Appendix C
EIT Data Acquisition Code SLOWEIT.BAS

This Microsoft® QuickBasic™ program functions in a similar manner to FASTEIT.BAS, but initializes the EIT system so that multiplexer commands are issued by the data acquisition card instead of the internal counters. Memory is statically allocated by the code, and voltages are stored in arrays after each projection set is collected, rather than being held in buffers until all projection sets are complete.

```
'***
'***  Steve and Ann's EIT code
'***    Revised 9/19/96 by dlg
'***    Revised 11/1/96 by kas to add symmetry checks
'***    Revised 6/97 by dlg to adapt to eight electrodes
'***    Revised 7/97-8/97 by slc and dlg for speed
'***    Revised 11/97 by dlg to move statistics code to separate module
'***

'$DYNAMIC

DECLARE SUB exit.error (status AS INTEGER, message AS STRING)
DECLARE SUB cleanup (dummy)
DECLARE SUB eistats (elec%, proj%, slength%, carrsum&(), quadsum&(), Vsum!(), resp$)

' Definition files for extended memory manager, error codes, counter/
'   timer subroutines and software tools; these files define several
'   constants used in subroutine calls, which are found in BOLDFACE
'   in the code

'$INCLUDE: 'd:\toolkit\dtst_xmm.bi'
'$INCLUDE: 'd:\toolkit\dtst_err.bi'
'$INCLUDE: 'd:\toolkit\dtst_ctr.bi'
'$INCLUDE: 'd:\toolkit\dtst_tls.bi'

'DECLARE SUB featur (iunit AS INTEGER, ihandle AS INTEGER) ' diagnostic subroutine
'DECLARE SUB putboard (iunit AS INTEGER, ihandle AS INTEGER, idma1 AS INTEGER, idma2
      AS INTEGER, intrpt AS INTEGER, adsetup AS INTEGER, dasetup AS INTEGER, itmout
      AS INTEGER) '*** replaced by direct call to dt.set.board
'DECLARE SUB getboard (iunit AS INTEGER, ihandle AS INTEGER) ' diagnostic subroutine

DIM digitalio AS DIGITALIOSTRUCT
DIM sap AS SETACQPARAMSSTRUCT
DIM board AS BOARDSTRUCT

DEFINT I-N
DEFSNG A-H
DEFLNG O-Z
```

```
' Size of channel/gain table changed so that one pass through the
'   table completely fills the buffer --- slc & dlg, 7/97

CONST blength% = 64        ' this is the buffer length
DIM ibuff(blength% - 1)    ' buffer array
DIM ichan(blength% - 1)    ' channel array for channel/gain table
DIM igain(blength% - 1)    ' gain array for channel/gain table

' Number of electrodes can now be set at 8 or 16 --- dlg, 5/27/97
CLS
PRINT
PRINT "This EIT program sets the individual electrodes as current, ground, or"
PRINT "measurement.  The voltage output of the demodulators is recorded for"
PRINT "analysis. "
PRINT
DO
  INPUT "Enter the number of electrodes (8 or 16): ", elec%
  IF elec% <> 8 AND elec% <> 16 THEN
    PRINT "Incorrect input."
  END IF
LOOP UNTIL elec% = 8 OR elec% = 16

DIM carrsum&(1 TO (elec% - 1), 2 TO elec%, 1 TO elec%) ' sums of carrier voltages
DIM quadsum&(1 TO (elec% - 1), 2 TO elec%, 1 TO elec%) ' sums of quadrature voltages
DIM Vsum!(1 TO 100, 1 TO elec%)   ' sums of off-centerline electrode voltages
                                  '   for 180-degree injection/ground cases
                                  '   indexed by projection and relative
                                  '   location

' Initialize system for data acquisition
' DO...LOOP WHILE added to eliminate bug in output options -- dlg, 5/28/97

DO
  INPUT "Input the number of projections to be taken (1 to 100): ", proj%
  IF proj% < 1 THEN proj% = 1
  IF proj% > 100 THEN proj% = 100

' PGA202 amplifier gain is now adjustable before execution --- dlg, 8/2/96
' Default gain changed to 10 with addition of AC coupling filters; gain
'   of 1 is now too low --- dlg, 8/97

  INPUT "Do you wish to change the amplifier gain from the default (Y/N) (default N)?
