITE(*,*)'FATAL ERROR IN CSOLU: ZERO ON THE DIAGONAL'
      STOP
      END
```

```
      SUBROUTINE FDF(N,M,XX)
      PARAMETER (MP=11,MX=40,MF=20)

c     organize calculation of error functions and their derivatives
c     called by the optimizer

      CHARACTER STRG*40,WRD1*4,WRD2*11
      DIMENSION XX(N),FS(8),DFS(8,MP),INDX(MX),FM(8,MF,3)
      DIMENSION INXMDA(3),PA(MP,3),X(MX),INPX(MP,3),BND(2,MX)
      DIMENSION FREQ(MF),XA(MX),WGT(8,MF,3)
      INTEGER*2 IKY1,KYCODE,IKY2,IHIGH
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/BLK1/PDIM(MP),DAT(8,MF,3),INPP(MP),PA0(MP,3)
      COMMON/BLK2/SNR(8,3),KFR(MF,3),DTM(8,MF,3),KOUNT,KODE
      COMMON/L11/MARK,F(250),DF(250,MX),FF(250),DFF(250,MX)
      COMMON/BLKPLT/IX(10),IY(10,8,3),YMIN(8,3),YR(8,3),JOUT
      SAVE FOBJ0


      KOUNT=KOUNT+1
      M1=0
      JSEN=1
      FOBJ=0.0
      CC1=90.0/ASIN(1.0)

c     reserve the exponential transformation on the variables

      DO 10 I=1,N
      XA(I)=EXP(XX(I))
      X(I)=XA(I)+BND(1,I)
      DO 10 J=1,M
10    DF(J,I)=0.0

c     loop for NCKT circuits

      DO 60 K=1,NCKT

c             update the variable parameters

              DO 20 I=1,MP
              IF(INPX(I,K).GT.0)PA(I,K)=X(INPX(I,K))
20            CONTINUE

c     sweep for all frequencies

      DO 60 J=1,NFR
              IF(KFR(J,K).EQ.0.AND.K.NE.JOUT)GOTO 60
              CALL SIMFET(FREQ(J),PA(1,K),FS,DFS,JSEN,K)

c             8 responses (real & imag. parts of 4 s-parameters)

              DO 40 L=1,8
                      IF(WGT(L,J,K).LE.1.E-5)GOTO 40
                      M1=M1+1
                      WT=WGT(L,J,K)*SNR(L,K)
```

```
                        F(M1)=WT*(FS(L)-DTM(L,J,K))

c                       compute the objective function

                        IF(KODE.EQ.1)THEN
                                FOBJ=FOBJ+ABS(F(M1))
                        ELSE
                                FOBJ=FOBJ+F(M1)*F(M1)
                        ENDIF

c                       assemble the gradients

                        DO 30 I=1,MP
                        I1=INPX(I,K)
                        IF(I1.GT.0)DF(M1,I1)=DF(M1,I1)+WT*DFS(L,I)*XA(I1)
30                      CONTINUE
40              CONTINUE

c               transform the responses to rectangular form

                IF(K.NE.JOUT)GOTO 60
                        DO 50 I=1,4
                        I2=2*I
                        I1=I2-1
                        FM(I1,J,JOUT)=SQRT(FS(I1)*FS(I1)+FS(I2)*FS(I2))

c                       avoid discontinuity in ATAN2

                                TEMPA=CC1*ATAN2(FS(I2),FS(I1))
                                TEMPB=DAT(I2,J,JOUT)
                                IF(ABS(TEMPA-TEMPB).GT.180.0)THEN
                                        IF(TEMPA.GT.TEMPB)THEN
                                                TEMPA=TEMPA-360.0
                                        ELSE
                                                TEMPA=TEMPA+360.0
                                        ENDIF
                                ENDIF
50                      FM(I2,J,JOUT)=TEMPA
60      CONTINUE

c       check for user interrupt (key <ESC>)

        CALL CHECKY(IKY1)
        IF(IKY1.EQ.27)THEN

c               interrupt handler

                CALL CLRGRF
70              WRITE(*,80)
80              FORMAT(///15X,'USER INTERRUPT'/6X,'Press  T  to terminate',
     1           ' optimization'/6X,'Or press <CR> to continue')
                IKY2=KYCODE(IKY2)
                IHIGH=IKY2/256
                IKY2=IKY2-IHIGH*256
                IF(IKY2.EQ.84.OR.IKY2.EQ.116)THEN
```

```
c                       'T' has been pressed: terminate optimization

                        MARK=0
                        RETURN
                ELSEIF(IKY2.NE.13)THEN

c                       neither 'T' not <CR>. wait for another key

                        GOTO 70
                ENDIF
                CALL FETPLT(FM,2)
        ENDIF

c       display the count of iteration and the value of objective function

        RM=M
        FOBJ=FOBJ*8.0/RM
        WRITE(WRD1,'(I4)')KOUNT
        IF(FOBJ.GT.999.0.OR.FOBJ.LT.1.E-6)THEN
                WRITE(WRD2,'(E11.5)')FOBJ
        ELSE
                WRITE(WRD2,'(F11.6)')FOBJ
        ENDIF
        STRG(1:16)='@L01@C01@7K=    '
        DO 90 I=1,4
        IF(WRD1(I:I).NE.' ')GOTO 100
90      CONTINUE
100     I1=17-I
        STRG(13:I1)=WRD1(I:4)
        STRG(17:33)=' ER=           _'
        DO 110 I=1,11
        IF(WRD2(I:I).NE.' ')GOTO 120
110     CONTINUE
120     I1=33-I
        STRG(21:I1)=WRD2(I:11)
        CALL DISP(STRG)
        IF(KOUNT.GT.1.AND.FOBJ.GT.FOBJ0)RETURN
        FOBJ0=FOBJ
        IF(IKY1.NE.27)CALL FETPLT(FM,3)
        RETURN
        END
```

```
      SUBROUTINE SIMFET(FREQ,PA,F,DF,JSEN,ICKT)
      PARAMETER (MP=11,MX=40,MF=20)

c     organize model simulation for a single frequency (FREQ)
c     expect 11 parameters in the following order:
c     RG(OH),  RD(OH),  GDS(/OH),  RI(OH),  RS(OH),  LS(nH),  CGS(pF)
c     CDG(pF), CDS(pF), G(/OH),    T(ps)

      DIMENSION PA(MP),PAP(MP),PP(MP),F(8),DF(8,MP),DFS(8,MP)
      COMMON/BLK1/PDIM(MP),DAT(8,MF,3),INPP(MP),PAO(MP,3)
      COMMON/FETCONST/F3DB(3),GGS(3),FDIM
      COMPLEX*8 SMX(4),DSMX(4,MP),PH(6),QH(6),P(6),Q(6)
      COMPLEX*8 Y(6,6),S,Z(9),Z2P(4),ZDELTA,DZ2P(4,MP)
      COMPLEX*8 SMXMU(4),DSMXT(4),C1,C2,C3,C4,C5,C6
      INTEGER INDEX1(7),INDEX2(7)
      SAVE INDEX1,INDEX2
      DATA INDEX1,INDEX2/1,3,5,4,2,2,4,2,4,6,5,5,3,5/

      DO 1 I=1,MP
1     PAP(I)=PA(I)
      DO 2 I=1,4
      IF(I.EQ.3)GOTO 2
      IF(PAP(I).GT.1.E-6)THEN
              PAP(I)=1.0/PAP(I)
      ELSE
              PAP(I)=1.E6
      ENDIF
2     CONTINUE
      DO 3 I=1,MP
      II=INPP(I)
3     PP(II)=PAP(I)

c     assemble the Y matrix

              DO 10 I=1,6
              DO 10 J=1,6
10            Y(J,I)=(0.0,0.0)

              TWOPIF=6.283185*FREQ
              C1=CMPLX(0.0,TWOPIF)
              C2=0.001*C1
              C3=CMPLX(1.0,FREQ/F3DB(ICKT))
              B=TWOPIF*0.001*PP(11)
              C4=CMPLX(COS(B),-SIN(B))/C3

              DO 20 I=1,4
20            Z(I)=PP(I)
              DO 30 I=5,7
30            Z(I)=C2*PP(I)
              Z(8)=1.0/(PP(8)+C1*PP(9))
              Z(9)=PP(10)*C4

              Y(1,1)=Z(1)
              Y(1,2)=-Z(1)
              Y(2,2)=Z(1)+Z(6)+Z(5)+GGS(ICKT)
```

```
                Y(2,3)=-Z(6)-GGS(ICKT)
                Y(2,5)=-Z(5)
                Y(3,3)=Z(2)+Z(6)+GGS(ICKT)
                Y(3,4)=-Z(2)
                Y(4,4)=Z(2)+Z(4)+Z(7)+Z(8)
                Y(4,5)=-Z(4)-Z(7)
                Y(5,5)=Z(5)+Z(4)+Z(7)+Z(3)
                Y(5,6)=-Z(3)
                Y(6,6)=Z(3)

                DO 40 I=1,5
                DO 40 J=I+1,6
40              Y(J,I)=Y(I,J)

                Y(5,2)=Y(5,2)+Z(9)
                Y(5,3)=Y(5,3)-Z(9)
                Y(4,2)=Y(4,2)-Z(9)
                Y(4,3)=Y(4,3)+Z(9)

c       solve the nodal equation for 4 solutions

        CALL CSOLU(Y,P,Q,PH,QH)

c       construct the two-port Z matrix

                Z2P(1)=P(1)/50.0
                Z2P(2)=P(6)/50.0
                Z2P(3)=Q(1)/50.0
                Z2P(4)=Q(6)/50.0
                ZDELTA=(Z2P(1)+1.0)*(Z2P(4)+1.0)-Z2P(2)*Z2P(3)

c       compute the two-port S matrix

                SMX(1)=1.0-2.0*(Z2P(4)+1.0)/ZDELTA
                SMX(2)=2.0*Z2P(2)/ZDELTA
                SMX(3)=2.0*Z2P(3)/ZDELTA
                SMX(4)=1.0-2.0*(Z2P(1)+1.0)/ZDELTA

        IF(JSEN.EQ.0)GOTO 150

c       sensitivity analysis

                DO 60 I=1,7
                J1=INDEX1(I)
                J2=INDEX2(I)
                DZ2P(1,I)=(PH(J1)-PH(J2))*(P(J2)-P(J1))
                DZ2P(2,I)=(QH(J1)-QH(J2))*(P(J2)-P(J1))
                DZ2P(3,I)=(PH(J1)-PH(J2))*(Q(J2)-Q(J1))
60              DZ2P(4,I)=(QH(J1)-QH(J2))*(Q(J2)-Q(J1))

                DO 70 I=5,7
                DO 70 J=1,4
70              DZ2P(J,I)=C2*DZ2P(J,I)

                C5=Z(8)*Z(8)
```

```
                DZ2P(1,8)=C5*PH(4)*P(4)
                DZ2P(2,8)=C5*QH(4)*P(4)
                DZ2P(3,8)=C5*PH(4)*Q(4)
                DZ2P(4,8)=C5*QH(4)*Q(4)

                DO 80 J=1,4
80              DZ2P(J,9)=C1*DZ2P(J,8)

                DZ2P(1,10)=C4*(PH(5)-PH(4))*(P(3)-P(2))
                DZ2P(2,10)=C4*(QH(5)-QH(4))*(P(3)-P(2))
                DZ2P(3,10)=C4*(PH(5)-PH(4))*(Q(3)-Q(2))
                DZ2P(4,10)=C4*(QH(5)-QH(4))*(Q(3)-Q(2))

                C6=-PP(10)*C2
                DO 90 J=1,4
90              DZ2P(J,11)=C6*DZ2P(J,10)

                SMXMU(1)=1.0-SMX(1)
                SMXMU(2)=-SMX(2)
                SMXMU(3)=-SMX(3)
                SMXMU(4)=1.0-SMX(4)

                DO 100 I=1,11
                DSMXT(1)=SMXMU(1)*DZ2P(1,I)+SMXMU(3)*DZ2P(2,I)
                DSMXT(2)=SMXMU(2)*DZ2P(1,I)+SMXMU(4)*DZ2P(2,I)
                DSMXT(3)=SMXMU(1)*DZ2P(3,I)+SMXMU(3)*DZ2P(4,I)
                DSMXT(4)=SMXMU(2)*DZ2P(3,I)+SMXMU(4)*DZ2P(4,I)

                DSMX(1,I)=(DSMXT(1)*SMXMU(1)+DSMXT(3)*SMXMU(2))/100.0
                DSMX(2,I)=(DSMXT(2)*SMXMU(1)+DSMXT(4)*SMXMU(2))/100.0
                DSMX(3,I)=(DSMXT(1)*SMXMU(3)+DSMXT(3)*SMXMU(4))/100.0
100             DSMX(4,I)=(DSMXT(2)*SMXMU(3)+DSMXT(4)*SMXMU(4))/100.0

                DO 110 K=1,4
                K1=2*K-1
                DO 110 I=1,11
                DFS(K1,I)=REAL(DSMX(K,I))
110             DFS(K1+1,I)=AIMAG(DSMX(K,I))

                DO 120 I=1,MP
                I1=INPP(I)
                DO 120 J=1,8
120             DF(J,I)=DFS(J,I1)
                DO 140 I=1,4
                IF(I.EQ.3)GOTO 140
                PR=-PAP(I)*PAP(I)
                DO 130 J=1,8
130             DF(J,I)=DF(J,I)*PR
140             CONTINUE

c       separate the real and imaginary parts of the S-parameters

150             DO 160 K=1,4
                K1=2*K-1
                F(K1)=REAL(SMX(K))
```

```
160             F(K1+1)=AIMAG(SMX(K))

        RETURN
        END
```

```
c*********************************************************************

c       file name: SCFILE.FOR

c       for program FETSCG (PC/AT version).
c       this group of subroutines constitutes a file handler to process
c       a circuit file and the associated measurement data files

c*********************************************************************

$NOFLOATCALLS


        BLOCK DATA A
        DIMENSION TYPDIM(6),IPTYP(18),IVTYP(40),XDIM(40),VDIM(40)
        COMMON/SC1/TYPDIM,IPTYP,XDIM,VDIM,IVTYP
        DATA TYPDIM/1.E-9,1.0,1.0,1.E9,1.E12,1.E12/
        DATA IPTYP/3,5,1,6,3,5,5,5,3,2,2,2,2,5,5,4,4,4/
        END
```

```
      SUBROUTINE BLKBLK(IERROR)
      PARAMETER (MP=11,MX=40,MF=20)

c     decode the 'BLK' block

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,WRD*80,KYWRD*3,CHARR
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3),X(MX)
      DIMENSION PA(MP,3),WGT(8,MF,3),INPX(MP,3),BND(2,MX),INDX(MX)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),FREQ(MF)
      DIMENSION TYPDIM(6),IPTYP(18),IVTYP(MX),XDIM(MX),VDIM(MX)
      COMMON/SC1/TYPDIM,IPTYP,XDIM,VDIM,IVTYP
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1      TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/FETCONST/F3DB(3),GGS(3),FDIM
      LTOP=LOCBLK(1,1)
      LBOTTOM=LOCBLK(2,1)
      IERROR=1
      IF(LTOP.EQ.0.OR.LBOTTOM.LE.LTOP)CALL ERMSG(1,I,J1,*80)
      DO 10 K=1,6
10    CKTNAME(K)='        '
      NX=0
      NVAR=0
      I=LTOP
      NCKT=0
      NFET=0
20    I=I+1
      IF(I.GT.LBOTTOM)GOTO 50
      J0=1
      CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
      IF(LENG.EQ.0)GOTO 20
      CALL GETSYMBL(WRD,LENG,IP,IFLAG)
      IF(NFET.EQ.1)GOTO 30
      IF(IFLAG.EQ.3)THEN
              LENG=IP-1
              J0=J1+IP
              IF(LENG.EQ.0.OR.LENG.GT.8)CALL ERMSG(6,I,J1,*80)
              NVAR=NVAR+1
              VNAME(NVAR)='        '
              VNAME(NVAR)(1:LENG)=WRD(1:LENG)
              CALL GETVAR(I,J0,NVAR,*80)
              IF(LOCV(3,NVAR).NE.0)THEN
                      NX=NX+1
                      INVAR(NVAR)=NX
                      X(NX)=VAL(1,NVAR)
                      XDIM(NX)=VDIM(NVAR)
                      LOCX(1,NX)=LOCV(1,NVAR)
                      LOCX(2,NX)=LOCV(2,NVAR)
                      INDX(NX)=LOCV(3,NVAR)
                      IF(INDX(NX).EQ.2)THEN
                              BND(1,NX)=VAL(2,NVAR)
                              BND(2,NX)=VAL(3,NVAR)
                      ENDIF
              ENDIF
```

```
            GOTO 20
      ENDIF
      IF(LENG.NE.3.OR.WRD(1:3).NE.'FET')CALL ERMSG(14,I,1,*80)
      CALL GETFET(I,J0,*80)
      NFET=1
      GOTO 20
30    IF(IFLAG.NE.3.OR.IP.NE.LENG.OR.LENG.LT.2)CALL ERMSG(16,I,1,*80)
      LENG=LENG-1
      NND=NCKT*3-2
      CKTNAME(NCKT)=WRD(1:LENG)
      CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
      IF(LENG.NE.4.OR.WRD(1:4).NE.'2POR')CALL ERMSG(18,I,J1,*80)
      CALL GETINT(TT(1,I),J0,J1,ND1,*90)
      IF(ND1.NE.NODE(NND))CALL ERMSG(15,I,J1,*80)
      CALL GETINT(TT(1,I),J0,J1,ND2,*90)
      IF(ND2.NE.NODE(NND+1))CALL ERMSG(15,I,J1,*80)
      NFET=0
      GOTO 20
50    IF(NFET.NE.0)CALL ERMSG(16,I,1,*80)
      IERROR=0
80    RETURN
90    CALL ERMSG(7,I,J1,*80)
      END
```

```
      SUBROUTINE CHKTYP(ITY,I,*)

c     make sure that a parameter and a symbolic variable that is assigned
c     to it have consistent types (capacitor, inductor, etc.)

      DIMENSION ITY(3)
      IF(ITY(2).EQ.7)GOTO 90
      IF(ITY(1).EQ.0.OR.ITY(1).EQ.7)GOTO 20
      IF(ITY(2).NE.0.AND.ITY(1).NE.ITY(2))GOTO 90
      IF(ITY(3).EQ.0.OR.ITY(3).EQ.7)GOTO 80
      IF(ITY(1).NE.ITY(3))GOTO 90
20    IF(ITY(3).EQ.0.OR.ITY(3).EQ.7)GOTO 80
      IF(ITY(2).EQ.0)GOTO 80
      IF(ITY(2).NE.ITY(3))GOTO 90
80    IF((ITY(1).NE.0).AND.(ITY(1).NE.7))ITY(2)=ITY(1)
      IF((ITY(3).NE.0).AND.(ITY(3).NE.7))ITY(2)=ITY(3)
      RETURN
90    CALL ERMSG(29,I,1,*100)
100   RETURN 1
      END
```

```
      SUBROUTINE DATABLK(IERROR)
      PARAMETER (MP=11,MX=40,MF=20)

c     decode the 'DATA' block

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,WRD*80
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3),X(MX)
      DIMENSION PA(MP,3),WGT(8,MF,3),INPX(MP,3),BND(2,MX),INDX(MX)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),FREQ(MF)
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1     TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      DO 10 I=1,6
10    CKTFIL(I)='            '
      IERROR=1
      LTOP=LOCBLK(1,4)
      LBOTTOM=LOCBLK(2,4)
      IF(LTOP.EQ.0.OR.LTOP.GE.LBOTTOM)CALL ERMSG(35,I,J1,*100)
      LTOP=LTOP+1
      NMDA=3
      DO 90 I=LTOP,LBOTTOM
      J0=1
      CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
      IF(LENG.EQ.0)GOTO 90
      CALL GETSYMBL(WRD,LENG,IP,IFLAG)
      IF(IP.EQ.1.OR.IFLAG.NE.3)CALL ERMSG(30,I,J1,*100)
      NMDA=NMDA+1
      IF(NMDA.GT.6)CALL ERMSG(5,I,J1,*100)
      LENG=IP-1
      CKTNAME(NMDA)(1:LENG)=WRD(1:LENG)
      J0=IP+1
      CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
      IF(LENG.EQ.0)CALL ERMSG(17,I,J1,*100)
      CKTFIL(NMDA)(1:LENG)=WRD(1:LENG)
90    CONTINUE
      IERROR=0
100   RETURN
      END
```

```
      SUBROUTINE ERMSG(IER,ILINE,ICOL,*)
      PARAMETER (MP=11,MX=40,MF=20)

c     analyze an error and display the corresponding message

      CHARACTER TT(100)*80,VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,T*79
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8)
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1     TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      IF(IER.LE.3)THEN
              WRITE(*,10)MSG(IER)
      ELSEIF(IER.LE.30)THEN
              T=TT(ILINE)(1:79)
              WRITE(*,10)MSG(IER)
              WRITE(*,20)ILINE,ICOL
              WRITE(*,'(1X,A79)')T
      ELSE
              WRITE(*,30)MSG(IER)
      ENDIF
      RETURN 1
10    FORMAT(5X,'Error in circuit file: ',A30)
20    FORMAT(5X,'Error in circuit file at line ',I3,' after column',I3)
30    FORMAT(5X,'Error: ',A30)
      END
```

```
      SUBROUTINE FERMSG(IER,FILNAM,ILINE,*)

c     display file-related error messages

      CHARACTER FILNAM*12
      IF(IER.EQ.1)WRITE(*,10)FILNAM
      IF(IER.EQ.2)WRITE(*,20)FILNAM,ILINE
      IF(IER.EQ.3)WRITE(*,30)FILNAM
      IF(IER.EQ.4)WRITE(*,40)FILNAM
      RETURN 1
10    FORMAT(5X,'File not found: ',A12)
20    FORMAT(5X,'Bad data in the file ',A12,' at line',I3)
30    FORMAT(5X,'Not enough data in file ',A12)
40    FORMAT(5X,'File missing: ',A12)
      END
```

```
      SUBROUTINE FREQBLK(IERROR)
      PARAMETER (MP=11,MX=40,MF=20)

c     decode the 'FREQ' block

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,WRD*80
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3),X(MX)
      DIMENSION PA(MP,3),WGT(8,MF,3),INPX(MP,3),BND(2,MX),INDX(MX)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),FREQ(MF)
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1     TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      IERROR=1
      LTOP=LOCBLK(1,2)
      LBOTTOM=LOCBLK(2,2)
      IF(LTOP.EQ.0.OR.LTOP.GE.LBOTTOM)CALL ERMSG(2,I,J1,*100)
      LTOP=LTOP+1
      NFR=0
      EPS=1.E-12
      DO 30 I=LTOP,LBOTTOM
      J0=1
10    J00=J0
      CALL GETWRD(TT(1,I),J00,J1,WRD,LENG)
      IF(LENG.EQ.0)GOTO 30
      II=ICHAR(WRD(1:1))
      IF(II.GE.65.AND.II.LE.122)THEN
              CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
              IF(LENG.EQ.4.AND.WRD(1:4).EQ.'STEP')THEN
                      CALL GETFRQ(I,J0,J1,FLOW,*100)
                      CALL GETFRQ(I,J0,J1,FHIGH,*100)
                      CALL GETFRQ(I,J0,J1,DFR,*100)
                      IF(DFR.LT.EPS)GOTO 120
                      IF(FLOW.LT.EPS)GOTO 120
                      IF(FHIGH.LT.EPS)GOTO 120
                      IF(FHIGH.LT.FLOW)THEN
                              TEMP=FHIGH
                              FHIGH=FLOW
                              FLOW=TEMP
                      ENDIF
                      N1=INT((FHIGH-FLOW)/DFR)+1
                      N2=NFR+N1
                      IF(N2.GT.MF)GOTO 110
                      DO 20 J=1,N1
                      FREQ(NFR+J)=FLOW
20                    FLOW=FLOW+DFR
                      NFR=N2
              ELSE
                      CALL ERMSG(8,I,J1,*100)
              ENDIF
      ELSE
              IF(NFR.GE.MF)GOTO 110
              NFR=NFR+1
              CALL GETFRQ(I,J0,J1,FREQ(NFR),*100)
      ENDIF
```

```
      GOTO 10
30    CONTINUE
      IF(NFR.EQ.0)CALL ERMSG(2,I,J1,*100)
      IERROR=0
100   RETURN
110   CALL ERMSG(19,I,J1,*100)
120   CALL ERMSG(20,I,J1,*100)
      END
```

```
      SUBROUTINE GETCHAR(STRG,ITOP,ICHR,CHARR,*)

c     get a single character from a line starting from the ITOPth position

      CHARACTER STRG(80),CHARR
      ICHR=0
      IF(ITOP.GT.80)RETURN 1
      DO 10 I=ITOP,80
      IF(STRG(I).NE.' ')GOTO 20
10    CONTINUE
      RETURN 1
20    CHARR=STRG(I)
      ICHR=I
      ITOP=I+1
      RETURN
      END
```

```
      SUBROUTINE GETFET(I,J0,*)
      PARAMETER (MP=11,MX=40,MF=20)

c     decode the FET element

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,CHARR,PARNAM(18)*3,WRD*80
      CHARACTER KYWRD*3,VLABL*8
      DIMENSION IDP(18),ICO(18),VAL(3,MX),LOCV(3,MX),INXMDA(3),X(MX)
      DIMENSION PA(MP,3),WGT(8,MF,3),INPX(MP,3),BND(2,MX),INDX(MX)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),FREQ(MF)
      DIMENSION TYPDIM(6),IPTYP(18),IVTYP(MX),XDIM(MX),VDIM(MX)
      COMMON/SC1/TYPDIM,IPTYP,XDIM,VDIM,IVTYP
      COMMON/TS1/LOCBLK(2,10),LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1       TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/FETCONST/F3DB(3),GGS(3),FDIM
      SAVE PARNAM,IDP,ICO,IL
      DATA PARNAM/'G  ','CGS','F  ','T  ','GGS','CDG','CDC','CDS',
     1'GDS','RI ','RG ','RD ','RS ','CGE','CDE','LG ','LD ','LS '/
      DATA IDP/10,7,6,11,6,8,6,9,3,4,1,2,5,6,6,6,6,6/
      DATA ICO/2,2,1,2,-1,2,0,2,2,2,2,2,2,0,0,0,0,2/
      IG=NCKT*3+1
      NCKT=NCKT+1
      IF(NCKT.GT.3)CALL ERMSG(5,I,1,*150)
      CALL GETINT(TT(1,I),J0,J1,NODE(IG),*140)
      CALL GETINT(TT(1,I),J0,J1,NODE(IG+1),*140)
      CALL GETCHAR(TT(1,I),J0,J1,CHARR,*10)
      IF(CHARR.EQ.'0')GOTO 10
      IF(CHARR.EQ.'G')THEN
              J0=J1
              GOTO 10
      ELSE
              CALL ERMSG(21,I,J1,*150)
      ENDIF
10    DO 20 K=1,MP
      PA(K,NCKT)=0.0
20    INPX(K,NCKT)=0
      F3DB(NCKT)=0.0
      GGS(NCKT)=0.0
      K=1
30    CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
      IF(LENG.EQ.0)THEN
              I=I+1
              J0=1
              CALL GETCHAR(TT(1,I),J0,J1,CHARR,*40)
              IF(CHARR.EQ.'+')GOTO 30
40            I=I-1
              GOTO 100
      ENDIF
      CALL NOSYMBL(WRD,LENG,J0,J1)
      IF(LENG.EQ.0.OR.LENG.GT.3)GOTO 60
      KYWRD='   '
      KYWRD(1:LENG)=WRD(1:LENG)
      DO 50 KK=K,18
```

```
        IF(KYWRD.EQ.PARNAM(KK))GOTO 70
50      CONTINUE
60      CALL ERMSG(8,I,J1,*150)
70      K=KK+1
        IPAR=IDP(KK)
        JTYP=IPTYP(KK)
        J00=J0
        CALL GETCHAR(TT(1,I),J00,J1,CHARR,*160)
        IF(CHARR.EQ.'=')J0=J00
        J00=J0
        CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
        IF(LENG.EQ.0)GOTO 160
        IA=ICHAR(WRD(1:1))
        IF(IA.GE.65.AND.IA.LE.122)THEN
                IF(NVAR.EQ.0)GOTO 170
                IF(LENG.GT.8)CALL ERMSG(6,I,J1,*150)
                VLABL='        '
                VLABL(1:LENG)=WRD(1:LENG)
                DO 80 L=1,NVAR
                IF(VLABL.EQ.VNAME(L))GOTO 90
80              CONTINUE
                GOTO 170
        ELSE
                J0=J00
                CALL GETVAR(I,J0,MX,*150)
                IF(LOCV(3,MX).NE.0)THEN
                        NX=NX+1
                        INVAR(MX)=NX
                        X(NX)=VAL(1,MX)
                        XDIM(NX)=VDIM(MX)
                        LOCX(1,NX)=LOCV(1,MX)
                        LOCX(2,NX)=LOCV(2,MX)
                        INDX(NX)=LOCV(3,MX)
                        IF(INDX(NX).EQ.2)THEN
                                BND(1,NX)=VAL(2,MX)
                                BND(2,NX)=VAL(3,MX)
                        ENDIF
                ENDIF
                L=MX
        ENDIF
90      IF(IVTYP(L).EQ.0)THEN
                IVTYP(L)=JTYP
                VAL(1,L)=VAL(1,L)*TYPDIM(JTYP)
                IF(LOCV(3,L).NE.0)THEN
                        JX=INVAR(L)
                        X(JX)=X(JX)*TYPDIM(JTYP)
                        XDIM(JX)=XDIM(JX)*TYPDIM(JTYP)
                        IF(INDX(JX).EQ.2)THEN
                                BND(1,JX)=BND(1,JX)*TYPDIM(JTYP)
                                BND(2,JX)=BND(2,JX)*TYPDIM(JTYP)
                        ENDIF
                ENDIF
        ELSEIF(IVTYP(L).NE.JTYP)THEN
                CALL ERMSG(29,I,J1,*150)
        ENDIF
```

```
      PA(IPAR,NCKT)=VAL(1,L)
      IF(LOCV(3,L).EQ.0)THEN
              INPX(IPAR,NCKT)=0
      ELSE
              INPX(IPAR,NCKT)=INVAR(L)
      ENDIF
      IJ=INPX(6,NCKT)
      ZZ=PA(6,NCKT)
      IF(ICO(KK).EQ.1)THEN
              F3DB(NCKT)=ZZ
              IF(IJ.NE.0)CALL ERMSG(11,I,J1,*150)
      ELSEIF(ICO(KK).EQ.-1)THEN
              GGS(NCKT)=ZZ
              IF(IJ.NE.0)CALL ERMSG(12,I,J1,*150)
      ELSEIF(ICO(KK).EQ.0)THEN
              IF(IJ.NE.0.OR.ZZ.NE.0.0)CALL ERMSG(13,I,J1,*150)
      ENDIF
      IF(K.LE.18)GOTO 30
100   RETURN
140   CALL ERMSG(7,I,J1,*150)
150   RETURN1
160   CALL ERMSG(4,I,J1,*150)
170   CALL ERMSG(10,I,J1,*150)
      END
```

```
      SUBROUTINE GETFRQ(I,J0,J1,TEMP,*)
      PARAMETER (MP=11,MX=40,MF=20)

c     decode a single frequency

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX), NODE(9),LOCX(2,MX)
      DIMENSION TYPDIM(6),IPTYP(18),IVTYP(MX),XDIM(MX),VDIM(MX)
      COMMON/SC1/TYPDIM,IPTYP,XDIM,VDIM,IVTYP
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR(MX),LOCX,VAL,DIM(8),
     1     TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      CALL GETVAR(I,J0,MX,*100)
      IF(IVTYP(MX).GT.1)CALL ERMSG(29,I,J1,*100)
      IF(LOCV(3,MX).NE.0)CALL ERMSG(4,I,J1,*100)
      TEMP=VAL(1,MX)
      RETURN
100   RETURN 1
      END
```

```
      SUBROUTINE GETINT(STRG,ITOP,IP,IVAL,*)

c     try to convert an ASCII string into an integer

      CHARACTER STRG(80),WRD*80,WRD1*20
      CALL GETWRD(STRG,ITOP,IP,WRD,LENG)
      IF(LENG.EQ.0)RETURN 1
      WRD1='                    '
      WRD1(1:LENG)=WRD(1:LENG)
      READ(WRD1,*,ERR=10,END=10)IVAL
      IF(IVAL.GE.0)RETURN
10    RETURN 1
      END
```

```
      SUBROUTINE GETREAL(WRD,LENG,VALUE,*)

C     try to convert an ASCII string into a real value

      CHARACTER WRD*80,WRD1*20
      IF(LENG.EQ.0)GOTO 10
      WRD1='                    '
      WRD1(1:LENG)=WRD(1:LENG)
      READ(WRD1,*,ERR=10,END=10)VALUE
      IF(VALUE.GE.0.0)RETURN
10    RETURN 1
      END
```

```
      SUBROUTINE GETSYMBL(WRD,LENG,IP,IFLAG)

c     detect special symbols in a word

      CHARACTER SYMBOLS(10),WRD*80,A
      SAVE SYMBOLS
      DATA SYMBOLS/'=','?',':','+','-','%','*','/','>','<'/
      IFLAG=0
      DO 10 I=1,LENG
      A=WRD(I:I)
      DO 10 J=1,4
      IF(A.EQ.SYMBOLS(J))GOTO 20
10    CONTINUE
      RETURN
20    IP=I
      IFLAG=J
      RETURN
      END
```

```
      SUBROUTINE GETUNIT(WRD,LENG,IP,ITYPE,VAL)

c     decode the unit attached to a numeral

      CHARACTER WRD*80,KYWRD*4,UNAM(6,7)*4
      DIMENSION DDIM(6,7)
      SAVE UNAM,DDIM
      DATA UNAM/'HZ  ','KHZ ','MHZ ','GHZ ','    ','    ','OH  ','KOH ',
     1'MOH ','    ','    ','    ','/OH ','/KOH','/MOH','    ','    ',
     2'    ','L   ','MH  ','UH  ','NH  ','PH  ','    ','F   ','MF  ',
     3'UF  ','NF  ','PF  ','FF  ','SEC ','MS  ','US  ','NS  ','    ',
     4'    ','%   ','    ','    ','    ','    ','    ','/
      DATA DDIM/1.,1.E3,1.E6,1.E9,0.,0.,1.,1.E3,1.E6,0.,0.,0.,1.,1.E-3,
     1 1.E-6,0.,0.,0.,1.,1.E-3,1.E-6,1.E-9,1.E-12,0.,1.,1.E-3,1.E-6,
     2 1.E-9,1.E-12,1.E-15,1.0,1.E-3,1.E-6,1.E-9,8*0.0/
      IP=0
      ITYPE=0
      VAL=1.0
      DO 10 I=LENG,1,-1
      IA=ICHAR(WRD(I:I))
      IF(IA.GE.48.AND.IA.LE.57)GOTO 20
       IF(IA.EQ.4)GOTO 20
10     CONTINUE
       RETURN
20     IF(I.EQ.LENG)RETURN
       IP=I+1
       IG=LENG-IP+1
       IF(IG.GT.4)GOTO 40
       KYWRD='    '
       KYWRD(1:IG)=WRD(IP:LENG)
       DO 30 I=1,7
       DO 30 J=1,6
       IF(KYWRD.EQ.UNAM(J,I))GOTO 50
30     CONTINUE
40     IP=-1
       RETURN
50     ITYPE=I
       VAL=DDIM(J,I)
       RETURN
       END
```

```
      SUBROUTINE GETVAR(I,J0,NVR,*)
      PARAMETER (MP=11,MX=40,MF=20)

c     decode infomation relating to a variable including
c     its value, type (bounded or not) and the bounds on it

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,WRD*80,A
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3),ITY(3)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),VU(3),VL(3)
      DIMENSION TYPDIM(6),IPTYP(18),IVTYP(MX),XDIM(MX),VDIM(MX)
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1     TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/SC1/TYPDIM,IPTYP,XDIM,VDIM,IVTYP
      J00=J0
      INDXV=0
      CALL GETCHAR(TT(1,I),J00,J1,A,*70)
      IF(A.EQ.'?')THEN
              INDXV=1
              J0=J00
              JX=J00
      ENDIF
      KOUNT=1
10    CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
      CALL NOSYMBL(WRD,LENG,J0,J1)
      IF(LENG.EQ.0)GOTO 70
      CALL GETUNIT(WRD,LENG,IP,ITY(KOUNT),VU(KOUNT))
      IF(IP.LT.0)CALL ERMSG(25,I,J1,*70)
      IF(IP.GT.0)LENG=IP-1
      CALL GETREAL(WRD,LENG,VL(KOUNT),*70)
      IF(INDXV.EQ.0)GOTO 25
      IF(KOUNT.EQ.1)THEN
              J00=J0
              CALL GETCHAR(TT(1,I),J00,J11,A,*80)
              IF(A.EQ.'?')GOTO 20
              JX=J0
              KOUNT=2
              LOCV(3,NVR)=2
      ELSEIF(KOUNT.EQ.2)THEN
              KOUNT=3
              JX1=J1+LENG
              CALL PAD(TT(1,I),JX,JX1,ISHIFT)
              J0=J0+ISHIFT
      ELSE
              CALL CHKTYP(ITY,I,*90)
              CALL GETCHAR(TT(1,I),J0,J1,A,*80)
              IF(A.NE.'?')GOTO 80
              GOTO 30
      ENDIF
      GOTO 10
20    JX1=J1+LENG
      CALL PAD(TT(1,I),JX,JX1,ISHIFT)
      J0=J00+ISHIFT
25    LOCV(3,NVR)=INDXV
      LOCV(2,NVR)=JX
```

