Appendix D
EIT Data Acquisition Subroutines

This appendix lists Microsoft® QuickBasic™ subroutines called by FASTEIT.BAS and SLOWEIT.BAS. These subroutines perform statistical checks on electrode voltages, display symmetry and antisymmetry checks for cases where injection and ground electrodes are directly opposite each other, and save the measured voltages in a variety of output formats. The file format used by EIT reconstruction codes is described in Appendix E; other output formats include an ASCII file readable by Microsoft Excel and a Postscript file which prints the normalized electrodes voltages in a graphic "clock face" format.

```
DECLARE SUB datmake (elec%, carr!(), quad!(), volt!(), file$)
DECLARE SUB symmake (elec%, symstd!, sym!(), asymstd!, asym!(), file$)
DECLARE SUB difmake (elec%, carr!(), quad!(), volt!())
DECLARE SUB xlsmake (elec%, volt!())
DECLARE SUB linmake (elec%, std.L!, volt.L!(), file$)
DECLARE SUB psmake (elec%, carr!(), quad!(), volt!())
DECLARE SUB promake (elec%, proj%, Vsum!(), czero!, qzero!, CEVmean!, CEVmax!,
       CEVmin!, CEVstd!, Vmean!, Vmax!, Vmin!, Vstd!, file$)

DEFINT I-N
DEFLNG O-Z



SUB datmake (elec%, carr!(), quad!(), volt!(), file$)

PRINT
PRINT "Input file name ('.dat' is appended automatically) or "
INPUT "type 'ls' to see a listing of current files: ", file$
IF file$ = "ls" OR file$ = "LS" THEN
  FILES "d:\data\*.dat"
  INPUT "Input file name ('.dat' is appended automatically): ", file$
END IF

OPEN "d:\data\" + file$ + ".dat" FOR OUTPUT AS #1
FOR iv1 = 1 TO elec% - 1
  FOR iv2 = iv1 + 1 TO elec%
    FOR iv3 = 1 TO elec%
      PRINT #1, USING "## ## ## ####.## ####.## ####.##"; iv1; iv2; iv3; volt!(iv1,
       iv2, iv3); carr!(iv1, iv2, iv3); quad!(iv1, iv2, iv3)
    NEXT iv3
  NEXT iv2
NEXT iv1
CLOSE #1

PRINT "Data saved to file d:\data\"; file$; ".dat"
```

```
END SUB


SUB difmake (elec%, carr!(), quad!(), volt!())

PRINT
PRINT "Use a reference file that involves the same number of electrodes as"
PRINT "the current data."
INPUT "Input the reference file ('.dat' is appended automatically): ", file$
OPEN "d:\data\" + file$ + ".dat" FOR INPUT AS #3

PRINT
INPUT "Input difference file name ('.dif' is appended automatically): ", file$
OPEN "d:\data\" + file$ + ".dif" FOR OUTPUT AS #1

' Take difference of data values

FOR iv1 = 1 TO elec% - 1
  FOR iv2 = iv1 + 1 TO elec%
    FOR iv3 = 1 TO elec%
      INPUT #3, i1temp%, i2temp%, i3temp%, vtemp!, ctemp!, qtemp!

      IF i3temp% <> iv3 THEN PRINT "Discrepancy in electrode count between files!"

      ctemp! = carr!(iv1, iv2, iv3) - ctemp!
      qtemp! = quad!(iv1, iv2, iv3) - qtemp!
      vtemp! = volt!(iv1, iv2, iv3) - vtemp!

      PRINT #1, USING "## ## ## #####.## #####.## #####.##"; iv1; iv2; iv3; vtemp!;
       ctemp!; qtemp!

    NEXT iv3
  NEXT iv2
NEXT iv1

CLOSE #3
CLOSE #1

PRINT "Difference file saved as d:\data\"; file$; ".dif"

END SUB


SUB eistats (elec%, proj%, slength%, carrsum&(), quadsum&(), Vsum!(), resp$)

DIM Vsum.elc!(1 TO elec%)         ' averages of Vsum!() for each electrode
DIM Vsum2.elc!(1 TO elec%)        ' standard deviations of Vsum!() for each
                                  '   electrode

'*** diagnostic to print free memory
'PRINT "Free memory in bytes = "; FRE(-1)

DIM carr!(1 TO (elec% - 1), 2 TO elec%, 1 TO elec%) 'averages of voltages saved
DIM quad!(1 TO (elec% - 1), 2 TO elec%, 1 TO elec%) '  in real arrays
DIM volt!(1 TO (elec% - 1), 2 TO elec%, 1 TO elec%)

DIM sym!(1 TO elec% / 2, 1 TO 2, 1 TO (elec% / 2 - 1)) 'symmetry data
DIM save!(1 TO elec%)                   'averages of symmetry data
DIM ssum!(1 TO (elec% - 1))             'sums of symmetry data
DIM ssum2!(1 TO (elec% - 1))            'sum of symmetry data squared
DIM asym!(1 TO elec% / 2, 1 TO 4, 1 TO elec% / 4) 'antisymmetry data
DIM asum!(1 TO elec% / 4)               'sums of antisymmetry data
DIM asum2!(1 TO elec% / 4)              'sum of antisymmetry data squared

DIM volt.L!(1 TO (elec% / 2 - 1), 2 TO elec% - 1, 1 TO elec%) 'linearity check arrays
DIM sum2.L!(1 TO (elec% / 2 - 1), 2 TO elec% - 1)

' Subtract off baseline voltage from projections and average;
'   since the ground electrode is used as a voltage reference,
'   carrsum(i,n,n) and quadsum(i,n,n) should equal zero for any
'   i and n (in theory)

  carrzero& = 0: quadzero& = 0

  FOR iv1 = 1 TO elec% - 1
    FOR iv2 = iv1 + 1 TO elec%

      carrzero& = carrzero& + carrsum&(iv1, iv2, iv2)
      quadzero& = quadzero& + quadsum&(iv1, iv2, iv2)

    NEXT iv2
  NEXT iv1

  nz! = (elec%) * (elec% - 1) / 2
  fact! = 1! / CSNG(proj% * slength%)
  czero! = CSNG(carrzero&) * (fact! / nz!)