      ", gainset$

  IF gainset$ = "Y" OR gainset$ = "y" THEN

    PRINT "Input the index for the amplifier gain:"
    PRINT "   N = 0       Gain =    1"
    PRINT "       1                 10"
    PRINT "       2                100"
    PRINT "       3               1000"
    INPUT "Recommended value is N = 1 (default). ", igainpwr

    IF igainpwr < 0 OR igainpwr > 3 THEN igainpwr = 1

  ELSE

    igainpwr = 1

  END IF

'******* Initialize summation and averaging arrays *******
```

```
  FOR icurrent = 1 TO elec% - 1          'icurrent = index of current injection
      electrode
    FOR iground = icurrent + 1 TO elec%  'iground = index of current return electrode
      FOR ivolt2 = 1 TO elec%            'ivolt2 = index of voltage meas. electrode
       carrsum&(icurrent, iground, ivolt2) = 0
       quadsum&(icurrent, iground, ivolt2) = 0
      NEXT ivolt2
    NEXT iground
  NEXT icurrent

  FOR i = 1 TO proj%
    FOR ivolt2 = 1 TO elec%
      Vsum!(i, ivolt2) = 0!
    NEXT ivolt2
  NEXT i

'******* Initialize and reset the DT283X board *******
'*** Diagnostic calls to 'exit.error' added after each 'dt.' call ---
'***   dlg, 8/25/97

  idrivername$ = "DT283X$0" + CHR$(0)
  iunit = 0       'Board is unit 0

' SADD and VARSEG functions pass the address of their arguments;
'   retained from example on p. 24 of software toolkit user's manual

  istat = dt.initialize(SADD(idrivername$), VARSEG(idrivername$), ihandle)
  IF istat <> 0 THEN CALL exit.error(istat, "dt.initialize")
  istat = dt.ct31.initialize(ihandle)
  IF istat <> 0 THEN CALL exit.error(istat, "dt.ct31.initialize")
  istat = dt.reset(iunit, ihandle)
  IF istat <> 0 THEN CALL exit.error(istat, "dt.reset")

' Get the board's feature list
'  CALL featur(iunit, ihandle) '*** diagnostic

' Get/set the board parameters
'  CALL getboard(iunit, ihandle) '*** diagnostic

  board.dmachannel1 = 5             'DMA channel 1 selection
'  board.dmachannel2 = 6             'DMA channel 2 selection '*** removed for single
      DMA
  board.interruptlevel = 10         'interrupt level
  board.boardtimeout = 10           '10-second wait before timeout error returned
  board.adsetupbits = 0             'a-to-d setup code
  board.dasetupbits = NODACS        'd-to-a setup code

  istat = dt.set.board(iunit, ihandle, board)
  IF istat <> 0 THEN CALL exit.error(istat, "dt.set.board")

'  CALL getboard(iunit, ihandle) '*** diagnostic

'******* Set board acquisition parameters *******

  isection = ADSECTION
  isize = blength%

  digitalio.command = SETXFER
  digitalio.direction = DIOOUTPUT

'******* Create a buffer *******
' 'dt.create.buffer' moved out from electrode loop, since it must be
'   created before channel-gain list --- dlg, 8/25/97
```

```
  istat = dt.create.buffer(iunit, isection, isize, ibuff(0), ibhandle)
  IF istat <> 0 THEN CALL exit.error(istat, "dt.create.buffer")

' Create a channel/gain list (since the board can apply a different
'   gain to each D/A channel)
'
'       DT2839 channel gains:   gain    igain()
'                                 1        0
'                                 2        1
'                                 4        2
'                                 8        3
'
'   Board range is set at +/- 10V.

  FOR jj = 0 TO 31
    ichan(2 * jj) = 0: ichan(2 * jj + 1) = 1
    igain(2 * jj) = 0: igain(2 * jj + 1) = 0
  NEXT jj

' Subroutine 'dt.create.cgl' expects ichan() and igain() arrays to be
'   passed by reference, not by value, so last two arguments are the
'   first elements in each array, not the arrays themselves
' Hardwired list size replaced by 'blength%' to match buffer and
'   list sizes --- dlg, 8/27/97

  istat = dt.create.cgl(blength%, ichan(0), igain(0))
  IF istat <> 0 THEN CALL exit.error(istat, "dt.create.cgl")

' Clock rate must be reduced from 416 kHz to 320 kHz when channel gain > 1.
'   If value of arate is not a possible conversion rate, driver rounds up
'   to the next highest available rate.