```
      VDIM(NVR)=VU(1)
      IF(ITY(1).EQ.7)THEN
              CALL ERMSG(29,I,1,*90)
      ELSEIF(ITY(1).NE.0)THEN
              VDIM(NVR)=VDIM(NVR)*TYPDIM(ITY(1))
      ENDIF
      IVTYP(NVR)=ITY(1)
      VAL(1,NVR)=VL(1)*VDIM(NVR)
      GOTO 40
30    LOCV(2,NVR)=JX
      VDIM(NVR)=VU(2)
      IF(ITY(2).NE.0)VDIM(NVR)=VDIM(NVR)*TYPDIM(ITY(2))
      VAL(1,NVR)=VL(2)*VDIM(NVR)
      IF(ITY(1).EQ.7)THEN
              VAL(2,NVR)=VAL(1,NVR)*VL(1)*0.01
      ELSE
              IF(ITY(2).NE.0)VU(1)=VU(1)*TYPDIM(ITY(2))
              VAL(2,NVR)=VL(1)*VU(1)
      ENDIF
      IF(ITY(3).EQ.7)THEN
              VAL(3,NVR)=VAL(1,NVR)*VL(3)*0.01
      ELSE
              IF(ITY(2).NE.0)VU(3)=VU(3)*TYPDIM(ITY(2))
              VAL(3,NVR)=VL(3)*VU(3)
      ENDIF
      IF(VAL(3,NVR).LE.VAL(2,NVR))CALL ERMSG(26,I,1,*90)
      IVTYP(NVR)=ITY(2)
40    LOCV(1,NVR)=I
      RETURN
70    CALL ERMSG(4,I,J1,*90)
80    CALL ERMSG(22,I,J1,*90)
90    RETURN1
      END
```

```
      SUBROUTINE GETWRD(STRG,ITOP,IWRD,WRD,LENG)

c     get a word from a line (STRG) starting from the ITOPth position.
c     words are separated by blanks. return the word in WRD
c     its length in LENG

      CHARACTER STRG(80),WRD*80,A
      LENG=0
      IF(ITOP.GT.80)RETURN
      DO 10 I=ITOP,80
      IF(STRG(I).NE.' ')GOTO 20
10    CONTINUE
      RETURN
20    IF(STRG(I).EQ.'*')RETURN
      IWRD=I
      IW=1
      DO 30 I=IWRD,80
      A=STRG(I)
      IF(A.EQ.' ')GOTO 40
      IA=ICHAR(A)
      IF(IA.GE.97.AND.IA.LE.122)A=CHAR(IA-32)
      WRD(IW:IW)=A
      IW=IW+1
30    CONTINUE
40    ITOP=I
      LENG=ITOP-IWRD
      RETURN
      END
```

```
      SUBROUTINE NODUMMY(ILINE)

c     skip over comment lines in the measurement files

      CHARACTER CLINE*80,DUMMY
1     READ(1,'(A)',END=4)CLINE
      J0=1
      ILINE=ILINE+1
2     CALL GETCHAR(CLINE,J0,J1,DUMMY,*1)
      IF(DUMMY.EQ.CHAR(9))GOTO 2
      IF(DUMMY.EQ.'*')GOTO 1
      IDUMMY=ICHAR(DUMMY)-48
      IF(IDUMMY.GE.0.AND.IDUMMY.LE.9)GOTO 3
      IF(DUMMY.NE.'+'.AND.DUMMY.NE.'-'.AND.DUMMY.NE.'.')GOTO 1
3     BACKSPACE 1
      ILINE=ILINE-1
4     RETURN
      END
```

```
      SUBROUTINE NOSYMBL(WRD,LENG,J0,J1)

c     detect special symbols within a word

      CHARACTER WRD*80
      IF(LENG.EQ.0)RETURN
      CALL GETSYMBL(WRD,LENG,IP,IFLAG)
      IF(IFLAG.GT.0)THEN
              J0=J1+IP-1
              LENG=IP-1
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE OPTBLK(IERROR)
      PARAMETER (MP=11,MX=40,MF=20)

c     decode the 'OPT' block

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8
      CHARACTER WRD*80,PNAME*8,CNAME*7,CMDA(8)*7,CHARR
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3),X(MX)
      DIMENSION PA(MP,3),WGT(8,MF,3),INPX(MP,3),BND(2,MX),INDX(MX)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),FREQ(MF),WW(4)
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1      TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      DO 10 I=1,3
      DO 10 J=1,8
      DO 10 K=1,NFR
10    WGT(K,J,I)=0.0
      ETERM=0.0
      IERROR=1
      LTOP=LOCBLK(1,3)
      LBOTTOM=LOCBLK(2,3)
      IF(LTOP.EQ.0.OR.LTOP.GE.LBOTTOM)GOTO 210
      I=LTOP
      JCKT=0
20    I=I+1
      IF(I.GT.LBOTTOM)GOTO 150
      J0=1
30    CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
      IF(LENG.EQ.0)GOTO 20
      CHARR=WRD(1:1)
      IF(CHARR.EQ.'F'.AND.JCKT.GT.0)THEN
              IF(LENG.GT.1.AND.WRD(2:2).NE.'=')GOTO 100
              J0=J1+1
              CALL GETCHAR(TT(1,I),J0,J1,CHARR,*240)
              IF(CHARR.NE.'=')GOTO 220
              CALL GETFRQ(I,J0,J1,FLOW,*200)
              FHIGH=FLOW
              J00=J0
              CALL GETCHAR(TT(1,I),J00,J1,CHARR,*20)
              II=ICHAR(CHARR)
              IF(II.GE.43.AND.II.LE.57)CALL GETFRQ(I,J0,J1,FHIGH,*200)
              GOTO 30
      ELSEIF(CHARR.EQ.'S'.AND.JCKT.GT.0)THEN
              IF(LENG.GT.1.AND.WRD(2:2).NE.'=')GOTO 100
              J0=J1+1
              CALL GETCHAR(TT(1,I),J0,J1,CHARR,*240)
              IF(CHARR.NE.'=')GOTO 220
              CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
              IF(LENG.EQ.0.OR.LENG.GT.8)GOTO 240
              PNAME='        '
              PNAME(1:LENG)=WRD(1:LENG)
              DO 40 K1=4,6
              IF(PNAME.EQ.CKTNAME(K1))GOTO 50
40            CONTINUE
```

```
                GOTO 220
50              INXMDA(JCKT)=K1
                J00=J0
                CALL GETCHAR(TT(1,I),J00,J1,CHARR,*80)
                IF(CHARR.EQ.'W')THEN
                        J0=J00
                        CALL GETCHAR(TT(1,I),J00,J1,CHARR,*60)
                        IF(CHARR.EQ.'=')J0=J00
60                      DO 70 J=1,4
                        CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
                        CALL GETREAL(WRD,LENG,WW(J),*230)
70                      CONTINUE
                        TEMP=WW(2)
                        WW(2)=WW(3)
                        WW(3)=TEMP
                ENDIF
80              DO 95 J=1,NFR
                IF(FREQ(J).LT.FLOW.OR.FREQ(J).GT.FHIGH)GOTO 95
                DO 90 L=1,4
                L2=2*L
                L1=L2-1
                WGT(L1,J,JCKT)=WW(L)
90              WGT(L2,J,JCKT)=WW(L)
95              CONTINUE
                GOTO 30
        ELSEIF(LENG.EQ.4.AND.WRD(1:4).EQ.'TERM')THEN
                J0=J1+4
                CALL GETCHAR(TT(1,I),J0,J1,CHARR,*250)
                IF(CHARR.NE.'=')GOTO 220
                CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
                CALL GETREAL(WRD,LENG,ETERM,*230)
                GOTO 30
        ENDIF
100     IF(LENG.GT.8)GOTO 250
        PNAME='        '
        PNAME(1:LENG)=WRD(1:LENG)
        DO 110 K0=1,3
        IF(PNAME.EQ.CKTNAME(K0))GOTO 120
110     CONTINUE
        GOTO 250
120     JCKT=K0
        FHIGH=1.E20
        FLOW=-100.0
        DO 130 L=1,4
130     WW(L)=1.0
        GOTO 30
150     DO 160 I=1,NCKT
        IF(INXMDA(I).GT.0)GOTO 160
                WRITE(*,260)CKTNAME(I)
                GOTO 200
160     CONTINUE
        IERROR=0
200     RETURN
210     CALL ERMSG(3,I,J1,*200)
220     CALL ERMSG(8,I,J1,*200)
```

```
230     CALL ERMSG(4,I,J1,*200)
240     CALL ERMSG(24,I,J1,*200)
250     IF(JCKT.EQ.0)CALL ERMSG(16,I,J1,*200)
        GOTO 220
260     FORMAT(5X,'Error in CKT file:  ',A8,' not used for modelling')
        END
```

```
      SUBROUTINE OUTBLK(IERROR,NCKT)
      PARAMETER (MP=11,MX=40,MF=20)

c     decode the 'OUT' block

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,WRD*80,WRD1*8,WRD2*3
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8)
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1     TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/BLKPLT/IX(10),IY(10,8,3),YMIN(8,3),YR(8,3),JOUT
      JOUT=1
      IF(NCKT.LE.1)RETURN
      LTOP=LOCBLK(1,5)
      LBOTTOM=LOCBLK(2,5)
      IF(LTOP.EQ.0.OR.LTOP.GE.LBOTTOM)RETURN
      LTOP=LTOP+1
      DO 20 I=LTOP,LBOTTOM
      J0=1
      CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
      IF(LENG.EQ.0)GOTO 20
      IF(LENG.NE.3)GOTO 40
      WRD2=WRD(1:3)
      IF(WRD2.EQ.'PRI')THEN
            CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
            IF(LENG.NE.3)GOTO 20
            WRD2=WRD(1:3)
            IF(WRD2.NE.'PLO')GOTO 20
      ENDIF
      IF(WRD2.EQ.'PLO')THEN
            CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
            IF(LENG.EQ.0.OR.LENG.GT.8)GOTO 50
            WRD1='        '
            WRD1(1:LENG)=WRD(1:LENG)
            DO 10 K=1,NCKT
            IF(WRD1.EQ.CKTNAME(K))GOTO 30
10          CONTINUE
            GOTO 50
      ELSE
            GOTO 40
      ENDIF
20    CONTINUE
      RETURN
30    JOUT=K
      RETURN
40    CALL ERMSG(8,I,J1,*60)
50    CALL ERMSG(16,I,J1,*60)
60    IERROR=1
      RETURN
      END
```

```
      SUBROUTINE PAD(STRG,JX,JX1,ISHIFT)

c     insert blanks between words so that 8 digits
c     can be accommodated

      CHARACTER STRG(80)
      ISHIFT=8-JX1+JX
      IF(ISHIFT.LE.0)RETURN
      I00=80-ISHIFT
      IF(I00.LE.JX1)RETURN
      DO 10 I=I00,JX1,-1
10    STRG(I+ISHIFT)=STRG(I)
      J3=JX+7
      DO 20 I=JX1,J3
20    STRG(I)='0'
      RETURN
      END
```

```
      SUBROUTINE RDCKT(FILNAM,OLDFIL,IERROR)
      PARAMETER (MP=11,MX=40,MF=20)

c     organize the process of reading circuit and measurement files
c     it is called by the main program.  It is the link between
c     this group of files and the rest of the program

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,A*80,KYWRD*2,STRG*17
      CHARACTER FILNAM*12,OLDFIL*12
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3)
      DIMENSION PA(MP,3),X(MX),INPX(MP,3),BND(2,MX),INDX(MX)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),FREQ(MF)
      DIMENSION PDIM(MP),DAT(8,MF,3),INPP(MP),WGT(8,MF,3)
      LOGICAL THERE
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1     TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/BLK1/PDIM,DAT,INPP,PAO(MP,3)
      COMMON/BLK2/SNR(8,3),KFR(MF,3),DTM(8,MF,3),KOUNT,KODE
      COMMON/FETCONST/F3DB(3),GGS(3),FDIM

      OPEN(1,FILE=FILNAM,STATUS='OLD')
      WRITE(*,'(5X,''Reading file: '',A12)')FILNAM

c     read circuit file

      CALL RDSCF(IERROR)
      WRITE(*,*)
      IF(IERROR.NE.1)CALL BLKBLK(IERROR)
      IF(IERROR.NE.1)CALL FREQBLK(IERROR)
      IF(IERROR.NE.1)CALL DATABLK(IERROR)
      IF(IERROR.NE.1)CALL OPTBLK(IERROR)
      IF(IERROR.NE.1)CALL OUTBLK(IERROR,NCKT)
      CLOSE(1)
      IF(IERROR.EQ.1)GOTO 230
      OLDFIL=FILNAM

c     read measurement files

      DO 40 K=1,NCKT
      FILNAM=CKTFIL(INXMDA(K))
      INQUIRE(FILE=FILNAM,EXIST=THERE)
      IF(.NOT.THERE)CALL FERMSG(1,FILNAM,I,*220)
      OPEN(1,FILE=FILNAM,STATUS='OLD')
      EPS=0.001*FREQ(1)
      ILINE=1
      DO 30 J=1,NFR
20    CALL NODUMMY(ILINE)
      READ(1,*,ERR=190,END=200)FRE,(DAT(M,J,K),M=1,8)
      ILINE=ILINE+1
      IF(ABS(FRE-FREQ(J)).GE.EPS)GOTO 20
30    CONTINUE
      CLOSE(1)
40    CONTINUE
```

```
      DO 50 J=1,6
50    DIM(J)=1.0
      DO 60 J=1,MP
60    PDIM(J)=1.0
      FDIM=1.0

c     transform the measurement data to rectangular form

      C1=ASIN(1.0)/90.0
      DO 70 I=1,NCKT
      DO 70 K=1,4
      K1=2*K-1
      K2=K1+1
      DO 70 J=1,NFR
      T1=DAT(K1,J,I)
      T2=DAT(K2,J,I)*C1
      DTM(K1,J,I)=T1*COS(T2)
      DTM(K2,J,I)=T1*SIN(T2)
70    CONTINUE
      DO 100 I=1,NCKT
      DO 100 K=1,8
      SNR(K,I)=1.0
      IF(NFR.EQ.1)GOTO 100
      SMAX=-2000.0
      SMIN=2000.0
      DO 80 J=1,NFR
      IF(DTM(K,J,I).GT.SMAX)SMAX=DTM(K,J,I)
      IF(DTM(K,J,I).LT.SMIN)SMIN=DTM(K,J,I)
80    CONTINUE
      SNR(K,I)=1.0/(SMAX-SMIN)
100   CONTINUE

c     count the error functions

      MM=0
      DO 120 I=1,NCKT
      DO 120 J=1,NFR
      M1=0
      DO 110 K=1,8
      IF(WGT(K,J,I).GT.1.E-5)M1=M1+1
110   CONTINUE
      MM=MM+M1
120   KFR(J,I)=M1
      IF(MM.GT.250)THEN
              WRITE(*,*)'     Too many functions'
              GOTO 220
      ENDIF
      CALL BRIEF(OLDFIL,CKTFIL,NCKT)
180   RETURN
190   CALL FERMSG(2,FILNAM,ILINE,*210)
200   CALL FERMSG(3,FILNAM,ILINE,*210)
210   CLOSE(1)
220   IERROR=1
230   NCKT=0
```

```
GOTO 180
END
```

```
      SUBROUTINE RDSCF(IERROR)
      PARAMETER (MP=11,MX=40,MF=20)

c     put the contents of a the circuit file in char array TT(80,100)
c     locate various blocks in the file
c     block i  begins at the LOCBLK(1,i)th line and ends at
c     the LOCBLK(2,i)th line

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,T*80,A,B,BLKNAME(6)*5,WRD*80
      CHARACTER KYWRD*5
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8)
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1     TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      SAVE BLKNAME
      DATA BLKNAME/'BLK  ','FREQ ','OPT  ','DATA ','OUT  ','     '/
      B=CHAR(9)
      DO 25 I=1,100
      READ(1,'(A)',ERR=30,END=30)T
      DO 10 J=1,80
10    TT(J,I)=' '
      K=1
      DO 20 J=1,80
      IF(T(J:J).EQ.B)THEN

c             a tab

              K=(K/9+1)*9
      ELSE
              IF(K.GT.80)GOTO 25
              TT(K,I)=T(J:J)
              K=K+1
      ENDIF
20    CONTINUE
25    CONTINUE
30    LINET=I-1
      DO 40 I=1,10
40    LOCBLK(1,I)=0
      IPRECED=0
      DO 80 I=1,LINET
      J0=1
      KYWRD='     '
      CALL GETWRD(TT(1,I),J0,J2,WRD,LENG)
      IF(LENG.EQ.0)GOTO 80
      IF(IPRECED.EQ.0)THEN
              IF(LENG.GT.4)GOTO 60
              KYWRD(1:LENG)=WRD(1:LENG)
              DO 50 J=1,5
              IF(KYWRD.EQ.BLKNAME(J))GOTO 70
50            CONTINUE
60            CALL ERMSG(27,I,1,*100)
70            IPRECED=J
              LOCBLK(1,IPRECED)=I
      ELSEIF(LENG.EQ.3.AND.WRD(1:3).EQ.'END')THEN
```

```
                LOCBLK(2,IPRECED)=I-1
                IPRECED=0
        ENDIF
80      CONTINUE
        IF(IPRECED.NE.0)CALL ERMSG(28,LINET,1,*100)
        IERROR=0
        RETURN
100     IERROR=1
        RETURN
        END
```

```
c*********************************************************************

c       file name: FETPLT.FOR

c       for program FETSCG (PC/AT version).
c      this group of files are responsible graphics and plotter
c       output

c*********************************************************************

$NOFLOATCALLS

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c       for program FETSCG.
c       subroutines FETPLT and HPPLT are responsible for plotting
c       FETPLT generates screen graphics
c       HPPLT controls a HP plotter to generate a hardcopy
c       they are valid for both TI or IBM/AT
c       the appropriate graphics library (TIGRF or IBMGRF) must be linked

c       the ISCREEN parameter must be set to distinguish TI or IBM/AT:
c       set ISCREEN=1 for TI and ISCREEN=2 for IBM/AT

c       Also, the device for the plotter is system dependent.  In this
c       particular code, LPT3 is used for the TI, and COM1 for the AT.

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc



        SUBROUTINE HPPLT(ICKT,MODNAM,IMO,MESNAM,IME,NF,IFPIX,IMODPIX,
     +IMESPIX,IXBPIX,IYBPIX)
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        PARAMETER (ISCREEN=2,MF=20)
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c HP plotting subroutine for FET modeling program
c
c  eight boxes are to be displayed on the plotter,
c  the top    4 boxes are numbered: 1 3 5 7  from left to right
c  the bottom 4 boxes are numbered: 2 4 6 8  from left to right
c
c
c arguments needed:     ICKT    circuit number (multiple cct modeling)
c                       MODNAM  model file name
c                       IMO     number of character in MODNAM
c                       MESNAM  measurement file name
c                       IME     number of characters in MESNAM
c                       NF      number of frequencies
c                       IFPIX()         pixels for x axis plot
c                       IMODPIX(,,)     pixels for y axis plot (model)
c                       IMESPIX(,,)     pixels for y axis plot (measurement)
c                       IXBPIX  pixel range for x axis of box
c                       IYBPIX  pixel range for y axis of box
```

```
      IMPLICIT REAL*4 (A-H,O-Z)
      IMPLICIT INTEGER*2 (I-N)
      DIMENSION IFPIX(MF),IMODPIX(MF,8,3),IMESPIX(MF,8,3),IBBB(4,8)
      DIMENSION XCEN(4),YCEN(2)
      CHARACTER BLAB(8)*4,MODNAM(IMO)*1,MESNAM(IME)*1
      CHARACTER BLNK*89,STRG*89
      DATA BLAB/'MS11','AS11','MS21','AS21','MS12','AS12','MS22','AS22'/

c glossary:
c
c XPAGE         the size of a page in the x direction (cm)
c YPAGE         the size of a page in the y direction (cm)
c XMARG         the x margin desired (cm)
c YMARG         the y margin desired (cm)
c XB            the x width  of each of the 8 boxes (cm)
c YB            the y height of each of the 8 boxes (cm)
c XCEN()        page location along x axis of the center of boxes (cm)
c YCEN()        page location along y axis of the center of boxes (cm)
c H             size of the character in % wrt XB and YB
c S             the x distance between boxes, in % wrt XB
c IBBB(i,j)     the x and y coord. of the corners of each box (plotter units)
c               i = 1 to 4  denoting xmin, xmax, ymin, ymax of the box
c               j = 1 to 8  denoting the box number

c set defaults

      XMARG=3.5
      YMARG=3.5
      H=0.035

c prompt user as to wether defaults are acceptable

      BLNK(1:8)='@L01@C01'
      DO 5 I=1,80
5     BLNK(I+8:I+8)=' '
      BLNK(89:89)='_'
      CALL DISP(BLNK)
      STRG(1:38)='@L01@C01@7Plot defaults: XMARG=3.5cm, '
      STRG(39:76)='YMARG=3.5cm, char=3.5% of box. (y/n)?_'
      CALL DISP(STRG)
      IKY=KYCODE(IKY)
      IDEF=1
      IF(ISCREEN.EQ.1) THEN
              IF(IKY.EQ.78.OR.IKY.EQ.110) IDEF=0
      ENDIF
      IF(ISCREEN.EQ.2) THEN
              IF(IKY.EQ.12622.OR.IKY.EQ.12654) IDEF=0
      ENDIF

c if defaults unacceptable, get desired values

      IF(IDEF.EQ.0) THEN
              CALL DISP(BLNK)
              STRG(1:37)='@L01@C01@7 Enter: XMARG cm, YMARG cm,'
```

```
                STRG(38:54)=' char (% box) : _'
                CALL DISP(STRG)
7               READ(*,*) XMARG,YMARG,H
                H=H/100.0
                STRG=BLNK
                STRG(1:4)='@L02'
                CALL DISP(STRG)
                Z=0.0
                IF(XMARG.LT.Z.OR.YMARG.LT.Z.OR.H.LT.Z) THEN
                        CALL BELL
                        GOTO 7
                ENDIF
        ENDIF
        CALL DISP(BLNK)
        STRG(1:49)='@L01@C01@7 Prepare plotter pens: P1 box, P2 model'
        STRG(50:87)=', P3 measurement.   <CR> when ready. _'
        CALL DISP(STRG)
        IKY=KYCODE(IKY)
        CALL DISP(BLNK)
        CALL DISP('@L01@C30@7@B ... HP plotting ... @b _')

c the rest is automatically adjusted

        S=2.0*H
        XPAGE=28.0
        YPAGE=21.6
        XB=(XPAGE-2.0*XMARG)/(4.0+3.0*S)
        YB=(YPAGE-2.0*YMARG)/(2.0+7.5*H)
        XLCLIP=1.0
        XMARG=XMARG-XLCLIP
        YBCLIP=0.35
        YMARG=YMARG-YBCLIP
        YCEN(1)=YMARG+0.5*YB
        YCEN(2)=YMARG+(1.5+3.0*H)*YB
        XCEN(1)=XMARG+0.5*XB
        XCEN(2)=XCEN(1)+(1.0+S)*XB
        XCEN(3)=XCEN(2)+(1.0+S)*XB
        XCEN(4)=XCEN(3)+(1.0+S)*XB
        DO 10 IC=1,4
        DO 10 IB=1,8
                IX=(IB+1)/2
                IY=2
                IF(MOD(IB,2).EQ.0) IY=1
                PM=-1.0
                IF(MOD(IC,2).EQ.0) PM=+1.0

c the conversion between plotter units and cm = 400.0 pu/cm

                IF(IC.LE.2) IBBB(IC,IB)=(XCEN(IX)+PM*0.5*XB)*400.0
                IF(IC.GE.3) IBBB(IC,IB)=(YCEN(IY)+PM*0.5*YB)*400.0
10      CONTINUE

c obtain the character height in plotter units

        HPU=H*YB*400.0
```

```
c start plotting:
c these devices are system dependent

        IF(ISCREEN.EQ.1) OPEN(2,FILE='LPT3',STATUS='OLD')
        IF(ISCREEN.EQ.2) OPEN(2,FILE='COM1',STATUS='OLD')

c initialize plotter and set character size: width & height (cm)

        WRITE(2,300) H*XB, H*YB
        DO 20 IB=1,8

c draw the box labels

                IYPU=IBBB(4,IB)+0.5*HPU
                IX=(IB+1)/2
                IXPU=XCEN(IX)*400.0
                WRITE(2,310) IXPU,IYPU,BLAB(IB),CHAR(3)

c then the box

                WRITE(2,320) IBBB(1,IB),IBBB(3,IB),IBBB(2,IB),IBBB(4,IB)
20      CONTINUE

c draw Model title

        IYPU=IBBB(4,1)+3.5*HPU
        IXPU=IBBB(1,1)
        WRITE(2,330) IXPU,IYPU,MODNAM,CHAR(3)

c then the model curves

        DO 30 IB=1,8

c       first extablish P1,P2; the corners of the box

                WRITE(2,350) IBBB(1,IB),IBBB(3,IB),IBBB(2,IB),IBBB(4,IB)

c       then scale it to the range of data to be plotted

                WRITE(2,360) 0,IXBPIX,0,IYBPIX

c       the plot the data

                DO 30 IF=1,NF
                        WRITE(2,370) IFPIX(IF), IMODPIX(IF,IB,ICKT)
30      CONTINUE

c draw Measurement title

        IXPU=IBBB(1,5)
        WRITE(2,340) IXPU,IYPU,MESNAM,CHAR(3)

c then the measurement curves
```

```
      DO 40 IB=1,8
            WRITE(2,350) IBBB(1,IB),IBBB(3,IB),IBBB(2,IB),IBBB(4,IB)
            WRITE(2,360) 0,IXBPIX,0,IYBPIX
            DO 40 IF=1,NF
                  WRITE(2,370) IFPIX(IF), IMESPIX(IF,IB,ICKT)
40    CONTINUE
      WRITE(2,380)
      CLOSE(2)
      CALL DISP(BLNK)
      RETURN

300   FORMAT(' IN ;SI ',2F10.3,' ;PU ;SP1 ;')
310   FORMAT(' PA ',2I10,' ;CP -2. 0. ;LB',A4,A1)
320   FORMAT(' PU ;PA ',2I10,' ;PD ;EA ',2I10,' ;PU ;')
330   FORMAT(' PU; SC; PA ',2I10,' ;SP2 ;LBMODEL: ',90A)
340   FORMAT(' PU; SC; PA ',2I10,' ;SP3 ;LBMEASUREMENT: ',90A)
350   FORMAT(' PU ;IP ',4I10,' ;')
360   FORMAT(' SC ',4I10,' ;')
370   FORMAT(' PA ',2I10,' ;PD ;')
380   FORMAT(' SP0 ;IN ;')
      END
```

```
      SUBROUTINE FETPLT(FM,JOB)
      PARAMETER (ISCREEN=2,MP=11,MX=40,MF=20)

c     job=1 coming from main program for simple plotting
c     job=2 ........................ preparing for optimization
c     job=3 coming from fdf during optimization
c     hp plotter interface should only deal with job=1

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,STRG*90
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INDX(MX),PA(MP,3)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),X(MX),BND(2,MX)
      DIMENSION FREQ(MF),WGT(8,MF,3),FM(8,MF,3),INXMDA(3),INPX(MP,3)
      INTEGER*2 IKY,IX(MF),IYO(MF,8,3),NPLT,IY(MF,8,3),IBB(4,9,2)
      INTEGER*2 KYCODE,IHIGH,MODE1
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     +           TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/BLK1/PDIM(MP),DAT(8,MF,3),INPP(MP),PAO(MP,3)
      COMMON/BLKPLT/IX,IY,YMIN(8,3),YR(8,3),JOUT
      SAVE IBB

c     IBB(I,J,K) stores pixel coordinates for screen display of boxes
c     I = 1 to 4   denoting xmin,xmax,ymin,ymax
c     J = 1 to 8   denoting the 8 boxes to be displayed
c     J = 9        to denote the size of the entire screen
c     K = 1 / 2    flag denoting  TI / AT  type of screen

      DATA IBB/15,165,160,280,15,165,15,135,195,345,160,280,
     +         195,345,15,135,375,525,160,280,375,525,15,135,
     +         555,705,160,280,555,705,15,135,0,719,0,299,
     +         5,155,186,326,5,155,17,157,165,315,186,326,
     +         165,315,17,157,325,475,186,326,325,475,17,157,
     +         485,635,186,326,485,635,17,157,0,639,0,349/

c x pixels per box

      IXBPIX=IBB(2,1,ISCREEN)-IBB(1,1,ISCREEN)
      XBPIX=FLOAT(IXBPIX)

c y pixels per box

      IYBPIX=IBB(4,1,ISCREEN)-IBB(3,1,ISCREEN)
      YBPIX=FLOAT(IYBPIX)

      IF (NCKT.LE.0) RETURN

      NPLT=NFR
      GOTO(10,10,100),JOB
10    XMIN=FREQ(1)
      XRATIO=XBPIX/(FREQ(NFR)-XMIN)
      DO 20 J=1,NFR
20          IX(J)=INT((FREQ(J)-XMIN)*XRATIO)
      DO 40 I=1,NCKT
      DO 40 K=1,8
```

```
                FMX=-200.0
                FMI=200.0
                DO 30 J=1,NFR
                        IF(DAT(K,J,I).GT.FMX)FMX=DAT(K,J,I)
                        IF(FM(K,J,I).GT.FMX)FMX=FM(K,J,I)
                        IF(DAT(K,J,I).LT.FMI)FMI=DAT(K,J,I)
                        IF(FM(K,J,I).LT.FMI)FMI=FM(K,J,I)
30              CONTINUE
                FD=FMX-FMI
                YMI=FMI-0.1*FD
                YRATIO=(YBPIX/1.2)/FD
                YMIN(K,I)=YMI
                YR(K,I)=YRATIO
        DO 40 J=1,NFR
                IYO(J,K,I)=INT((DAT(K,J,I)-YMI)*YRATIO)
                IY(J,K,I)=INT((FM(K,J,I)-YMI)*YRATIO)
40      CONTINUE
        IF(JOB.EQ.2)THEN
                CALL CLEAR
        ELSE
                CALL SAVSCR
        ENDIF
        CALL MODE(MODE1,16)
        IF(JOB.EQ.2)THEN
                I=JOUT
        ELSE
                I=1
        ENDIF
        CALL PTCOLR(7)
        IF(JOB.EQ.2)CALL SETXOR
50      STRG(1:38)= '@C09@L13@7MS11@S17MS21@S16MS12@S17MS22'
        STRG(39:75)='@C09@L25AS11@S17AS21@S16AS12@S17AS22_'
        CALL DISP(STRG)
        STRG(1:18)='@C10@L01@I@6Model:'
        STRG(19:26)=CKTNAME(I)
        IF (ISCREEN.EQ.1) STRG(27:37)='    @2File:'
        IF (ISCREEN.EQ.2) STRG(27:37)='    @4File:'
        STRG(38:49)=CKTFIL(INXMDA(I))
        IF(JOB.EQ.2)THEN
                STRG(50:75)='@7@i   <ESC> to interrupt_'
                STRG(1:4)='@C24'
        ELSE
                STRG(50:80)='@7@i@S10<RETURN>  or  P=plot  _'
        ENDIF
        CALL DISP(STRG)
        DO 60 K=1,8
                CALL BOX(IBB(1,K,ISCREEN),'L')
                CALL WINDOW(IBB(1,K,ISCREEN))
                IF (ISCREEN.EQ.1) CALL PTCOLR(2)
                IF (ISCREEN.EQ.2) CALL PTCOLR(12)
                CALL LINE(NPLT,IX,IYO(1,K,I))
                IF (ISCREEN.EQ.1) CALL PTCOLR(6)
                IF (ISCREEN.EQ.2) CALL PTCOLR(14)
                CALL LINE(NPLT,IX,IY(1,K,I))
                CALL WINDOW(IBB(1,9,ISCREEN))
```

```
                CALL PTCOLR(7)
60      CONTINUE
        IF(JOB.EQ.2)GOTO 90

c wait for the user to respond

70      IKY=KYCODE(IKY)
        IF (ISCREEN.EQ.2) IHIGH=IKY/256
        IF (ISCREEN.EQ.2) IKY=IKY-IHIGH*256
        IF(IKY.NE.13.AND.IKY.NE.80.AND.IKY.NE.112) GOTO 70
        IF(IKY.EQ.13) GOTO 75

c HP plotter routine:

        IC1=8
        IC2=12
        CALL HPPLT(I,CKTNAME(I),IC1,CKTFIL(INXMDA(I)),IC2,NFR,IX,
     +             IY,IY0,IXBPIX,IYBPIX)

75      CALL CLRGRF
        IF(I.EQ.NCKT)GOTO 80
        I=I+1
        GOTO 50

c all circuits done, ready for return to the main program

80      CALL MODE(MODE1,3)
        CALL RSTSCR
        RETURN

c here when JOB=2

90      IF (ISCREEN.EQ.1) CALL PTCOLR(6)
        IF (ISCREEN.EQ.2) CALL PTCOLR(14)
        RETURN

c here when JOB=3 (during optimization)

100     NPLT=NFR
        DO 120 K=1,8
                CALL WINDOW(IBB(1,K,ISCREEN))
                CALL LINE(NPLT,IX,IY(1,K,JOUT))
                YMI=YMIN(K,JOUT)
                YRATIO=YR(K,JOUT)
                DO 110 J=1,NFR
                        ITEMP=INT((FM(K,J,JOUT)-YMI)*YRATIO)
                        IF (ITEMP.LT.0)       ITEMP=0
                        IF (ITEMP.GT.IYBPIX) ITEMP=IYBPIX
                        IY(J,K,JOUT)=ITEMP
110             CONTINUE
                CALL LINE(NPLT,IX,IY(1,K,JOUT))
                CALL WINDOW(IBB(1,9,ISCREEN))
120     CONTINUE

        RETURN
```

END

```
c*********************************************************************

c       file name: FETSCG1.FOR

c       for program FETSCG1 (PC/AT version).
c       this file contains the main program itself: FETMDL and other
c       related subroutines

c*********************************************************************

$NOFLOATCALLS



c       prepare interface to C utility.   ref: Microsoft CEXEC.LIB

        interface to integer*2 function spawn
     +          [c,varying,alias:'spawnlp']
     +          (mode)
        integer*2 mode
        end
        interface to integer*2 function system [c]
     +           (string[reference])
        character*1 string
        end



        SUBROUTINE BRIEF(FILNAM,CKTFIL,NCKT)

c       display a summary of the input infomation

        CHARACTER*12 FILNAM,CKTFIL(6)
        IF(NCKT.EQ.0)CALL ERMSG(32,I,J,*20)
        KK=NCKT+3
        WRITE(*,10)FILNAM,NCKT,(CKTFIL(K),K=4,KK)
10      FORMAT(5X,'Working file: ',A12/5X,'Number of circuits: ',I2/5X,
     *  'Measurement files: ',3(A12,3X))
20      RETURN
        END
```

```
      SUBROUTINE CMPARE(JOPT,NCKT,PA,PA0,PDIM,CKTNAME)
      PARAMETER (MP=13,MX=40,MF=20)

c     compare the optimal solution with the starting point
c     (command 'COMPARE')    starting point in PA0, solution in PA

      DIMENSION PA(MP,3),PDIM(MP),PP(6),PA0(MP,3)
      CHARACTER CKTNAME(6)*8,PN(MP)*8,FMT1(3)*15,FMT3(3)*22
      CHARACTER C1*8,C2*8,FMT2(3)*19
      SAVE PN,FMT1,FMT2,FMT3

ccc   DATA PN/'RG  (OH)','RD  (OH)','GDS(/OH)','RI  (OH)','RS  (OH)',
ccc  *'LS  (nH)','CGS (pF)','CDG (pF)','CDS (pF)','G  (/OH)','T   (ps)'/
      DATA PN/'RG  (OH)','RD  (OH)','GDS(/OH)','RI  (OH)','RS  (OH)',
     *'LS  (nH)','CGS (pF)','CDG (pF)','CDS (pF)','G  (/OH)','T   (ps)',
     *'LG  (nH)','LD  (nH)'/

      DATA FMT1/'(13X,A8)       ','(11X,A8,22X,A8)','(7X,3(A8,14X)) '/
      DATA FMT2,FMT3/'(7X,A8,7X,A8)      ','(2(5X,A8,5X,A8,4X))',
     * '(3(2X,A8,2X,A8,2X))','(2(1X,F14.5),2X,A8)   ',
     *'(2(2(3X,F10.4),4X),A8)','(3(2(2X,F8.4),2X),A8) '/

      IF(NCKT.LE.0)CALL ERMSG(32,I,J,*30)
      IF(JOPT.EQ.0)CALL ERMSG(35,I,J,*30)
      C1='starting'
      C2='solution'
      WRITE(*,*)'          FET MODEL PARAMETERS'
      WRITE(*,FMT1(NCKT))(CKTNAME(K),K=1,NCKT)
      WRITE(*,FMT2(NCKT))(C1,C2,K=1,NCKT)
      KK=2*NCKT
      DO 20 I=1,MP
      DO 10 K=1,NCKT
      K2=2*K
      K1=K2-1
      PP(K1)=PA0(I,K)
10    PP(K2)=PA(I,K)
      WRITE(*,FMT3(NCKT))(PP(K),K=1,KK),PN(I)
20    CONTINUE
      WRITE(*,*)
30    RETURN
      END
```

```
      PROGRAM FETMDL

c     main program of FETSCG1
c     A FET modelling program developed for Raytheon Company
c     equivalent circuit has 13 parameters including LS and LD
c     symbol 'ccc' in columns 1-3 indicates lines that have been
c     changed from FETSCG. Also, in each subroutine the statement
c             PARAMETER (MP=11,MX=40,MF=20)
c     has been changed to
c             PARAMETER (MP=13,MX=40,MF=20)