  qzero! = CSNG(quadzero&) * (fact! / nz!)

  FOR iv1 = 1 TO elec% - 1
    FOR iv2 = iv1 + 1 TO elec%
      FOR iv3 = 1 TO elec%

        carr!(iv1, iv2, iv3) = fact! * CSNG(carrsum&(iv1, iv2, iv3)) - czero!
        quad!(iv1, iv2, iv3) = fact! * CSNG(quadsum&(iv1, iv2, iv3)) - qzero!
        volt!(iv1, iv2, iv3) = SQR(carr!(iv1, iv2, iv3) ^ 2 + quad!(iv1, iv2, iv3) ^
       2)

'        PRINT
'        PRINT "carr, quad, volt   "; carr!(iv1, iv2, iv3), quad!(iv1, iv2, iv3),
       volt!(iv1, iv2, iv3)
'        PRINT


      NEXT iv3
    NEXT iv2
  NEXT iv1

'************* CALCULATE STATISTICS AND OUTPUT TO SCREEN *************

' Calculate mean and extremes of voltages recorded for each projection
'   for 180 degree current injection and ground cases. Also, calculate
'   standard deviation between projections. NOTE: CEV stands for cross
'   electrode voltage and is the voltage between the current and the ground.
'   The other statistics are for the remaining voltages.

  FOR iv3 = 1 TO elec%
    Vsum.elc!(iv3) = 0!
    Vsum2.elc!(iv3) = 0!
  NEXT iv3

  CEVmean! = 0!: CEVmin! = 100000!: CEVmax! = 0!: CEVstd! = 0!
  Vmean! = 0!: Vmin! = 100000!: Vmax! = 0!: Vstd! = 0!

  FOR i = 1 TO proj%
```

```
    IF Vsum!(i, 1) > CEVmax! THEN CEVmax! = Vsum!(i, 1)
    IF Vsum!(i, 1) < CEVmin! THEN CEVmin! = Vsum!(i, 1)
    CEVmean! = CEVmean! + Vsum!(i, 1)
    CEVstd! = CEVstd! + Vsum!(i, 1) ^ 2

    FOR iv3 = 2 TO elec%
      IF iv3 <> (elec% / 2 + 1) THEN
        IF Vsum!(i, iv3) > Vmax! THEN Vmax! = Vsum!(i, iv3)
        IF Vsum!(i, iv3) < Vmin! THEN Vmin! = Vsum!(i, iv3)
        Vsum.elc!(iv3) = Vsum.elc!(iv3) + Vsum!(i, iv3)
        Vsum2.elc!(iv3) = Vsum2.elc!(iv3) + Vsum!(i, iv3) ^ 2
      END IF
    NEXT iv3

  NEXT i

  IF proj% > 1 THEN
    temp! = CSNG(proj%) * CEVstd! - CEVmean! ^ 2
    IF temp! < 0 THEN
      PRINT "ERROR computing standard deviation in CEV"
      PRINT "N sum(Xi^2) - (sum(Xi))^2 = "; temp!
      INPUT "Hit <return> to continue. ", resp$
      CEVstd! = 0!
    ELSE
      CEVstd! = SQR(temp! / CSNG(proj% * (proj% - 1)))
    END IF
    CEVmean! = CEVmean! / CSNG(proj%)
    FOR iv3 = 2 TO elec%
      IF iv3 <> (elec% / 2 + 1) THEN
        temp! = CSNG(proj%) * Vsum2.elc!(iv3) - Vsum.elc!(iv3) ^ 2
        IF temp! < 0 THEN
          PRINT "ERROR computing standard deviation in voltages"
          PRINT "N sum(Xi^2) - (sum(Xi))^2 = "; temp!
          INPUT "Hit <return> to continue. ", resp$
        ELSE
          Vstd! = Vstd! + SQR(temp! / CSNG(proj% * (proj% - 1))) / CSNG(elec% - 2)
        END IF
        Vmean! = Vmean! + Vsum.elc!(iv3) / CSNG((elec% - 2) * proj%)
      END IF
    NEXT iv3
  ELSE
    CEVstd! = 0!
    Vstd! = 0!
    FOR iv3 = 2 TO elec%
      IF iv3 <> (elec% / 2 + 1) THEN
        Vmean! = Vmean! + Vsum.elc!(iv3) / CSNG(elec% - 2)
      END IF
    NEXT iv3
  END IF

' Print data to screen and file

  CLS
  PRINT " Average Data Set Characteristics"
  PRINT
  PRINT USING " Carrier Zero Offset            #####"; czero!
  PRINT USING " Quad Zero Offset               #####"; qzero!
  PRINT
  PRINT " Statistics for cross electrode voltages comparing data"
  PRINT " recorded for each projection:"
  PRINT
  PRINT USING " MEAN Cross Electrode Voltage   #####.##"; CEVmean!
  PRINT USING " MAX Cross Electrode Voltage    #####.##"; CEVmax!
  PRINT USING " MIN Cross Electrode Voltage    #####.##"; CEVmin!
  PRINT USING " STDEV Cross Electrode Voltage   ####.##"; CEVstd!
  PRINT
  PRINT " Statistics for electrode voltages (not including current injection"
  PRINT " and ground electrodes) for 180 degree injection/ground cases"
  PRINT " comparing data recorded for each projection"
  PRINT
  PRINT USING " MEAN Electrode Voltage         #####.##"; Vmean!
  PRINT USING " MAX Electrode Voltage          #####.##"; Vmax!
  PRINT USING " MIN Electrode Voltage          #####.##"; Vmin!
  PRINT USING " STDEV Electrode Voltage         ####.##"; Vstd!
  PRINT
  INPUT " Hit <return> to continue. ", resp$

'*********** CHECK SYMMETRY AND ANTISYMMETRY ************************

  FOR iv1 = 1 TO elec% / 2
    iv2 = iv1 + elec% / 2
    save!(iv1) = 0!
    FOR k = 1 TO (elec% / 2 - 1)
      k1 = iv1 + k
      k2 = iv1 - k
      IF k2 < 1 THEN k2 = elec% - (k - iv1)
      sym!(iv1, 1, k) = volt!(iv1, iv2, k1)
      sym!(iv1, 2, k) = volt!(iv1, iv2, k2)
      save!(iv1) = save!(iv1) + (sym!(iv1, 1, k) + sym!(iv1, 2, k)) / CSNG(elec% - 2)
    NEXT k
  NEXT iv1

  FOR k = 1 TO (elec% / 2 - 1)
    ssum!(k) = 0!
    ssum2!(k) = 0!
  NEXT k
  smax! = 0!: smin! = 10000!
  FOR iv1 = 1 TO elec% / 2
    FOR j = 1 TO 2
      FOR k = 1 TO (elec% / 2 - 1)
        sym!(iv1, j, k) = sym!(iv1, j, k) - save!(iv1)
        IF sym!(iv1, j, k) > smax! THEN smax! = sym!(iv1, j, k)
        IF sym!(iv1, j, k) < smin! THEN smin! = sym!(iv1, j, k)
        ssum!(k) = ssum!(k) + sym!(iv1, j, k)
        ssum2!(k) = ssum2!(k) + sym!(iv1, j, k) ^ 2
      NEXT k
    NEXT j
  NEXT iv1

  symstd! = 0!
  FOR k = 1 TO (elec% / 2 - 1)
    temp! = CSNG(elec%) * ssum2!(k) - ssum!(k) ^ 2
    IF temp! < 0 THEN
      PRINT "ERROR computing standard deviation in symmetry data"
      PRINT "N sum(Xi^2) - (sum(Xi))^2 = "; temp!
      INPUT "Hit <return> to continue. ", resp$
    ELSE
      symstd! = symstd! + SQR(temp! / CSNG(elec% * (elec% - 1))) / CSNG(elec% / 2 - 1)
    END IF
  NEXT k

  FOR k = 1 TO elec% / 4
    asum!(k) = 0!
    asum2!(k) = 0!
  NEXT k
  amin! = 10000!: amax! = 0!
  FOR iv1 = 1 TO elec% / 2
    iv2 = iv1 + elec% / 2
    n1 = (iv1 + iv2) / 2                  'antisymmetry nodes
    IF n1 <= (elec% / 2) THEN
      n2 = n1 + elec% / 2
      C! = 1!
    ELSE
      ntemp = n1
      n1 = n1 - elec% / 2
      n2 = ntemp
      C! = -1!
    END IF
    FOR k = 1 TO elec% / 4
      k1 = n1 - (k - 1)
      IF k1 < 1 THEN k1 = elec% + n1 - (k - 1)
      k2 = n1 + (k - 1)
      k3 = n2 - (k - 1)
      k4 = n2 + (k - 1)
      IF k4 > elec% THEN k4 = (k - 1) - (elec% - n2)
      asym!(iv1, 1, k) = C! * (volt!(iv1, iv2, k1) - save!(iv1))
      asym!(iv1, 2, k) = C! * (save!(iv1) - volt!(iv1, iv2, k2))
      asym!(iv1, 3, k) = C! * (save!(iv1) - volt!(iv1, iv2, k3))
      asym!(iv1, 4, k) = C! * (volt!(iv1, iv2, k4) - save!(iv1))
      FOR j = 1 TO 4
        IF asym!(iv1, j, k) > amax! THEN amax! = asym!(iv1, j, k)
        IF asym!(iv1, j, k) < amin! THEN amin! = asym!(iv1, j, k)
        asum!(k) = asum!(k) + asym!(iv1, j, k)
        asum2!(k) = asum2!(k) + asym!(iv1, j, k) ^ 2
      NEXT j
    NEXT k
  NEXT iv1

  asymstd! = 0!
  FOR k = 1 TO (elec% / 4)
    temp! = CSNG(2 * elec%) * asum2!(k) - asum!(k) ^ 2
    IF temp! < 0 THEN
      PRINT "ERROR computing standard deviation in axisymmetry data"
      PRINT "N sum(Xi^2) - (sum(Xi))^2 = "; temp!