      PARAMETER (MP=13,MX=40,MF=20)
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     DIMENSIONAL LIMITS:     MP  maximum number of circuit parameters
c                             MX  maximum number of variables
c                             MF  maximum number of frequencies
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,A*80,KYWRD*2,STRG*19
      CHARACTER FILNAM*12,OLDFIL*12,FILEC*27
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3),X(MX)
      DIMENSION PA(MP,3),INPX(MP,3),BND(2,MX),INDX(MX),WGT(8,MF,3)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),FREQ(MF),INPO(MP)
      DIMENSION PDIM(MP),DAT(8,MF,3),FM(8,MF,3),F1(8,MF,3),INPP(MP)
      INTEGER*2 SYSTEM,SPAWN,ICXC
      LOGICAL THERE
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1      TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/BLK1/PDIM,DAT,INPP,PAO(MP,3)
ccc   DATA INPO/1,3,4,2,8,9,6,5,7,10,11/
      DATA INPO/1,3,4,2,8,9,6,5,7,10,11,12,13/

c     make sure the auxiliary files are present

      INQUIRE(FILE='FETSCG.MSG',EXIST=THERE)
      IF(.NOT.THERE)CALL FERMSG(4,'FETSCG.MSG  ',I,*2)
      INQUIRE(FILE='CED.EXE',EXIST=THERE)
      IF(.NOT.THERE)CALL FERMSG(4,'CED.EXE     ',I,*2)

      CALL HEAD
      CALL HELP
      OPEN(1,FILE='FETSCG.MSG',STATUS='OLD')
      DO 200 I=1,40
200   READ(1,'(A)')MSG(I)
      CLOSE(1)
      DO 210 I=1,MP
210   INPP(I)=INPO(I)

c     JOPT: have we done optimization?  0/1 = no/yes
c     JERR: have we done simulation?  0/1 = no/yes
c     NCKT: have we read circuit files?  0/(1,2,3)  = no/yes

      JOPT=0
      JERR=0
```

```
      NCKT=0

c     loop. read, interprete and act upon a command

100   WRITE(*,'(A$)')' COMD>'
      READ(*,'(A)')A
      CALL PARSER(A,KYCD,FILNAM)
      GOTO(1,1,2,3,4,5,6,7,7,15,8,9,14,10,11,12,13,100),KYCD

c     commands 'L1' and 'L2'

1     CALL OPTIMI(JOPT,JERR,FM,F1,KYCD)
      IF(JOPT.EQ.0)GOTO 100
      GOTO 6

2     STOP

c     command 'DIR'

3     icxc=system('dir/w'c)
      WRITE(*,*)
      GOTO 100

4     CALL HELP
      GOTO 100

c     command 'PARAMETER'

5     CALL WPA(NCKT,PA,PDIM,OLDFIL,CKTNAME)
      GOTO 100

c     command 'COMPARE'

6     CALL CMPARE(JOPT,NCKT,PA,PAO,PDIM,CKTNAME)
      GOTO 100

c     commands 'SWEEP' and '%ER'

7     CALL FETSIM(FM,F1,JERR)
      CALL WDA(FM,F1,KYCD)
      GOTO 100

c     command 'SENSITIVITY'

8     CALL GRAD(PDIM,CKTNAME)
      GOTO 100

c     command 'PRESENT'

9     CALL PRESENT
      GOTO 100

c     command 'SAVE'

10    CALL SAVCKT(FILNAM,OLDFIL,NX,X)
```

```
      GOTO 100

c     command 'READ'

11    CALL RDCKT(FILNAM,OLDFIL,IERROR)
      JOPT=0
      JERR=0
      GOTO 100

c     command 'EDIT'

12    filec(1:12)=filnam
      filec(13:13)=' '
      filec(14:25)=filnam
      filec(26:26)=char(0)
      filec(27:27)=char(0)
      icxc=spawn(0,loc('ced'c),loc('ced'c),loc(filec),int4(0))
      GOTO 100

c     command 'TYPE'

13    STRG(1:5)='TYPE '
      STRG(6:17)=FILNAM
      STRG(18:18)=CHAR(0)
      STRG(19:19)=CHAR(0)
      ICXC=SYSTEM(STRG)
      GOTO 100

c     command 'BRIEFING'

14    CALL BRIEF(OLDFIL,CKTFIL,NCKT)
      GOTO 100

c     command 'PLOT'

15    CALL FETSIM(FM,F1,JERR)
      CALL FETPLT(FM,1)
      GOTO 100

      END
```

```
      SUBROUTINE FETSIM(FM,F1,JERR)
      PARAMETER (MP=13,MX=40,MF=20)

c     organize model simulation for all frequency points
c     call SIMFET for each frequency

      DIMENSION INXMDA(3),PA(MP,3),INPX(MP,3),BND(2,MX),WGT(8,MF,3)
      DIMENSION INDX(MX),FREQ(MF),F(8),DF(8,MP),INPP(MP),X(MX)
      DIMENSION PDIM(MP),DAT(8,MF,3),FM(8,MF,3),F1(8,MF,3)
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/BLK1/PDIM,DAT,INPP,PAO(MP,3)

      IF(NCKT.LE.0)CALL ERMSG(32,I,J,*100)
      IF(JERR.GT.0)RETURN
      C1=90.0/ASIN(1.0)
      DO 90 I=1,NCKT
      DO 50 J=1,NFR
      CALL SIMFET(FREQ(J),PA(1,I),F,DF,JERR,I)
      DO 40 K=1,4
      K1=2*K-1
      K2=K1+1
      FM(K1,J,I)=SQRT(F(K1)*F(K1)+F(K2)*F(K2))

c     avoid discontinuity in ATAN2

              TEMPA=C1*ATAN2(F(K2),F(K1))
              TEMPB=DAT(K2,J,I)
              IF(ABS(TEMPA-TEMPB).GT.180.0)THEN
                      IF(TEMPA.GT.TEMPB)THEN
                              TEMPA=TEMPA-360.0
                      ELSE
                              TEMPA=TEMPA+360.0
                      ENDIF
              ENDIF
40    FM(K2,J,I)=TEMPA

      DO 50 K=1,8
      T=DAT(K,J,I)
      IF(ABS(T).GT.0.001)THEN
              T=100.0*ABS((FM(K,J,I)/T)-1.0)
              F1(K,J,I)=AMIN1(999.0,T)
      ELSE
              F1(K,J,I)=0.0
      ENDIF
50    CONTINUE
90    CONTINUE
      JERR=1
100   RETURN
      END
```

```
      SUBROUTINE GRAD(PDIM,CKTNAME)
      PARAMETER (MP=13,MX=40,MF=20)

c     organize model sensitivity evaluation and least-squares objective
c     call SIMFET

      CHARACTER PN(MP)*8,FMT1(3)*11,FMT2(3)*19,C*8,CKTNAME(6)*8
      DIMENSION PA(MP,3),X(MX),INPX(MP,3),BND(2,MX),INDX(MX),G(MP,3)
      DIMENSION INXMDA(3),FREQ(MF),WGT(8,MF,3),PDIM(MP),F(8),DF(8,MP)
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/BLK2/SNR(8,3),KFR(MF,3),DTM(8,MF,3),KOUNT,KODE
      SAVE PN,FMT1,FMT2

ccc   DATA PN/'RG  (OH)','RD  (OH)','GDS(/OH)','RI  (OH)','RS  (OH)',
ccc  *'LS  (nH)','CGS (pF)','CDG (pF)','CDS (pF)','G  (/OH)','T   (ps)'/
      DATA PN/'RG  (OH)','RD  (OH)','GDS(/OH)','RI  (OH)','RS  (OH)',
     *'LS  (nH)','CGS (pF)','CDG (pF)','CDS (pF)','G  (/OH)','T   (ps)',
     *'LG  (nH)','LD  (nH)'/

      DATA FMT1/'(12X,A8)   ','(2(12X,A8))','(3(12X,A8))'/
      DATA FMT2/'(5X,F15.6,5X,A8)   ','(2(5X,F15.6),5X,A8)',
     *'(3(5X,F15.6),5X,A8)'/

      IF(NCKT.EQ.0)CALL ERMSG(32,I,J,*100)
      M=0
      JSEN=1
      FOBJ=0.0
      DO 10 I=1,NCKT
      DO 10 IP=1,MP
10    G(IP,I)=0.0
      DO 20 I=1,NCKT
      DO 20 J=1,NFR
      CALL SIMFET(FREQ(J),PA(1,I),F,DF,JSEN,I)
      M=M+KFR(J,I)
      DO 20 K=1,8
      WF=SNR(K,I)*WGT(K,J,I)*(F(K)-DTM(K,J,I))
      FOBJ=FOBJ+WF*WF
      WF=2.0*WF
      DO 20 IP=1,MP
20    G(IP,I)=G(IP,I)+WF*DF(K,IP)
      RM=M
      FOBJ=FOBJ*8.0/RM
      C='gradient'
      DO 30 I=1,3
      DO 30 IP=1,MP
30    G(IP,I)=G(IP,I)*PA(IP,I)*8.0/RM
      WRITE(*,90)FOBJ
      WRITE(*,FMT1(NCKT))(CKTNAME(I),I=1,NCKT)
      WRITE(*,FMT1(NCKT))(C,I=1,NCKT)
      DO 40 IP=1,MP
40    WRITE(*,FMT2(NCKT))(G(IP,I),I=1,NCKT),PN(IP)
      WRITE(*,*)
90    FORMAT(10X,'SENSITIVITY OF THE CURRENT MODEL'/10X,
     * 'least-squares error:',F15.8)
100   RETURN
```

END

```
      SUBROUTINE HEAD
      CALL CLEAR
      WRITE(*,1)
1     FORMAT(33X,'FET MODELLING'//25X,'Interactive Command System'/)
      RETURN
      END
```

```
      SUBROUTINE HELP
      WRITE(*,10)
10    FORMAT(10X,'Commands Available')
      WRITE(*,20)
20    FORMAT(5X,'L1',12X,'start modelling using L1 optimization'/5X,
     *'L2',12X,'start modelling using least-squares optimization'/5X,
     *'STOP          stop this program and return to DOS'/5X,
     *'DIR           list the file directory'/5X,
     *'HELP          display the available commands'/5X,
     *'PARAMETER     display the current model parameters'/5X,
     *'COMPARE       compare starting point with optimization solution'/
     *5X,'%ERROR',8X,'display percentage errors in S-parameter matching'
     */5X,'SWEEP         generate S-parameters of the current model'/
     *5X,'PLOT          plot the match between model and measurement'/
     *5X,'SENSITIVITY   sensitivity analysis for the current model'/
     *5X,'SAVE fname    save current data in a CKT file'/5X,
     *'READ fname    read desired data from a file'/5X,
     *'EDIT fn.ext   edit a disk file using full sreen editor'/5X,
     *'TYPE fn.ext   display the content of a disk file'/5X,
     *'PRESENTATION  call the user-defined routine PRESENT'/5X,
     *'BRIEFING      display the current status'/)
      RETURN
      END
```

```
      SUBROUTINE OPTIMI(JOPT,JERR,FM,F1,KYCD)
      PARAMETER (MP=13,MX=40,MF=20)

c     prepare for optimization. call L1C or QUASI2

      DIMENSION PA(MP,3),X(MX),INPX(MP,3),BND(2,MX),INDX(MX),PAO(MP,3)
      DIMENSION INXMDA(3),FREQ(MF),WGT(8,MF,3),XX(MX),INPP(MP)
      DIMENSION PDIM(MP),DAT(8,MF,3),FM(8,MF,3),F1(8,MF,3)
      INTEGER*2 MODE1
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/BLK1/PDIM,DAT,INPP,PAO
      COMMON/BLK2/SNR(8,3),KFR(MF,3),DTM(8,MF,3),KOUNT,KODE
      COMMON/L11/MARK,FF1(250),DFF1(250,MX),FF(250),DFF(250,MX)
      COMMON/L12/C(MX),DC(MX,MX),CLOC(MX)

      IF(NCKT.LE.0)CALL ERMSG(32,I,J,*100)

      N=NX
      IF(NX.LE.0)CALL ERMSG(33,I,J,*100)
      JOPT=0
      L=0

c     exponential transformation on the variables
c     lower bounds on the original variables are conveniently handled

      DO 20 I=1,N
      TEMP=X(I)-BND(1,I)
      IF(TEMP.GT.4.5E-5)THEN
              XX(I)=ALOG(TEMP)
      ELSE
              XX(I)=-10.0
      ENDIF
      IF(INDX(I).EQ.2)THEN

c     transform the upper bounds on the original variables

              L=L+1
              DO 10 J=1,N
10            DC(L,J)=0.0
              DC(L,I)=-1.0
              C(L)=ALOG(BND(2,I)-BND(1,I))
      ENDIF
20    CONTINUE

      M=0
      DO 60 I=1,NCKT
      DO 60 J=1,NFR
60    M=M+KFR(J,I)

c     save the starting point

      DO 70 I=1,NCKT
      DO 70 J=1,MP
70    PAO(J,I)=PA(J,I)
```

```
      IF(M.EQ.0)CALL ERMSG(34,I,J,*100)
      JOPT=1
      KOUNT=0
      KODE=KYCD
      DX=0.2
      MAXF=1000
      EPS=0.01
      IF(ETERM.GE.1.E-8)EPS=ETERM
      LEQ=0
      CALL FETSIM(FM,F1,JERR)
      CALL FETPLT(FM,2)
      JERR=0

c     start optimization

      IF(KODE.EQ.1)CALL L1C(N,M,L,LEQ,XX,DX,EPS,MAXF)
      IF(KODE.EQ.2)CALL QUASI2(N,M,L,LEQ,XX,DX,EPS,MAXF)
      CALL RSTXOR
      CALL CLRGRF
      CALL MODE(MODE1,3)
      WRITE(*,*)
      IF(MARK.GT.0)THEN
              WRITE(*,*)'      Optimization Completed'
      ELSE
              WRITE(*,*)'      Optimization Interrupted'
      ENDIF
      WRITE(*,'(6X,''after '',I4,'' iterations'')')MAXF

c     reverse the exponential transformation on the variables

      DO 80 I=1,NX
80    X(I)=EXP(XX(I))+BND(1,I)
      DO 90 K=1,NCKT
      DO 90 I=1,MP
      IF(INPX(I,K).GT.0)PA(I,K)=X(INPX(I,K))
90    CONTINUE

100   RETURN
      END
```

```
      SUBROUTINE PARSER(A,KYCD,FILNAM)

c     interprete a input command contained in the ASCII string A
c     return a integer code KYCD

      CHARACTER A*80,WRD*80,COMD(17)*2,KYWRD*2,FILNAM*12
      LOGICAL THERE
      SAVE COMD
      DATA COMD/'L1','L2','ST','DI','HE','PA','CO','%E','SW','PL',
     *          'SE','PR','BR','SA','RE','ED','TY'/
      KYCD=18
      J0=1
      CALL GETWRD(A,J0,J1,WRD,LENG)
      IF(LENG.EQ.0)RETURN
      IF(LENG.EQ.1)GOTO 20
      KYWRD=WRD(1:2)
      DO 10 I=1,17
      IF(KYWRD.EQ.COMD(I))GOTO 30
10    CONTINUE
20    CALL ERMSG(31,I,J,*100)

c     is it a file related command?

30    IF(I.GE.14)THEN

c     yes, search the input string for a file name

40            CALL GETWRD(A,J0,J1,WRD,LENG)
              IF(LENG.EQ.0)THEN

c             file name missing, ask for it

                      WRITE(*,'(A$)')'     File name: '
                      READ(*,'(A)')A
                      J0=1
                      GOTO 40
              ENDIF
              FILNAM='              '
              FILNAM=WRD(1:LENG)
              IF(I.GE.16)GOTO 60

c             command 'save' or 'read'. put in the omitted extension '.DAT'

              DO 50 K=1,LENG
              IF(WRD(K:K).EQ.'.')GOTO 60
50            CONTINUE
              L1=LENG+1
              IF(L1.GT.9)L1=9
              L2=L1+3
              FILNAM(L1:L2)='.DAT'

60            IF(I.EQ.17.OR.I.EQ.15)THEN

c             command 'read' or 'type'. make sure the file exits
```

```
                    INQUIRE(FILE=FILNAM,EXIST=THERE)
                    IF(.NOT.THERE)CALL FERMSG(1,FILNAM,I,*100)
              ENDIF
        ENDIF

        KYCD=I
100     RETURN
        END
```

```
      SUBROUTINE PRESENT
      PARAMETER (MP=13,MX=40,MF=20)

c     gateway for user implemented operations. See manual
c     basic information:
c             NCKT:   number of circuits
c             PA:     current values of circuit parameters

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3),X(MX)
      DIMENSION PA(MP,3),INPX(MP,3),BND(2,MX),INDX(MX),WGT(8,MF,3)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),FREQ(MF),PDIM(MP)
      DIMENSION DAT(8,MF,3),INPP(MP),PAO(MP,3)
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1      TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/BLK1/PDIM,DAT,INPP,PAO

      RETURN
      END
```

```
      SUBROUTINE SAVCKT(FILNAM,OLDFIL,NX,X)
      PARAMETER (MP=13,MX=40,MF=20)

c     save the current infomation to a Super-Compact file

      CHARACTER TT(100)*80,VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,FILNAM*12,OLDFIL*12,AX*8,AT*80
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),X(MX)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8)
      INTEGER*2 LENG
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1     TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/SC1/TD(6),IPT(18),XDIM(MX),VDIM(MX),IVTYP(MX)

       IF(N.LE.0)GOTO 20
       DO 10 I=1,NX
       TEMP=X(I)/XDIM(I)
       IF(TEMP.GT.99.9999.OR.TEMP.LT.0.001)THEN
               WRITE(AX,'(E8.3)')TEMP
       ELSE
               WRITE(AX,'(F8.5)')TEMP
       ENDIF
       J1=LOCX(2,I)
       J2=J1+7
       TT(LOCX(1,I))(J1:J2)=AX
10     CONTINUE
20     OPEN(2,FILE=FILNAM,STATUS='NEW')
       OLDFIL=FILNAM
       DO 30 I=1,LINET
       AT=TT(I)(1:80)
       LENG=80
       CALL STRIM(AT,LENG)
       WRITE(2,'(A)')AT(1:LENG)
30     CONTINUE
       CLOSE(2)
       RETURN
       END
```

```
      SUBROUTINE WDA(FM,F1,KYCD)

c     display simulation results (commands 'SWEEP' and '%ER')

      PARAMETER (MP=13,MX=40,MF=20)
      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,TN(8)*4,CHR
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3),X(MX)
      DIMENSION PA(MP,3),WGT(8,MF,3),INPX(MP,3),BND(2,MX),INDX(MX)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),FREQ(MF)
      DIMENSION PDIM(MP),DAT(8,MF,3),FM(8,MF,3),F1(8,MF,3),INPP(MP)
      INTEGER*2 KYCODE,IKY
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1     TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/BLK1/PDIM,DAT,INPP,PA0(MP,3)
      SAVE TN
      DATA TN/'MS11','AS11','MS21','AS21','MS12','AS12','MS22','AS22'/
      IF(NCKT.LE.0)RETURN
      I=1
10    IF(KYCD.EQ.8)THEN
              WRITE(*,110)CKTNAME(I),CKTFIL(INXMDA(I))
              WRITE(*,'(6X,''(GHZ)'',8(4X,A4))')(TN(K),K=1,8)
              DO 20 J=1,NFR
20            WRITE(*,120)FREQ(J),(F1(K,J,I),K=1,8)
      ELSE
              J0=0
30            WRITE(J0,130)CKTNAME(I)
              WRITE(J0,'(1X,''*    (GHZ)'',8(4X,A4))')(TN(K),K=1,8)
              DO 40 J=1,NFR
40            WRITE(J0,140)FREQ(J),(FM(K,J,I),K=1,8)
              IF(CKTFIL(I)(1:1).NE.' ')THEN
                      IF(J0.EQ.2)THEN
                              CLOSE(2)
                      ELSE
                              OPEN(2,FILE=CKTFIL(I),STATUS='NEW')
                              J0=2
                              GOTO 30
                      ENDIF
              ENDIF
      ENDIF
      IF(I.GE.NCKT)GOTO 100
      WRITE(*,'(/5X,''Press any key to continue ...'')')
      IKY=KYCODE(IKY)
      I=I+1
      WRITE(*,*)
      GOTO 10
100   RETURN
110   FORMAT(/19X,'PERCENTAGE ERRORS IN S-PARAMETER MATCHING'/20X,
     *'ABS[ (Model-Measurement)/Measurement ] %'/3X,'Frequency',7X,
     *'model: ',A8,6X,'measurement: ',A12)
120   FORMAT(1X,F9.1,2X,8(F7.2,'%'))
130   FORMAT(1X,'*',15X,'S-PARAMETERS OF MODEL ',A8/1X,'*  Frequency')
140   FORMAT(1X,F9.1,2X,4(F8.4,F8.2))
      END
```

```
      SUBROUTINE WPA(NCKT,PA,PDIM,FILNAM,CKTNAME)
      PARAMETER (MP=13,MX=40,MF=20)

c     display the current values of the parameters (command 'PA')

      DIMENSION PA(MP,3),PDIM(MP),PP(3)
      CHARACTER CKTNAME(6)*8,PN(MP)*8,FMT1(3)*15,FMT2(3)*19
      CHARACTER FILNAM*12
      SAVE PN,FMT1,FMT2

ccc   DATA PN/'RG  (OH)','RD  (OH)','GDS(/OH)','RI  (OH)','RS  (OH)',
ccc  *'LS  (nH)','CGS (pF)','CDG (pF)','CDS (pF)','G  (/OH)','T   (ps)'/
      DATA PN/'RG  (OH)','RD  (OH)','GDS(/OH)','RI  (OH)','RS  (OH)',
     *'LS  (nH)','CGS (pF)','CDG (pF)','CDS (pF)','G  (/OH)','T   (ps)',
     *'LG  (nH)','LD  (nH)'/

      DATA FMT1/'(16X,A8)       ','(12X,A8,10X,A8)','(5X,3(A8,10X)) '/
      DATA FMT2/'(5X,F18.5,5X,A8)   ','(1X,2F18.5,5X,A8)  ',
     *'(1X,3(F12.5,6X),A8)'/

      IF(NCKT.LE.0)CALL ERMSG(32,I,J,*30)
      WRITE(*,'(6X,''FET MODEL PARAMETERS    file: '',A12)')FILNAM
      WRITE(*,FMT1(NCKT))(CKTNAME(K),K=1,NCKT)
      DO 20 I=1,MP
      DO 10 K=1,NCKT
10    PP(K)=PA(I,K)*PDIM(I)
      WRITE(*,FMT2(NCKT))(PP(K),K=1,NCKT),PN(I)
20    CONTINUE
      WRITE(*,*)
30    RETURN
      END
```

```
c***********************************************************************

c       file name: FETFDFG1.FOR

c       for program FETSCG1 (PC/AT version).
c       this group of files are responsible for model simulation

c***********************************************************************

$NOFLOATCALLS


        SUBROUTINE CSOLU(Y,A,B,C,D)

c       solve a complex system of linear equations using LU factorization

        COMPLEX*8 Y(6,6),A(6),B(6),C(6),D(6),Z,T1,T2,T3,T4
        DO 20 I=1,5
        Z=Y(I,I)
        IF(Z.EQ.(0.0,0.0))GO TO 110
        M=I+1
        DO 20 J=M,6
        Y(I,J)=Y(I,J)/Z
        DO 10 K=M,6
10      Y(K,J)=Y(K,J)-Y(K,I)*Y(I,J)
20      CONTINUE

        A(1)=1.0/Y(1,1)
        C(1)=1.0
        DO 40 I=2,6
        T1=0.0
        T2=0.0
        DO 30 K=1,I-1
        T1=T1-Y(I,K)*A(K)
30      T2=T2-Y(K,I)*C(K)
        A(I)=T1/Y(I,I)
40      C(I)=T2

        B(6)=1.0/Y(6,6)
        C(6)=C(6)/Y(6,6)
        D(6)=1.0/Y(6,6)
        DO 60 L=2,6
        I=7-L
        T1=A(I)
        T2=0.0
        T3=C(I)
        T4=0.0
        DO 50 K=I+1,6
        T1=T1-Y(I,K)*A(K)
        T2=T2-Y(I,K)*B(K)
        T3=T3-Y(K,I)*C(K)
50      T4=T4-Y(K,I)*D(K)
        A(I)=T1
        B(I)=T2
```

```
      C(I)=T3/Y(I,I)
60    D(I)=T4/Y(I,I)

      RETURN

110   WRITE(*,*)'FATAL ERROR IN CSOLU: ZERO ON THE DIAGONAL'
      STOP
      END
```

```
      SUBROUTINE FDF(N,M,XX)
      PARAMETER (MP=13,MX=40,MF=20)

c     organize calculation of error functions and their derivatives
c     called by the optimizer

      CHARACTER STRG*40,WRD1*4,WRD2*11
      DIMENSION XX(N),FS(8),DFS(8,MP),INDX(MX),FM(8,MF,3)
      DIMENSION INXMDA(3),PA(MP,3),X(MX),INPX(MP,3),BND(2,MX)
      DIMENSION FREQ(MF),XA(MX),WGT(8,MF,3)
      INTEGER*2 IKY1,KYCODE,IKY2,IHIGH
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/BLK1/PDIM(MP),DAT(8,MF,3),INPP(MP),PA0(MP,3)
      COMMON/BLK2/SNR(8,3),KFR(MF,3),DTM(8,MF,3),KOUNT,KODE
      COMMON/L11/MARK,F(250),DF(250,MX),FF(250),DFF(250,MX)
      COMMON/BLKPLT/IX(10),IY(10,8,3),YMIN(8,3),YR(8,3),JOUT
      SAVE FOBJ0

      KOUNT=KOUNT+1
      M1=0
      JSEN=1
      FOBJ=0.0
      CC1=90.0/ASIN(1.0)

c     reserve the exponential transformation on the variables

      DO 10 I=1,N
      XA(I)=EXP(XX(I))
      X(I)=XA(I)+BND(1,I)
      DO 10 J=1,M
10    DF(J,I)=0.0

c     loop for NCKT circuits

      DO 60 K=1,NCKT

c             update the variable parameters

              DO 20 I=1,MP
              IF(INPX(I,K).GT.0)PA(I,K)=X(INPX(I,K))
20            CONTINUE

c     sweep for all frequencies

      DO 60 J=1,NFR
              IF(KFR(J,K).EQ.0.AND.K.NE.JOUT)GOTO 60
              CALL SIMFET(FREQ(J),PA(1,K),FS,DFS,JSEN,K)

c             8 responses (real & imag. parts of 4 s-parameters)

              DO 40 L=1,8
                      IF(WGT(L,J,K).LE.1.E-5)GOTO 40
                      M1=M1+1
                      WT=WGT(L,J,K)*SNR(L,K)
                      F(M1)=WT*(FS(L)-DTM(L,J,K))
```

```
c                       compute the objective function

                        IF(KODE.EQ.1)THEN
                                FOBJ=FOBJ+ABS(F(M1))
                        ELSE
                                FOBJ=FOBJ+F(M1)*F(M1)
                        ENDIF

c                       assemble the gradients

                        DO 30 I=1,MP
                        I1=INPX(I,K)
                        IF(I1.GT.0)DF(M1,I1)=DF(M1,I1)+WT*DFS(L,I)*XA(I1)
30                      CONTINUE
40              CONTINUE

c               transform the responses to rectangular form

                IF(K.NE.JOUT)GOTO 60
                        DO 50 I=1,4
                        I2=2*I
                        I1=I2-1
                        FM(I1,J,JOUT)=SQRT(FS(I1)*FS(I1)+FS(I2)*FS(I2))

c                       avoid discontinuity in ATAN2

                                TEMPA=CC1*ATAN2(FS(I2),FS(I1))
                                TEMPB=DAT(I2,J,JOUT)
                                IF(ABS(TEMPA-TEMPB).GT.180.0)THEN
                                        IF(TEMPA.GT.TEMPB)THEN
                                                TEMPA=TEMPA-360.0
                                        ELSE
                                                TEMPA=TEMPA+360.0
                                        ENDIF
                                ENDIF

50                      FM(I2,J,JOUT)=TEMPA
60      CONTINUE

c       check for user interrupt (key <ESC>)

        CALL CHECKY(IKY1)
        IF(IKY1.EQ.27)THEN

c               interrupt handler

                CALL CLRGRF
70              WRITE(*,80)
80              FORMAT(///15X,'USER INTERRUPT'/6X,'Press  T  to terminate',
     1           ' optimization'/6X,'Or press <CR> to continue')
                IKY2=KYCODE(IKY2)
                IHIGH=IKY2/256
                IKY2=IKY2-IHIGH*256
                IF(IKY2.EQ.84.OR.IKY2.EQ.116)THEN
```

```
c                       'T' has been pressed: terminate optimization

                        MARK=0
                        RETURN
                ELSEIF(IKY2.NE.13)THEN

c                       neither 'T' nor <CR>. wait for another key

                        GOTO 70
                ENDIF
                CALL FETPLT(FM,2)
        ENDIF

c       display the count of iteration and the value of objective function

        RM=M
        FOBJ=FOBJ*8.0/RM
        WRITE(WRD1,'(I4)')KOUNT
        IF(FOBJ.GT.999.0.OR.FOBJ.LT.1.E-6)THEN
                WRITE(WRD2,'(E11.5)')FOBJ
        ELSE
                WRITE(WRD2,'(F11.6)')FOBJ
        ENDIF
        STRG(1:16)='@L01@C01@7K=     '
        DO 90 I=1,4
        IF(WRD1(I:I).NE.' ')GOTO 100
90      CONTINUE
100     I1=17-I
        STRG(13:I1)=WRD1(I:4)
        STRG(17:33)=' ER=            _'
        DO 110 I=1,11
        IF(WRD2(I:I).NE.' ')GOTO 120
110     CONTINUE
120     I1=33-I
        STRG(21:I1)=WRD2(I:11)
        CALL DISP(STRG)
        IF(KOUNT.GT.1.AND.FOBJ.GT.FOBJ0)RETURN
        FOBJ0=FOBJ
        IF(IKY1.NE.27)CALL FETPLT(FM,3)
        RETURN
        END
```

```
      SUBROUTINE SIMFET(FREQ,PA,F,DF,JSEN,ICKT)
      PARAMETER (MP=13,MX=40,MF=20)

c     organize model simulation for a single frequency (FREQ)
c     expecting 13 parameters in the following order:
c     RG(OH),  RD(OH),  GDS(/OH),  RI(OH),  RS(OH),  LS(nH),  CGS(pF),
c     CDG(pF), CDS(pF), G(/OH),    T(ps),   LG(nH),  LD(nH)

      DIMENSION PA(MP),PAP(MP),PP(MP),F(8),DF(8,MP),DFS(8,MP)
      COMMON/BLK1/PDIM(MP),DAT(8,MF,3),INPP(MP),PAO(MP,3)
      COMMON/FETCONST/F3DB(3),GGS(3),FDIM
      COMPLEX*8 SMX(4),DSMX(4,MP),PH(6),QH(6),P(6),Q(6)
      COMPLEX*8 Y(6,6),S,Z(9),Z2P(4),ZDELTA,DZ2P(4,MP)
      COMPLEX*8 SMXMU(4),DSMXT(4),C1,C2,C3,C4,C5,C6
      INTEGER INDEX1(7),INDEX2(7)
      SAVE INDEX1,INDEX2
      DATA INDEX1,INDEX2/1,3,5,4,2,2,4,2,4,6,5,5,3,5/

      DO 1 I=1,MP
1     PAP(I)=PA(I)

ccc   because of inclusion of LG and LD, the inversions of RG -> GG
ccc   and RD -> GD are no longer appropriate, only RI (PA(4)) needs
ccc   to be inverted

ccc   DO 2 I=1,4
ccc   IF(I.EQ.3)GOTO 2
      I=4
      IF(PAP(I).GT.1.E-6)THEN
              PAP(I)=1.0/PAP(I)
      ELSE
              PAP(I)=1.E6
      ENDIF
ccc2  CONTINUE

      DO 3 I=1,MP
      II=INPP(I)
3     PP(II)=PAP(I)

c     assemble the Y matrix

              DO 10 I=1,6
              DO 10 J=1,6
10            Y(J,I)=(0.0,0.0)

              TWOPIF=6.283185*FREQ
              C1=CMPLX(0.0,TWOPIF)
              C2=0.001*C1
              C3=CMPLX(1.0,FREQ/F3DB(ICKT))
              B=TWOPIF*0.001*PP(11)
              C4=CMPLX(COS(B),-SIN(B))/C3

ccc           DO 20 I=1,4
ccc20         Z(I)=PP(I)
c             the following four lines replace the above two lines
```

```
                    Z(1)=1.0/(PP(1)+C1*PP(12))
                    Z(2)=PP(2)
                    Z(3)=1.0/(PP(3)+C1*PP(13))
                    Z(4)=PP(4)

              DO 30 I=5,7
30            Z(I)=C2*PP(I)
              Z(8)=1.0/(PP(8)+C1*PP(9))
              Z(9)=PP(10)*C4

              Y(1,1)=Z(1)
              Y(1,2)=-Z(1)
              Y(2,2)=Z(1)+Z(6)+Z(5)+GGS(ICKT)
              Y(2,3)=-Z(6)-GGS(ICKT)
              Y(2,5)=-Z(5)
              Y(3,3)=Z(2)+Z(6)+GGS(ICKT)
              Y(3,4)=-Z(2)
              Y(4,4)=Z(2)+Z(4)+Z(7)+Z(8)
              Y(4,5)=-Z(4)-Z(7)
              Y(5,5)=Z(5)+Z(4)+Z(7)+Z(3)
              Y(5,6)=-Z(3)
              Y(6,6)=Z(3)

              DO 40 I=1,5
              DO 40 J=I+1,6
40            Y(J,I)=Y(I,J)

              Y(5,2)=Y(5,2)+Z(9)
              Y(5,3)=Y(5,3)-Z(9)
              Y(4,2)=Y(4,2)-Z(9)
              Y(4,3)=Y(4,3)+Z(9)

c       solve the nodal equation for 4 solutions

        CALL CSOLU(Y,P,Q,PH,QH)

c       construct the two-port Z matrix

              Z2P(1)=P(1)/50.0
              Z2P(2)=P(6)/50.0
              Z2P(3)=Q(1)/50.0
              Z2P(4)=Q(6)/50.0
              ZDELTA=(Z2P(1)+1.0)*(Z2P(4)+1.0)-Z2P(2)*Z2P(3)

c       compute the two-port S matrix

              SMX(1)=1.0-2.0*(Z2P(4)+1.0)/ZDELTA
              SMX(2)=2.0*Z2P(2)/ZDELTA
              SMX(3)=2.0*Z2P(3)/ZDELTA
              SMX(4)=1.0-2.0*(Z2P(1)+1.0)/ZDELTA

        IF(JSEN.EQ.0)GOTO 150

c       sensitivity analysis
```

```
                DO 60 I=1,7
                J1=INDEX1(I)
                J2=INDEX2(I)
                DZ2P(1,I)=(PH(J1)-PH(J2))*(P(J2)-P(J1))
                DZ2P(2,I)=(QH(J1)-QH(J2))*(P(J2)-P(J1))
                DZ2P(3,I)=(PH(J1)-PH(J2))*(Q(J2)-Q(J1))
60              DZ2P(4,I)=(QH(J1)-QH(J2))*(Q(J2)-Q(J1))

ccc     the following 7 lines are newly added to account for LG and LD
                        C5=-Z(1)*Z(1)
                        C6=-Z(3)*Z(3)
                        DO 65 J=1,4
                        DZ2P(J,1)=DZ2P(J,1)*C5
                        DZ2P(J,12)=DZ2P(J,1)*C1
                        DZ2P(J,3)=DZ2P(J,3)*C6
65                      DZ2P(J,13)=DZ2P(J,3)*C1
ccc     end of changes

                DO 70 I=5,7
                DO 70 J=1,4
70              DZ2P(J,I)=C2*DZ2P(J,I)

                C5=Z(8)*Z(8)
                DZ2P(1,8)=C5*PH(4)*P(4)
                DZ2P(2,8)=C5*QH(4)*P(4)
                DZ2P(3,8)=C5*PH(4)*Q(4)
                DZ2P(4,8)=C5*QH(4)*Q(4)

                DO 80 J=1,4
80              DZ2P(J,9)=C1*DZ2P(J,8)

                DZ2P(1,10)=C4*(PH(5)-PH(4))*(P(3)-P(2))
                DZ2P(2,10)=C4*(QH(5)-QH(4))*(P(3)-P(2))
                DZ2P(3,10)=C4*(PH(5)-PH(4))*(Q(3)-Q(2))
                DZ2P(4,10)=C4*(QH(5)-QH(4))*(Q(3)-Q(2))

                C6=-PP(10)*C2
                DO 90 J=1,4
90              DZ2P(J,11)=C6*DZ2P(J,10)

                SMXMU(1)=1.0-SMX(1)
                SMXMU(2)=-SMX(2)
                SMXMU(3)=-SMX(3)
                SMXMU(4)=1.0-SMX(4)

                DO 100 I=1,MP
                DSMXT(1)=SMXMU(1)*DZ2P(1,I)+SMXMU(3)*DZ2P(2,I)
                DSMXT(2)=SMXMU(2)*DZ2P(1,I)+SMXMU(4)*DZ2P(2,I)
                DSMXT(3)=SMXMU(1)*DZ2P(3,I)+SMXMU(3)*DZ2P(4,I)
                DSMXT(4)=SMXMU(2)*DZ2P(3,I)+SMXMU(4)*DZ2P(4,I)

                DSMX(1,I)=(DSMXT(1)*SMXMU(1)+DSMXT(3)*SMXMU(2))/100.0
                DSMX(2,I)=(DSMXT(2)*SMXMU(1)+DSMXT(4)*SMXMU(2))/100.0
                DSMX(3,I)=(DSMXT(1)*SMXMU(3)+DSMXT(3)*SMXMU(4))/100.0
100             DSMX(4,I)=(DSMXT(2)*SMXMU(3)+DSMXT(4)*SMXMU(4))/100.0
```

```
                DO 110 K=1,4
                K1=2*K-1
                DO 110 I=1,MP
                DFS(K1,I)=REAL(DSMX(K,I))
110             DFS(K1+1,I)=AIMAG(DSMX(K,I))

                DO 120 I=1,MP
                I1=INPP(I)
                DO 120 J=1,8
120             DF(J,I)=DFS(J,I1)

ccc             DO 140 I=1,4
ccc             IF(I.EQ.3)GOTO 140
                I=4
                PR=-PAP(I)*PAP(I)
                DO 130 J=1,8
130             DF(J,I)=DF(J,I)*PR
ccc140          CONTINUE

c       separate the real and imaginary parts of the S-parameters

150             DO 160 K=1,4
                K1=2*K-1
                F(K1)=REAL(SMX(K))
160             F(K1+1)=AIMAG(SMX(K))

        RETURN
        END
```

```
c***********************************************************************

c       file name: SCFILE1.FOR

c       for program FETSCG1 (PC/AT version).
c       this group of subroutines constitutes a file handler to process
c       a circuit file and the associated measurement data files

c***********************************************************************

$NOFLOATCALLS


        BLOCK DATA A
        DIMENSION TYPDIM(6),IPTYP(18),IVTYP(40),XDIM(40),VDIM(40)
        COMMON/SC1/TYPDIM,IPTYP,XDIM,VDIM,IVTYP
        DATA TYPDIM/1.E-9,1.0,1.0,1.E9,1.E12,1.E12/
        DATA IPTYP/3,5,1,6,3,5,5,5,3,2,2,2,2,5,5,4,4,4/
        END
```

```
      SUBROUTINE BLKBLK(IERROR)
      PARAMETER (MP=13,MX=40,MF=20)

c     decode the 'BLK' block

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,WRD*80,KYWRD*3,CHARR
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3),X(MX)
      DIMENSION PA(MP,3),WGT(8,MF,3),INPX(MP,3),BND(2,MX),INDX(MX)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),FREQ(MF)
      DIMENSION TYPDIM(6),IPTYP(18),IVTYP(MX),XDIM(MX),VDIM(MX)
      COMMON/SC1/TYPDIM,IPTYP,XDIM,VDIM,IVTYP
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1     TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/FETCONST/F3DB(3),GGS(3),FDIM
      LTOP=LOCBLK(1,1)
      LBOTTOM=LOCBLK(2,1)
      IERROR=1
      IF(LTOP.EQ.0.OR.LBOTTOM.LE.LTOP)CALL ERMSG(1,I,J1,*80)
      DO 10 K=1,6
10    CKTNAME(K)='        '
      NX=0
      NVAR=0
      I=LTOP
      NCKT=0
      NFET=0
20    I=I+1
      IF(I.GT.LBOTTOM)GOTO 50
      J0=1
      CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
      IF(LENG.EQ.0)GOTO 20
      CALL GETSYMBL(WRD,LENG,IP,IFLAG)
      IF(NFET.EQ.1)GOTO 30
      IF(IFLAG.EQ.3)THEN
              LENG=IP-1
              J0=J1+IP
              IF(LENG.EQ.0.OR.LENG.GT.8)CALL ERMSG(6,I,J1,*80)
              NVAR=NVAR+1
              VNAME(NVAR)='        '
              VNAME(NVAR)(1:LENG)=WRD(1:LENG)
              CALL GETVAR(I,J0,NVAR,*80)
              IF(LOCV(3,NVAR).NE.0)THEN
                      NX=NX+1
                      INVAR(NVAR)=NX
                      X(NX)=VAL(1,NVAR)
                      XDIM(NX)=VDIM(NVAR)
                      LOCX(1,NX)=LOCV(1,NVAR)
                      LOCX(2,NX)=LOCV(2,NVAR)
                      INDX(NX)=LOCV(3,NVAR)
                      IF(INDX(NX).EQ.2)THEN
                              BND(1,NX)=VAL(2,NVAR)
                              BND(2,NX)=VAL(3,NVAR)
                      ENDIF
              ENDIF
```

```
          GOTO 20
      ENDIF
      IF(LENG.NE.3.OR.WRD(1:3).NE.'FET')CALL ERMSG(14,I,1,*80)
      CALL GETFET(I,J0,*80)
      NFET=1
      GOTO 20
30    IF(IFLAG.NE.3.OR.IP.NE.LENG.OR.LENG.LT.2)CALL ERMSG(16,I,1,*80)
      LENG=LENG-1
      NND=NCKT*3-2
      CKTNAME(NCKT)=WRD(1:LENG)
      CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
      IF(LENG.NE.4.OR.WRD(1:4).NE.'2POR')CALL ERMSG(18,I,J1,*80)
      CALL GETINT(TT(1,I),J0,J1,ND1,*90)
      IF(ND1.NE.NODE(NND))CALL ERMSG(15,I,J1,*80)
      CALL GETINT(TT(1,I),J0,J1,ND2,*90)
      IF(ND2.NE.NODE(NND+1))CALL ERMSG(15,I,J1,*80)
      NFET=0
      GOTO 20
50    IF(NFET.NE.0)CALL ERMSG(16,I,1,*80)
      IERROR=0
80    RETURN
90    CALL ERMSG(7,I,J1,*80)
      END
```

```
      SUBROUTINE CHKTYP(ITY,I,*)

c     make sure that a parameter and a symbolic variable that is assigned
c     to it have consistent types (capacitor, inductor, etc.)

      DIMENSION ITY(3)
      IF(ITY(2).EQ.7)GOTO 90
      IF(ITY(1).EQ.0.OR.ITY(1).EQ.7)GOTO 20
      IF(ITY(2).NE.0.AND.ITY(1).NE.ITY(2))GOTO 90
      IF(ITY(3).EQ.0.OR.ITY(3).EQ.7)GOTO 80
      IF(ITY(1).NE.ITY(3))GOTO 90
20    IF(ITY(3).EQ.0.OR.ITY(3).EQ.7)GOTO 80
      IF(ITY(2).EQ.0)GOTO 80
      IF(ITY(2).NE.ITY(3))GOTO 90
80    IF((ITY(1).NE.0).AND.(ITY(1).NE.7))ITY(2)=ITY(1)
      IF((ITY(3).NE.0).AND.(ITY(3).NE.7))ITY(2)=ITY(3)
      RETURN
90    CALL ERMSG(29,I,1,*100)
100   RETURN 1
      END
```

```
      SUBROUTINE DATABLK(IERROR)
      PARAMETER (MP=13,MX=40,MF=20)

c     decode the 'DATA' block

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,WRD*80
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3),X(MX)
      DIMENSION PA(MP,3),WGT(8,MF,3),INPX(MP,3),BND(2,MX),INDX(MX)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),FREQ(MF)
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1      TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      DO 10 I=1,6
10    CKTFIL(I)='            '
      IERROR=1
      LTOP=LOCBLK(1,4)
      LBOTTOM=LOCBLK(2,4)
      IF(LTOP.EQ.0.OR.LTOP.GE.LBOTTOM)CALL ERMSG(35,I,J1,*100)
      LTOP=LTOP+1
      NMDA=3
      DO 90 I=LTOP,LBOTTOM
      J0=1
      CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
      IF(LENG.EQ.0)GOTO 90
      CALL GETSYMBL(WRD,LENG,IP,IFLAG)
      IF(IP.EQ.1.OR.IFLAG.NE.3)CALL ERMSG(30,I,J1,*100)
      NMDA=NMDA+1
      IF(NMDA.GT.6)CALL ERMSG(5,I,J1,*100)
      LENG=IP-1
      CKTNAME(NMDA)(1:LENG)=WRD(1:LENG)
      J0=IP+1
      CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
      IF(LENG.EQ.0)CALL ERMSG(17,I,J1,*100)
      CKTFIL(NMDA)(1:LENG)=WRD(1:LENG)
90    CONTINUE
      IERROR=0
100   RETURN
      END
```

```
      SUBROUTINE ERMSG(IER,ILINE,ICOL,*)
      PARAMETER (MP=13,MX=40,MF=20)

c     analyze an error and display the corresponding message

      CHARACTER TT(100)*80,VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,T*79
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8)
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1      TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      IF(IER.LE.3)THEN
              WRITE(*,10)MSG(IER)
      ELSEIF(IER.LE.30)THEN
              T=TT(ILINE)(1:79)
              WRITE(*,10)MSG(IER)
              WRITE(*,20)ILINE,ICOL
              WRITE(*,'(1X,A79)')T
      ELSE
              WRITE(*,30)MSG(IER)
      ENDIF
      RETURN 1
10    FORMAT(5X,'Error in circuit file: ',A30)
20    FORMAT(5X,'Error in circuit file at line ',I3,' after column',I3)
30    FORMAT(5X,'Error: ',A30)
      END
```

```
      SUBROUTINE FERMSG(IER,FILNAM,ILINE,*)

c     display file-related error messages

      CHARACTER FILNAM*12
      IF(IER.EQ.1)WRITE(*,10)FILNAM
      IF(IER.EQ.2)WRITE(*,20)FILNAM,ILINE
      IF(IER.EQ.3)WRITE(*,30)FILNAM
      IF(IER.EQ.4)WRITE(*,40)FILNAM
      RETURN 1
10    FORMAT(5X,'File not found: ',A12)
20    FORMAT(5X,'Bad data in the file ',A12,' at line',I3)
30    FORMAT(5X,'Not enough data in file ',A12)
40    FORMAT(5X,'File missing: ',A12)
      END
```

```
      SUBROUTINE FREQBLK(IERROR)
      PARAMETER (MP=13,MX=40,MF=20)

c     decode the 'FREQ' block

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,WRD*80
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3),X(MX)
      DIMENSION PA(MP,3),WGT(8,MF,3),INPX(MP,3),BND(2,MX),INDX(MX)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),FREQ(MF)
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1     TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      IERROR=1
      LTOP=LOCBLK(1,2)
      LBOTTOM=LOCBLK(2,2)
      IF(LTOP.EQ.0.OR.LTOP.GE.LBOTTOM)CALL ERMSG(2,I,J1,*100)
      LTOP=LTOP+1
      NFR=0
      EPS=1.E-12
      DO 30 I=LTOP,LBOTTOM
      J0=1
10    J00=J0
      CALL GETWRD(TT(1,I),J00,J1,WRD,LENG)
      IF(LENG.EQ.0)GOTO 30
      II=ICHAR(WRD(1:1))
      IF(II.GE.65.AND.II.LE.122)THEN
              CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
              IF(LENG.EQ.4.AND.WRD(1:4).EQ.'STEP')THEN
                      CALL GETFRQ(I,J0,J1,FLOW,*100)
                      CALL GETFRQ(I,J0,J1,FHIGH,*100)
                      CALL GETFRQ(I,J0,J1,DFR,*100)
                      IF(DFR.LT.EPS)GOTO 120
                      IF(FLOW.LT.EPS)GOTO 120
                      IF(FHIGH.LT.EPS)GOTO 120
                      IF(FHIGH.LT.FLOW)THEN
                              TEMP=FHIGH
                              FHIGH=FLOW
                              FLOW=TEMP
                      ENDIF
                      N1=INT((FHIGH-FLOW)/DFR)+1
                      N2=NFR+N1
                      IF(N2.GT.MF)GOTO 110
                      DO 20 J=1,N1
                      FREQ(NFR+J)=FLOW
20                    FLOW=FLOW+DFR
                      NFR=N2
              ELSE
                      CALL ERMSG(8,I,J1,*100)
              ENDIF
      ELSE
              IF(NFR.GE.MF)GOTO 110
              NFR=NFR+1
              CALL GETFRQ(I,J0,J1,FREQ(NFR),*100)
      ENDIF
```

```
      GOTO 10
30    CONTINUE
      IF(NFR.EQ.0)CALL ERMSG(2,I,J1,*100)
      IERROR=0
100   RETURN
110   CALL ERMSG(19,I,J1,*100)
120   CALL ERMSG(20,I,J1,*100)
      END
```

```
      SUBROUTINE GETCHAR(STRG,ITOP,ICHR,CHARR,*)

c     get a single character from a line starting from the ITOPth position

      CHARACTER STRG(80),CHARR
      ICHR=0
      IF(ITOP.GT.80)RETURN 1
      DO 10 I=ITOP,80
      IF(STRG(I).NE.' ')GOTO 20
10    CONTINUE
      RETURN 1
20    CHARR=STRG(I)
      ICHR=I
      ITOP=I+1
      RETURN
      END
```

```
      SUBROUTINE GETFET(I,J0,*)
      PARAMETER (MP=13,MX=40,MF=20)

c     decode the FET element

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,CHARR,PARNAM(18)*3,WRD*80
      CHARACTER KYWRD*3,VLABL*8
      DIMENSION IDP(18),ICO(18),VAL(3,MX),LOCV(3,MX),INXMDA(3),X(MX)
      DIMENSION PA(MP,3),WGT(8,MF,3),INPX(MP,3),BND(2,MX),INDX(MX)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),FREQ(MF)
      DIMENSION TYPDIM(6),IPTYP(18),IVTYP(MX),XDIM(MX),VDIM(MX)
      COMMON/SC1/TYPDIM,IPTYP,XDIM,VDIM,IVTYP
      COMMON/TS1/LOCBLK(2,10),LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1      TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/FETCONST/F3DB(3),GGS(3),FDIM
      SAVE PARNAM,IDP,ICO,IL
      DATA PARNAM/'G  ','CGS','F  ','T  ','GGS','CDG','CDC','CDS',
     1'GDS','RI ','RG ','RD ','RS ','CGE','CDE','LG ','LD ','LS '/
ccc   DATA IDP/10,7,6,11,6,8,6,9,3,4,1,2,5,6,6,6,6,6/
      DATA IDP/10,7,6,11,6,8,6,9,3,4,1,2,5,6,6,12,13,6/
ccc   DATA ICO/2,2,1,2,-1,2,0,2,2,2,2,2,2,0,0,0,0,2/
      DATA ICO/2,2,1,2,-1,2,0,2,2,2,2,2,2,0,0,2,2,2/

      IG=NCKT*3+1
      NCKT=NCKT+1
      IF(NCKT.GT.3)CALL ERMSG(5,I,1,*150)
      CALL GETINT(TT(1,I),J0,J1,NODE(IG),*140)
      CALL GETINT(TT(1,I),J0,J1,NODE(IG+1),*140)
      CALL GETCHAR(TT(1,I),J0,J1,CHARR,*10)
      IF(CHARR.EQ.'0')GOTO 10
      IF(CHARR.EQ.'G')THEN
              J0=J1
              GOTO 10
      ELSE
              CALL ERMSG(21,I,J1,*150)
      ENDIF
10    DO 20 K=1,MP
      PA(K,NCKT)=0.0
20    INPX(K,NCKT)=0
      F3DB(NCKT)=0.0
      GGS(NCKT)=0.0
      K=1
30    CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
      IF(LENG.EQ.0)THEN
              I=I+1
              J0=1
              CALL GETCHAR(TT(1,I),J0,J1,CHARR,*40)
              IF(CHARR.EQ.'+')GOTO 30
40            I=I-1
              GOTO 100
      ENDIF
      CALL NOSYMBL(WRD,LENG,J0,J1)
      IF(LENG.EQ.0.OR.LENG.GT.3)GOTO 60
```

```
      KYWRD='    '
      KYWRD(1:LENG)=WRD(1:LENG)
      DO 50 KK=K,18
      IF(KYWRD.EQ.PARNAM(KK))GOTO 70
50    CONTINUE
60    CALL ERMSG(8,I,J1,*150)
70    K=KK+1
      IPAR=IDP(KK)
      JTYP=IPTYP(KK)
      J00=J0
      CALL GETCHAR(TT(1,I),J00,J1,CHARR,*160)
      IF(CHARR.EQ.'=')J0=J00
      J00=J0
      CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
      IF(LENG.EQ.0)GOTO 160
      IA=ICHAR(WRD(1:1))
      IF(IA.GE.65.AND.IA.LE.122)THEN
              IF(NVAR.EQ.0)GOTO 170
              IF(LENG.GT.8)CALL ERMSG(6,I,J1,*150)
              VLABL='        '
              VLABL(1:LENG)=WRD(1:LENG)
              DO 80 L=1,NVAR
              IF(VLABL.EQ.VNAME(L))GOTO 90
80            CONTINUE
              GOTO 170
      ELSE
              J0=J00
              CALL GETVAR(I,J0,MX,*150)
              IF(LOCV(3,MX).NE.0)THEN
                      NX=NX+1
                      INVAR(MX)=NX
                      X(NX)=VAL(1,MX)
                      XDIM(NX)=VDIM(MX)
                      LOCX(1,NX)=LOCV(1,MX)
                      LOCX(2,NX)=LOCV(2,MX)
                      INDX(NX)=LOCV(3,MX)
                      IF(INDX(NX).EQ.2)THEN
                              BND(1,NX)=VAL(2,MX)
                              BND(2,NX)=VAL(3,MX)
                      ENDIF
              ENDIF
              L=MX
      ENDIF
90    IF(IVTYP(L).EQ.0)THEN
              IVTYP(L)=JTYP
              VAL(1,L)=VAL(1,L)*TYPDIM(JTYP)
              IF(LOCV(3,L).NE.0)THEN
                      JX=INVAR(L)
                      X(JX)=X(JX)*TYPDIM(JTYP)
                      XDIM(JX)=XDIM(JX)*TYPDIM(JTYP)
                      IF(INDX(JX).EQ.2)THEN
                              BND(1,JX)=BND(1,JX)*TYPDIM(JTYP)
                              BND(2,JX)=BND(2,JX)*TYPDIM(JTYP)
                      ENDIF
              ENDIF
```

```
        ELSEIF(IVTYP(L).NE.JTYP)THEN
                CALL ERMSG(29,I,J1,*150)
        ENDIF
        PA(IPAR,NCKT)=VAL(1,L)
        IF(LOCV(3,L).EQ.0)THEN
                INPX(IPAR,NCKT)=0
        ELSE
                INPX(IPAR,NCKT)=INVAR(L)
        ENDIF
        IJ=INPX(6,NCKT)
        ZZ=PA(6,NCKT)
        IF(ICO(KK).EQ.1)THEN
                F3DB(NCKT)=ZZ
                IF(IJ.NE.0)CALL ERMSG(11,I,J1,*150)
        ELSEIF(ICO(KK).EQ.-1)THEN
                GGS(NCKT)=ZZ
                IF(IJ.NE.0)CALL ERMSG(12,I,J1,*150)
        ELSEIF(ICO(KK).EQ.0)THEN
                IF(IJ.NE.0.OR.ZZ.NE.0.0)CALL ERMSG(13,I,J1,*150)
        ENDIF
        IF(K.LE.18)GOTO 30
100     RETURN
140     CALL ERMSG(7,I,J1,*150)
150     RETURN1
160     CALL ERMSG(4,I,J1,*150)
170     CALL ERMSG(10,I,J1,*150)
        END
```

```
      SUBROUTINE GETFRQ(I,J0,J1,TEMP,*)
      PARAMETER (MP=13,MX=40,MF=20)

c     decode a single frequency

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX), NODE(9),LOCX(2,MX)
      DIMENSION TYPDIM(6),IPTYP(18),IVTYP(MX),XDIM(MX),VDIM(MX)
      COMMON/SC1/TYPDIM,IPTYP,XDIM,VDIM,IVTYP
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR(MX),LOCX,VAL,DIM(8),
     1      TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      CALL GETVAR(I,J0,MX,*100)
      IF(IVTYP(MX).GT.1)CALL ERMSG(29,I,J1,*100)
      IF(LOCV(3,MX).NE.0)CALL ERMSG(4,I,J1,*100)
      TEMP=VAL(1,MX)
      RETURN
100   RETURN 1
      END
```

```
      SUBROUTINE GETINT(STRG,ITOP,IP,IVAL,*)

c     try to convert an ASCII string into an integer

      CHARACTER STRG(80),WRD*80,WRD1*20
      CALL GETWRD(STRG,ITOP,IP,WRD,LENG)
      IF(LENG.EQ.0)RETURN 1
      WRD1='                    '
      WRD1(1:LENG)=WRD(1:LENG)
      READ(WRD1,*,ERR=10,END=10)IVAL
      IF(IVAL.GE.0)RETURN
10    RETURN 1
      END
```

```
      SUBROUTINE GETREAL(WRD,LENG,VALUE,*)

c     try to convert an ASCII string into a real value

      CHARACTER WRD*80,WRD1*20
      IF(LENG.EQ.0)GOTO 10
      WRD1='                    '
      WRD1(1:LENG)=WRD(1:LENG)
      READ(WRD1,*,ERR=10,END=10)VALUE
      IF(VALUE.GE.0.0)RETURN
10    RETURN 1
      END
```

```
      SUBROUTINE GETSYMBL(WRD,LENG,IP,IFLAG)

c     detect special symbols in a word

      CHARACTER SYMBOLS(10),WRD*80,A
      SAVE SYMBOLS
      DATA SYMBOLS/'=','?',':','+','-','%','*','/','>','<'/
      IFLAG=0
      DO 10 I=1,LENG
      A=WRD(I:I)
      DO 10 J=1,4
      IF(A.EQ.SYMBOLS(J))GOTO 20
10    CONTINUE
      RETURN
20    IP=I
      IFLAG=J
      RETURN
      END
```

```
      SUBROUTINE GETUNIT(WRD,LENG,IP,ITYPE,VAL)

c     decode the unit attached to a numeral

      CHARACTER WRD*80,KYWRD*4,UNAM(6,7)*4
      DIMENSION DDIM(6,7)
      SAVE UNAM,DDIM
      DATA UNAM/'HZ  ','KHZ ','MHZ ','GHZ ','    ','    ','OH  ','KOH ',
     1'MOH ','    ','    ','    ','/OH ','/KOH','/MOH','    ','    ',
     2'    ','L   ','MH  ','UH  ','NH  ','PH  ','    ','F   ','MF  ',
     3'UF  ','NF  ','PF  ','FF  ','SEC ','MS  ','US  ','NS  ','    ',
     4'    ','%   ','    ','    ','    ','    ','    ','    '/
      DATA DDIM/1.,1.E3,1.E6,1.E9,0.,0.,1.,1.E3,1.E6,0.,0.,0.,1.,1.E-3,
     1  1.E-6,0.,0.,0.,1.,1.E-3,1.E-6,1.E-9,1.E-12,0.,1.,1.E-3,1.E-6,
     2  1.E-9,1.E-12,1.E-15,1.0,1.E-3,1.E-6,1.E-9,8*0.0/
      IP=0
      ITYPE=0
      VAL=1.0
      DO 10 I=LENG,1,-1
      IA=ICHAR(WRD(I:I))
      IF(IA.GE.48.AND.IA.LE.57)GOTO 20
       IF(IA.EQ.4)GOTO 20
10     CONTINUE
       RETURN
20     IF(I.EQ.LENG)RETURN
       IP=I+1
       IG=LENG-IP+1
       IF(IG.GT.4)GOTO 40
       KYWRD='    '
       KYWRD(1:IG)=WRD(IP:LENG)
       DO 30 I=1,7
       DO 30 J=1,6
       IF(KYWRD.EQ.UNAM(J,I))GOTO 50
30     CONTINUE
40     IP=-1
       RETURN
50     ITYPE=I
       VAL=DDIM(J,I)
       RETURN
       END
```

```
      SUBROUTINE GETVAR(I,J0,NVR,*)
      PARAMETER (MP=13,MX=40,MF=20)

c     decode infomation relating to a variable including
c     its value, type (bounded or not) and the bounds on it

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,WRD*80,A
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3),ITY(3)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),VU(3),VL(3)
      DIMENSION TYPDIM(6),IPTYP(18),IVTYP(MX),XDIM(MX),VDIM(MX)
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1     TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/SC1/TYPDIM,IPTYP,XDIM,VDIM,IVTYP
      J00=J0
      INDXV=0
      CALL GETCHAR(TT(1,I),J00,J1,A,*70)
      IF(A.EQ.'?')THEN
              INDXV=1
              J0=J00
              JX=J00
      ENDIF
      KOUNT=1
10    CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
      CALL NOSYMBL(WRD,LENG,J0,J1)
      IF(LENG.EQ.0)GOTO 70
      CALL GETUNIT(WRD,LENG,IP,ITY(KOUNT),VU(KOUNT))
      IF(IP.LT.0)CALL ERMSG(25,I,J1,*70)
      IF(IP.GT.0)LENG=IP-1
      CALL GETREAL(WRD,LENG,VL(KOUNT),*70)
      IF(INDXV.EQ.0)GOTO 25
      IF(KOUNT.EQ.1)THEN
              J00=J0
              CALL GETCHAR(TT(1,I),J00,J11,A,*80)
              IF(A.EQ.'?')GOTO 20
              JX=J0
              KOUNT=2
              LOCV(3,NVR)=2
      ELSEIF(KOUNT.EQ.2)THEN
              KOUNT=3
              JX1=J1+LENG
              CALL PAD(TT(1,I),JX,JX1,ISHIFT)
              J0=J0+ISHIFT
      ELSE
              CALL CHKTYP(ITY,I,*90)
              CALL GETCHAR(TT(1,I),J0,J1,A,*80)
              IF(A.NE.'?')GOTO 80
              GOTO 30
      ENDIF
      GOTO 10
20    JX1=J1+LENG
      CALL PAD(TT(1,I),JX,JX1,ISHIFT)
      J0=J00+ISHIFT
25    LOCV(3,NVR)=INDXV
      LOCV(2,NVR)=JX
```

```
      VDIM(NVR)=VU(1)
      IF(ITY(1).EQ.7)THEN
              CALL ERMSG(29,I,1,*90)
      ELSEIF(ITY(1).NE.0)THEN
              VDIM(NVR)=VDIM(NVR)*TYPDIM(ITY(1))
      ENDIF
      IVTYP(NVR)=ITY(1)
      VAL(1,NVR)=VL(1)*VDIM(NVR)
      GOTO 40
30    LOCV(2,NVR)=JX
      VDIM(NVR)=VU(2)
      IF(ITY(2).NE.0)VDIM(NVR)=VDIM(NVR)*TYPDIM(ITY(2))
      VAL(1,NVR)=VL(2)*VDIM(NVR)
      IF(ITY(1).EQ.7)THEN
              VAL(2,NVR)=VAL(1,NVR)*VL(1)*0.01
      ELSE
              IF(ITY(2).NE.0)VU(1)=VU(1)*TYPDIM(ITY(2))
              VAL(2,NVR)=VL(1)*VU(1)
      ENDIF
      IF(ITY(3).EQ.7)THEN
              VAL(3,NVR)=VAL(1,NVR)*VL(3)*0.01
      ELSE
              IF(ITY(2).NE.0)VU(3)=VU(3)*TYPDIM(ITY(2))
              VAL(3,NVR)=VL(3)*VU(3)
      ENDIF
      IF(VAL(3,NVR).LE.VAL(2,NVR))CALL ERMSG(26,I,1,*90)
      IVTYP(NVR)=ITY(2)
40    LOCV(1,NVR)=I
      RETURN
70    CALL ERMSG(4,I,J1,*90)
80    CALL ERMSG(22,I,J1,*90)
90    RETURN1
      END
```

```
      SUBROUTINE GETWRD(STRG,ITOP,IWRD,WRD,LENG)

c     get a word from a line (STRG) starting from the ITOPth position.
c     words are separated by blanks. return the word in WRD
c     its length in LENG

      CHARACTER STRG(80),WRD*80,A
      LENG=0
      IF(ITOP.GT.80)RETURN
      DO 10 I=ITOP,80
      IF(STRG(I).NE.' ')GOTO 20
10    CONTINUE
      RETURN
20    IF(STRG(I).EQ.'*')RETURN
      IWRD=I
      IW=1
      DO 30 I=IWRD,80
      A=STRG(I)
      IF(A.EQ.' ')GOTO 40
      IA=ICHAR(A)
      IF(IA.GE.97.AND.IA.LE.122)A=CHAR(IA-32)
      WRD(IW:IW)=A
      IW=IW+1
30    CONTINUE
40    ITOP=I
      LENG=ITOP-IWRD
      RETURN
      END
```

```
      SUBROUTINE NODUMMY(ILINE)

c     skip over comment lines in the measurement files

      CHARACTER CLINE*80,DUMMY
1     READ(1,'(A)',END=4)CLINE
      J0=1
      ILINE=ILINE+1
2     CALL GETCHAR(CLINE,J0,J1,DUMMY,*1)
      IF(DUMMY.EQ.CHAR(9))GOTO 2
      IF(DUMMY.EQ.'*')GOTO 1
      IDUMMY=ICHAR(DUMMY)-48
      IF(IDUMMY.GE.0.AND.IDUMMY.LE.9)GOTO 3
      IF(DUMMY.NE.'+'.AND.DUMMY.NE.'-'.AND.DUMMY.NE.'.')GOTO 1
3     BACKSPACE 1
      ILINE=ILINE-1
4     RETURN
      END
```

```
      SUBROUTINE NOSYMBL(WRD,LENG,J0,J1)

c     detect special symbols within a word

      CHARACTER WRD*80
      IF(LENG.EQ.0)RETURN
      CALL GETSYMBL(WRD,LENG,IP,IFLAG)
      IF(IFLAG.GT.0)THEN
              J0=J1+IP-1
              LENG=IP-1
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE OPTBLK(IERROR)
      PARAMETER (MP=13,MX=40,MF=20)

c     decode the 'OPT' block

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8
      CHARACTER WRD*80,PNAME*8,CNAME*7,CMDA(8)*7,CHARR
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3),X(MX)
      DIMENSION PA(MP,3),WGT(8,MF,3),INPX(MP,3),BND(2,MX),INDX(MX)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),FREQ(MF),WW(4)
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1     TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      DO 10 I=1,3
      DO 10 J=1,8
      DO 10 K=1,NFR
10    WGT(K,J,I)=0.0
      ETERM=0.0
      IERROR=1
      LTOP=LOCBLK(1,3)
      LBOTTOM=LOCBLK(2,3)
      IF(LTOP.EQ.0.OR.LTOP.GE.LBOTTOM)GOTO 210
      I=LTOP
      JCKT=0
20    I=I+1
      IF(I.GT.LBOTTOM)GOTO 150
      J0=1
30    CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
      IF(LENG.EQ.0)GOTO 20
      CHARR=WRD(1:1)
      IF(CHARR.EQ.'F'.AND.JCKT.GT.0)THEN
              IF(LENG.GT.1.AND.WRD(2:2).NE.'=')GOTO 100
              J0=J1+1
              CALL GETCHAR(TT(1,I),J0,J1,CHARR,*240)
              IF(CHARR.NE.'=')GOTO 220
              CALL GETFRQ(I,J0,J1,FLOW,*200)
              FHIGH=FLOW
              J00=J0
              CALL GETCHAR(TT(1,I),J00,J1,CHARR,*20)
              II=ICHAR(CHARR)
              IF(II.GE.43.AND.II.LE.57)CALL GETFRQ(I,J0,J1,FHIGH,*200)
              GOTO 30
      ELSEIF(CHARR.EQ.'S'.AND.JCKT.GT.0)THEN
              IF(LENG.GT.1.AND.WRD(2:2).NE.'=')GOTO 100
              J0=J1+1
              CALL GETCHAR(TT(1,I),J0,J1,CHARR,*240)
              IF(CHARR.NE.'=')GOTO 220
              CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
              IF(LENG.EQ.0.OR.LENG.GT.8)GOTO 240
              PNAME='        '
              PNAME(1:LENG)=WRD(1:LENG)
              DO 40 K1=4,6
              IF(PNAME.EQ.CKTNAME(K1))GOTO 50
40            CONTINUE
```

```
                GOTO 220
50              INXMDA(JCKT)=K1
                J00=J0
                CALL GETCHAR(TT(1,I),J00,J1,CHARR,*80)
                IF(CHARR.EQ.'W')THEN
                        J0=J00
                        CALL GETCHAR(TT(1,I),J00,J1,CHARR,*60)
                        IF(CHARR.EQ.'=')J0=J00
60                      DO 70 J=1,4
                        CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
                        CALL GETREAL(WRD,LENG,WW(J),*230)
70                      CONTINUE
                        TEMP=WW(2)
                        WW(2)=WW(3)
                        WW(3)=TEMP
                ENDIF
80              DO 95 J=1,NFR
                IF(FREQ(J).LT.FLOW.OR.FREQ(J).GT.FHIGH)GOTO 95
                DO 90 L=1,4
                L2=2*L
                L1=L2-1
                WGT(L1,J,JCKT)=WW(L)
90              WGT(L2,J,JCKT)=WW(L)
95              CONTINUE
                GOTO 30
        ELSEIF(LENG.EQ.4.AND.WRD(1:4).EQ.'TERM')THEN
                J0=J1+4
                CALL GETCHAR(TT(1,I),J0,J1,CHARR,*250)
                IF(CHARR.NE.'=')GOTO 220
                CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
                CALL GETREAL(WRD,LENG,ETERM,*230)
                GOTO 30
        ENDIF
100     IF(LENG.GT.8)GOTO 250
        PNAME='        '
        PNAME(1:LENG)=WRD(1:LENG)
        DO 110 K0=1,3
        IF(PNAME.EQ.CKTNAME(K0))GOTO 120
110     CONTINUE
        GOTO 250
120     JCKT=K0
        FHIGH=1.E20
        FLOW=-100.0
        DO 130 L=1,4
130     WW(L)=1.0
        GOTO 30
150     DO 160 I=1,NCKT
        IF(INXMDA(I).GT.0)GOTO 160
                WRITE(*,260)CKTNAME(I)
                GOTO 200
160     CONTINUE
        IERROR=0
200     RETURN
210     CALL ERMSG(3,I,J1,*200)
220     CALL ERMSG(8,I,J1,*200)
```

```
230     CALL ERMSG(4,I,J1,*200)
240     CALL ERMSG(24,I,J1,*200)
250     IF(JCKT.EQ.0)CALL ERMSG(16,I,J1,*200)
        GOTO 220
260     FORMAT(5X,'Error in CKT file:  ',A8,' not used for modelling')
        END
```

```
      SUBROUTINE OUTBLK(IERROR,NCKT)
      PARAMETER (MP=13,MX=40,MF=20)

c     decode the 'OUT' block

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,WRD*80,WRD1*8,WRD2*3
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8)
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1     TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/BLKPLT/IX(10),IY(10,8,3),YMIN(8,3),YR(8,3),JOUT
      JOUT=1
      IF(NCKT.LE.1)RETURN
      LTOP=LOCBLK(1,5)
      LBOTTOM=LOCBLK(2,5)
      IF(LTOP.EQ.0.OR.LTOP.GE.LBOTTOM)RETURN
      LTOP=LTOP+1
      DO 20 I=LTOP,LBOTTOM
      J0=1
      CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
      IF(LENG.EQ.0)GOTO 20
      IF(LENG.NE.3)GOTO 40
      WRD2=WRD(1:3)
      IF(WRD2.EQ.'PRI')THEN
          CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
          IF(LENG.NE.3)GOTO 20
          WRD2=WRD(1:3)
          IF(WRD2.NE.'PLO')GOTO 20
      ENDIF
      IF(WRD2.EQ.'PLO')THEN
          CALL GETWRD(TT(1,I),J0,J1,WRD,LENG)
          IF(LENG.EQ.0.OR.LENG.GT.8)GOTO 50
          WRD1='        '
          WRD1(1:LENG)=WRD(1:LENG)
          DO 10 K=1,NCKT
          IF(WRD1.EQ.CKTNAME(K))GOTO 30
10        CONTINUE
          GOTO 50
      ELSE
          GOTO 40
      ENDIF
20    CONTINUE
      RETURN
30    JOUT=K
      RETURN
40    CALL ERMSG(8,I,J1,*60)
50    CALL ERMSG(16,I,J1,*60)
60    IERROR=1
      RETURN
      END
```

```
      SUBROUTINE PAD(STRG,JX,JX1,ISHIFT)

c     insert blanks between words so that 8 digits can be accomodated

      CHARACTER STRG(80)
      ISHIFT=8-JX1+JX
      IF(ISHIFT.LE.0)RETURN
      I00=80-ISHIFT
      IF(I00.LE.JX1)RETURN
      DO 10 I=I00,JX1,-1
10    STRG(I+ISHIFT)=STRG(I)
      J3=JX+7
      DO 20 I=JX1,J3
20    STRG(I)='0'
      RETURN
      END
```

```
      SUBROUTINE RDCKT(FILNAM,OLDFIL,IERROR)
      PARAMETER (MP=13,MX=40,MF=20)

c     organize the process of reading circuit and measurement files
c     it is called by the main program.  It is the link between
c     this group of files and the rest of the program

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,A*80,KYWRD*2,STRG*17
      CHARACTER FILNAM*12,OLDFIL*12
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3)
      DIMENSION PA(MP,3),X(MX),INPX(MP,3),BND(2,MX),INDX(MX)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),FREQ(MF)
      DIMENSION PDIM(MP),DAT(8,MF,3),INPP(MP),WGT(8,MF,3)
      LOGICAL THERE
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1     TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/BLK1/PDIM,DAT,INPP,PA0(MP,3)
      COMMON/BLK2/SNR(8,3),KFR(MF,3),DTM(8,MF,3),KOUNT,KODE
      COMMON/FETCONST/F3DB(3),GGS(3),FDIM

      OPEN(1,FILE=FILNAM,STATUS='OLD')
      WRITE(*,'(5X,''Reading file: '',A12)')FILNAM

c     read circuit file

      CALL RDSCF(IERROR)
      WRITE(*,*)
      IF(IERROR.NE.1)CALL BLKBLK(IERROR)
      IF(IERROR.NE.1)CALL FREQBLK(IERROR)
      IF(IERROR.NE.1)CALL DATABLK(IERROR)
      IF(IERROR.NE.1)CALL OPTBLK(IERROR)
      IF(IERROR.NE.1)CALL OUTBLK(IERROR,NCKT)
      CLOSE(1)
      IF(IERROR.EQ.1)GOTO 230
      OLDFIL=FILNAM

c     read measurement files

      DO 40 K=1,NCKT
      FILNAM=CKTFIL(INXMDA(K))
      INQUIRE(FILE=FILNAM,EXIST=THERE)
      IF(.NOT.THERE)CALL FERMSG(1,FILNAM,I,*220)
      OPEN(1,FILE=FILNAM,STATUS='OLD')
      EPS=0.001*FREQ(1)
      ILINE=1
      DO 30 J=1,NFR
20    CALL NODUMMY(ILINE)
      READ(1,*,ERR=190,END=200)FRE,(DAT(M,J,K),M=1,8)
      ILINE=ILINE+1
      IF(ABS(FRE-FREQ(J)).GE.EPS)GOTO 20
30    CONTINUE
      CLOSE(1)
40    CONTINUE
```

```
      DO 50 J=1,6
50    DIM(J)=1.0
      DO 60 J=1,MP
60    PDIM(J)=1.0
      FDIM=1.0

c     transform the measurement data to rectangular form

      C1=ASIN(1.0)/90.0
      DO 70 I=1,NCKT
      DO 70 K=1,4
      K1=2*K-1
      K2=K1+1
      DO 70 J=1,NFR
      T1=DAT(K1,J,I)
      T2=DAT(K2,J,I)*C1
      DTM(K1,J,I)=T1*COS(T2)
      DTM(K2,J,I)=T1*SIN(T2)
70    CONTINUE
      DO 100 I=1,NCKT
      DO 100 K=1,8
      SNR(K,I)=1.0
      IF(NFR.EQ.1)GOTO 100
      SMAX=-2000.0
      SMIN=2000.0
      DO 80 J=1,NFR
      IF(DTM(K,J,I).GT.SMAX)SMAX=DTM(K,J,I)
      IF(DTM(K,J,I).LT.SMIN)SMIN=DTM(K,J,I)
80    CONTINUE
      SNR(K,I)=1.0/(SMAX-SMIN)
100   CONTINUE

c     count the error functions

      MM=0
      DO 120 I=1,NCKT
      DO 120 J=1,NFR
      M1=0
      DO 110 K=1,8
      IF(WGT(K,J,I).GT.1.E-5)M1=M1+1
110   CONTINUE
      MM=MM+M1
120   KFR(J,I)=M1
      IF(MM.GT.250)THEN
              WRITE(*,*)'     Too many functions'
              GOTO 220
      ENDIF
      CALL BRIEF(OLDFIL,CKTFIL,NCKT)
180   RETURN
190   CALL FERMSG(2,FILNAM,ILINE,*210)
200   CALL FERMSG(3,FILNAM,ILINE,*210)
210   CLOSE(1)
220   IERROR=1
230   NCKT=0
```

```
GOTO 180
END
```

```
      SUBROUTINE RDSCF(IERROR)
      PARAMETER (MP=13,MX=40,MF=20)

c     put the contents of a the circuit file in char array TT(80,100)
c     locate various blocks in the file
c     block i  begins at the LOCBLK(1,i)th line and ends at
c     the LOCBLK(2,i)th line

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,T*80,A,B,BLKNAME(6)*5,WRD*80
      CHARACTER KYWRD*5
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INXMDA(3)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8)
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     1      TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      SAVE BLKNAME
      DATA BLKNAME/'BLK  ','FREQ ','OPT  ','DATA ','OUT  ','     '/
      B=CHAR(9)
      DO 25 I=1,100
      READ(1,'(A)',ERR=30,END=30)T
      DO 10 J=1,80
10    TT(J,I)=' '
      K=1
      DO 20 J=1,80
      IF(T(J:J).EQ.B)THEN

c             a tab

              K=(K/9+1)*9
      ELSE
              IF(K.GT.80)GOTO 25
              TT(K,I)=T(J:J)
              K=K+1
      ENDIF
20    CONTINUE
25    CONTINUE
30    LINET=I-1
      DO 40 I=1,10
40    LOCBLK(1,I)=0
      IPRECED=0
      DO 80 I=1,LINET
      J0=1
      KYWRD='     '
      CALL GETWRD(TT(1,I),J0,J2,WRD,LENG)
      IF(LENG.EQ.0)GOTO 80
      IF(IPRECED.EQ.0)THEN
              IF(LENG.GT.4)GOTO 60
              KYWRD(1:LENG)=WRD(1:LENG)
              DO 50 J=1,5
              IF(KYWRD.EQ.BLKNAME(J))GOTO 70
50            CONTINUE
60            CALL ERMSG(27,I,1,*100)
70            IPRECED=J
              LOCBLK(1,IPRECED)=I
      ELSEIF(LENG.EQ.3.AND.WRD(1:3).EQ.'END')THEN
```

```
                LOCBLK(2,IPRECED)=I-1
                IPRECED=0
        ENDIF
80      CONTINUE
        IF(IPRECED.NE.0)CALL ERMSG(28,LINET,1,*100)
        IERROR=0
        RETURN
100     IERROR=1
        RETURN
        END
```

```
c***********************************************************************

c       file name: FETPLT1.FOR

c       for program FETSCG1 (PC/AT version).
c       this group of files are responsible for model simulation

c***********************************************************************

$NOFLOATCALLS

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c       for program FETSCG.
c       subroutines FETPLT and HPPLT are responsible for plotting
c       FETPLT generates screen graphics
c       HPPLT controls a HP plotter to generate a hardcopy
c       they are valid for both TI or IBM/AT
c       the appropriate graphics library (TIGRF or IBMGRF) must be linked

c       the ISCREEN parameter must be set to distinguish TI or IBM/AT:
c       set ISCREEN=1 for TI and ISCREEN=2 for IBM/AT

c       Also, the device for the plotter is system dependent.  In this
c       particular code, LPT3 is used for the TI, and COM1 for the AT.