      INPUT "Hit <return> to continue. ", resp$
    ELSE
      asymstd! = asymstd! + SQR(temp! / CSNG((2 * elec%) * (2 * elec% - 1))) /
       CSNG(elec% / 4)
    END IF
  NEXT k

' Display symmetry data to screen

  CLS

  SCREEN 12                       'set screen to 640x480 resolution
  dt.p! = 5!                      'length of tick marks in pixels
  view.p! = 240!                  'viewport length and width in pixels

  x1.p% = (640! - 2! * view.p!) / 3! - 1
  y1.p% = (480! - view.p!) / 2! - 1
  x2.p% = 2 * x1.p% + view.p!
  y2.p% = y1.p%

'****************** SYMMETRY PLOT ********************

' Find limits for y axis
```

```
 ymax! = 10 * CINT((smax! + 10) / 10)
 ymin! = 10 * CINT((smin! - 10) / 10)
 n = CINT((ymax! - ymin!) / 10)
 dy! = (ymax! - ymin!) / n

 VIEW (x1.p%, y1.p%)-(x1.p% + view.p!, y1.p% + view.p!)
 WINDOW (0, ymin!)-(elec% / 2, ymax!)

 LINE (0, ymin!)-(elec% / 2, ymax!), , B        'draw box around graph

 dt.x! = (dt.p! / view.p!) * (ymax! - ymin!)    'calculate length of x-ticks
 FOR i = 1 TO (elec% / 2 - 1)                   'draw x ticks
   LINE (i, ymin!)-(i, ymin! + dt.x!)
 NEXT i

 dt.y! = (dt.p! / view.p!) * CSNG(elec% / 2)    'calculate length of y-ticks
 FOR i = 1 TO n - 1                             'draw y ticks
   y! = ymin! + CSNG(i) * dy!
   LINE (0, y!)-(0 + dt.y!, y!)
 NEXT i

 FOR i = 1 TO elec% / 2
   FOR j = 1 TO 2
     PSET (1, sym!(i, j, 1)), i                 'plot first point
     FOR k = 2 TO (elec% / 2 - 1)
       LINE -(k, sym!(i, j, k)), i              'join remaining points
     NEXT k
   NEXT j
 NEXT i

 COL% = (x1.p% / 640!) * 80 + 1
 ROW% = (y1.p% / 480!) * 30 - 1
 LOCATE ROW%, COL%
 PRINT "SYMMETRY PLOT"
 LOCATE ROW% + 1, COL%
 PRINT USING "(####.## STANDARD DEVIATION)"; symstd!

 COL% = (x1.p% / 640!) * 80 - 5
 ROW% = (y1.p% / 480!) * 30 + 1
 LOCATE ROW%, COL%
 PRINT USING "####"; ymax!

 COL% = (x1.p% / 640!) * 80 - 5
 ROW% = ((y1.p% + view.p!) / 480!) * 30 + 1
 LOCATE ROW%, COL%
 PRINT USING "####"; ymin!

 FOR i = 1 TO (elec% / 2 - 1)
   COL% = ((x1.p% + i * (view.p! / CSNG(elec% / 2))) / 640!) * 80 + 1
   ROW% = ((y1.p% + view.p!) / 480!) * 30 + 2
   LOCATE ROW%, COL%
   PRINT USING "#"; i
 NEXT i

'****************** ANTISYMMETRY PLOT ********************

' Find limits for y axis

 ymax! = 10 * CINT((amax! + 10) / 10)
 ymin! = 10 * CINT((amin! - 10) / 10)
 n = CINT((ymax! - ymin!) / 10)
 dy! = (ymax! - ymin!) / n
```

```
  VIEW (x2.p%, y2.p%)-(x2.p% + view.p!, y2.p% + view.p!)
  WINDOW (0, ymin!)-((elec% / 4 + 1), ymax!)

  LINE (0, ymin!)-((elec% / 4 + 1), ymax!), , B    'draw box around graph

  dt.x! = (dt.p! / view.p!) * (ymax! - ymin!)      'calculate length of x-ticks
  FOR i = 1 TO elec% / 4                           'draw x ticks
    LINE (i, ymin!)-(i, ymin! + dt.x!)
  NEXT i

  dt.y! = (dt.p! / view.p!) * CSNG(elec% / 4 + 1) 'calculate length of y-ticks
  FOR i = 1 TO n - 1                               'draw y ticks
    y! = ymin! + CSNG(i) * dy!
    LINE (0, y!)-(0 + dt.y!, y!)
  NEXT i

  FOR i = 1 TO elec% / 2
    FOR j = 1 TO 4
      PSET (1, asym!(i, j, 1)), i                  'plot first point
      FOR k = 2 TO elec% / 4
        LINE -(k, asym!(i, j, k)), i               'join remaining points
      NEXT k
    NEXT j
  NEXT i

  COL% = (x2.p% / 640!) * 80 + 1
  ROW% = (y2.p% / 480!) * 30 - 1
  LOCATE ROW%, COL%
  PRINT "ANTISYMMETRY PLOT"
  LOCATE ROW% + 1, COL%
  PRINT USING "(####.## STANDARD DEVIATION)"; asymstd!

  COL% = (x2.p% / 640!) * 80 - 5
  ROW% = (y2.p% / 480!) * 30 + 1
  LOCATE ROW%, COL%
  PRINT USING "####"; ymax!

  COL% = (x2.p% / 640!) * 80 - 4
  ROW% = ((y2.p% + view.p!) / 480!) * 30 + 1
  LOCATE ROW%, COL%
  PRINT USING "###"; ymin!

  FOR i = 1 TO elec% / 4
    COL% = ((x2.p% + i * (view.p! / CSNG(elec% / 4 + 1))) / 640!) * 80 + 1
    ROW% = ((y2.p% + view.p!) / 480!) * 30 + 2
    LOCATE ROW%, COL%
    PRINT USING "#"; i
  NEXT i

  LOCATE 28, 1
  INPUT " Hit <return> to continue. ", resp$

'**************** CHECK LINEARITY OF DATA *****************

  max.L! = 0!: min.L! = 10000!
  FOR iv1 = 1 TO elec% / 2 - 1
    FOR iv2 = iv1 + 1 TO elec% - iv1
      avg.L! = 0!
      FOR iv3 = 1 TO elec%
        IF iv3 <> iv1 AND iv3 <> iv2 THEN
          temp! = volt!(iv1, elec% + 1 - iv1, iv3) - volt!(iv2, elec% + 1 - iv1, iv3)
          volt.L!(iv1, iv2, iv3) = volt!(iv1, iv2, iv3) - temp!
          avg.L! = avg.L! + volt.L!(iv1, iv2, iv3) / CSNG(elec% - 2)
        END IF
      NEXT iv3
      sum2.L!(iv1, iv2) = 0!
      FOR iv3 = 1 TO elec%
        IF iv3 <> iv1 AND iv3 <> iv2 THEN
          volt.L!(iv1, iv2, iv3) = volt.L!(iv1, iv2, iv3) - avg.L!
          IF volt.L!(iv1, iv2, iv3) > max.L! THEN max.L! = volt.L!(iv1, iv2, iv3)
          IF volt.L!(iv1, iv2, iv3) < min.L! THEN min.L! = volt.L!(iv1, iv2, iv3)
          sum2.L!(iv1, iv2) = sum2.L!(iv1, iv2) + volt.L!(iv1, iv2, iv3) ^ 2
        ELSE
          volt.L!(iv1, iv2, iv3) = 0!
        END IF
      NEXT iv3
    NEXT iv2
  NEXT iv1
  std.L! = 0!
  idenom = 0
  FOR iv1 = 1 TO elec% / 2 - 1
    idenom = idenom + iv1
  NEXT iv1
  denom = CSNG((elec% * (elec% / 2 - 1)) - 2 * idenom)
  FOR iv1 = 1 TO elec% / 2 - 1
    FOR iv2 = iv1 + 1 TO elec% - iv1
      std.L! = std.L! + SQR(sum2.L!(iv1, iv2) / CSNG(elec% - 3)) / denom
    NEXT iv2
  NEXT iv1

'****************** LINEARITY PLOT ********************

  CLS 0
  dt.p! = 5!                       'length of tick marks in pixels
  view.p! = 240!                   'viewport length and width/2 in pixels

  xoff.p% = (640! - 2! * view.p!) / 2! - 1
  yoff.p% = (480! - view.p!) / 2! - 1

' Find limits for y axis

  ymax! = 10 * CINT((max.L! + 10) / 10)
  ymin! = 10 * CINT((min.L! - 10) / 10)
  n = CINT((ymax! - ymin!) / 10)
  dy! = (ymax! - ymin!) / n

  VIEW (xoff.p%, yoff.p%)-(xoff.p% + 2 * view.p!, yoff.p% + view.p!)
  WINDOW (0, ymin!)-(elec% + 1, ymax!)

  LINE (0, ymin!)-(elec% + 1, ymax!), , B          'draw box around graph

  dt.x! = (dt.p! / view.p!) * (ymax! - ymin!)      'calculate length of x-ticks
  FOR i = 1 TO elec%                               'draw x ticks
    LINE (i, ymin!)-(i, ymin! + dt.x!)
  NEXT i

  dt.y! = (dt.p! / (2 * view.p!)) * CSNG(elec% + 1) 'calculate length of y-ticks
  FOR i = 1 TO n - 1                               'draw y ticks
    y! = ymin! + CSNG(i) * dy!
    LINE (0, y!)-(0 + dt.y!, y!)
  NEXT i

  FOR iv1 = 1 TO elec% / 2 - 1
    FOR iv2 = iv1 + 1 TO elec% - iv1
      PSET (1, volt.L!(iv1, iv2, 1)), iv1          'plot first point
      FOR iv3 = 2 TO elec%
        LINE -(iv3, volt.L!(iv1, iv2, iv3)), iv1  'join remaining points
      NEXT iv3
    NEXT iv2
  NEXT iv1

  COL% = (xoff.p% / 640!) * 80 + 1
  ROW% = (yoff.p% / 480!) * 30 - 1
  LOCATE ROW%, COL%
  PRINT "LINEARITY PLOT"
  LOCATE ROW% + 1, COL%
  PRINT USING "(####.## STANDARD DEVIATION)"; std.L!

  COL% = (xoff.p% / 640!) * 80 - 5
  ROW% = (yoff.p% / 480!) * 30 + 1
  LOCATE ROW%, COL%
  PRINT USING "####"; ymax!