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc



        SUBROUTINE HPPLT(ICKT,MODNAM,IMO,MESNAM,IME,NF,IFPIX,IMODPIX,
     +IMESPIX,IXBPIX,IYBPIX)
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        PARAMETER (ISCREEN=2,MF=20)
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c HP plotting subroutine for FET modeling program
c
c  eight boxes are to be displayed on the plotter,
c  the top    4 boxes are numbered: 1 3 5 7  from left to right
c  the bottom 4 boxes are numbered: 2 4 6 8  from left to right
c
c
c arguments needed:     ICKT    circuit number (multiple cct modeling)
c                       MODNAM  model file name
c                       IMO     number of character in MODNAM
c                       MESNAM  measurement file name
c                       IME     number of characters in MESNAM
c                       NF      number of frequencies
c                       IFPIX()         pixels for x axis plot
c                       IMODPIX(,,)     pixels for y axis plot (model)
c                       IMESPIX(,,)     pixels for y axis plot (measurement)
c                       IXBPIX  pixel range for x axis of box
c                       IYBPIX  pixel range for y axis of box
```

```
      IMPLICIT REAL*4 (A-H,O-Z)
      IMPLICIT INTEGER*2 (I-N)
      DIMENSION IFPIX(MF),IMODPIX(MF,8,3),IMESPIX(MF,8,3),IBBB(4,8)
      DIMENSION XCEN(4),YCEN(2)
      CHARACTER BLAB(8)*4,MODNAM(IMO)*1,MESNAM(IME)*1
      CHARACTER BLNK*89,STRG*89
      DATA BLAB/'MS11','AS11','MS21','AS21','MS12','AS12','MS22','AS22'/

c glossary:
c
c XPAGE         the size of a page in the x direction (cm)
c YPAGE         the size of a page in the y direction (cm)
c XMARG         the x margin desired (cm)
c YMARG         the y margin desired (cm)
c XB            the x width  of each of the 8 boxes (cm)
c YB            the y height of each of the 8 boxes (cm)
c XCEN()        page location along x axis of the center of boxes (cm)
c YCEN()        page location along y axis of the center of boxes (cm)
c H             size of the character in % wrt XB and YB
c S             the x distance between boxes, in % wrt XB
c IBBB(i,j)     the x and y coord. of the corners of each box (plotter units)
c               i = 1 to 4  denoting xmin, xmax, ymin, ymax of the box
c               j = 1 to 8  denoting the box number

c set defaults

      XMARG=3.5
      YMARG=3.5
      H=0.035

c prompt user as to wether defaults are acceptable

      BLNK(1:8)='@L01@C01'
      DO 5 I=1,80
5     BLNK(I+8:I+8)=' '
      BLNK(89:89)='_'
      CALL DISP(BLNK)
      STRG(1:38)='@L01@C01@7Plot defaults: XMARG=3.5cm, '
      STRG(39:76)='YMARG=3.5cm, char=3.5% of box. (y/n)?_'
      CALL DISP(STRG)
      IKY=KYCODE(IKY)
      IDEF=1
      IF(ISCREEN.EQ.1) THEN
              IF(IKY.EQ.78.OR.IKY.EQ.110) IDEF=0
      ENDIF
      IF(ISCREEN.EQ.2) THEN
              IF(IKY.EQ.12622.OR.IKY.EQ.12654) IDEF=0
      ENDIF

c if defaults unacceptable, get desired values

      IF(IDEF.EQ.0) THEN
              CALL DISP(BLNK)
              STRG(1:37)='@L01@C01@7 Enter: XMARG cm, YMARG cm,'
              STRG(38:54)=' char (% box) : _'
```

```
                CALL DISP(STRG)
7               READ(*,*) XMARG,YMARG,H
                H=H/100.0
                STRG=BLNK
                STRG(1:4)='@L02'
                CALL DISP(STRG)
                Z=0.0
                IF(XMARG.LT.Z.OR.YMARG.LT.Z.OR.H.LT.Z) THEN
                        CALL BELL
                        GOTO 7
                ENDIF
        ENDIF
        CALL DISP(BLNK)
        STRG(1:49)='@L01@C01@7 Prepare plotter pens: P1 box, P2 model'
        STRG(50:87)=', P3 measurement.   <CR> when ready. _'
        CALL DISP(STRG)
        IKY=KYCODE(IKY)
        CALL DISP(BLNK)
        CALL DISP('@L01@C30@7@B ... HP plotting ... @b _')

c the rest is automatically adjusted

        S=2.0*H
        XPAGE=28.0
        YPAGE=21.6
        XB=(XPAGE-2.0*XMARG)/(4.0+3.0*S)
        YB=(YPAGE-2.0*YMARG)/(2.0+7.5*H)
        XLCLIP=1.0
        XMARG=XMARG-XLCLIP
        YBCLIP=0.35
        YMARG=YMARG-YBCLIP
        YCEN(1)=YMARG+0.5*YB
        YCEN(2)=YMARG+(1.5+3.0*H)*YB
        XCEN(1)=XMARG+0.5*XB
        XCEN(2)=XCEN(1)+(1.0+S)*XB
        XCEN(3)=XCEN(2)+(1.0+S)*XB
        XCEN(4)=XCEN(3)+(1.0+S)*XB
        DO 10 IC=1,4
        DO 10 IB=1,8
                IX=(IB+1)/2
                IY=2
                IF(MOD(IB,2).EQ.0) IY=1
                PM=-1.0
                IF(MOD(IC,2).EQ.0) PM=+1.0

c the conversion between plotter units and cm = 400.0 pu/cm

                IF(IC.LE.2) IBBB(IC,IB)=(XCEN(IX)+PM*0.5*XB)*400.0
                IF(IC.GE.3) IBBB(IC,IB)=(YCEN(IY)+PM*0.5*YB)*400.0
10      CONTINUE

c obtain the character height in plotter units

        HPU=H*YB*400.0
```

```
c start plotting:
c these devices are system dependent

        IF(ISCREEN.EQ.1) OPEN(2,FILE='LPT3',STATUS='OLD')
        IF(ISCREEN.EQ.2) OPEN(2,FILE='COM1',STATUS='OLD')

c initialize plotter and set character size: width & height (cm)

        WRITE(2,300) H*XB, H*YB
        DO 20 IB=1,8

c draw the box labels

                IYPU=IBBB(4,IB)+0.5*HPU
                IX=(IB+1)/2
                IXPU=XCEN(IX)*400.0
                WRITE(2,310) IXPU,IYPU,BLAB(IB),CHAR(3)

c then the box

                WRITE(2,320) IBBB(1,IB),IBBB(3,IB),IBBB(2,IB),IBBB(4,IB)
20      CONTINUE

c draw Model title

        IYPU=IBBB(4,1)+3.5*HPU
        IXPU=IBBB(1,1)
        WRITE(2,330) IXPU,IYPU,MODNAM,CHAR(3)

c then the model curves

        DO 30 IB=1,8

c       first extablish P1,P2; the corners of the box

                WRITE(2,350) IBBB(1,IB),IBBB(3,IB),IBBB(2,IB),IBBB(4,IB)

c       then scale it to the range of data to be plotted

                WRITE(2,360) 0,IXBPIX,0,IYBPIX

c       the plot the data

                DO 30 IF=1,NF
                        WRITE(2,370) IFPIX(IF), IMODPIX(IF,IB,ICKT)
30      CONTINUE

c draw Measurement title

        IXPU=IBBB(1,5)
        WRITE(2,340) IXPU,IYPU,MESNAM,CHAR(3)

c then the measurement curves

        DO 40 IB=1,8
```

```
                WRITE(2,350) IBBB(1,IB),IBBB(3,IB),IBBB(2,IB),IBBB(4,IB)
                WRITE(2,360) 0,IXBPIX,0,IYBPIX
                DO 40 IF=1,NF
                        WRITE(2,370) IFPIX(IF), IMESPIX(IF,IB,ICKT)
40      CONTINUE
        WRITE(2,380)
        CLOSE(2)
        CALL DISP(BLNK)
        RETURN

300     FORMAT(' IN ;SI ',2F10.3,' ;PU ;SP1 ;')
310     FORMAT(' PA ',2I10,' ;CP -2. 0. ;LB',A4,A1)
320     FORMAT(' PU ;PA ',2I10,' ;PD ;EA ',2I10,' ;PU ;')
330     FORMAT(' PU; SC; PA ',2I10,' ;SP2 ;LBMODEL: ',90A)
340     FORMAT(' PU; SC; PA ',2I10,' ;SP3 ;LBMEASUREMENT: ',90A)
350     FORMAT(' PU ;IP ',4I10,' ;')
360     FORMAT(' SC ',4I10,' ;')
370     FORMAT(' PA ',2I10,' ;PD ;')
380     FORMAT(' SP0 ;IN ;')
        END
```

```
      SUBROUTINE FETPLT(FM,JOB)
      PARAMETER (ISCREEN=2,MP=13,MX=40,MF=20)

c     job=1 coming from main program for simple plotting
c     job=2 ........................ preparing for optimization
c     job=3 coming from fdf during optimization
c     hp plotter interface should only deal with job=1

      CHARACTER TT(80,100),VNAME(MX)*8,CKTNAME(6)*8,CKTFIL(6)*12
      CHARACTER MSG(40)*30,PROC(3)*8,STRG*90
      DIMENSION LOCBLK(2,10),VAL(3,MX),LOCV(3,MX),INDX(MX),PA(MP,3)
      DIMENSION NODE(9),LOCX(2,MX),INVAR(MX),DIM(8),X(MX),BND(2,MX)
      DIMENSION FREQ(MF),WGT(8,MF,3),FM(8,MF,3),INXMDA(3),INPX(MP,3)
      INTEGER*2 IKY,IX(MF),IYO(MF,8,3),NPLT,IY(MF,8,3),IBB(4,9,2)
      INTEGER*2 KYCODE,IHIGH,MODE1
      COMMON/TS1/LOCBLK,LINET,NVAR,LOCV,NODE,INVAR,LOCX,VAL,DIM,
     +           TT,VNAME,CKTNAME,CKTFIL,PROC,MSG
      COMMON/TS2/NX,NCKT,NFR,INXMDA,INPX,INDX,PA,X,BND,FREQ,WGT,ETERM
      COMMON/BLK1/PDIM(MP),DAT(8,MF,3),INPP(MP),PAO(MP,3)
      COMMON/BLKPLT/IX,IY,YMIN(8,3),YR(8,3),JOUT
      SAVE IBB

c     IBB(I,J,K) stores pixel coordinates for screen display of boxes
c     I = 1 to 4   denoting xmin,xmax,ymin,ymax
c     J = 1 to 8   denoting the 8 boxes to be displayed
c     J = 9        to denote the size of the entire screen
c     K = 1 / 2    flag denoting  TI / AT  type of screen

      DATA IBB/15,165,160,280,15,165,15,135,195,345,160,280,
     +         195,345,15,135,375,525,160,280,375,525,15,135,
     +         555,705,160,280,555,705,15,135,0,719,0,299,
     +         5,155,186,326,5,155,17,157,165,315,186,326,
     +         165,315,17,157,325,475,186,326,325,475,17,157,
     +         485,635,186,326,485,635,17,157,0,639,0,349/

c x pixels per box

      IXBPIX=IBB(2,1,ISCREEN)-IBB(1,1,ISCREEN)
      XBPIX=FLOAT(IXBPIX)

c y pixels per box

      IYBPIX=IBB(4,1,ISCREEN)-IBB(3,1,ISCREEN)
      YBPIX=FLOAT(IYBPIX)

      IF (NCKT.LE.0) RETURN

      NPLT=NFR
      GOTO(10,10,100),JOB
10    XMIN=FREQ(1)
      XRATIO=XBPIX/(FREQ(NFR)-XMIN)
      DO 20 J=1,NFR
20          IX(J)=INT((FREQ(J)-XMIN)*XRATIO)
      DO 40 I=1,NCKT
      DO 40 K=1,8
```

```
                FMX=-200.0
                FMI=200.0
                DO 30 J=1,NFR
                        IF(DAT(K,J,I).GT.FMX)FMX=DAT(K,J,I)
                        IF(FM(K,J,I).GT.FMX)FMX=FM(K,J,I)
                        IF(DAT(K,J,I).LT.FMI)FMI=DAT(K,J,I)
                        IF(FM(K,J,I).LT.FMI)FMI=FM(K,J,I)
30              CONTINUE
                FD=FMX-FMI
                YMI=FMI-0.1*FD
                YRATIO=(YBPIX/1.2)/FD
                YMIN(K,I)=YMI
                YR(K,I)=YRATIO
        DO 40 J=1,NFR
                IYO(J,K,I)=INT((DAT(K,J,I)-YMI)*YRATIO)
                IY(J,K,I)=INT((FM(K,J,I)-YMI)*YRATIO)
40      CONTINUE
        IF(JOB.EQ.2)THEN
                CALL CLEAR
        ELSE
                CALL SAVSCR
        ENDIF
        CALL MODE(MODE1,16)
        IF(JOB.EQ.2)THEN
                I=JOUT
        ELSE
                I=1
        ENDIF
        CALL PTCOLR(7)
        IF(JOB.EQ.2)CALL SETXOR
50      STRG(1:38)= '@C09@L13@7MS11@S17MS21@S16MS12@S17MS22'
        STRG(39:75)='@C09@L25AS11@S17AS21@S16AS12@S17AS22_'
        CALL DISP(STRG)
        STRG(1:18)='@C10@L01@I@6Model:'
        STRG(19:26)=CKTNAME(I)
        IF (ISCREEN.EQ.1) STRG(27:37)='    @2File:'
        IF (ISCREEN.EQ.2) STRG(27:37)='    @4File:'
        STRG(38:49)=CKTFIL(INXMDA(I))
        IF(JOB.EQ.2)THEN
                STRG(50:75)='@7@i   <ESC> to interrupt_'
                STRG(1:4)='@C24'
        ELSE
                STRG(50:80)='@7@i@S10<RETURN>  or  P=plot  _'
        ENDIF
        CALL DISP(STRG)
        DO 60 K=1,8
                CALL BOX(IBB(1,K,ISCREEN),'L')
                CALL WINDOW(IBB(1,K,ISCREEN))
                IF (ISCREEN.EQ.1) CALL PTCOLR(2)
                IF (ISCREEN.EQ.2) CALL PTCOLR(12)
                CALL LINE(NPLT,IX,IYO(1,K,I))
                IF (ISCREEN.EQ.1) CALL PTCOLR(6)
                IF (ISCREEN.EQ.2) CALL PTCOLR(14)
                CALL LINE(NPLT,IX,IY(1,K,I))
                CALL WINDOW(IBB(1,9,ISCREEN))
```

```
                CALL PTCOLR(7)
60      CONTINUE
        IF(JOB.EQ.2)GOTO 90

c wait for the user to respond

70      IKY=KYCODE(IKY)
        IF (ISCREEN.EQ.2) IHIGH=IKY/256
        IF (ISCREEN.EQ.2) IKY=IKY-IHIGH*256
        IF(IKY.NE.13.AND.IKY.NE.80.AND.IKY.NE.112) GOTO 70
        IF(IKY.EQ.13) GOTO 75

c HP plotter routine:

        IC1=8
        IC2=12
        CALL HPPLT(I,CKTNAME(I),IC1,CKTFIL(INXMDA(I)),IC2,NFR,IX,
     +             IY,IY0,IXBPIX,IYBPIX)

75      CALL CLRGRF
        IF(I.EQ.NCKT)GOTO 80
        I=I+1
        GOTO 50

c all circuits done, ready for return to the main program

80      CALL MODE(MODE1,3)
        CALL RSTSCR
        RETURN

c here when JOB=2

90      IF (ISCREEN.EQ.1) CALL PTCOLR(6)
        IF (ISCREEN.EQ.2) CALL PTCOLR(14)
        RETURN

c here when JOB=3 (during optimization)

100     NPLT=NFR
        DO 120 K=1,8
                CALL WINDOW(IBB(1,K,ISCREEN))
                CALL LINE(NPLT,IX,IY(1,K,JOUT))
                YMI=YMIN(K,JOUT)
                YRATIO=YR(K,JOUT)
                DO 110 J=1,NFR
                        ITEMP=INT((FM(K,J,JOUT)-YMI)*YRATIO)
                        IF (ITEMP.LT.0)       ITEMP=0
                        IF (ITEMP.GT.IYBPIX) ITEMP=IYBPIX
                        IY(J,K,JOUT)=ITEMP
110             CONTINUE
                CALL LINE(NPLT,IX,IY(1,K,JOUT))
                CALL WINDOW(IBB(1,9,ISCREEN))
120     CONTINUE

        RETURN
```

```
END
```

```
c***********************************************************************

c       file name: L1ABC.FOR

c       for programs FETSCG and FETSCG1 (PC/AT version).
c       this group of subroutines constitutes the L1 optimization
c       package

c***********************************************************************

$NOFLOATCALLS


      SUBROUTINE ADDCOL(N,KCOL,KNEW,RIGHT,W,ACCUM,LRIGHT,EPS)
      PARAMETER (NX=40,MX=250,LX=40,NX2=NX+NX)
      COMMON/L13/XX(NX2),X1(NX),R(NX,NX),B(NX,NX),W0(NX),W1(NX),W2(NX),
     1    ASET(NX),U(NX,NX),KSET(NX),KSET0(NX),KSTATC(LX),KSTATF(MX)
      DIMENSION RIGHT(N),W(N)
      LOGICAL ACCUM,LRIGHT
      K1=KCOL+1
      K2=KCOL+KNEW
      DO 200 K=K1,K2
      S=0.0
      T=0.0
      IF(K.EQ.1) GOTO 20
      KK=K-1
      DO 10 I=1,KK
   10 T=T+R(I,K)**2
   20 DO 30 I=K,N
   30 S=S+R(I,K)**2
      T=T+S
      T=SQRT(T)
      S=SQRT(S)
      IF(T.EQ.0.0) RETURN
      IF(S.LT.T*EPS) RETURN
      TT=R(K,K)
      T=ABS(TT)
      ALFA=SQRT(S*(S+T))
      BETA=-SIGN(S,TT)
      R(K,K)=BETA
      W(K)=(TT-BETA)/ALFA
      IF(K.EQ.N) GOTO 80
      KK=K+1
      DO 40 I=KK,N
   40 W(I)=R(I,K)/ALFA
      IF(K.EQ.K2) GOTO 80
      DO 70 J=KK,K2
      T=0.0
      DO 50 I=K,N
   50 T=T+W(I)*R(I,J)
      DO 60 I=K,N
   60 R(I,J)=R(I,J)-T*W(I)
   70 CONTINUE
   80 IF(.NOT.LRIGHT) GOTO 110
```

```
      T=0.0
      DO 90 I=K,N
   90 T=T+W(I)*RIGHT(I)
      DO 100 I=K,N
  100 RIGHT(I)=RIGHT(I)-T*W(I)
  110 IF(ACCUM) GOTO 130
      DO 120 I=K,N
  120 U(I,K)=W(I)
      GOTO 200
  130 DO 160 I=1,N
      T=0.0
      DO 140 J=K,N
  140 T=T+U(I,J)*W(J)
      DO 150 J=K,N
  150 U(I,J)=U(I,J)-T*W(J)
  160 CONTINUE
  200 KCOL=KCOL+1
      RETURN
      END
```

```
      SUBROUTINE BFGS(N,SEPS)
      PARAMETER (NX=40,MX=250,LX=40,NX2=NX+NX)
      COMMON/L13/XX(NX2),X1(NX),R(NX,NX),B(NX,NX),W(NX),W1(NX),Y(NX),
     1    ASET(NX),U(NX,NX),KSET(NX),KSETO(NX),KSTATC(LX),KSTATF(MX)
      EPS=(N+10)*SEPS
      DO 20 I=1,N
      T=0.0
      DO 10 J=1,N
   10 T=T+B(I,J)*XX(J)
   20 W(I)=T
      YXX=0.0
      WXX=0.0
      DO 30 I=1,N
      YXX=YXX+Y(I)*XX(I)
   30 WXX=WXX+W(I)*XX(I)
      IF(WXX.LT.1.E-12.OR.YXX.LT.EPS*WXX) RETURN
      DO 40 I=1,N
   40 B(I,I)=B(I,I)+Y(I)**2/YXX-W(I)**2/WXX
      IF(N.EQ.1) RETURN
      DO 50 I=2,N
      I1=I-1
      DO 50 J=1,I1
      B(I,J)=B(I,J)+Y(I)*Y(J)/YXX-W(I)*W(J)/WXX
   50 B(J,I)=B(I,J)
      RETURN
      END
```

```
      SUBROUTINE DELCOL(K,N,KCOL)
      PARAMETER (NX=40,MX=250,LX=40,NX2=NX+NX)
      COMMON/L13/X00(NX2),X1(NX),R(NX,NX),B(NX,NX),W(NX),H(NX),W2(NX),
     1    ASET(NX),U(NX,NX),KSET(NX),KSET0(NX),KSTATC(LX),KSTATF(MX)
      COMMON/L14/RIGHT(NX),FUP(MX),DLDF(MX),CUP(LX),DLDC(LX),
     1       STEPF(MX),KSTATL(NX),DSTA2(NX2,NX2)

      KCOL=KCOL-1
      IF(K.GT.KCOL) RETURN
      DO 10 J=K,KCOL
      J1=J+1
      DO 10 I=1,J1
   10 R(I,J)=R(I,J1)
      DO 100 KK=K,KCOL
      K1=KK+1
      X=R(KK,KK)
      Y=R(K1,KK)
      A=SQRT(X*X+Y*Y)
      C=X/A
      S=Y/A
      R(KK,KK)=C*X+S*Y
      IF(KK.EQ.KCOL) GOTO 60
      DO 50 J=K1,KCOL
      X=R(KK,J)
      Y=R(K1,J)
      R(KK,J)=C*X+S*Y
   50 R(K1,J)=C*Y-S*X
   60 X=RIGHT(KK)
      Y=RIGHT(K1)
      RIGHT(KK)=C*X+S*Y
      RIGHT(K1)=C*Y-S*X
   70 DO 80 I=1,N
      X=U(I,KK)
      Y=U(I,K1)
      U(I,KK)=C*X+S*Y
   80 U(I,K1)=C*Y-S*X
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE HACCUM(N,KCOL)
      PARAMETER (NX=40,MX=250,LX=40,NX2=NX+NX)
      COMMON/L13/X(NX2),X1(NX),R(NX,NX),B(NX,NX),W(NX),H(NX),W2(NX),
     1    ASET(NX),U(NX,NX),KSET(NX),KSETO(NX),KSTATC(LX),KSTATF(MX)
      K1=KCOL+1
      DO 10 I=KCOL,N
   10 W(I)=U(I,KCOL)
      DO 20 I=KCOL,N
   20 U(I,I)=1.0-W(I)**2
      IF(KCOL.EQ.N) GOTO 40
      DO 30 I=K1,N
      I1=I-1
      T=W(I)
      DO 30 J=KCOL,I1
      S=-T*W(J)
      U(I,J)=S
   30 U(J,I)=S
   40 IF(KCOL.EQ.1) RETURN
      DO 100 KK=2,KCOL
      K=K1-KK
      DO 50 I=K,N
   50 W(I)=U(I,K)
      T=W(K)
      KP1=K+1
      U(K,K)=1.0-T**2
      DO 60 I=KP1,N
   60 U(I,K)=-T*W(I)
      DO 90 L=KP1,N
      S=0.0
      DO 70 I=KP1,N
   70 S=S+W(I)*U(I,L)
      U(K,L)=-T*S
      DO 80 I=KP1,N
   80 U(I,L)=U(I,L)-S*W(I)
   90 CONTINUE
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE L1A(N,M,X,MAXF)
      DIMENSION X(N)
      DX=0.0
      DO 10 I=1,N
10    DX=DX+X(I)*X(I)
      DX=0.05*SQRT(DX)
      IF(DX.LT.0.005)DX=0.005
      EPS=1.E-6
      MAXF=200
      CALL L1B(N,M,X,DX,EPS,MAXF)
      RETURN
      END
```

```
SUBROUTINE L1B(N,M,X,DX,EPS,MAXF)
DIMENSION X(N)
L=0
LEQ=0
CALL L1C(N,M,L,LEQ,X,DX,EPS,MAXF)
RETURN
END
```

```
      SUBROUTINE L1C(N,M,L,LEQ,X,DX,EPS,MAXF)
      PARAMETER (NX=40,MX=250,LX=40,NX2=NX+NX)
      DIMENSION X(N)
      COMMON/L11/MARK,F1(MX),DF1(MX,NX),F(MX),DF(MX,NX)
      COMMON/L12/C(LX),DC(LX,NX),CLOC(LX)
      COMMON/L13/XX(NX2),X1(NX),R(NX,NX),B(NX,NX),W(NX),W1(NX),W2(NX),
     1    ASET(NX),U(NX,NX),KSET(NX),KSETO(NX),KSTATC(LX),KSTATF(MX)

      SEPS=1.E-12
      IF(N.GT.NX.OR.M.GT.MX.OR.L.GT.LX)THEN
        WRITE(*,*)' Fatal error: dimensional limits exceeded'
        RETURN
      ENDIF
      IF(DX.LE.SEPS.OR.EPS.LE.SEPS)THEN
        WRITE(*,*)' Fatal error: DX or EPS too small '
        RETURN
      ENDIF

      LI=L-LEQ
      LNN=L+N+N
      KEQSET=0
      NSTEP=0
      NACTO=0
      MARK=1
      NCALL=1
      DXO=DX
      FL1REF=1.E16
      XN=0.0
      DO 20 I=1,N
      XN=XN+X(I)**2
      DO 10 J=1,N
10    B(I,J)=0.0
20    B(I,I)=1.0
      XN=SQRT(XN)
      IF(L.GT.0) THEN
              DO 30 K=1,L
              T=0.0
              DO 25 I=1,N
25            T=T+DC(K,I)*X(I)
30            CLOC(K)=T+C(K)
      ENDIF
      CALL FDF(N,M,X)
        IF(MARK.EQ.0)GOTO 300
      FL10=0.0
      DO 40 J=1,M
      F(J)=F1(J)
      DO 35 I=1,N
35    DF(J,I)=DF1(J,I)
40    FL10=FL10+ABS(F(J))

c    iterative loop

50    NACT=NACTO
      IF(NACT.GT.0) THEN
              DO 60 I=1,NACT
```

```
60              KSET(I)=KSET0(I)
        ENDIF

        CALL L1LP(M,N,L,LEQ,DX,XXN,NACT,FL1,IFALL)
        IF(FL1.GE.FL10) GOTO 300

        DO 70 I=1,N
70      X1(I)=X(I)+XX(I)
        CALL FDF(N,M,X1)
        IF(MARK.NE.1)GOTO 300
        NCALL=NCALL+1
        FL11=0.0
        DO 80 I=1,M
80      FL11=FL11+ABS(F1(I))

        IF((FL10-FL11).GT.0.75*(FL10-FL1)) DX=2.0*XXN
        IF((FL10-FL11).LT.0.25*(FL10-FL1)) DX=0.25*XXN

        K=0
        IF(NACT.GT.0)THEN
                DO 110 I=1,NACT
                II=KSET(I)
                IF(II.GT.LNN)THEN
                        K=K+1
                        KSET(K)=KSET(I)
                        II=II-LNN
                        DO 90 J=1,N
90                      R(J,K)=DF(II,J)
                ELSEIF(II.LE.L)THEN
                        K=K+1
                        KSET(K)=KSET(I)
                        DO 100 J=1,N
100                     R(J,K)=DC(II,J)
                ENDIF
110             CONTINUE
        ENDIF

        DO 120 I=1,N
        W(I)=0.D0
120     W1(I)=0.D0
        DO 150 I=1,M
        IF(KSTATF(I).LT.0) THEN
                DO 130 J=1,N
                W(J)=W(J)-DF(I,J)
130             W1(J)=W1(J)-DF1(I,J)
        ELSEIF(KSTATF(I).GT.0) THEN
                DO 140 J=1,N
                W(J)=W(J)+DF(I,J)
140             W1(J)=W1(J)+DF1(I,J)
        ENDIF
150     CONTINUE
        DO 160 I=1,N
160     ASET(I)=-W(I)
        NACT=0
        IF(K.GT.0) THEN
```

```
                CALL ADDCOL(N,NACT,K,ASET,W2,.FALSE.,.TRUE.,SEPS)
                IF(NACT.LT.K) KEQSET=1
                CALL RSOLV(N,NACT,ASET,ASET)
                DO 180 I=1,NACT
                K=KSET(I)
                IF(K.GT.LNN) THEN
                        KK=K-LNN
                        T=ASET(I)
                        DO 170 J=1,N
                        W1(J)=W1(J)+T*DF1(KK,J)
170                     W(J)=W(J)+T*DF(KK,J)
                ENDIF
180             CONTINUE
        ENDIF

        DO 190 I=1,N
190     W2(I)=W1(I)-W(I)
        CALL BFGS(N,SEPS)

        IF((FL10-FL11).LE.0.01*(FL10-FL1)) GOTO 270

        IF(NACT0.NE.NACT) THEN
                KEQSET=1
        ELSE
                IF(NACT.GT.0) THEN
                        DO 200 I=1,NACT
                        K=KSET(I)
                        DO 210 J=1,NACT
                        IF(K.EQ.KSET0(J)) GOTO 200
210                     CONTINUE
                        KEQSET=1
                        GOTO 220
200                     CONTINUE
                ENDIF
                IF (FL11.GE.0.99*FL10) KEQSET=KEQSET+1
        ENDIF

220     NSTEP=NSTEP+1
        XN=0.0
        FL10=FL11
        NACT0=NACT
        DO 230 I=1,N
        X(I)=X1(I)
        XN=XN+X(I)**2
        DO 230 J=1,M
230     DF(J,I)=DF1(J,I)
        XN=SQRT(XN)
        DO 240 I=1,M
240     F(I)=F1(I)
        IF(L.GT.0) THEN
                DO 250 K=1,L
                T=0.0
                DO 245 I=1,N
245             T=T+DC(K,I)*X(I)
250             CLOC(K)=T+C(K)
```

```
      ENDIF
      IF(NACT.GT.0) THEN
              DO 260 I=1,NACT
260           KSETO(I)=KSET(I)
      ENDIF

270   IF(NCALL.GT.MAXF.OR.XXN.LE.EPS*XN.OR.XXN.LE.SEPS*XN) GOTO 300

      IF(NSTEP.LT.N.OR.FL10.GT.FL1REF.OR.KEQSET.LT.3) GOTO 50
      IF(NACT.GT.LEQ) THEN
              DO 280 I=1,NACT
              K=KSET(I)
              T=ASET(I)
              IF(K.GT.LEQ.AND.K.LE.L.AND.T.GE.0.0) GOTO 50
              IF(K.GT.LNN.AND.ABS(T).GE.1.0) GOTO 50
280           CONTINUE
      ENDIF

      NSHIFT=NSHIFT+1
      FL1REF=FL10*0.999999
      XXNMAX=AMAX1(DX0,10.0*DX)
      CALL L1STA2(N,M,L,LEQ,X,XXNMAX,NACT,EPS,MAXF,NCALL,
     1      XXN,NSTEP,IFALL)
      IF(IFALL.LT.3) GOTO 300
      FL10=0.0
      DO 290 I=1,M
290   FL10=FL10+ABS(F(I))
      DX=AMAX1(DX,XXN*0.5)
      KEQSET=1
      GOTO 50

300   MAXF=NCALL
      EPS=XXN
      RETURN
      END
```

```
      SUBROUTINE L1LP(M,N,L,LE,XNMAX,XN,NACT,FL1,IFALL)
      PARAMETER (NX=40,MX=250,LX=40,NX2=NX+NX)
      COMMON/L11/MARK,F1(MX),DF1(MX,NX),F(MX),DF(MX,NX)
      COMMON/L12/CO(LX),DC(LX,NX),C(LX)
      COMMON/L13/X(NX2),X1(NX),R(NX,NX),B(NX,NX),W(NX),H(NX),W2(NX),
     1    ASET(NX),U(NX,NX),KSET(NX),KSETO(NX),KSTATC(LX),KSTATF(MX)
      COMMON/L14/DL(NX),FUP(MX),DLDF(MX),CUP(LX),DLDC(LX),
     1       STEPF(MX),KSTATL(NX),DSTA2(NX2,NX2)
      LOGICAL ACCUM,LINDEP