  COL% = (xoff.p% / 640!) * 80 - 4
  ROW% = ((yoff.p% + view.p!) / 480!) * 30 + 1
  LOCATE ROW%, COL%
  PRINT USING "###"; ymin!

  FOR i = 1 TO elec%
    COL% = ((xoff.p% + i * (2 * view.p! / CSNG(elec% + 1))) / 640!) * 80
    ROW% = ((yoff.p% + view.p!) / 480!) * 30 + 2
    LOCATE ROW%, COL%
    PRINT USING "##"; i
  NEXT i

  LOCATE 28, 1
  INPUT " Hit <return> to continue. ", resp$

'***************** PRINT DATA TO FILES *********************

  SCREEN 0
  CLS

' Output options DO loop corrected to ensure data is saved -- dlg, 5/28/97

  ftest% = 0
  cont$ = "N"

  DO

    PRINT : PRINT : PRINT "Output options:"
    PRINT "  Input 'F' to save data to files."
    PRINT "  Input 'D' to generate difference files."
    PRINT "  Input 'E' to generate an ASCII file for Excel."
    PRINT "  Input 'P' to generate a graphical output file."
    PRINT "  Input 'C' to collect a new data set."
    PRINT "  Input 'Q' to quit."
    INPUT resp$
    IF resp$ = "F" OR resp$ = "f" THEN
      CALL datmake(elec%, carr!(), quad!(), volt!(), file$)
      CALL symmake(elec%, symstd!, sym!(), asymstd!, asym!(), file$)
      CALL linmake(elec%, std.L!, volt.L!(), file$)
      CALL promake(elec%, proj%, Vsum!(), czero!, qzero!, CEVmean!, CEVmax!, CEVmin!,
       CEVstd!, Vmean!, Vmax!, Vmin!, Vstd!, file$)
      ftest% = 1
    ELSEIF resp$ = "D" OR resp$ = "d" THEN
      CALL difmake(elec%, carr!(), quad!(), volt!())
    ELSEIF resp$ = "E" OR resp$ = "e" THEN
```

```
      CALL xlsmake(elec%, volt!())
      ftest% = 1
    ELSEIF resp$ = "P" OR resp$ = "p" THEN
      CALL psmake(elec%, carr!(), quad!(), volt!())
    ELSEIF resp$ = "C" OR resp$ = "c" OR resp$ = "Q" OR resp$ = "q" THEN
      IF ftest% = 0 THEN
        PRINT "The data set has not been saved to a file yet."
        INPUT "Are you sure you want to do this (Y/N - default N) ?", cont$
      ELSE
        cont$ = "Y"
      END IF
    END IF

  LOOP UNTIL cont$ = "Y" OR cont$ = "y"

  ERASE carr!, quad!, volt!
  ERASE Vsum.elc!, Vsum2.elc!, sym!, save!, ssum!, ssum2!
  ERASE asym!, asum!, asum2!, volt.L!, sum2.L!

END SUB



SUB linmake (elec%, std.L!, volt.L!(), file$)

PRINT
INPUT "Do you want to save linearity data? (default Y):", save$

IF save$ <> "N" AND save$ <> "n" THEN

  OPEN "d:\data\" + file$ + ".lin" FOR OUTPUT AS #1
  PRINT #1, USING "Standard Deviation                #####.## "; std.L!
  PRINT #1, ""
  FOR iv1 = 1 TO (elec% / 2 - 1)
    FOR iv2 = iv1 + 1 TO elec% - iv1
      PRINT #1, USING "## , ## , "; iv1; iv2;
      FOR iv3 = 1 TO elec%
        PRINT #1, USING "####.## , "; volt.L!(iv1, iv2, iv3);
      NEXT iv3
      PRINT #1, ""
    NEXT iv2
  NEXT iv1
  CLOSE #1

  PRINT "Linearity data saved to d:\data\"; file$; ".lin"

END IF

END SUB



SUB promake (elec%, proj%, Vsum!(), czero!, qzero!, CEVmean!, CEVmax!, CEVmin!,
       CEVstd!, Vmean!, Vmax!, Vmin!, Vstd!, file$)

PRINT
INPUT "Do you want to save projection statistics data? (default Y):", save$

IF save$ <> "N" AND save$ <> "n" THEN

  OPEN "d:\data\" + file$ + ".pro" FOR OUTPUT AS #1
  PRINT #1, " Average Data Set Characteristics"
  PRINT #1, ""
  PRINT #1, USING " Carrier Zero Offset              #####"; czero!
  PRINT #1, USING " Quad Zero Offset                 #####"; qzero!
  PRINT #1, ""
  PRINT #1, " Statistics for cross-electrode voltages comparing data"
  PRINT #1, " recorded for each projection:"
  PRINT #1, ""
  PRINT #1, USING " MEAN Cross Electrode Voltage     #####.##"; CEVmean!
  PRINT #1, USING " MAX Cross Electrode Voltage      #####.##"; CEVmax!
  PRINT #1, USING " MIN Cross Electrode Voltage      #####.##"; CEVmin!
  PRINT #1, USING " STDEV Cross Electrode Voltage     ####.##"; CEVstd!
  PRINT #1, ""
  PRINT #1, " Statistics for electrode voltages (not including current injection"
  PRINT #1, " and ground electrodes) for 180 degree injection/ground cases"
  PRINT #1, " comparing data recorded for each projection:"
  PRINT #1, ""
  PRINT #1, USING " MEAN Electrode Voltage           #####.##"; Vmean!