      LE1=LE+1
      LI=L-LE
      LN=L+N
      LNN=LN+N
      XN=0.0
      SEPS=1.E-12
      EPS=N*SEPS
      ACCUM=.FALSE.
      LINDEP=.FALSE.
      NCYC=0
      IFALL=0
      FL1=1.E16
      DO 10 I=1,N
      KSTATL(I)=0
   10 X(I)=0.0
      DO 15 I=1,M
      KSTATF(I)=1
      IF(F(I).LT.0.0) KSTATF(I)=-1
   15 FUP(I)=F(I)

c     activate initial active set of constraints

      NACTIN=NACT
      NACT=0
      IF(L.EQ.0) GOTO 100
      DO 20 I=1,L
      KSTATC(I)=0
   20 CUP(I)=C(I)
      IF(LE.EQ.0) GOTO 50
      DO 40 I=1,LE
      DO 30 J=1,N
   30 R(J,I)=DC(I,J)
      KSET(I)=I
   40 KSTATC(I)=1
      CALL ADDCOL(N,NACT,LE,H,W,ACCUM,.FALSE.,EPS)
      IF(NACT.EQ.LE) GOTO 50
      IFALL=3
      RETURN
   50 IF(NACTIN.LT.LE1) GOTO 100
      DO 70 K=LE1,NACTIN
      KK=KSET(K)
      IF(KK.LT.LE1.OR.KK.GT.L) GOTO 70
      NACT1=NACT+1
      IF(NACT1.GT.N) GOTO 100
      DO 60 I=1,N
```

```
   60 R(I,NACT1)=DC(KK,I)
      CALL UTTRNS(N,NACT,ACCUM,R(1,NACT1))
      CALL ADDCOL(N,NACT,1,H,W,ACCUM,.FALSE.,EPS)
      IF(NACT.LT.NACT1) GOTO 70
      EPS=EPS+2.0*SEPS
      KSET(NACT1)=KK
      KSTATC(KK)=1
   70 CONTINUE

c     calculate the restricted gradient

  100 FL1=0.0
      DO 105 I=1,N
  105 H(I)=0.0
      DO 140 I=1,M
      FL1=FL1+ABS(FUP(I))
      K=KSTATF(I)
      IF(K.GT.-1) GOTO 120
      DO 110 J=1,N
  110 H(J)=H(J)-DF(I,J)
  120 IF(K.LT.1) GOTO 140
      DO 130 J=1,N
  130 H(J)=H(J)+DF(I,J)
  140 CONTINUE

c     transform the restricted gradient

      DO 150 I=1,N
  150 DL(I)=-H(I)
      CALL UTTRNS(N,NACT,ACCUM,DL)

c     check if the current point is a dead point

  200 T=0.0
      NACT1=NACT+1
      DLN2=0.0
      IF(NACT.EQ.N) GOTO 220
      DO 210 I=NACT1,N
  210 DLN2=DLN2+DL(I)**2
  220 IF(NACT1.EQ.1) GOTO 240
      DO 230 I=1,NACT
  230 T=T+DL(I)**2
  240 T=T+DLN2
      IF(T.EQ.0.0) GOTO 900
      IF(DLN2.GT.EPS*EPS*T) GOTO 395

c     dead point - calculate multipliers

      IF(NACT.EQ.LE) GOTO 900
      CALL RSOLV(N,NACT,DL,ASET)
      AMAX=-1.E16
      DO 250 K=LE1,NACT
      A=ASET(K)
      IF(KSET(K).GT.LNN) A=ABS(A)-1.0
      IF(A.LE.AMAX) GOTO 250
```

```
      KMAX=K
      AMAX=A
  250 CONTINUE
      IF(AMAX.LE.0.0) GOTO 900

c     delete a function from the active set

      KK=KSET(KMAX)
      IF(KK.LE.LNN) GOTO 310
      KK=KK-LNN
      A=ASET(KMAX)
      IF(A.LT.0.0) GOTO 280
      KSTATF(KK)=1
      DO 260 I=1,N
  260 H(I)=H(I)+DF(KK,I)
      DO 270 I=1,KMAX
  270 DL(I)=DL(I)-R(I,KMAX)
      GOTO 330
  280 KSTATF(KK)=-1
      DO 290 I=1,N
  290 H(I)=H(I)-DF(KK,I)
      DO 300 I=1,KMAX
  300 DL(I)=DL(I)+R(I,KMAX)
      GOTO 330

c     deactivate a constraint

  310 IF(KK.GT.L) GOTO 320
      KSTATC(KK)=0
      GOTO 330
  320 IF(KK.GT.LN) GOTO 325
      KSTATL(KK-L)=0
      GOTO 330
  325 KSTATL(KK-LN)=0

c     remove the corresponding column in r

  330 IF(ACCUM) GOTO 340
      ACCUM=.TRUE.
      CALL HACCUM(N,NACT)
  340 CALL DELCOL(KMAX,N,NACT)
      EPS=EPS+2.0*SEPS
      IF(KMAX.GT.NACT) GOTO 360
      DO 350 I=KMAX,NACT
  350 KSET(I)=KSET(I+1)

c     remove linear dependence labels

  360 IF(.NOT.LINDEP) GOTO 200
      LINDEP=.FALSE.
      DO 370 I=1,M
      IF(KSTATF(I).EQ.2) KSTATF(I)=1
  370 IF(KSTATF(I).EQ.-2) KSTATF(I)=-1
      DO 380 I=1,N
  380 IF(KSTATL(I).EQ.2) KSTATL(I)=0
```

```
      IF(LI.EQ.0) GOTO 200
      DO 390 I=LE1,L
  390 IF(KSTATC(I).EQ.2) KSTATC(I)=0
      GOTO 200

c     calculate the direction of search

  395 IF(NACT.EQ.0) GOTO 405
      DO 400 I=1,NACT
  400 DL(I)=0.0
  405 CALL UTRNS(N,NACT,ACCUM)
      DLH=0.0
      DO 410 I=1,N
  410 DLH=DLH+DL(I)*H(I)

c     project gradients on the direction of search

      DO 415 J=1,M
      T=0.0
      DO 412 I=1,N
412   T=T+DF(J,I)*DL(I)
415   DLDF(J)=T
      IF(L.GT.0) THEN
                DO 419 K=1,L
                T=0.0
                DO 418 I=1,N
418             T=T+DC(K,I)*DL(I)
419             DLDC(K)=T
      ENDIF

c     calculate the step lengths

      DO 420 I=1,M
      S=DLDF(I)
      K=KSTATF(I)
      STEPF(I)=1.E16
      IF(IABS(K).NE.1.OR.K*S.GE.-1.E-12) GOTO 420
      STEPF(I)=-FUP(I)/S
  420 CONTINUE
  425 SLP=1.E16
      SLN=1.E16
      DO 440 I=1,N
      IF(KSTATL(I).NE.0) GOTO 440
      S=DL(I)
      IF(S*S.LE.EPS*EPS*DLN2) GOTO 440
      T=(XNMAX-X(I))/S
      IF(S.LT.0.0.OR.T.GE.SLP) GOTO 430
      SLP=T
      KLP=I
  430 T=-(XNMAX+X(I))/S
      IF(S.GT.0.0.OR.T.GE.SLN) GOTO 440
      SLN=T
      KLN=I
  440 CONTINUE
      SC=1.E16
```

```
      IF(LI.EQ.0) GOTO 460
      DO 450 I=LE1,L
      S=DLDC(I)
      IF(KSTATC(I).NE.0.OR.S.GE.-1.E-12) GOTO 450
      T=-CUP(I)/S
      IF(T.GE.SC) GOTO 450
      KC=I
      SC=T
  450 CONTINUE
  460 STEPLC=AMIN1(SLP,SLN)
      STEPLC=AMIN1(STEPLC,SC)
      NACT1=NACT+1

c     do line search

  500 SF=1.E16
      DO 510 I=1,M
      S=STEPF(I)
      IF(S.GE.SF) GOTO 510
      KF=I
      SF=S
  510 CONTINUE
      STEP=AMIN1(STEPLC,SF)
      IF(SF.GT.STEP) GOTO 630
      T=DLH-2*KSTATF(KF)*DLDF(KF)
      IF(T.GE.0.0) GOTO 600
      DLH=T
      KSTATF(KF)=-KSTATF(KF)
      STEPF(KF)=1.E16
      GOTO 500

c     activate a function

  600 DO 610 I=1,N
  610 R(I,NACT1)=DF(KF,I)
      CALL UTTRNS(N,NACT,ACCUM,R(1,NACT1))
      CALL ADDCOL(N,NACT,1,DL,W,ACCUM,.FALSE.,EPS)
      IF(NACT.EQ.NACT1) GOTO 620
      KSTATF(KF)=2*KSTATF(KF)
      STEPF(KF)=1.E16
      LINDEP=.TRUE.
      GOTO 500
  620 KSET(NACT)=KF+LNN
      KSTATF(KF)=0
      GOTO 700

c     -  or a constraint

  630 IF(SC.GT.STEP) GOTO 660
      DO 640 I=1,N
  640 R(I,NACT1)=DC(KC,I)
      CALL UTTRNS(N,NACT,ACCUM,R(1,NACT1))
      CALL ADDCOL(N,NACT,1,DL,W,ACCUM,.FALSE.,EPS)
      IF(NACT.EQ.NACT1) GOTO 650
      KSTATC(KC)=2
```

```
      LINDEP=.TRUE.
      GOTO 425
  650 KSTATC(KC)=1
      KSET(NACT)=KC
      GOTO 700

c     - or a bound

  660 DO 670 I=1,N
  670 R(I,NACT1)=0.0
      IF(SLN.EQ.SLP) SLN=2.E16
      IF(STEP.EQ.SLN) R(KLN,NACT1)=1.0
      IF(STEP.EQ.SLP) R(KLP,NACT1)=-1.0
      CALL UTTRNS(N,NACT,ACCUM,R(1,NACT1))
      CALL ADDCOL(N,NACT,1,DL,W,ACCUM,.FALSE.,EPS)
      IF(NACT.EQ.NACT1) GOTO 680
      LINDEP=.TRUE.
      IF(STEP.EQ.SLN) KSTATL(KLN)=2
      IF(STEP.EQ.SLP) KSTATL(KLP)=2
      GOTO 425
  680 IF(STEP.EQ.SLN) GOTO 690
      KSET(NACT)=KLP+L
      KSTATL(KLP)=1
      GOTO 700
  690 KSET(NACT)=KLN+LN
      KSTATL(KLN)=-1
  700 EPS=EPS+2.0*SEPS

c     introduce the new point

      XN=0.0
      XXN=0.0
      NCYC=NCYC+1
      IF(NCYC.GT.2*N) GOTO 900
      DO 800 I=1,N
      S=STEP*DL(I)
      XXN=AMAX1(XXN,ABS(S))
      X(I)=X(I)+S
  800 XN=AMAX1(XN,ABS(X(I)))
      IF(XXN.GT.EPS*XN) NCYC=0
      DO 810 I=1,M
  810 FUP(I)=FUP(I)+STEP*DLDF(I)
      IF(L.EQ.0) GOTO 100
      DO 820 I=1,L
  820 CUP(I)=CUP(I)+STEP*DLDC(I)
      GOTO 100

c     return

  900 IF(NACT.LT.N) IFALL=2
      DO 910 I=1,NACT
  910 IF(KSET(I).GT.L.AND.KSET(I).LE.LNN) IFALL=1
      IF(IFALL.EQ.1) XN=XNMAX
      IF(NCYC.GT.2*N) IFALL=4
      RETURN
```

```
END
```

```
      SUBROUTINE L1STA2(N,M,L,LEQ,X,XXNMAX,NACT,EPS,MAXF,NCALL,
     1       XXN,NSTEP,IFALL)
      PARAMETER (NX=40,MX=250,LX=40,NX2=NX+NX)
      DIMENSION X(N)
      COMMON/L11/MARK,F1(MX),DF1(MX,NX),F(MX),DF(MX,NX)
      COMMON/L12/C(LX),DC(LX,NX),CLOC(LX)
      COMMON/L13/ZZ(NX2),X1(NX),R(NX,NX),B(NX,NX),W(NX),WO(NX),W1(NX),
     1    ASET(NX),U(NX,NX),KSET(N),KSETO(NX),KSTATC(LX),KSTATF(MX)
      COMMON/L14/ASET1(NX),WM(MX),DLDF(MX),WL(LX),DLDC(LX),
     1       STEPF(MX),KSTATL(NX),DZ(NX2,NX2)

      LI=L-LEQ
      LE1=LEQ+1
      LNN=L+N+N
      N1=N+1
      IFALL=0
      SEPS=1.E-12
      SSEPS=SQRT(SEPS)
      NZ=N+NACT
      NSTEP2=0
      XXN=0.0
      RESREF=1.E16

c     iterative loop starts here
c     set up the iteration matrix and the righthand side

   50 DO 70 I=1,N
      DO 60 J=1,N
   60 DZ(I,J)=B(I,J)
   70 ZZ(I)=0.0
      DO 100 I=1,M
      IF(KSTATF(I).GE.0) GOTO 80
      DO 75 J=1,N
   75 ZZ(J)=ZZ(J)+DF(I,J)
      GOTO 100
   80 IF(KSTATF(I).EQ.0) GOTO 100
      DO 90 J=1,N
   90 ZZ(J)=ZZ(J)-DF(I,J)
  100 CONTINUE
      IF(NACT.EQ.0) GOTO 200
      DO 190 J=1,NACT
      K=KSET(J)
      JN=J+N
      IF(K.GT.L) GOTO 140
      ZZ(JN)=-CLOC(K)
      DO 130 I=1,N
      DZ(I,JN)=DC(K,I)
  130 DZ(JN,I)=DZ(I,JN)
      GOTO 160
  140 KK=K-LNN
      ZZ(JN)=-F(KK)
      DO 150 I=1,N
      DZ(I,JN)=DF(KK,I)
  150 DZ(JN,I)=DZ(I,JN)
  160 DO 170 I=N1,NZ
```

```
  170 DZ(I,JN)=0.0
      T=ASET(J)
      DO 180 I=1,N
  180 ZZ(I)=ZZ(I)-T*DZ(JN,I)
  190 CONTINUE

c     calculate the quasi-newton step

  200 CALL LINSYS(NZ,K,SEPS)
      IF(K.EQ.NZ) GOTO 205
      IFALL=3
      RETURN

c     control step length

  205 XXN1=0.0
      ALFA=1.0
      DO 210 I=1,N
  210 XXN1=XXN1+ZZ(I)**2
      XXN1=SQRT(XXN1)
      IF(XXN1.GT.XXNMAX) ALFA=XXNMAX/XXN1
      IF(ALFA.LT.0.1) IFALL=4

c     will other constraints or functions become active?

      DO 220 J=1,M
      T=0.0
      DO 215 I=1,N
215   T=T+DF(J,I)*ZZ(I)
220   WM(J)=T
      IF(L.GT.0) THEN
                 DO 225 K=1,L
                 T=0.0
                 DO 222 I=1,N
222              T=T+DC(K,I)*ZZ(I)
225              WL(K)=T
      ENDIF
      STEP=1.E16
      IF(LI.EQ.0) GOTO 240
      DO 230 I=LE1,L
      IF(KSTATC(I).NE.0) GOTO 230
      T=WL(I)
      IF(T.GE.-1.E-12) GOTO 230
      T=-CLOC(I)/T
      IF(T.GT.STEP) GOTO 230
      STEP=T
  230 CONTINUE
  240 DO 250 I=1,M
      T=WM(I)
      K=KSTATF(I)
      IF(IABS(K).NE.1.OR.K*T.GE.-1.E-12) GOTO 250
      T=-2.5*F(I)/T
      IF(T.GE.STEP) GOTO 250
      STEP=T
  250 CONTINUE
```

```
      IF(STEP.GT.ALFA) GOTO 275
      IFALL=4
      ALFA=STEP

c     scale the step

  275 DO 280 I=1,NZ
  280 ZZ(I)=ALFA*ZZ(I)
      XXN1=ABS(ALFA)*XXN1

c     calculate function values in the new point

      XN1=0.0
      DO 290 I=1,N
      X1(I)=X(I)+ZZ(I)
  290 XN1=XN1+X1(I)**2
      XN1=SQRT(XN1)
      NCALL=NCALL+1
      CALL FDF(N,M,X1)
      IF(MARK.NE.1)GOTO 560
      IF(NACT.GT.0) THEN
              DO 295 I=1,NACT
  295         ASET1(I)=ASET(I)+ZZ(I+N)
      ENDIF
      IF(L.GT.0) THEN
                DO 305 K=1,L
                T=0.0
                DO 300 I=1,N
300             T=T+DC(K,I)*X1(I)
305             WL(I)=T+C(I)
      ENDIF

c     calculate residuals

      DO 310 I=1,N
      W(I)=0.0
  310 W1(I)=0.0
      DO 330 I=1,M
      IF(KSTATF(I).GE.0) GOTO 320
      DO 315 J=1,N
      W(J)=W(J)-DF(I,J)
  315 W1(J)=W1(J)-DF1(I,J)
      GOTO 330
  320 IF(KSTATF(I).EQ.0) GOTO 330
      DO 325 J=1,N
      W(J)=W(J)+DF(I,J)
  325 W1(J)=W1(J)+DF1(I,J)
  330 CONTINUE
      RES=0.0
      IF(NACT.EQ.0) GOTO 370
      DO 360 J=1,NACT
      K=KSET(J)
      T=ASET1(J)
      IF(K.GT.L) GOTO 340
      DO 335 I=1,N
```

```
      S=T*DC(K,I)
      W(I)=W(I)+S
  335 W1(I)=W1(I)+S
      RES=RES+WL(K)**2
      GOTO 360
  340 KK=K-LNN
      DO 350 I=1,N
      W(I)=W(I)+T*DF(KK,I)
  350 W1(I)=W1(I)+T*DF1(KK,I)
      RES=RES+F1(KK)**2
  360 CONTINUE
  370 DO 380 I=1,N
  380 RES=RES+W1(I)**2
      RES=SQRT(RES)

c     update the hessian approximation

      IF(XXN1.LT.100.0*SEPS*XN1) GOTO 395
      DO 390 I=1,N
  390 W1(I)=W1(I)-W(I)
      CALL BFGS(N,SEPS)

c     test if the residual has decreased

  395 IF(RES.LE.0.999*RESREF) GOTO 480
      IF(IFALL.GT.2) RETURN
      IF(XXN1.GT.SSEPS*(XXNMAX+XN1).OR.NSTEP2.LT.2) GOTO 400
      IFALL=2
      IF(XXN1.GT.EPS*XN1) RETURN
      IFALL=1
      IF(NACT.LT.N) RETURN
      IFALL=0
      RETURN
  400 IFALL=5
      RETURN
  480 NSTEP2=NSTEP2+1
      NSTEP=NSTEP+1
      DO 490 I=1,N
      X(I)=X1(I)
      DO 490 J=1,M
  490 DF(J,I)=DF1(J,I)
      XN=XN1
      XXN=XXN1
      FL1=0.0
      DO 500 I=1,M
      FL1=FL1+ABS(F1(I))
  500 F(I)=F1(I)
      RESREF=RES
      IF(NACT.EQ.0) GOTO 520
      DO 510 I=1,NACT
      K=KSET(I)
      S=ASET1(I)
      ASET(I)=S
      IF(K.GT.L) S=ABS(S)-1.0
  510 IF(K.GT.LEQ.AND.S.GT.0.0) IFALL=6
```

```
520 IF(L.EQ.0) GOTO 535
    DO 530 J=1,L
530 CLOC(J)=WL(J)
535 DO 540 I=1,M
    IF(KSTATF(I).GT.0.AND.F(I).LT.-RES) IFALL=7
540 IF(KSTATF(I).LT.0.AND.F(I).GT.RES) IFALL=7
    IF(IFALL.GT.2) RETURN
    IF(XXN.GT.EPS*XN) GOTO 550
    IF(NACT.LT.N) IFALL=1
    RETURN
550 IF(XXN.GT.SEPS*XN.AND.NCALL.LE.MAXF) GOTO 50
560 IFALL=2
    RETURN
    END
```

```
      SUBROUTINE LINSYS(N,NR,SEPS)
      PARAMETER (NX=40,MX=250,LX=40,NX2=NX+NX)
      COMMON/L13/B(NX2),X1(NX),R(NX,NX),B0(NX,NX),W(NX),W0(NX),W1(NX),
     1    ASET(NX),U(NX,NX),KSET(NX),KSET0(NX),KSTATC(LX),KSTATF(MX)
      COMMON/L14/ASET1(NX),WM(MX),DLDF(MX),WL(LX),DLDC(LX),
     1       STEPF(MX),KSTATL(NX),A(NX2,NX2)
      NR=0

      IF(N-1) 200,1,10
    1 IF(ABS(A(1,1)).LT.1.E-12) RETURN
      NR=1
      B(1)=B(1)/A(1,1)
      RETURN
   10 DO 30 I=1,N
      AM=ABS(A(I,1))
      DO 20 J=2,N
      S=ABS(A(I,J))
   20 IF(AM.LT.S) AM=S
      IF(AM.LT.1.E-12) AM=1.0
      B(I)=B(I)/AM
      DO 30 J=1,N
   30 A(I,J)=A(I,J)/AM
      N1=N-1
      DO 80 K=1,N1
      NR=K-1
      AM=ABS(A(K,K))
      I0=K
      K1=K+1
      DO 40 I=K1,N
      S=ABS(A(I,K))
      IF(S.LE.AM) GOTO 40
      AM=S
      I0=I
   40 CONTINUE
      IF(AM.LT.2*K*SEPS) RETURN
      IF(I0.EQ.K) GOTO 60
      DO 50 J=K,N
      S=A(K,J)
      A(K,J)=A(I0,J)
   50 A(I0,J)=S
      S=B(K)
      B(K)=B(I0)
      B(I0)=S
   60 AM=A(K,K)
      DO 80 I=K1,N
      S=A(I,K)/AM
      DO 70 J=K1,N
   70 A(I,J)=A(I,J)-S*A(K,J)
   80 B(I)=B(I)-S*B(K)
      NR=N1
      IF(ABS(A(N,N)).LT.2*N*SEPS) RETURN
      NR=N
      B(N)=B(N)/A(N,N)
      K=N
      DO 100 I=2,N
```

```
      K1=K
      K=K-1
      S=B(K)
      DO 90 J=K1,N
   90 S=S-A(K,J)*B(J)
  100 B(K)=S/A(K,K)
  200 RETURN
      END
```

```
      SUBROUTINE RSOLV(N,KCOL,RIGHT,X)
      PARAMETER (NX=40,MX=250,LX=40,NX2=NX+NX)
      COMMON/L13/XX(NX2),X1(NX),R(NX,NX),B(NX,NX),W(NX),W1(NX),W2(NX),
     1    ASET(NX),U(NX,NX),KSET(NX),KSETO(NX),KSTATC(LX),KSTATF(MX)
      DIMENSION RIGHT(N),X(N)
      K=KCOL
      K1=K+1
   10 IF(K.EQ.0) RETURN
      T=RIGHT(K)
      IF(K1.GT.KCOL) GOTO 30
      DO 20 J=K1,KCOL
   20 T=T-X(J)*R(K,J)
   30 X(K)=T/R(K,K)
      K1=K
      K=K-1
      GOTO 10
      END
```

```
      SUBROUTINE UTRNS(N,KCOL,ACCUM)
      PARAMETER (NX=40,MX=250,LX=40,NX2=NX+NX)
      COMMON/L13/X(NX2),X1(NX),R0(NX,NX),B(NX,NX),W(NX),H(NX),W2(NX),
     1    ASET(NX),U(NX,NX),KSET(NX),KSET0(NX),KSTATC(LX),KSTATF(MX)
      COMMON/L14/R(NX),FUP(MX),DLDF(MX),CUP(LX),DLDC(LX),
     1       STEPF(MX),KSTATL(NX),DSTA2(NX2,NX2)
      LOGICAL ACCUM
      K1=KCOL+1
      IF(ACCUM) GOTO 100
      IF(KCOL.EQ.0) RETURN
      DO 30 KK=1,KCOL
      K=K1-KK
      T=0.0
      DO 10 J=K,N
   10 T=T+U(J,K)*R(J)
      DO 20 J=K,N
   20 R(J)=R(J)-T*U(J,K)
   30 CONTINUE
      RETURN
  100 DO 110 I=1,N
  110 W(I)=R(I)
      DO 130 I=1,N
      T=0.0
      DO 120 J=1,N
  120 T=T+U(I,J)*W(J)
  130 R(I)=T
      RETURN
      END
```

```
      SUBROUTINE UTTRNS(N,KCOL,ACCUM,R)
      PARAMETER (NX=40,MX=250,LX=40,NX2=NX+NX)
      COMMON/L13/X(NX2),X1(NX),RO(NX,NX),B(NX,NX),W(NX),H(NX),W2(NX),
     1    ASET(NX),U(NX,NX),KSET(NX),KSETO(NX),KSTATC(LX),KSTATF(MX)
      DIMENSION R(N)
      LOGICAL ACCUM

      IF(ACCUM) GOTO 100
      IF(KCOL.EQ.0) RETURN
      DO 30 K=1,KCOL
      T=0.0
      DO 10 I=K,N
   10 T=T+R(I)*U(I,K)
      DO 20 I=K,N
   20 R(I)=R(I)-T*U(I,K)
   30 CONTINUE
      RETURN
  100 DO 110 I=1,N
  110 W(I)=R(I)
      DO 130 K=1,N
      T=0.0
      DO 120 I=1,N
  120 T=T+U(I,K)*W(I)
  130 R(K)=T
      RETURN
      END
```

```
c***********************************************************************

c       file name: QUASI2.FOR

c       for programs FETSCG and FETSCG1 (PC/AT version).
c       this group of subroutines constitutes the L2 optimization
c       package

c***********************************************************************

$NOFLOATCALLS


        SUBROUTINE PENALTY(N,L,X,V,G)
        PARAMETER (NX=40,MX=250,LX=40)
        DIMENSION X(N),G(NX)
        COMMON/L12/C(LX),DC(LX,NX),CLOC(LX)
        IF(L.LE.0)RETURN
        DO 30 K=1,L
        CLC=C(K)
        DO 10 I=1,N
10      CLC=CLC+DC(K,I)*X(I)
        IF(CLC.LT.0.0)THEN
                V=V+10.0*CLC*CLC
                DO 20 I=1,N
20              G(I)=G(I)+20.0*CLC*DC(K,I)
        ENDIF
30      CONTINUE
        RETURN
        END
```

```
      SUBROUTINE QUASI2(N,M,L,LEQ,X,DX,EPS,MAXF)
      PARAMETER (NX=40,MX=250,LX=40)
      DIMENSION X(N),H(NX,NX),X1(NX),G(NX),G1(NX)
      DIMENSION W(NX),W1(NX),S(NX)
      LOGICAL CONV
      COMMON/L11/MARK,F(MX),DF(MX,NX),F1(MX),DF1(MX,NX)
      NCALL=1
      MARK=1
      DMIN=1.0
      ALP=0.5
      SEPS=1.E-7
      NP=N+1
      N1=N-1
      NN=N*NP/2
      DO 20 I=1,N
      DO 10 J=1,N
10    H(J,I)=0.
20    H(I,I)=1.
      CALL FDF(N,M,X)
      IF(MARK.EQ.0)GOTO 370
      V=0.0
      DO 30 I=1,N
30    G(I)=0.0
      CALL PENALTY(N,L,X,V,G)
      DO 40 J=1,M
      V=V+F(J)*F(J)
      DO 40 I=1,N
40    G(I)=G(I)+2.0*F(J)*DF(J,I)

c     iterative loop
c     find direction S

60    W1(1)=-G(1)
      IF(N.EQ.1)GOTO 90
      DO 80 I=2,N
      I1=I-1
      Z=-G(I)
      DO 70 J=1,I1
70    Z=Z-H(I,J)*W1(J)
80    W1(I)=Z
90    S(N)=W1(N)/H(N,N)
      IF(N.EQ.1)GOTO 120
      N1=N-1
      DO 110 I=1,N1
      I1=N-I
      I2=I1+1
      Z=0.
      DO 100 J=I2,N
100   Z=Z+H(I1,J)*S(J)
110   S(I1)=W1(I1)/H(I1,I1)-Z

c     check the directional gradient

120   GS=0.
      DO 130 I=1,N
```

```
130     GS=GS+S(I)*G(I)
        IF(GS.GE.-SEPS)GOTO 350
140     GS0=GS
        ALAMDA=-2.*V/GS
        IF(ALAMDA.GT.1.)ALAMDA=1.
        TOT=0.
        VDIFF=V

c       small loop, line search

150     DXX=0.0
        DO 160 I=1,N
        DX=ALAMDA*S(I)
        IF(ABS(DX).GT.DXX)DXX=ABS(DX)
160     X1(I)=X(I)+DX
        CALL FDF(N,M,X1)
        NCALL=NCALL+1
        V1=0.0
        DO 170 I=1,N
170     G1(I)=0.0
        CALL PENALTY(N,L,X1,V1,G1)
        DO 180 J=1,M
        V1=V1+F(J)*F(J)
        DO 180 I=1,N
180     G1(I)=G1(I)+2.0*F(J)*DF(J,I)

        CONV=MARK.EQ.0.OR.V1.LT.SEPS.OR.NCALL.GE.MAXF.OR.DXX.LE.EPS
        IF(CONV)GOTO 350

        GYS=0.
        DO 190 I=1,N
190     GYS=GYS+S(I)*G1(I)
        IF(V1.GE.V)GOTO 220
        IF(ABS(GYS/GS0).LE.0.9)GOTO 230
        IF(GYS.GT.0.)GOTO 220

c       linear extrapolation

        TOT=TOT+ALAMDA
        Z=10.
        IF(GS.GE.GYS)GOTO 200
        Z1=10.0*ABS(GS-GYS)
        Z2=ABS(GYS)
        IF(Z2.GT.Z1)GOTO 200
        Z=10.0*Z2/Z1
200     ALAMDA=ALAMDA*Z
        V=V1
        GS=GYS
        DO 210 I=1,N
210     X(I)=X1(I)
        GOTO 150

c       cubic interpolation

220     Z=3.*(V-V1)/ALAMDA+GYS+GS
```

```
      ZZ=SQRT(Z*Z-GS*GYS)
      GZ=GYS+ZZ
      Z=1.0-(GZ-Z)/(ZZ+GZ-GS)
      ALAMDA=ALAMDA*Z
      GOTO 150

c     line search completed

230   ALAMDA=ALAMDA+TOT
      V=V1
      VDIFF=VDIFF-V
      DGS=GYS-GS0
      LINK=1

c     updating H

      IF(DGS+ALAMDA*GS0.GT.0.)GOTO 250
      DO 240 I=1,N
240   S(I)=G1(I)-G(I)
      SIG=1./(ALAMDA*DGS)
      GOTO 270

250   ZZ=ALAMDA/(DGS-ALAMDA*GS0)
      Z=DGS*ZZ-1.
      DO 260 I=1,N
260   S(I)=Z*G(I)+G1(I)
      SIG=1./(ZZ*DGS*DGS)
270   W1(1)=S(1)
      DO 290 I=2,N
      I1=I-1
      Z=S(I)
      DO 280 J=1,I1
280   Z=Z-H(I,J)*W1(J)
290   W1(I)=Z
      DO 300 I=1,N
      Z=H(I,I)+SIG*W1(I)*W1(I)
      IF(Z.LE.0.)Z=DMIN
      IF(Z.LT.DMIN)DMIN=Z
      H(I,I)=Z
      W(I)=W1(I)*SIG/Z
300   SIG=SIG-W(I)*W(I)*Z
      N1=N-1
      DO 310 I=1,N1
      I1=I+1
      DO 310 J=I1,N
      S(J)=S(J)-H(I,J)*W1(I)
      H(I,J)=H(I,J)+W(I)*S(J)
310   H(J,I)=H(I,J)
      IF(LINK.EQ.2)GOTO 330
      LINK=2
      DO 320 I=1,N
320   S(I)=G(I)
      SIG=1./GS0
      IF(DGS+ALAMDA*GS0.GT.0.)SIG=-ZZ
      GOTO 270
```

```
c       Updating of H completed

330     V=V1
        DO 340 I=1,N
        X(I)=X1(I)
340     G(I)=G1(I)
        GOTO 60
350     IF(V1.GE.V)GOTO 370
        DO 360 I=1,N
360     X(I)=X1(I)
370     MAXF=NCALL
        RETURN
        END
```

```
c*********************************************************************

c       file name: CED.ASM

c       for program FETSCG (PC/AT version).
c       these assembly language routines are used for the screen editor

c*********************************************************************



;=========================================================================
S_SEG   SEGMENT STACK
        DB      64      DUP     ('stack   ')
S_SEG   ENDS
;=========================================================================
D_SEG   SEGMENT
        PATH1   DB      30      DUP(0)
        FINLMSG DB      '  File saved: '
        PATH2   DB      30      DUP(0)
        FBUF    DB      80      DUP(?)
        F5BUF   DB      81      DUP(32)
        F3BUF   DB      81      DUP(32)
        BLNKL   DB      80      DUP(32)
        STATUS  DB      10      DUP(0)
        BUFSEG  DW      (?)
        LBUF    DW      0
        TOPBUF  DW      0
        SIZ     DW      0
        LSIZ    DB      1001    DUP(0)
        ERMSG1  DB      'NOT ENOUGH MEMORY$'
        ERMSG2  DB      'INVALID FILE NAME$'
        ERMSG3  DB      'FILE TOO LARGE$'
        MENU    DB      '          F1=Quit  F2=Exit  F3=UndoF4  F4=DelEOL'
                DB      '  F5=UndoF6  F6=DelLine          '
D_SEG   ENDS
;=========================================================================
Z_SEG   SEGMENT
Z_SEG   ENDS
C_SEG   SEGMENT
MAIN    PROC    FAR
        ASSUME  CS:C_SEG,DS:D_SEG
;
        push    ds
        sub     ax,ax
        push    ax
        mov     ax,D_SEG
        mov     ds,ax
;
        MOV     AL,0
        MOV     AH,05H
        INT     10H
        CALL    GET_BUF
        JNC     STEP1
        JMP     FINISH
```

```
STEP1:  CALL    GET_FN
;
        CALL    RD_FL
        JNC     STEP2
        JMP     FINISH
;
STEP2:  CALL    WR_FL
STEP3:  MOV     AH,0
        INT     16H
        CMP     AL,0
        JNE     NORML
        JMP     NONCHR
NORML:  CMP     AL,08H
        JNE     S_UP
        CALL    BKSPC
        JMP     SHORT   STEP3
S_UP:   CMP     AX,1848$_8$
        JNE     S_DN
        JMP     PGDN
        JMP     SHORT   STEP3
S_DN:   CMP     AX,2053$_0$
        JNE     S_LEFT
        JMP     PGUP
        JMP     SHORT   STEP3
S_LEFT: CMP     AX,1925$_2$
        JNE     S_RIGT
        CALL    WRDLT
        JMP     SHORT   STEP3
S_RIGT: CMP     AX,1976$_6$
        JNE     TABKEY
        CALL    WRDRT
        JMP     STEP3
TABKEY: CMP     AL,9
        JNE     CRLF
        CALL    TAB2
        JMP     STEP3
CRLF:   CMP     AL,0DH
        JNE     WR_CHR
        CALL    NEW_LN
        JMP     SHORT   GO_WR
WR_CHR: CALL    CINSRT
GO_WR:  MOV     DL,AL
        MOV     AH,02H
        INT     21H
        JMP     STEP3
NONCHR: CMP     AH,60
        JNE     CON1
        JMP     STEP4
CON1:   CMP     AH,59
        JNE     CON2
        JMP     STEP5
CON2:   CMP     AH,80
        JE      C_MOVE
        CMP     AH,72
        JE      C_MOVE
```

```
        CMP     AH,75
        JE      C_MOVE
        CMP     AH,77
        JNE     CUREOL
C_MOVE: CALL    MOV_CUR
        JMP     STEP3
CUREOL: CMP     AH,116
        JNE     CURBOL
        CALL    CUR_EOL
        JMP     STEP3
CURBOL: CMP     AH,115
        JNE     NONCUR
        CALL    CUR_BOL
        JMP     STEP3
NONCUR: CMP     AH,83
        JNE     INST
        CALL    DELT
        JMP     STEP3
INST:   CMP     AH,82
        JNE     FKEY
        XOR     BYTE PTR STATUS,0FFH
        JMP     STEP3
FKEY:   CMP     AH,64
        JA      MOVPG
        CMP     AH,61
        JB      MOVPG
        CALL    FKEYS
        JMP     STEP3
MOVPG:  CMP     AH,73
        JNE     MOVPGU
PGDN:   MOV     CX,15
LPGDN:  CALL    LIN_DN
        LOOP    LPGDN
        JMP     STEP3
MOVPGU: CMP     AH,81
        JNE     CHOME
PGUP:   MOV     CX,15
LPGUP:  CALL    LIN_UP
        LOOP    LPGUP
        JMP     STEP3
CHOME:  CMP     AH,71
        JNE     CCORN
        MOV     BH,0
        MOV     DX,0
        MOV     AH,02H
        INT     10H
        JMP     STEP3
CCORN:  CMP     AH,79
        JNE     RBELL
        MOV     BH,0
        MOV     DH,23
        MOV     DL,0
        MOV     AH,02H
        INT     10H
        JMP     STEP3
```

```
RBELL:  CALL    BELL
        JMP     STEP3
;
STEP4:  CALL    SAV_FL
        JMP     SHORT   FINISH
STEP5:  CALL    QUIT
;
FINISH: MOV     AH,49H
        INT     21H
        RET
MAIN    ENDP
;
QUIT    PROC    NEAR
        SUB     DX,DX
        MOV     DH,23
        MOV     BH,0
        MOV     AH,02H
        INT     10H
        MOV     CX,160
        MOV     AL,32
        MOV     AH,09H
        MOV     BL,7
        INT     10H
        RET
QUIT    ENDP
;
CLRS    PROC    NEAR
        MOV     CX,0
        MOV     DH,24
        MOV     DL,79
        MOV     AL,0
        MOV     AH,06H
        MOV     AL,0
        MOV     BH,07H
        INT     10H
        RET
CLRS    ENDP
;
GET_BUF PROC    NEAR
        PUSH    ES
        MOV     BX,Z_SEG
        MOV     AX,ES
        SUB     BX,AX
        MOV     AH,4AH
        INT     21H
        JC      ME_ER
        MOV     BX,4000
        MOV     AH,48H
        INT     21H
        JC      ME_ER
        MOV     BUFSEG,AX
        MOV     ES,AX
        MOV     DI,0
        MOV     AL,20H
        MOV     CX,64000
```