  PRINT #1, USING " MAX Electrode Voltage            #####.##"; Vmax!
  PRINT #1, USING " MIN Electrode Voltage            #####.##"; Vmin!
  PRINT #1, USING " STDEV Electrode Voltage           ####.##"; Vstd!
  PRINT #1, ""
  PRINT #1, "Average data for each projection recorded. Voltages are"
  PRINT #1, "averages for each electrode for cases where current and"
  PRINT #1, USING "ground are 180 degrees opposed (# total). Column 1 is the"; elec% /
      2
  PRINT #1, USING "projection number and columns 2 through ## are data from"; elec% +
      1
  PRINT #1, USING "electrodes 1 through ## (1 current and # ground)."; elec%; elec% /
      2 + 1
  PRINT #1, ""
  FOR i = 1 TO proj%
    PRINT #1, USING "### , "; i;
    FOR k = 1 TO elec% - 1
      PRINT #1, USING "#####.##, "; Vsum!(i, k);
    NEXT k
    PRINT #1, USING "#####.##"; Vsum!(i, elec%)
  NEXT i
  CLOSE #1

  PRINT "Projection statistics saved to d:\data\"; file$; ".pro"

END IF

END SUB



SUB psmake (elec%, carr!(), quad!(), volt!())
DIM x!(15), y!(15), cur%(3850), gnd%(3850)
DIM mag%(3850)

DO

  PRINT "Enter 'M' for magnitude, 'C' for carrier, or 'Q' for quadrature signal plot.
      "
  INPUT "  Hit <return> for default (magnitude plot). ", pl$
  IF pl$ = "" THEN pl$ = "M"

  INPUT "Enter Postscript data file name ('.ps' appended automatically): ", file$

  file$ = "d:\data\" + file$ + ".ps"
  OPEN file$ FOR OUTPUT AS #2

  icount = 1
```

```
FOR iv1 = 1 TO elec% - 1
  FOR iv2 = iv1 + 1 TO elec%
    FOR iv3 = 1 TO elec%

      cur%(icount) = iv1: gnd%(icount) = iv2

      IF pl$ = "M" OR pl$ = "m" THEN
        mag%(icount) = volt!(iv1, iv2, iv3)
      ELSEIF pl$ = "C" OR pl$ = "c" THEN
        mag%(icount) = carr!(iv1, iv2, iv3)
      ELSEIF pl$ = "Q" OR pl$ = "q" THEN
        mag%(icount) = quad!(iv1, iv2, iv3)
      END IF

      icount = icount + 1

    NEXT iv3
  NEXT iv2
NEXT iv1

max% = 0
min% = 5000

FOR i = 1 TO icount - 1

  IF mag%(i) > max% THEN max% = mag%(i)
  IF mag%(i) < min% THEN min% = mag%(i)

NEXT i

x!(1) = 1.5
x!(2) = 2.2
x!(3) = 2.2
x!(4) = -3.3
x!(5) = 2.2
x!(6) = 2.2
x!(7) = -5.5
x!(8) = 2.2
x!(9) = 2.2
x!(10) = -3.3
x!(11) = 2.2
x!(12) = 2.2
x!(13) = -5.5
x!(14) = 2.2
x!(15) = 2.2

y!(1) = 9.5
y!(2) = 0!
y!(3) = 0!
y!(4) = -2!
y!(5) = 0!
y!(6) = 0!
y!(7) = -2!
y!(8) = 0!
y!(9) = 0!
y!(10) = -2!
y!(11) = 0!
y!(12) = 0!
y!(13) = -2!
y!(14) = 0!
y!(15) = 0!
```

```
  PRINT #2, "%!PS-Adobe-3.0 EPSF-3.0"
  PRINT #2, "%%Page: 8"
  PRINT #2, "/inch { 72 mul } bind def"
  PRINT #2, "/inner 0.80 inch def"
  PRINT #2, "/outer 1.0 inch def"

  l = 0

' Total number of voltage circles is 1+2+...+(elec%-1); indices
'   changed to keep 15 voltage circles per page.  Actual LSB values
'   are printed now without being normalized to a maximum of 1000 ---
'   dlg, 5/28/97

  icrcls = 0
  icrcct = 0
  FOR n = 1 TO (elec% - 1)
    icrcls = icrcls + n
  NEXT n
  npages = FIX((icrcls + 14) / 15)

  FOR i = 1 TO npages

    FOR k = 1 TO 15

      IF icrcct < icrcls THEN

        PRINT #2, USING "##.# inch ##.# inch translate"; x!(k); y!(k)
        PRINT #2, "/Times-Roman findfont 0.15 inch scalefont setfont"

        FOR j = 1 TO elec%

          l = l + 1
          linew! = .015 + mag%(l) * .3 / (max% - min%)

' Normalization to 1000 can be restored on the next line if needed

          PRINT #2, USING "0.5 inch -0.05 inch moveto (####) show"; mag%(l)
          PRINT #2, USING "#.### inch setlinewidth"; linew!
          PRINT #2, "0 cos inner mul 0 sin inner mul moveto"
          PRINT #2, "0 cos outer mul 0 sin outer mul lineto"
          PRINT #2, "stroke"
          PRINT #2, USING "###.# rotate"; -360! / CSNG(elec%)

        NEXT j

        cangle! = (cur%(l) - 1) * -360! / CSNG(elec%)
        cnangle! = -1! * cangle!

        gangle! = (gnd%(l) - 1) * -360! / CSNG(elec%)
        gnangle! = -1! * gangle!

        PRINT #2, "/Times-Roman findfont 0.2 inch scalefont setfont"

        PRINT #2, USING "####.# rotate 1.0 inch -0.05 inch moveto (J) show"; cangle!
        PRINT #2, USING "####.# rotate"; cnangle!

        PRINT #2, USING "####.# rotate 1.0 inch -0.05 inch moveto (G) show"; gangle!
        PRINT #2, USING " ####.# rotate"; gnangle!

        PRINT #2, "/Times-Roman findfont 0.15 inch scalefont setfont"

        icrcct = icrcct + 1
      ELSE
      END IF

    NEXT k

    PRINT #2, "showpage"
    PRINT #2, USING "%%Page: ## ##"; i; i

  NEXT i

  CLOSE #2

  PRINT "Postscript file saved to "; file$
  INPUT "Do you wish to make another plot? (Y/N) ", rs$

LOOP WHILE rs$ = "y" OR rs$ = "Y"

END SUB



SUB symmake (elec%, symstd!, sym!(), asymstd!, asym!(), file$)

PRINT
INPUT "Do you want to save symmetry & antisymmetry data? (default Y):", save$

IF save$ <> "N" AND save$ <> "n" THEN

  OPEN "d:\data\" + file$ + ".sym" FOR OUTPUT AS #1
  PRINT #1, "This file contains data used to determine if the test case"
  PRINT #1, USING "has the appropriate symmetry and antisymmetry. The # cases tested";
       elec% / 2
  PRINT #1, "are those where the current and ground are 180 degrees opposed."
  PRINT #1, "The first column is the number of electrode 'steps' from the injection"
  PRINT #1, "point to the measurement point.  For each data point, the mean voltage"
  PRINT #1, "has been subtracted to allow for differences in the ground"
  PRINT #1, "electrode's contact resistance."
  PRINT #1, ""
  PRINT #1, "Symmetry Data: The data is given for each case, first clockwise,"
  PRINT #1, "then counter-clockwise from the injection point for a total of"
  PRINT #1, USING "## curves. The STDEV between the curves is: ###.##"; elec%; symstd!
  PRINT #1, ""
  PRINT #1, "Injection pt:";
  FOR i = 1 TO elec% / 2
    PRINT #1, USING "#                   "; i;
  NEXT i
  PRINT #1, ""
  PRINT #1, "Steps";
  FOR i = 1 TO elec% / 2
    PRINT #1, "   CW        CCW    ";
  NEXT i
  PRINT #1, ""
  FOR k = 1 TO elec% / 2 - 1
    PRINT #1, USING "   # , "; k;
    FOR i = 1 TO elec% / 2
      FOR j = 1 TO 2
        PRINT #1, USING "#####.##, "; sym!(i, j, k);
      NEXT j
    NEXT i
    PRINT #1, ""
  NEXT k
  PRINT #1, ""
  PRINT #1, "Antisymmetry Data: The data is given for each case, first counter-"
  PRINT #1, "clockwise, then clockwise from both antisymmetry electrode nodes for"
  PRINT #1, USING "a total of ## curves.  Antisymmetry nodes are 90 degrees from the";
       elec% * 2
  PRINT #1, "injection and ground points.  The first values are measured at the
       asymmetry"
  PRINT #1, "points, the second from one step away, etc.  The STDEV between the"
  PRINT #1, USING "curves is: ###.##"; asymstd!
  PRINT #1, ""
  PRINT #1, "Injection pt:";
  FOR i = 1 TO elec% / 4
    PRINT #1, USING " #                                      "; i;
  NEXT i
  PRINT #1, ""
  PRINT #1, "From inj. ";
  FOR i = 1 TO elec% / 4
    PRINT #1, "  CW         CW       CCW       CCW     ";
  NEXT i
  PRINT #1, ""
  PRINT #1, "From asym.";
  FOR i = 1 TO elec% / 4
    PRINT #1, " CCW         CW       CCW        CW     ";
  NEXT i
  PRINT #1, ""
  FOR k = 1 TO elec% / 4
    PRINT #1, USING "# , "; k;
    FOR i = 1 TO elec% / 2
      FOR j = 1 TO 4
        PRINT #1, USING "#####.##, "; asym!(i, j, k);
      NEXT j
    NEXT i
    PRINT #1, ""
  NEXT k
  CLOSE #1

  PRINT "Symmetry and antisymmetry data saved to d:\data\"; file$; ".sym"

END IF

END SUB



SUB xlsmake (elec%, volt!())
  INPUT "Input Excel file name ('.xls' is appended automatically): ", file$
  OPEN "d:\data\" + file$ + ".xls" FOR OUTPUT AS #1

  FOR iv1 = 1 TO elec% - 1
    FOR iv2 = iv1 + 1 TO elec%

      PRINT #1, iv1; "to"; iv2; ",";

      FOR iv3 = 1 TO elec% - 1
        PRINT #1, USING "#####.## ,"; volt!(iv1, iv2, iv3)
      NEXT iv3

      PRINT #1, USING "#####.##"; volt!(iv1, iv2, elec%)

    NEXT iv2
  NEXT iv1

  PRINT "Data saved in excel format as d:\data\"; file$; ".xls"
  CLOSE #1
END SUB
```