```
        CLD
REP     STOSB
        CLC
        JMP     OKG1
ME_ER:  LEA     DX,ERMSG1
        MOV     AH,09H
        INT     21H
        STC
OKG1:   POP     ES
        RET
GET_BUF ENDP
;
GET_FN  PROC    NEAR
        MOV     BYTE PTR STATUS[1],0    ;assume no output file name
        MOV     DI,80H
        SUB     CX,CX
        MOV     CL,ES:[DI]              ;number of parameter bytes
        JCXZ    GTFN4
        LEA     SI,PATH1
GTFN0:  INC     DI
        CMP     BYTE PTR ES:[DI],' '    ;skip the leading spaces
        JNE     GTFN1
        LOOP    GTFN0
        JMP     SHORT   GTFN4
GTFN1:  MOV     AL,ES:[DI]
        CMP     AL,' '                  ;a space would mark the
        JE      GTFN2                   ;end of the input file name
        MOV     [SI],AL
        INC     DI                      ;get the input file name
        INC     SI                      ;in PATH1
        LOOP    GTFN1
        JMP     SHORT   GTFN4           ;no output file name
GTFN2:  CMP     BYTE PTR ES:[DI],' '    ;skip the spaces separating
        JNE     GTFN3                   ;two file names
        INC     DI
        LOOP    GTFN2
        JMP     SHORT   GTFN4           ;no output file name
GTFN3:  LEA     SI,PATH2
        CLD
REP     MOVSB                           ;get output file name in PATH2
        MOV     BYTE PTR STATUS[1],1    ;set flag that PATH2 exists
GTFN4:  RET
GET_FN  ENDP
;
FNBAK   PROC    NEAR
        PUSH    ES
        PUSH    DS
        POP     ES
        LEA     SI,PATH1
        LEA     DI,PATH2
        CLD
        MOV     BX,0
        MOV     CX,30
FNBK1:  CMP     BYTE PTR [SI],0
        JE      FNBK3
```

```
        CMP     BYTE PTR [SI],'.'
        JNE     FNBK2
        MOV     BX,SI
FNBK2:  MOVSB
        LOOP    FNBK1
FNBK3:  CMP     BX,0
        JNE     FNBK4
        MOV     BX,SI
FNBK4:  MOV     WORD PTR DS:[BX],422EH
        MOV     WORD PTR DS:[BX+2],4B41H
        POP     ES
        RET
FNBAK   ENDP
;
RD_FL   PROC    NEAR
        MOV     AX,BUFSEG
        MOV     ES,AX
        MOV     AH,3DH
        MOV     AL,0
        LEA     DX,PATH1
        INT     21H             ;open the file
        JNC     OKR1
        CMP     AX,2
        JNE     FFER
        MOV     AX,1920
        MOV     SIZ,AX
        MOV     BYTE PTR STATUS[1],0
        JMP     OKR11
FFER:   LEA     DX,ERMSG2
        MOV     AH,09H
        INT     21H
        STC
        JMP     OKR12
OKR1:   MOV     BX,AX
        MOV     DI,0
OKR2:   LEA     DX,FBUF
        MOV     CX,80
        MOV     AH,3FH
        INT     21H
        MOV     CX,AX
        CMP     CX,0
        JA      OKR3
        JMP     OKR8
OKR3:   LEA     SI,FBUF
        PUSH    CX
OKR4:   MOV     AL,[SI]
        INC     SI
        CMP     AL,0DH
        JE      OKR7
        CMP     AL,0AH
        JNE     OKR5
        CALL    END_LN
        JNC     OKR7
        LEA     DX,ERMSG3
        MOV     AH,09H
```

```
        INT     21H
        POP     CX
        JMP     OKR12
OKR5:   CMP     AL,09H
        JNE     OKR6
        CALL    TAB1
        JMP     SHORT   OKR7
OKR6:   MOV     ES:[DI],AL
        INC     DI
OKR7:   LOOP    OKR4
        POP     CX
        CMP     CX,80
        JNE     OKR8
        JMP     OKR2
OKR8:   DEC     DI
        MOV     AL,20H
        MOV     ES:[DI],AL
        MOV     AX,LBUF
        ADD     AX,80
        CMP     AX,1920
        JAE     OKR9
        MOV     AX,1920
OKR9:   MOV     SIZ,AX
        MOV     AH,3EH
        INT     21H
OKR10:  CMP     BYTE PTR STATUS[1],0
        JA      OKR11
        CALL    FNBAK
OKR11:  CLC
OKR12:  RET
RD_FL   ENDP
;
WR_FL   PROC    NEAR
        PUSH    ES
        PUSH    DS
        CALL    CLRS
        MOV     CX,1920
        MOV     DI,0
        MOV     AX,0B800H
        MOV     ES,AX
        MOV     AX,BUFSEG
        MOV     DS,AX
        MOV     SI,0
        CLD
WRFL1:  MOVSB
        INC     DI
        LOOP    WRFL1
        POP     DS
        LEA     SI,MENU
        MOV     CX,80
        MOV     AL,1FH
WRFL2:  MOVSB
        MOV     ES:[DI],AL
        INC     DI
        LOOP    WRFL2
```

```
        POP     ES
        MOV     BH,0
        MOV     DX,0
        MOV     AH,02H
        INT     10H
        PUSH    ES
        SUB     AX,AX
        MOV     ES,AX
        MOV     AL,ES:[417H]
        MOV     STATUS[2],AL
        MOV     AL,11011111B
        AND     ES:[417H],AL
        POP     ES
        RET
WR_FL   ENDP
;
MOV_CUR PROC    NEAR
        CALL    GETCUR
        CMP     AH,80
        JNE     NO_DN
        CMP     DH,23
        JB      DNCUR
        CALL    LIN_UP
        JMP     OKMOV
DNCUR:  INC     DH
        JMP     SHORT   MOV_IT
NO_DN:  CMP     AH,72
        JNE     NO_UP
        CMP     DH,0
        JA      UPCUR
        CALL    LIN_DN
        JMP     OKMOV
UPCUR:  DEC     DH
        JMP     SHORT   MOV_IT
NO_UP:  CMP     AH,75
        JNE     NO_LF
        CMP     DL,0
        JBE     OKMOV
        DEC     DL
        JMP     SHORT   MOV_IT
NO_LF:  CMP     DL,79
        JAE     OKMOV
        INC     DL
MOV_IT: MOV     BH,0
        MOV     AH,02H
        INT     10H
OKMOV:  RET
MOV_CUR ENDP
;
LIN_UP  PROC    NEAR
        MOV     AX,TOPBUF
        MOV     DI,AX
        MOV     DX,AX
        ADD     DX,80
        ADD     AX,1920
```

```
        CMP     AX,WORD PTR SIZ
        JAE     OKUPL
        PUSH    CX
        MOV     TOPBUF,DX
        MOV     DX,0
        CALL    STOLIN
        CALL    SHTU
        MOV     SI,TOPBUF
        ADD     SI,1840
        PUSH    DS
        PUSH    ES
        POP     DS
        MOV     DH,23
        MOV     DL,0
        CALL    PUTLIN
        POP     DS
        POP     CX
OKUPL:  RET
LIN_UP  ENDP
;
LIN_DN  PROC    NEAR
        CMP     WORD PTR TOPBUF,0
        JBE     OKDNL
        PUSH    CX
        MOV     DX,0
        CALL    SHTD
        MOV     DI,TOPBUF
        SUB     DI,80
        MOV     TOPBUF,DI
        MOV     SI,DI
        PUSH    DS
        PUSH    ES
        POP     DS
        CALL    PUTLIN
        POP     DS
        POP     CX
OKDNL:  RET
LIN_DN  ENDP
;
BELL    PROC    NEAR
        MOV     AH,02H
        MOV     DL,07H
        INT     21H
        RET
BELL    ENDP
;
TAB1    PROC    NEAR
        PUSH    CX
        MOV     AX,LBUF
        SUB     DI,AX
        MOV     CL,3
        SHR     DI,CL
        INC     DI
        MOV     CL,3
        SHL     DI,CL
```

```
        ADD     DI,AX
        POP     CX
        RET
TAB1    ENDP
;
TAB2    PROC    NEAR
        CALL    GETCUR
        MOV     CL,3
        SHR     DL,CL
        INC     DL
        MOV     CL,3
        SHL     DL,CL
        MOV     BH,0
        MOV     AH,02
        INT     10H
        RET
TAB2    ENDP
;
END_LN  PROC    NEAR
        MOV     AX,LBUF
        ADD     AX,80
        MOV     DI,AX
        MOV     LBUF,AX
        CMP     AX,60000
        JB      EDLN1
        STC
        JC      EDLN2
EDLN1:  CLC
EDLN2:  RET
END_LN  ENDP
;
SAV_FL  PROC    NEAR
        PUSH    DS
        MOV     SI,0
        MOV     AX,BUFSEG
        MOV     ES,AX
        MOV     DI,TOPBUF
        MOV     AX,0B800H
        MOV     DS,AX
        MOV     CX,1920
        CLD
SVFL1:  MOVSB
        INC     SI
        LOOP    SVFL1
        POP     DS
        MOV     CX,SIZ
        MOV     DI,CX
OKSF1:  DEC     DI
        CMP     BYTE PTR ES:[DI],' '
        JNE     OKSF2
        LOOP    OKSF1
        MOV     SIZ,CX
        JMP     OKSF10
OKSF2:  MOV     AX,CX
        SUB     DX,DX
```

```
        MOV     SI,80
        DIV     SI
        CMP     DX,0
        JE      OKSF3
        NEG     DX
        ADD     DX,80
        ADD     CX,DX
        INC     AX
OKSF3:  MOV     BX,AX
        PUSH    BX
        MOV     SI,CX
        DEC     SI
OKSF4:  MOV     CX,80
        MOV     DI,SI
OKSF5:  CMP     BYTE PTR ES:[DI],' '
        JNE     OKSF6
        DEC     DI
        LOOP    OKSF5
OKSF6:  MOV     LSIZ[BX],CL
        DEC     BX
        JZ      OKSF7
        SUB     SI,80
        JMP     SHORT   OKSF4
OKSF7:  POP     CX
        MOV     BX,1
        MOV     DI,0
        MOV     DX,0
OKSF8:  PUSH    CX
        SUB     CX,CX
        MOV     CL,LSIZ[BX]
        JCXZ    OKSF9
        MOV     SI,DX
        PUSH    DS
        PUSH    ES
        POP     DS
        CLD
REP     MOVSB
        POP     DS
OKSF9:  MOV     AH,0AH
        MOV     AL,0DH
        MOV     ES:[DI],AX
        INC     BX
        ADD     DI,2
        ADD     DX,80
        POP     CX
        LOOP    OKSF8
        MOV     SIZ,DI
;
OKSF10: CMP     BYTE PTR STATUS[1],0
        JA      OKSF00
        PUSH    ES
        PUSH    DS
        POP     ES
        LEA     DX,PATH1
        MOV     AH,41H
```

```
        INT     21H
        LEA     DX,PATH2
        LEA     DI,PATH1
        MOV     AH,56H
        INT     21H
        POP     ES
OKSF00: MOV     AH,3CH
        MOV     CX,0
        LEA     DX,PATH2
        INT     21H                 ;open the file
        JNC     OKSF11
        CALL    FNBAK
        MOV     BYTE PTR STATUS[1],0
        JMP     SHORT   OKSF10
OKSF11: MOV     BX,AX
        MOV     CX,SIZ
        JCXZ    OKSF12
        PUSH    DS
        PUSH    ES
        POP     DS
        MOV     DX,0
        MOV     AH,40H
        INT     21H
        POP     DS
OKSF12: MOV     AH,3EH
        INT     21H
        CALL    CLEANUP
OKSF20: RET
SAV_FL  ENDP
;
CLEANUP PROC    NEAR
        CALL    CLRS
        PUSH    ES
        SUB     AX,AX
        MOV     ES,AX
        MOV     AL,STATUS[2]
        MOV     ES:[417H],AL
        MOV     AX,0B800H
        MOV     ES,AX
        MOV     DI,160
        LEA     SI,FINLMSG
        MOV     CX,44
        MOV     AL,1FH
        CLD
CLNUP1: MOVSB
        MOV     ES:[DI],AL
        INC     DI
        LOOP    CLNUP1
        POP     ES
        MOV     BH,0
        SUB     DX,DX
        MOV     DH,3
        MOV     AH,02H
        INT     10H
        RET
```

```
CLEANUP ENDP
;
GETCUR  PROC    NEAR
        PUSH    AX
        MOV     AH,03H
        MOV     BH,0
        INT     10H
        POP     AX
        RET
GETCUR  ENDP
;
DELT    PROC    NEAR
        CALL    GETCUR
        CMP     DL,79
        JE      OKD1
        CALL    SHTL
OKD1:   RET
DELT    ENDP
;
BKSPC   PROC    NEAR
        CALL    GETCUR
        CMP     DL,0
        JE      OKB1
        DEC     DL
        PUSH    DX
        MOV     BH,0
        MOV     AH,02H
        INT     10H
        POP     DX
        CALL    SHTL
OKB1:   RET
BKSPC   ENDP
;
VADRS   PROC    NEAR
        MOV     AL,DH
        MOV     BL,160
        MUL     BL
        MOV     DI,AX
        MOV     DH,0
        ADD     DI,DX
        ADD     DI,DX
        MOV     AX,0B800H
        RET
VADRS   ENDP
;
SHTL    PROC    NEAR
        PUSH    DS
        PUSH    ES
        CALL    VADRS
        MOV     SI,DI
        ADD     SI,2
        MOV     CX,79
        SUB     CX,DX
        MOV     DS,AX
        MOV     ES,AX
```

```
        CLD
REP     MOVSW
        MOV     AX,0720H
        MOV     ES:[DI],AX
        POP     ES
        POP     DS
        RET
SHTL    ENDP
;
SHTR    PROC    NEAR
        PUSH    DS
        PUSH    ES
        CALL    VADRS
        MOV     CX,79
        SUB     CX,DX
        ADD     DI,CX
        ADD     DI,CX
        MOV     SI,DI
        SUB     SI,2
        MOV     DS,AX
        MOV     ES,AX
        STD
REP     MOVSW
        CLD
        POP     ES
        POP     DS
        RET
SHTR    ENDP
;
FKEYS   PROC    NEAR
        CALL    GETCUR
        CMP     AH,63
        JNE     NO_F5
        CALL    UDELL
        JMP     SHORT   FDONE
NO_F5:  CMP     AH,64
        JNE     NO_F6
        CALL    DELL
        JMP     SHORT   FDONE
NO_F6:  CMP     AH,61
        JNE     NO_F3
        CALL    UDEOL
        JMP     SHORT   FDONE
NO_F3:  CALL    DEOL
FDONE:  RET
FKEYS   ENDP
;
STOLIN  PROC    NEAR
        PUSH    DS
        PUSH    DX
        PUSH    DI
        CALL    VADRS
        MOV     SI,DI
        POP     DI
        MOV     DS,AX
```

```
        MOV     CX,80
        CLD
STLN1:  MOVSB
        INC     SI
        LOOP    STLN1
        POP     DX
        POP     DS
        RET
STOLIN  ENDP
;
LIN_OUT PROC    NEAR
        MOV     AL,DH
        MOV     BL,80
        MUL     BL
        MOV     DI,TOPBUF
        ADD     DI,AX
        MOV     SI,DI
        ADD     SI,80
        MOV     CX,SIZ
        CMP     CX,SI
        JBE     OKLO3
        SUB     CX,80
        CMP     CX,1920
        JB      MINSIZ
        MOV     SIZ,CX
MINSIZ: SUB     CX,DI
        ADD     CX,80
        PUSH    DS
        PUSH    ES
        POP     DS
        CLD
REP     MOVSB
        POP     DS
OKLO3:  RET
LIN_OUT ENDP
;
DELL    PROC    NEAR
        MOV     F5BUF,AH
        MOV     DL,0
        CALL    LIN_OUT
        LEA     DI,F5BUF
        INC     DI
        PUSH    ES
        PUSH    DS
        POP     ES
        CALL    STOLIN
        POP     ES
        CALL    SHTU
        MOV     DH,23
        MOV     DL,0
        MOV     SI,TOPBUF
        ADD     SI,1840
        PUSH    DS
        PUSH    ES
        POP     DS
```

```
        CALL    PUTLIN
        POP     DS
        RET
DELL    ENDP
;
SHTU    PROC    NEAR
        CMP     DH,23
        JAE     OKS11
        MOV     CH,DH
;       INC     CH
        MOV     CL,0
        MOV     DL,79
        MOV     DH,23
        MOV     BH,07H
        MOV     AH,06H
        MOV     AL,1H
        INT     10H
OKS11:  RET
SHTU    ENDP
;
SHTD    PROC    NEAR
        PUSH    DX
        PUSH    DI
        MOV     DI,TOPBUF
        ADD     DI,1840
        MOV     DH,23
        MOV     DL,0
        CALL    STOLIN
        POP     DI
        POP     DX
        PUSH    DX
        CMP     DH,23
        JAE     OKS21
        MOV     CH,DH
        MOV     CL,0
        MOV     DH,23
        MOV     DL,79
        MOV     BH,07H
        MOV     AH,07H
        MOV     AL,1H
        INT     10H
OKS21:  POP     DX
        RET
SHTD    ENDP
;
PUTLIN  PROC    NEAR
        PUSH    ES
        CALL    VADRS
        MOV     ES,AX
        MOV     CX,80
        CLD
PTLN1:  MOVSB
        INC     DI
        LOOP    PTLN1
        POP     ES
```

```
        RET
PUTLIN  ENDP
;
LIN_IN  PROC    NEAR
        MOV     AX,TOPBUF
        ADD     AX,1839
        MOV     CX,SIZ
        MOV     SI,CX
        MOV     DI,SI
        ADD     DI,80
        MOV     SIZ,DI
        CMP     CX,AX
        JBE     OKLIN
        SUB     CX,AX
        PUSH    DS
        PUSH    ES
        POP     DS
        STD
REP     MOVSB
        POP     DS
OKLIN:  RET
LIN_IN  ENDP
;
UDELL   PROC    NEAR
        LEA     SI,F5BUF
        CMP     BYTE PTR [SI],64
        JNE     OKU1
        MOV     DL,0
        PUSH    SI
        CALL    SHTD
        CALL    LIN_IN
        POP     SI
        INC     SI
        CALL    PUTLIN
OKU1:   RET
UDELL   ENDP
;
DEOL    PROC    NEAR
        PUSH    DS
        PUSH    ES
        MOV     F3BUF,AH
        CALL    VADRS
        MOV     SI,DI
        LEA     DI,F3BUF
        INC     DI
        PUSH    DS
        POP     ES
        MOV     DS,AX
        MOV     CX,80
        SUB     CX,DX
        PUSH    DS
        PUSH    SI
        PUSH    CX
        CLD
DEOL1:  MOVSB
```

```
        INC     SI
        LOOP    DEOL1
        POP     CX
        POP     DI
        POP     ES
        MOV     AX,0720H
REP     STOSW
        POP     ES
        POP     DS
        RET
DEOL    ENDP
;
UDEOL   PROC    NEAR
        LEA     SI,F3BUF
        CMP     BYTE PTR [SI],62
        JNE     OKU11
        PUSH    ES
        CALL    VADRS
        INC     SI
        MOV     ES,AX
        MOV     CX,80
        SUB     CX,DX
        CLD
UDEOL1: MOVSB
        INC     DI
        LOOP    UDEOL1
        POP     ES
OKU11:  RET
UDEOL   ENDP
;
CINSRT  PROC    NEAR
        PUSH    AX
        CMP     BYTE PTR STATUS,0
        JE      OKC1
        CALL    GETCUR
        CMP     DL,79
        JE      OKC1
        CALL    SHTR
OKC1:   POP     AX
        RET
CINSRT  ENDP
;
WRDRT   PROC    NEAR
        CALL    GETCUR
        PUSH    ES
        CMP     DL,79
        JE      OKW6
        PUSH    DX
        CALL    VADRS
        MOV     ES,AX
        MOV     CX,79
        SUB     CX,DX
        PUSH    DI
OKW1:   CMP     BYTE PTR ES:[DI],' '
        JE      OKW2
```

```
        ADD     DI,2
        LOOP    OKW1
        JMP     SHORT   OKW5
OKW2:   JCXZ    OKW5
OKW3:   ADD     DI,2
        CMP     BYTE PTR ES:[DI],' '
        JNE     OKW4
        LOOP    OKW3
        JMP     SHORT   OKW5
OKW4:   POP     SI
        SUB     DI,SI
        SHR     DI,1
        POP     DX
        ADD     DX,DI
        MOV     BH,0
        MOV     AH,02H
        INT     10H
        JMP     SHORT   OKW6
OKW5:   POP     DI
        POP     DX
OKW6:   POP     ES
        RET
WRDRT   ENDP
;
WRDLT   PROC    NEAR
        CALL    GETCUR
        PUSH    ES
        CMP     DL,0
        JE      OKW12
        PUSH    DX
        CALL    VADRS
        MOV     ES,AX
        MOV     CX,DX
        PUSH    DI
OKW7:   SUB     DI,2
        CMP     BYTE PTR ES:[DI],' '
        JNE     OKW8
        LOOP    OKW7
OKW8:   JCXZ    OKW11
OKW9:   SUB     DI,2
        CMP     BYTE PTR ES:[DI],' '
        JE      OKW10
        LOOP    OKW9
        SUB     DI,2
OKW10:  ADD     DI,2
        POP     AX
        SUB     AX,DI
        SHR     AX,1
        POP     DX
        SUB     DL,AL
        MOV     BH,0
        MOV     AH,02H
        INT     10H
        JMP     SHORT   OKW12
OKW11:  POP     DI
```

```
        POP     DX
OKW12:  POP     ES
        RET
WRDLT   ENDP
;
CUR_BOL PROC    NEAR
        CALL    GETCUR
        MOV     DL,0
        MOV     BH,0
        MOV     AH,02H
        INT     10H
        RET
CUR_BOL ENDP
;
CUR_EOL PROC    NEAR
        CALL    GETCUR
        PUSH    ES
        MOV     DL,79
        PUSH    DX
        CALL    VADRS
        MOV     ES,AX
        POP     DX
        MOV     CX,79
LPC1:   CMP     BYTE PTR ES:[DI],' '
        JNE     OKC11
        SUB     DI,2
        LOOP    LPC1
        JMP     SHORT   OKC12
OKC11:  CMP     CX,79
        JE      OKC12
        INC     CX
OKC12:  MOV     DL,CL
        MOV     BH,0
        MOV     AH,02H
        INT     10H
        POP     ES
        RET
CUR_EOL ENDP
;
NEW_LN  PROC    NEAR
        PUSH    AX
        CALL    GETCUR
        CMP     DL,0
        JE      BEFOR
        MOV     DL,0
        CMP     DH,23
        JE      AFTER
        INC     DH
        PUSH    DX
        MOV     BH,0
        MOV     AH,02H
        INT     10H
        POP     DX
BEFOR:  CALL    SHTD
OKNL1:  CALL    LIN_IN
```

```
        LEA     SI,BLNKL
        CALL    PUTLIN
        JMP     OKNL2
AFTER:  MOV     AX,TOPBUF
        MOV     DI,AX
        ADD     AX,80
        MOV     TOPBUF,AX
        PUSH    DX
        MOV     DX,0
        CALL    STOLIN
        CALL    SHTU
        POP     DX
        JMP     OKNL1
OKNL2:  POP     AX
        RET
NEW_LN  ENDP
;-----------------------------------------------------------------------------
C_SEG   ENDS
        END
```

```
c*********************************************************************

c       file name: IBMTXT.ASM

c       for program FETSCG (PC/AT version).
c       library of assembly language routines for text

c*********************************************************************


DATA    SEGMENT PUBLIC 'DATA'
        STR00   DB      1BH,'[0m$'      ;escape sequence, normal attribute
        STR11   DB      1BH,'[1m$'      ;escape sequence, high intensity
        STR22   DB      1BH,'[3 m$'     ;escape sequence, set colour
        CRLF    DB      0DH,0AH,'$'     ;carriage return + line feed
        L1      DW      0H
        L2      DW      0H
        C1      DW      0H
        C2      DW      0H
        PGN     DB      0H
        ATTRIBUTE       DB      7       ;default attribute for DISP
        OLDCUR  DW      26      DUP(?)  ;buffer for save cursor
        OLDSCR  DW      2000    DUP(720H)       ;buffer for save screen
DATA    ENDS
DGROUP  GROUP DATA

CODE    SEGMENT 'CODE'
        ASSUME CS:CODE,DS:DGROUP,SS:DGROUP

PUBLIC  BELL,CHECKY,CLEAR,COLOUR,DISP,ERSEOL,KYCODE,LOCATE
PUBLIC  PAGER,PAGIN,PAGOUT,RSTSCR,SAVCUR,SAVSCR,SETCUR,STRIM,WRKY

;=====================================================================
;       CLEAR
;       Function: clear the text screen and home the cursor
;       Access:   CALL CLEAR
;=====================================================================
CLEAR   PROC    FAR
        PUSH    BP
        PUSH    AX
        PUSH    BX
        PUSH    CX              ;use BIOS scroll up routine to clear
        PUSH    DX
        MOV     CX,0            ;from (0,0)
        MOV     DH,24
        MOV     DL,79           ;to (24,79)
        MOV     AX,0600H        ;AH=6 and  AL=0: scroll out
        MOV     BH,7            ;normal attribute for blank lines
        INT     10H
        MOV     AH,0FH
        INT     10H             ;get page number
        MOV     DX,0            ;home cursor, (0,0) screen position
        MOV     AH,2
        INT     10H
```

```
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        POP     BP
        RET
CLEAR   ENDP

;=========================================================================
;       ERSEOL
;       Function: erase from cursor position to end of line
;       Access:   CALL ERSEOL
;=========================================================================
ERSEOL  PROC    FAR
        PUSH    BP
        MOV     AH,0FH
        INT     10H                     ;get the active page number in BH
        CALL    ERS_EOL
        POP     BP
        RET
ERSEOL  ENDP

;=========================================================================
;       DISP
;       Function: powerful screen writer
;       Access:   CALL DISP(STRG)
;                       STRG is a character string ended with '_'
;       Imbeded commands available:
;       @B , @b         blinking on, off
;       @I , @i         high intensity on, off
;       @0 to @7        set foreground color
;                       (dark,blue,green,cyan,red,magenta,brown,white)
;       @R0 to @R7      set background color
;       $               go to start of next line
;       @Lnn            go to line nn  (column unchanged)
;       @Cnn            go to column nn (line unchanged)
;       @Snn            skip nn columns
;       @@ , @$         display reserved symbols @, $
;       @X              clear screen and home cursor
;       @e              erase from cursor to the end of line
;=========================================================================
DISP    PROC    FAR
        PUSH    BP
        MOV     BP,SP
        LES     DI,DWORD PTR[BP+6]      ;input string starts at ES:DI
        MOV     AH,0FH
        INT     10H             ;get active page number in BH
        MOV     BL,ATTRIBUTE    ;get the current attribute in BL
TOP:    MOV     AL,ES:[DI]      ;get a char
        INC     DI              ;increase the offset
        CMP     AL,'_'          ;end of string?
        JNE     CON1
        JMP     FINISH          ;yes, exit
CON1:   CMP     AL,'$'          ;go to the 1st column of the next line?
        JNE     CON2
```

```
        LEA     DX,CRLF
        MOV     AH,09H
        INT     21H             ;yes, write CR+LF
        JMP     TOP
CON2:   CMP     AL,'@'          ;a command begin with @ ?
        JE      CON4
CON3:   MOV     CX,1
        MOV     AH,09H
        INT     10H             ;no, just display the char
        MOV     AH,3
        INT     10H             ;read cursor position
        INC     DL              ;advance one column
        MOV     AH,2
        INT     10H             ;set cursor
        JMP     TOP
CON4:   MOV     AL,ES:[DI]      ;get the command following '@'
        INC     DI
        CMP     AL,'@'          ;display the reserved word '@'?
        JE      CON3
        CMP     AL,'$'          ;display the reserved word '$'?
        JE      CON3
        CMP     AL,'I'          ;set high intensity?
        JNE     CON5
        OR      BL,8H           ;turn on bit 3
        JMP     TOP
CON5:   CMP     AL,'i'          ;reset to normal intensity?
        JNE     CON6
        AND     BL,0F7H         ;turn off bit 3
        JMP     TOP
CON6:   CMP     AL,'B'          ;set blinking?
        JNE     CON7
        OR      BL,80H          ;turn on bit 7
        JMP     TOP
CON7:   CMP     AL,'b'          ;disable blinking?
        JNE     CON8
        AND     BL,7FH          ;turn off bit 7
        JMP     TOP
CON8:   CMP     AL,'X'          ;clear screen?
        JNE     CON9
        CALL    CLEAR
        JMP     TOP
CON9:   CMP     AL,'e'          ;erase to the end of line?
        JNE     CON10
        CALL    ERS_EOL
        JMP     TOP
CON10:  CMP     AL,'L'          ;move cursor to a new line?
        JE      CON11
        CMP     AL,'C'          ;move cursor to a new column?
        JE      CON11
        CMP     AL,'S'          ;skip some columns?
        JNE     CON12
CON11:  CALL    CHGCUR
        JMP     TOP
CON12:  CMP     AL,'R'          ;set background colour?
        JNE     CON13
```

```
        MOV     AL,ES:[DI]      ;get the new background colour (ASCII)
        INC     DI
        SUB     AL,30H          ;convert to a number
        MOV     CL,4
        SHL     AL,CL           ;shift it to bit 4-6
        AND     BL,8FH
        ADD     BL,AL           ;and set it
        JMP     TOP
CON13:  CMP     AL,'r'          ;reset reverse video?
        JNE     CON14
        JMP     TOP             ;not available on IBM
CON14:  CMP     AL,'U'          ;set underline?
        JNE     CON15
        JMP     TOP             ;not available on IBM
CON15:  CMP     AL,'u'          ;reset underline?
        JNE     CON16
        JMP     TOP             ;not available on IBM
CON16:  AND     BL,0F8H         ;set foreground colour
        SUB     AL,30H
        ADD     BL,AL
        JMP     TOP
FINISH: MOV     ATTRIBUTE,BL    ;save the current attribute setting
        POP     BP              ;done
        RET     4H
DISP    ENDP

;=====================================================================
;       CHGCUR
;       Function: set cursor to new location
;       Access:   internal, called by DISP
;=====================================================================
CHGCUR  PROC    NEAR            ;routine for changing cursor position
        PUSH    CX
        PUSH    AX
        MOV     AH,3H
        INT     10H             ;read the current cursor position in DH,DL
        MOV     AL,ES:[DI]      ;\
        INC     DI              ; get the high digit of new position
        AND     AL,0FH          ;/
        MOV     CL,10D          ;\
        MUL     CL              ;/ multiply it by 10
        MOV     CL,AL           ;save the partial result in CL
        MOV     AL,ES:[DI]
        INC     DI
        AND     AL,0FH          ;get the low digit
        ADD     CL,AL           ;add up the partial results
        POP     AX              ;restore AX
        CMP     AL,'L'          ;change line?
        JE      SETL
        CMP     AL,'C'          ;or change column?
        JE      SETC
        ADD     DL,CL           ;neither, skip columns
        JMP     MOVIT
SETC:   DEC     CL              ;normalize w.r.t. origin (0,0)
        MOV     DL,CL           ;change column
```

```
        JMP     MOVIT
SETL:   DEC     CL              ;normalize w.r.t. origin (0,0)
        MOV     DH,CL           ;change line
MOVIT:  PUSH    AX
        MOV     AH,02H
        INT     10H
        POP     AX
        POP     CX
        RET
CHGCUR  ENDP

;=========================================================================
;       ERS_EOL
;       Function: erase from cursor position to end of line
;       Access:   internal, called by ERSEOL and DISP
;       Note:     assume page number in BH
;=========================================================================
ERS_EOL PROC    NEAR
        PUSH    AX
        PUSH    CX
        PUSH    DX
        MOV     AH,03H
        INT     10H             ;read the current cursor position in DH,DL
        PUSH    DX
        MOV     CX,80
        SUB     CL,DL           ;get the distance to the end of line
        MOV     AL,20H          ;put a space into AL
        MOV     AH,09H
        INT     10H             ;repeat writing spaces to the end of line
        POP     DX
        MOV     AH,02H
        INT     10H             ;restore the cursor position
        POP     DX
        POP     CX
        POP     AX
        RET
ERS_EOL ENDP

;=========================================================================
;       CHECKY
;       Function: get a keyboard input without waiting
;       Access:   CALL CHECKY(IKY)
;                 IKY integer*2
;       Return:   IKY=0   no input available
;                 otherwise IKY holds the ASCII code of the input
;=========================================================================
CHECKY  PROC    FAR
        PUSH    BP
        MOV     BP,SP
        MOV     AH,0BH
        INT     21H                     ;check keyboard buffer
        CMP     AL,0                    ;buffer is empty, no input
        JE      EXITT
        MOV     AH,08H
        INT     21H                     ;get the input
```

```
EXITT:  LES     BX,DWORD PTR[BP+6]
        MOV     AH,0
        MOV     ES:[BX],AX
        POP     BP
        RET     04H
CHECKY  ENDP

;=====================================================================
;       KYCODE
;       Function: get a keyboard input
;       Access:   IKY=KYCODE(IKY)
;                 IKY,KYCODE integer*2
;       Return:   KYCODE (function value) holds the FULL CODE
;                       high byte holds scan code
;                       low  byte holds ASCII code
;=====================================================================
KYCODE  PROC    FAR
        PUSH    BP
        MOV     AH,0
        INT     16H
        POP     BP
        RET     04H
KYCODE  ENDP

;=====================================================================
;       SAVSCR
;       Function: save the text screen and then clear the text screen
;       Access:   CALL SAVSCR
;       Note:     retrieve the saved screen by calling RSTSCR
;=====================================================================
SAVSCR  PROC    FAR
        PUSH    BP
        MOV     AH,0FH
        INT     10H             ;get page number
        MOV     AH,3
        INT     10H             ;read cursor position
        MOV     OLDCUR[50],DX   ;save cursor
        LEA     DI,OLDSCR
        MOV     AX,DS
        MOV     ES,AX           ;ES:[DI] points at save screen buffer
        PUSH    DS
        MOV     AH,0H
        MOV     AL,BH           ;get page number in AX
        MOV     BX,1000H
        MUL     BX              ;figure out the page offset
        MOV     SI,AX
        MOV     AX,0B800H       ;video buffer segment
        MOV     DS,AX           ;DS:[SI] points at active screen buffer
        MOV     CX,2000D        ;save all 2000 screen positions by copying
        CLD                     ;from screen buffer to memory buffer
        REP     MOVSW
        POP     DS
        CALL    CLEAR           ;clear screen, home cursor
        POP     BP
        RET
```

```
SAVSCR  ENDP

;=====================================================================
;       RSTSCR
;       Function: restore the text screen saved by SAVSCR
;       Access:   CALL RSTSCR
;=====================================================================
RSTSCR  PROC    FAR
        PUSH    BP
        MOV     AH,0FH
        INT     10H             ;get page number
        MOV     DX,OLDCUR[50]   ;retrieve cursor position saved by SAVSCR
        MOV     AH,2
        INT     10H             ;restore cursor
        MOV     AH,0H
        MOV     AL,BH           ;get page number in AX
        MOV     BX,1000H
        MUL     BX              ;figure out the page offset
        MOV     DI,AX
        MOV     AX,0B800H       ;video buffer segment
        MOV     ES,AX           ;ES:[DI] points at active video buffer
        LEA     SI,OLDSCR       ;DS:[SI] points at save screen buffer
        MOV     CX,2000D        ;restore all 2000 screen positions by
        CLD                     ;copying from memory buffer to video buffer
        REP     MOVSW
        POP     BP
        RET
RSTSCR  ENDP

;=====================================================================
;       BELL
;       Function: ring the bell
;       Access:   CALL BELL
;=====================================================================
BELL    PROC    FAR
        MOV     AH,02H
        MOV     DL,07H
        INT     21H
        RET
BELL    ENDP

;=====================================================================
;       WRKY
;       Function: write a single character
;       Access:   CALL WRKY(IKY)
;                 IKY integer*2   holding an ASCII code,  or
;                 IKY character*1
;=====================================================================
WRKY    PROC    FAR
        PUSH    BP
        MOV     BP,SP
        LES     BX,DWORD PTR[BP+6]
        MOV     DL,ES:[BX]              ;get the character
        MOV     AH,02H                  ;display it
        INT     21H
```

```
        POP     BP
        RET     04H
WRKY    ENDP

;=====================================================================
;       PAGER
;       Function: set video display to a new page
;       Access:   CALL PAGER(IPAGE)
;                 IPAGE integer*2   holding the page number
;       Note:     validity of IPAGE is not checked
;=====================================================================
PAGER   PROC    FAR
        PUSH    BP
        MOV     BP,SP
        LES     BX,DWORD PTR[BP+6]
        MOV     AL,ES:[BX]
        MOV     AH,05H
        INT     10H
        POP     BP
        RET     4H
PAGER   ENDP

;=====================================================================
;       LOCATE
;       Function: move the cursor to location (irow,icol)
;       Access:   CALL LOCATE(IROW,ICOL)
;                 IROW  integer*2   holding the line number  ( 1 - 25 )
;                 ICOL  integer*2   holding the column number( 1 - 80 )
;=====================================================================
LOCATE  PROC    FAR
        PUSH    BP
        MOV     BP,SP
        LES     BX,DWORD PTR[BP+10]
        MOV     AX,ES:[BX]              ;get the first parameter
        DEC     AX                      ;convert into row index w.r.t. 0
        LES     BX,DWORD PTR[BP+6]
        MOV     DX,ES:[BX]              ;get the second parameter
        DEC     DX                      ;convert into column index w.r.t. 0
        MOV     DH,AL                   ;DH,DL : row,column
        MOV     AH,0FH
        INT     10H                     ;get page number into BH
        MOV     AH,02H
        INT     10H                     ;set cursor position
        POP     BP
        RET     08H
LOCATE  ENDP

;=====================================================================
;       PAGIN
;       Function: save a window of display screen in an array
;       Access:   CALL PAGIN(L1,C1,L2,C2,ARRAY)
;                 L1,C1,L2,C2  integer*2  defining the window
;                                  (L1,C1) top left corner inclusive
;                                  (L2,C2) bottom right corner inclusive
;                 ARRAY  any type of array with a minimum size of
```

```
;                                   2*number of screen positions  (in bytes)
;=========================================================================
PAGIN   PROC    FAR
        PUSH    BP
        MOV     AH,0FH
        INT     10H
        MOV     PGN,BH
        MOV     BP,Sp
        LES     BX,DWORD PTR[BP+22]     ;get the first argument
        MOV     AX,ES:[BX]
        DEC     AX                      ;first row is 0
        MOV     L1,AX                   ;starting from this row
        LES     BX,DWORD PTR[BP+18]
        MOV     AX,ES:[BX]
        MOV     L2,AX                   ;end at row L2-1
        LES     BX,DWORD PTR[BP+14]
        MOV     AX,ES:[BX]
        DEC     AX
        MOV     C1,AX                   ;starting from this column
        LES     BX,DWORD PTR[BP+10]
        MOV     AX,ES:[BX]
        MOV     C2,AX                   ;end at column C2-1
        LES     DI,DWORD PTR[BP+6]      ;destination, array
        MOV     AX,L1                   ;row offset
        MOV     BX,160D
        MUL     BX
        MOV     SI,AX
        ADD     SI,C1                   ;column offset
        ADD     SI,C1                   ;twice, attribute byte
        MOV     CX,C2
        SUB     CX,C1                   ;CX: number of columns to copy
        MOV     DX,L2
        SUB     DX,L1                   ;DX: number of rows to copy
        PUSH    DX
        MOV     AH,0H
        MOV     AL,PGN
        MOV     BX,1000H
        MUL     BX                      ;page offset
        ADD     SI,AX
        PUSH    DS
        MOV     AX,0B800H               ;video buffer segment
        MOV     DS,AX
        MOV     AX,CX                   ;store column counter
        POP     DX
        CLD
        MOV     BX,SI                   ;store starting address
ROWW:   REP     MOVSW                   ;copy one row
        ADD     BX,160D                 ;next row
        MOV     SI,BX                   ;new starting address
        MOV     CX,AX                   ;restore column counter
        DEC     DX
        JNZ     ROWW                    ;all rows copied?
        POP     DS                      ;yes, return
        POP     BP
        RET     20d
```

```
PAGIN   ENDP

;=====================================================================
;       PAGOUT
;       Function: restore a window of display screen from an array
;       Access:   CALL PAGOUT(L1,C1,L2,C2,ARRAY)
;                 L1,C1,L2,C2  integer*2  defining the window
;                                 (L1,C1) top left corner inclusive
;                                 (L2,C2) bottom right corner inclusive
;                 ARRAY  any type of array with a minimum size of
;                        2*number of screen positions  (in bytes)
;=====================================================================
PAGOUT  PROC    FAR
        PUSH    BP
        MOV     BP,SP
        MOV     AH,0FH
        INT     10H
        MOV     PGN,BH
        LES     BX,DWORD PTR[BP+22]     ;get the first argument
        MOV     AX,ES:[BX]
        DEC     AX                      ;first row is 0
        MOV     L1,AX                   ;starting from this row
        LES     BX,DWORD PTR[BP+18]
        MOV     AX,ES:[BX]
        MOV     L2,AX                   ;end at row L2-1
        LES     BX,DWORD PTR[BP+14]
        MOV     AX,ES:[BX]
        DEC     AX
        MOV     C1,AX                   ;starting from this column
        LES     BX,DWORD PTR[BP+10]
        MOV     AX,ES:[BX]
        MOV     C2,AX                   ;end at column C2-1
        MOV     AX,L1                   ;row offset
        MOV     BX,160D
        MUL     BX
        MOV     DI,AX
        ADD     DI,C1                   ;column offset
        ADD     DI,C1                   ;twice, attribute byte
        MOV     CX,C2
        SUB     CX,C1                   ;CX: number of columns to copy
        MOV     DX,L2
        SUB     DX,L1                   ;DX: number of rows to copy
        PUSH    DX
        MOV     AH,0H
        MOV     AL,PGN
        MOV     BX,1000H
        MUL     BX                      ;page offset
        ADD     DI,AX
        MOV     AX,0B800H               ;video buffer segment
        MOV     ES,AX
        MOV     AX,CX                   ;store column counter
        POP     DX
        PUSH    DS
        LDS     SI,DWORD PTR[BP+6]      ;source, input array
        CLD
```

```
        MOV     BX,DI                   ;store starting address
ROW:    REP     MOVSW                   ;copy one row
        ADD     BX,160D                 ;next row
        MOV     DI,BX                   ;new starting address
        MOV     CX,AX                   ;restore column counter
        DEC     DX
        JNZ     ROW                     ;all rows copied?
        POP     DS                      ;yes, return
        POP     BP
        RET     20d
PAGOUT  ENDP

;=========================================================================
;       COLOUR
;       Function: set the DOS display color
;       Access:   CALL COLOUR(COLRID)
;                 COLRID integer*2   holding the color number  ( 0 - 15 )
;=========================================================================
COLOUR  PROC    FAR
        PUSH    BP
        MOV     BP,SP
        LES     BX,DWORD PTR[BP+6]
        MOV     AL,ES:[BX]              ;get the input color id
        CMP     AL,0
        JB      EXIT                    ;id < 0,invalid
        CMP     AL,15
        JA      EXIT                    ;id >15,invalid
        PUSH    AX
        LEA     DX,STR00
        MOV     AH,09H
        INT     21H                     ;clear attribute
        POP     AX
        CMP     AL,8                    ;high intensity?
        JB      LOWC
        PUSH    AX
        LEA     DX,STR11                ;yes
        MOV     AH,09H
        INT     21H                     ;set high intensity
        POP     AX
        SUB     AL,8H                   ;find corresponding color
LOWC:   ADD     AL,30H                  ;ASCII code for the numeral
        LEA     DX,STR22
        MOV     BX,DX
        MOV     [BX+3],AL               ;encode AL into STR22
        MOV     AH,09H
        INT     21H                     ;set color
EXIT:   POP     BP
        RET     4H
COLOUR  ENDP

;=========================================================================
;       SAVCUR
;       Function: save indexed cursor position
;       Access:   CALL SAVCUR(ICUR)
;                 ICUR  integer*2   holding the index ( 0 - 25 )
```

```
;=========================================================================
SAVCUR  PROC    FAR
        PUSH    BP
        MOV     BP,SP
        MOV     AH,0FH
        INT     10H                     ;get the active page number in BH
        MOV     AH,03H
        INT     10H                     ;read cursor position in DH,DL
        LES     BX,DWORD PTR[BP+6]
        MOV     AX,ES:[BX]              ;get the calling index
        LEA     BX,OLDCUR               ;get offset of OLDCUR
        ADD     BX,AX
        ADD     BX,AX                   ;offset of the indexed cell
        MOV     [BX],DX                 ;save the cursor position
        POP     BP
        RET     4H
SAVCUR  ENDP

;=========================================================================
;       SETCUR
;       Function: restore indexed cursor position
;       Access:   CALL SETCUR(ICUR)
;                 ICUR  integer*2   holding the index ( 0 - 25 )
;=========================================================================
SETCUR  PROC    FAR
        PUSH    BP
        MOV     BP,SP
        LES     BX,DWORD PTR[BP+6]
        MOV     AX,ES:[BX]              ;get the calling index
        LEA     BX,OLDCUR               ;get offset of OLDCUR
        ADD     BX,AX
        ADD     BX,AX                   ;offset of the indexed cell
        MOV     DX,[BX]                 ;retrieve the cursor position
        MOV     AH,0FH
        INT     10H                     ;get the active page number in BH
        MOV     AH,02H
        INT     10H                     ;set the cursor to DH,DL
        POP     BP
        RET     4H
SETCUR  ENDP

;=========================================================================
;       STRIM
;       Function: trim a string of its trailing spaces
;       Access:   CALL STRIM(STRG,LENGTH)
;                 STRG  character*? string of any length
;                 LENGTH  integer*2   holding the length of STRG upon entry
;                                     returning with the trimed length
;=========================================================================
STRIM   PROC    FAR
        PUSH    BP
        MOV     BP,SP
        LES     BX,DWORD PTR[BP+6]
        MOV     CX,ES:[BX]              ;get the initial length of string
        LES     BX,DWORD PTR[BP+10]     ;get address of the string
```

```
        ADD     BX,CX                   ;let's search backwards
        DEC     BX                      ;starting from the end
STRM1:  CMP     BYTE PTR ES:[BX],' '    ;loop to skip the trailing spaces
        JNE     STRM2                   ;break the loop at the first
        DEC     BX                      ;non-space char
        LOOP    STRM1
STRM2:  LES     BX,DWORD PTR [BP+6]
        MOV     ES:[BX],CX              ;return the actual length
        POP     BP
        RET     08H
STRIM   ENDP

CODE    ENDS
END
```

```
c*********************************************************************

c       file name: IBMGRF.ASM

c       for program FETSCG (PC/AT version).
c       library of assembly language routines for graphics

c*********************************************************************


        PAGE    60,132
        TITLE   SUBROUTINE LINE for IBM PC/AT
        PUBLIC  LINE,MODE,WINDOW,PTCOLR,BOX,SETXOR,RSTXOR,CLRGRF,AXIS
;=====================================================================
;                               DATA SEGMENTS
;                              ---------------

;---------------------------------------------------------------------
DATA    SEGMENT PUBLIC  'DATA'
        X1      DW      ?               ;\
        Y1      DW      ?               ; \ x,y coordinate of first
        X2      DW      ?               ; / and second point
        Y2      DW      ?               ;/
        ESIX    DW      ?               ;storage for segment
                                        ;  address of array IX
        ESIY    DW      ?               ;storage for segment
                                        ;  address of array IY
        delta_X DW      ?               ;horizontal distance between
                                        ;  two points
        delta_Y DW      ?               ;vertical distance between
                                        ;  two points
        half_X  DW      ?               ;half of horizontal distance
        half_Y  DW      ?               ;half of vertical distance
        X_inc   DW      1               ;horizontal increment
        Y_inc   DW      1               ;vertical increment
        XMAX    DW      639d            ;maximum horizontal pixel no.
        XMIN    DW      0d              ;minimum horizontal pixel no.
        YMAX    DW      349d            ;maximum vertical pixel no.
        YMIN    DW      0d              ;minimum vertical pixel no.
        GCOLR   DB      7d              ;default color is white
        Cmode   DB      ?               ;store the current mode
        CXOR    DB      0               ;default color XOR off
DATA    ENDS
;---------------------------------------------------------------------
DGROUP  GROUP   DATA
;=====================================================================
;                               CODE SEGMENT
;                              --------------
CODE    SEGMENT 'CODE'
        assume  cs:CODE,ds:DGROUP,ss:DGROUP
;---------------------------------------------------------------------
;                         SUBROUTINE LINE
;                        -----------------
;
```

```
;  Purpose       : plot the curve specified by the input array, using
;                  Bresengam's algorithm
;                  if XOR is set on then the curve is erased
;  Input         : N...........  no. of points to be plotted
;                  IX..........  x-axis values of points
;                  IY..........  y-axis values pf points
;  Output        : none
;  Reg. modified: none
;  SUBs called   : SETLINE, PLOTPT
;  Sample call   : From MS_Fortran 3.3
;                  CALL LINE(N,IX,IY)
;                  N,IX,IY are integer*2 variables
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

LINE    PROC    FAR

;save all registers used
        push    ax
        push    bx
        push    cx
        push    dx
        push    si
        push    di
        push    es

;save BP, and set BP=SP
        push    bp
        mov     bp,sp

;get addresses point to the input parameters
        les     di,dword ptr [bp+28]    ;get address of N into ES:DI
        mov     cx,es:[di]              ;CX=N
        dec     cx                      ;N-1 segments between N points

        les     di,dword ptr [bp+24]    ;get starting address of
                                        ;  array IX into ES:DI
        mov     ESIX,es                 ;save ES of IX
        les     si,dword ptr [bp+20]    ;get starting address of
                                        ;  array IY into DS:SI
        mov     ESIY,es                 ;save EX of IY

;check 0 or 1 point
        cmp     cx,0ffffh               ;0 - 1 = ffffh
        je      escln                   ;if N = 0 then escape

        mov     ax,es:[si]              ;plot one point
        mov     es,ESIX
        mov     bx,es:[di]
        call    PLOTPT
        cmp     cx,0                    ;1 - 1 = 0
        je      escln                   ;if N = 1 then exit

;set loop to plot all points
repeat:
        mov     es,ESIY                 ;set ES=segment address of IY
```

```
        mov     ax,es:[si]              ;get Y1
        add     si,2d
        mov     bx,es:[si]              ;get Y2
        mov     Y1,ax                   ;store Y1
        mov     Y2,bx                   ;store Y2

        mov     es,ESIX                 ;set ES=segment address of IX
        mov     ax,es:[di]              ;get X1
        add     di,2d
        mov     bx,es:[di]              ;get X2
        mov     X1,ax                   ;store X1
        mov     X2,bx                   ;store X2

        call    SETLINE                 ;set a line between two points

        loop    repeat                  ;repeat until points exhausted

;restore all registers saved
escln:  pop     bp                      ;restore BP
        pop     es
        pop     di
        pop     si
        pop     dx
        pop     cx
        pop     bx
        pop     ax

        ret     12d

LINE    ENDP
;-----------------------------------------------------------------------_
;                        SUBROUTINE SETLINE
;                       --------------------
;
;  Purpose       : fill in points between two points
;  Input         : X1..........  x coordinate of the first point
;                  Y1..........  y coordinate of the first point
;                  X2..........  x coordinate of the second point
;                  Y2..........  y coordinate of the second point
;  Output        : none
;  Reg. modified: none
;  SUBs called   : ALONG_X
;                  ALONG_Y
;  Sample call   : From ASM86 Routines
;                  CALL SETLINE
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
SETLINE PROC    NEAR

;save registers used
        push    ax
        push    bx
        push    cx
        push    dx

;initialize X_inc, Y_inc
```

```
        mov     X_inc,1                 ;X_inc=+1
        mov     Y_inc,1                 ;Y_inc=+1

;calculate horizontal distance |X2-X1| between two adjacent points
        mov     ax,X2                   ;get X2
        sub     ax,X1                   ;X2-X1
        jge     save_X                  ;if pos., save the distance
        mov     X_inc,-1                ;else change the horizontal
                                        ;  increment to -1
        neg     ax                      ;take absolute value of X2-X1
save_X:
        mov     delta_X,ax              ;delta_X=|X2-X1|=AX
        mov     cx,ax                   ;calculate |X2-X1|/2
        shr     cx,1
        mov     half_X,cx               ;half_X=|X2-X1|/2

;calculate vertical distance |Y2-Y1| between two adjacent points
        mov     bx,Y2                   ;get Y2
        sub     bx,Y1                   ;Y2-Y1
        jge     save_Y                  ;if pos., save the distance
        mov     Y_inc,-1                ;else change the vertical
                                        ;  increment to -1
        neg     bx                      ;take absolute value of Y2-Y1
save_Y:
        mov     delta_Y,bx              ;delta_Y=|Y2-Y1|=BX
        mov     cx,bx                   ;calculate |Y2-Y1|/2
        shr     cx,1
        mov     half_Y,cx               ;half_Y=|Y2-Y1|/2

;sort |X2-X1| and |Y2-Y1|
        cmp     ax,bx                   ;compare |X2-X1| with |Y2-Y1|
        jl      steep                   ;if |Y2-Y1| > |X2-X1| increase
                                        ;  vertically
        call    ALONG_X                 ;else increase horizontally
        jmp     escset1                 ;exit the routine

steep:
        call    ALONG_Y                 ;increase along vertical axis

;restore all registers saved
escset1:
        pop     dx
        pop     cx
        pop     bx
        pop     ax

        ret

SETLINE ENDP
;-----------------------------------------------------------------------_
;                        SUBROUTINE PLOTPT
;                       -------------------_
;
;  Purpose      : plot a point according to given coordinate
;  Input        : AX..........  Y coordinate
```

```
;                 BX..........  X coordinate
;  Output       : none
;  Reg. modified: none
;  SUBs called  : none
;  BIOS called  : function 0Ch of int 10h
;  Sample call  : From ASM86 Routines
;                 CALL PLOTPT
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

PLOTPT  PROC    NEAR

;save registers used
        push    ax
        push    bx
        push    cx
        push    dx

;add window offset
        add     ax,Ymin
        add     bx,Xmin

;escape out of bound points
        cmp     ax,Ymax                 ;\
        ja      $escl                   ; \ check Y coordinate
        cmp     ax,Ymin                 ; /
        jb      $escl                   ;/
        cmp     bx,Xmax                 ;\
        ja      $escl                   ; \ check X coordinate
        cmp     bx,Xmin                 ; /
        jb      $escl                   ;/

;change origin from top left to bottom left
        neg     ax
        add     ax,349d

        mov     dx,ax                   ;DX = row no. of the pixel
        mov     cx,bx                   ;CX = column no. of the pixel
        mov     bh,0                    ;BH = page number
        mov     al,GCOLR                ;AL = color

        cmp     CXOR,0                  ;check if XOR is on
        je      xoroff                  ;skip if XOR is off
        or      al,80H                  ;set XOR

xoroff: mov     ah,0CH                  ;BIOS function of drawing a dot
        int     10h

;restore all registers saved
$escl:  pop     dx
        pop     cx
        pop     bx
        pop     ax

        ret
```

```
PLOTPT  ENDP
;-----------------------------------------------------------------------_
;                        SUBROUTINE ALONG_X
;                        ------------------_
;
;  Purpose       : implement Bresengam's algorithm along X axis
;  Input         : X1..........  x coordinate of the first point
;                  Y1..........  y coordinate of the first point
;                  delta_X.....  horizontal distance between two points
;                  delta_Y.....  vertical distance between two points
;                  half_X......  on half of the horizontal distance
;                  X_inc.......  increment along X axis
;                  Y_inc.......  increment along Y axis
;  Output        : none
;  Reg. modified: none
;  SUBs called   : PLOTPT
;  Sample call   : From ASM86 Routines
;                  CALL ALONG_X
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

ALONG_X PROC    NEAR

;save registers used
        push    ax
        push    bx
        push    cx
        push    dx

        mov     dx,0d                   ;initialize ERROR=0
        mov     ax,Y1                   ;AX=Y1 (y coord. of 1st point)
        mov     bx,X1                   ;BX=X1 (x coord. of 1st point)

        mov     cx,delta_X              ;number of points to be
                                        ;  plotted in CX

;plot all the points up to (X2,Y2) using Bresengam's algorithm
lineX:  add     dx,delta_Y              ;ERROR <-- ERROR + delta_Y
        cmp     dx,half_X               ;compare ERROR with half_X
        jle     smoothX                 ;if ERROR < half_X then
                                        ;  increase X only
        add     ax,Y_inc                ;else increase Y as well
        sub     dx,delta_X              ;ERROR <-- ERROR - delta_X

smoothX:
        add     bx,X_inc                ;increase X
        call    PLOTPT                  ;plot the point
        loop    lineX                   ;repeat

;restore all registers saved
        pop     dx
        pop     cx
        pop     bx
        pop     ax

        ret
```

```
ALONG_X ENDP
;-----------------------------------------------------------------------_
;                        SUBROUTINE ALONG_Y
;                        ------------------
;
;  Purpose       : implement Bresengam's algorithm along Y axis
;  Input         : X1..........  x coordinate of the first point
;                  Y1..........  y coordinate of the first point
;                  delta_X.....  horizontal distance between two points
;                  delta_Y.....  vertical distance between two points
;                  half_Y......  on half of the vertical distance
;                  X_inc.......  increment along X axis
;                  Y_inc.......  increment along Y axis
;  Output        : none
;  Reg. modified: none
;  SUBs called   : PLOTPT
;  Sample call   : From ASM86 Routines
;                  CALL ALONG_Y
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

ALONG_Y PROC    NEAR

;save registers used
        push    ax
        push    bx
        push    cx
        push    dx

        mov     dx,0d                   ;initialize ERROR = 0
        mov     ax,Y1                   ;AX=Y1 (Y coord. of 1st point)
        mov     bx,X1                   ;BX=X1 (X coord. of 1st point)

        mov     cx,delta_Y              ;CX=no. of points to plot

;plot all the points up to (X2,Y2) using Bresengam's algorithm
lineY:  add     dx,delta_X              ;ERROR <-- ERROR + delta_X
        cmp     dx,half_Y               ;compare ERROR with half_Y
        jle     smoothY                 ;if ERROR < half_Y then
                                        ;  increase Y only
        add     bx,X_inc                ;else increase X as well
        sub     dx,delta_Y              ;ERROR <-- ERROR - delta_Y
smoothY:
        add     ax,Y_inc                ;increase Y
        call    PLOTPT                  ;plot the point
        loop    lineY                   ;repeat

;restore all registers saved
        pop     dx
        pop     cx
        pop     bx
        pop     ax

        ret
```

```
ALONG_Y ENDP
;-----------------------------------------------------------------------_
;                          SUBROUTINE MODE
;                         ----------------_
;
;  Purpose       : set the video mode on IBM PC
;  Input         : newmod......  new video mode to be set
;  output        : curmod......  current video mode
;  Reg. modified: none
;  SUBs called   : none
;  Sample call   : From MS FORTRAN 3.3
;                  CALL MODE(CURMOD,NEWMOD)
;                  CURMOD,NEWMOD are integer*2 variables
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

MODE    PROC    FAR

;save registers used
        push    ax
        push    bx
        push    es
        push    bp                      ;save BP

        mov     ah,0fh                  ;BIOS function 0Fh of
        int     10h                     ;int 10h to get current mode

        mov     bp,sp                   ;set BP = SP
        les     bx,[bp+16]              ;get the address of CURMOD
                                        ;  in DS:BX
        mov     es:[bx],al              ;return the current video mode
                                        ;  to FORTRAN

        les     bx,[bp+12]              ;get the address of NEWMOD
                                        ;  in DS:BX
        mov     al,es:[bx]              ;AL = video mode to be set
        mov     Cmode,al                ;save the NEW mode
        mov     ah,0h                   ;BIOS function 0h of int 10h
        int     10h                     ;to set new video mode

;restore all registers saved
        pop     bp
        pop     es
        pop     bx
        pop     ax

        ret     8d

MODE    ENDP
;-----------------------------------------------------------------------_
;                          SUBROUTINE WINDOW
;                         -------------------_
;
;  Purpose       : set the physical window for graghics
;  Input         : W...........  window size
;                                W(1) = Xmin
```

```
;                                W(2) = Xmax
;                                W(3) = Ymin
;                                W(4) = Ymax
;  Output        : none
;  Reg. modified: none
;  SUBs called   : none
;  Sample call   : From MS FORTRAN 3.3
;                  CALL WINDOW(W)
;                  W is array of integer*2 with 4 elements
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

WINDOW  PROC    FAR

;save registers used
        push    ax
        push    bx
        push    es
        push    bp

        mov     bp,sp                   ;set BP = SP

        les     bx,dword ptr [bp+12]    ;get the address of
                                        ;  input variable in DS:BX
        mov     ax,word ptr es:[bx]     ;get first element
        mov     Xmin,ax                 ;update Xmin
        mov     ax,word ptr es:[bx+2]   ;get second element
        mov     Xmax,ax                 ;update Xmax
        mov     ax,word ptr es:[bx+4]   ;get third element
        mov     Ymin,ax                 ;update Ymin
        mov     ax,word ptr es:[bx+6]   ;get forth element
        mov     Ymax,ax                 ;update Ymax

;restore all registers saved
        pop     bp
        pop     es
        pop     bx
        pop     ax

        ret     4d

WINDOW  ENDP
;-----------------------------------------------------------------------
;                        SUBROUTINE PTCOLR
;                       ------------------
;
;  Purpose       : set the pixel color for draw line
;  Input         : Col.........  color, one of  0 -- black
;                                                1 -- blue
;                                                2 -- green
;                                                3 -- cyan
;                                                4 -- red
;                                                5 -- magenta
;                                                6 -- brown
;                                                7 -- white
;                                                8 -- gray
```

```
;                                                   9 -- light blue
;                                                   10-- light green
;                                                   11-- light cyan
;                                                   12-- pink
;                                                   13-- light Magenta
;                                                   14-- yellow
;                                                   15-- bright white
;  Output        : none
;  Reg. modified: none
;  SUBs called   : none
;  Sample call   : From MS FORTRAN 3.3
;                  CALL PTCOLR(Col) or CALL PTCOLR(2)
;                  Col is a integer*2 variable
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

PTCOLR  PROC    FAR

;save registers used
        push    ax
        push    es
        push    bp

        mov     bp,sp                   ;set BP = SP

        les     di,dword ptr [bp+10]    ;get address of the input
                                        ;  variable in ES:DI
        mov     al,byte ptr es:[di]     ;get the input value in AL
        cmp     al,0fh                  ;check if color is in range
        ja      $outc                   ;no change in color
                                        ;  if out of range
        mov     GCOLR,al                ;update Colour

;restore all registers saved
$outc:  pop     bp
        pop     es
        pop     ax

        ret     4d

PTCOLR  ENDP
;-----------------------------------------------------------------------_
;                        SUBROUTINE BOX
;                       ----------------_
;
;  Purpose       : draw a box frame according to the given size
;  Input         : B...........  box size
;                                B(1) = Xmin
;                                B(2) = Xmax
;                                B(3) = Ymin
;                                B(4) = Ymax
;                  TYPE........  line type of the box
;                                only solid line implemented here
;  output        : none
;  Reg. modified: none
;  SUBs called   : SETLINE
```

```
;  Sample call  : From MS FORTRAN 3.3
;                 CALL BOX(B,TYPE)
;                 B is an array of 4 integer*2 variables,
;                 TYPE is a character*1 variable
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

BOX     PROC    FAR

;save registers used
        push    ax
        push    bx
        push    es
        push    bp                      ;save BP
        mov     bp,sp                   ;set BP = SP

        les     bx,dword ptr [bp+16]    ;get the starting address
                                        ;  of array B into ES:BX

;draw line from (Xmin,Ymin) - (Xmax,Ymin)
        mov     ax,es:[bx]              ;X1 = Xmin
        mov     X1,ax
        mov     ax,es:[bx+2]            ;X2 = Xmax
        mov     X2,ax
        mov     ax,es:[bx+4]            ;Y1 = Ymin
        mov     Y1,ax
        mov     ax,es:[bx+4]            ;Y2 = Ymin
        mov     Y2,ax
        call    SETLINE

;draw line from (Xmax,Ymin) - (Xmax,Ymax)
        mov     ax,es:[bx+2]            ;X1 = Xmax
        mov     X1,ax
        mov     ax,es:[bx+2]            ;X2 = Xmax
        mov     X2,ax
        mov     ax,es:[bx+4]            ;Y1 = Ymin
        mov     Y1,ax
        mov     ax,es:[bx+6]            ;Y2 = Ymax
        mov     Y2,ax
        call    SETLINE

;draw line from (Xmax,Ymax) - (Xmin,Ymax)
        mov     ax,es:[bx+2]            ;X1 = Xmax
        mov     X1,ax
        mov     ax,es:[bx]              ;X2 = Xmin
        mov     X2,ax
        mov     ax,es:[bx+6]            ;Y1 = Ymax
        mov     Y1,ax
        mov     ax,es:[bx+6]            ;Y2 = Ymax
        mov     Y2,ax
        call    SETLINE

;draw line from (Xmin,Ymax) - (Xmin,Ymin)
        mov     ax,es:[bx]              ;X1 = Xmin
        mov     X1,ax
        mov     ax,es:[bx]              ;X2 = Xmin
```

```
        mov     X2,ax
        mov     ax,es:[bx+6]            ;Y1 = Ymax
        mov     Y1,ax
        mov     ax,es:[bx+4]            ;Y2 = Ymin
        mov     Y2,ax
        call    SETLINE

;restore all registers saved
        pop     bp
        pop     es
        pop     bx
        pop     ax

        ret     8d
BOX     ENDP
;-----------------------------------------------------------------------_
;                        SUBROUTINE CLRGRF
;                       ------------------_
;
;  Purpose       : clear the screen using scrolling
;  Input         : none
;  output        : none
;  Reg. modified: none
;  SUBs called   : none
;  Sample call   : From MS FORTRAN 3.3
;                  CALL CLRGRF
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

CLRGRF  PROC    FAR

;save registers used
        push    ax
        push    bx
        push    cx
        push    dx

;scrolling the screen up
        mov     bh,0h                   ;set color to black
        mov     cx,0h                   ;set the frame to be scrolled
                                        ;  from (0,0)
        mov     dh,24d                  ;to (80,25)
        mov     dl,79d
        mov     ah,6h                   ;AH = 6 scroll up
        mov     al,0h                   ;AL = 0 to scroll the
                                        ;  whole screen
        int     10h

;restore registers saved
        pop     dx
        pop     cx
        pop     bx
        pop     ax

        RET
CLRGRF  ENDP
```

```
;-----------------------------------------------------------------------
;                        SUBROUTINE SETXOR
;                        -----------------
;
;  Purpose       : turn the color XOR on
;  Input         : none
;  output        : none
;  Reg. modified: none
;  SUBs called   : none
;  Sample call   : From MS FORTRAN 3.3
;                  CALL SETXOR
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

SETXOR  PROC    FAR

        mov     CXOR,1                  ;set color XOR

        ret
SETXOR  ENDP
;-----------------------------------------------------------------------
;                        SUBROUTINE RSTXOR
;                        -----------------
;
;  Purpose       : turn the color XOR off
;  Input         : none
;  output        : none
;  Reg. modified: none
;  SUBs called   : none
;  Sample call   : From MS FORTRAN 3.3
;                  CALL RSTXOR
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

RSTXOR  PROC    FAR

        mov     CXOR,0                  ;clear color XOR

        ret
RSTXOR  ENDP
;-----------------------------------------------------------------------
;                        SUBROUTINE AXIS
;                        ---------------
;
;  Purpose       : draw X,Y axis
;  Input         : AXtype....... type of axis
;                                 'X' -- X_axis
;                                 'Y' -- Y_axis
;                  MIN.......... starting point of the axis
;                  MAX.......... terminating point of the axis
;                  OTH.......... coordinate value of the other axis
;                  TIC.......... distance between each tics
;  output        : none
;  Reg. modified: none
;  SUBs called   : X_AXIS
;                  Y_AXIS
;  Sample call   : From MS FORTRAN 3.3
```

```
;                 CALL AXIS(AXtype,MIN,MAX,OTH,TIC)
;                 AXtype is a character*1 variable
;                 MIN,MAX,OTH,TIC are integer*2 variables
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

AXIS    PROC    FAR

;save registers used
        push    ax
        push    bx

        push    bp                      ;save BP
        mov     bp,sp                   ;set BP = SP

        les     bx,dword ptr [bp+26]    ;get address of the
                                        ;  AXtype into ES:BX
        mov     al,es:[bx]              ;get the AXtype
        cmp     al,'X'                  ;check if X_axis
        jne     $Y                      ;if not X_axis draw Y_axis
        call    X_AXIS                  ;else draw X_axis
        jmp     esc_axis                ;exit

$Y:     call    Y_AXIS                  ;draw Y_axis

esc_axis:
;restore registers saved
        pop     bp
        pop     bx
        pop     ax

        RET     20d
AXIS    ENDP
;-----------------------------------------------------------------------
;                       SUBROUTINE X_AXIS
;                      ------------------
;
;  Purpose      : draw X axis
;  Input        : MIN........... starting point of the axis
;                 MAX........... terminating point of the axis
;                 OTH........... coordinate value of the other axis
;                 TIC........... distance between each tics
;  output       : none
;  Reg. modified: none
;  SUBs called  : none
;  Sample call  : From ASM86
;                 CALL X_AXIS
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

X_AXIS  PROC    NEAR

;save registers used
        push    ax
        push    bx
        push    cx
        push    dx
```

```
;get input parameters, and draw the axis line
        les     bx,dword ptr [bp+22]    ;get address of MIN
                                        ;  into ES:BX
        mov     ax,es:[bx]              ;get MIN
        mov     X1,ax                   ;set X1 = MIN
        les     bx,dword ptr [bp+18]    ;get address of MAX
                                        ;  into ES:BX
        mov     ax,es:[bx]              ;get MAX
        mov     X2,ax                   ;set X2 = MAX
        les     bx,dword ptr [bp+14]    ;get address of OTH
                                        ;  into ES:BX
        mov     ax,es:[bx]              ;get OTH
        mov     Y1,ax                   ;set Y1 = Y2 = OTH
        mov     Y2,ax
        call    setline                 ;draw the line

;calculate tic length
        mov     ax,Ymax                 ;AX = Ymax - Ymin
        sub     ax,Ymin
        inc     ax
        mov     cl,6d                   ;set counter for shifting
        shr     ax,cl                   ;tic length = 1.5%
                                        ;  of window size
        cmp     ax,3                    ;if tic length is less
                                        ;  than minimum
        ja      $TL                     ;if not saved it
        mov     ax,3                    ;else set minimum tic
                                        ;  length be 3
$TL:    mov     dx,ax                   ;save tic length in DX

;calculate number of tics to be drawn
        mov     ax,X2                   ;AX = X2 - X1
        sub     ax,X1
        les     bx,dword ptr [bp+10]    ;get address of TIC
                                        ;  into ES:BX
        mov     cx,es:[bx]              ;CL = TIC
        div     cl                      ;AX / CL
        cbw                             ;AX = AL
        xchg    cx,ax                   ;CX = no. of tics
        inc     cx

;draw tics
        mov     bx,Y1                   ;BX = Y1
        add     bx,dx                   ;Y2 = Y1 + tic length
        mov     Y2,bx
        mov     bx,X1                   ;BX = X1
X_tic:  mov     X2,bx                   ;X2 = X1
        call    setline                 ;draw a tic
        add     bx,ax                   ;X1 = X2 = X1 + TIC
                                        ;  (distance between tics)
        mov     X1,bx
        loop    X_tic                   ;repeat until all tics
                                        ;  have been drawn
```

```
;restore all registers saved
        pop     dx
        pop     cx
        pop     bx
        pop     ax

        RET
X_AXIS  ENDP
;-----------------------------------------------------------------------
;                        SUBROUTINE Y_AXIS
;                       ------------------
;
;  Purpose       : draw Y axis
;  Input         : MIN........... starting point of the axis
;                  MAX........... terminating point of the axis
;                  OTH........... coordinate value of the other axis
;                  TIC........... distance between each tics
;  output        : none
;  Reg. modified: none
;  SUBs called   : none
;  Sample call   : From ASM86
;                  CALL Y_AXIS
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Y_AXIS  PROC    NEAR

;save registers used
        push    ax
        push    bx
        push    cx
        push    dx

;get input parameters, and draw the axis line
        les     bx,dword ptr [bp+22]    ;get address of MIN
                                        ;  into ES:BX
        mov     ax,es:[bx]              ;get MIN
        mov     Y1,ax                   ;set Y1 = MIN
        les     bx,dword ptr [bp+18]    ;get address of MAX
                                        ;  into ES:BX
        mov     ax,es:[bx]              ;get MAX
        mov     Y2,ax                   ;set Y2 = MAX
        les     bx,dword ptr [bp+14]    ;get address of OTH
                                        ;  into ES:BX
        mov     ax,es:[bx]              ;get OTH
        mov     X1,ax                   ;set X1 = X2 = OTH
        mov     X2,ax
        call    setline                 ;draw the line

;calculate tic length
        mov     ax,Xmax                 ;AX = Xmax - Xmin
        sub     ax,Xmin
        inc     ax
        mov     cl,7d                   ;set counter for shifting
        shr     ax,cl                   ;tic length = 1.5%
                                        ;  of window size
```

```
        cmp     ax,3                    ;if tic length is
                                        ;  less than minimum
        ja      $TLL                    ;if not save the tic length
        mov     ax,3                    ;else set minimum
                                        ;  tic length be 3
$TLL:   mov     dx,ax                   ;save tic length in DX

;calculate number of tics to be drawn
        mov     ax,Y2                   ;AX = Y2 - Y1
        sub     ax,Y1
        les     bx,dword ptr [bp+10]    ;get address of TIC
                                        ;  into ES:BX
        mov     cx,es:[bx]              ;CL = TIC
        div     cl                      ;AX / CL
        cbw                             ;AX = AL
        xchg    cx,ax                   ;CX = no. of tics
        inc     cx

;draw tics
        mov     bx,X1                   ;BX = X1
        add     bx,dx                   ;X2 = X1 + tic length
        mov     X2,bx
        mov     bx,Y1                   ;BX = Y1

Y_tic:  mov     Y2,bx                   ;Y2 = Y1
        call    setline                 ;draw a tic
        add     bx,ax                   ;Y1 = Y2 = Y1 + TIC
                                        ;  (distance between tics)
        mov     Y1,bx
        loop    Y_tic                   ;repeat until all tics
                                        ;  have been drawn

;restore all registers saved
        pop     dx
        pop     cx
        pop     bx
        pop     ax

        RET
Y_AXIS  ENDP
;-----------------------------------------------------------------------
CODE    ENDS
;=======================================================================
        END
```