  IF igain(0) = 0 AND igain(1) = 0 THEN
    arate = 400000
  ELSE
    arate = 300000
  END IF

' Transfer mode changed from BDUAL (dual channel DMA) to BSINGLE (single
'   channel DMA) to speed things up --- slc and dlg, 7/97
' Maximum possible trigger rate to scan channel-gain list (CGL) is
'   clockrate/(CGL size + 2).  If sap.trigrate is set too high,
'   'dt.set.acq' will round down to the highest possible rate ---
'   dlg, 8/22/97

  sap.section = isection
  sap.transfertype = BSINGLE
  sap.clockrate = arate
  sap.trigrate = arate / 4
  sap.cutoff = 0

  istat = dt.set.acq(iunit, ihandle, isection, sap)
'  PRINT "New clockrate = "; sap.clockrate '*** diagnostic
'  PRINT "New trigger rate = "; sap.trigrate '*** diagnostic
  IF istat <> 0 THEN CALL exit.error(istat, "dt.set.acq")

'*****
'
' Rules to set current injection and ground electrodes and voltage
'   amplifier:
'
' There are (N-1)*N/2 possible projections, where N is the number of
```

```
'   electrodes.  "icurrent" is the injection electrode, and "iground"
'   is the return electrode.
'
' All linearly independent combinations of electrode pairs will be used.
'   When only eight electrodes are used, leads 9 through 16 must be
'   connected, with 1 through 8 isolated; this is needed for the
'   internal counters to operate in fast mode, so the same convention
'   was placed in this slow version of the code.
'
' Arrangement of ports which communicate to the EIT electronics:
'
' Port 0 Bit:  00     01     02     03     04     05     06     07
'           |     electrode address     |enable|   latch address    |
'
' Port 1 Bit:  10     11     12     13     14     15     16     17
'           |     electrode address     |enable| PGA202 gain | not  |
'                                                             | used |
'
' To ready electrodes for current injection or voltage measurement,
'   the digital I/O lines must be set as follows to commumicate with
'   the correct muxes:
'
'                                     Port 0: Bit 05  Bit 06  Bit 07
'   current injection or ground mux                L       L       L
'   voltage measurement or reference mux           H       L       L
'
' To select an electrode, the following i/o lines must be set:
'
'                                                                 Bit 04
'                     Port 0: Bit 00  Bit 01  Bit 02  Bit 03  (enable)
'   current injection           A0      A1      A2      A3       H
'   voltage reference (ground)  A0      A1      A2      A3       H
'                                                                 Bit 14
'                     Port 1: Bit 10  Bit 11  Bit 12  Bit 13  (enable)
'   current return (ground)     A0      A1      A2      A3       H
'   voltage measurement         A0      A1      A2      A3       H
'
' A0, A1, A2, A3 are set high or low according to the table below:
'
'     A3   A2   A1   A0   Electrode
'      L    L    L    L       1
'      L    L    L    H       2
'      L    L    H    L       3
'      L    L    H    H       4
'      L    H    L    L       5
'      L    H    L    H       6
'      L    H    H    L       7
'      L    H    H    H       8
'      H    L    L    L       9
'      H    L    L    H      10
'      H    L    H    L      11
'      H    L    H    H      12
'      H    H    L    L      13
'      H    H    L    H      14
'      H    H    H    L      15
'      H    H    H    H      16
'
' To set the PGA202 amplifier gain, the following digital i/o lines must
'   be set:
'           Port 0: Bit 05  Bit 06  Bit 07
'                     L       H       L
'           Port 1: Bit 15  Bit 16
'                     A0      A1
```

```
'   Port 0 values changed with the addition of fast selection counters
'   to the EIT hardware, 7/97.
'
' A0 and A1 are set high or low according to the table below:
'
'                              Value added to integer which
'       A1     A0      Gain    selects voltage channels
'        L      L        1         0
'        L      H       10        32
'        H      L      100        64
'        H      H     1000        96
'
'*****

'******* Set amplifier gain *******
' This set of commands latches the gain on the PGA202 amplifier and
'   instructs the system not to use the new internal electrode
'   counters.  The '64' in .diovalue selects the gain mux.

  digitalio.dioport = PORT0
  digitalio.diovalue = 64
  istat = dt.set.dio(iunit, ihandle, digitalio)  'sends word to di/o.
  IF istat <> 0 THEN CALL exit.error(istat, "dt.set.dio (PGA202 PORT0)")

  gain% = 32 * igainpwr

  digitalio.dioport = PORT1
  digitalio.diovalue = gain%
  istat = dt.set.dio(iunit, ihandle, digitalio)  'sends word to di/o.
  IF istat <> 0 THEN CALL exit.error(istat, "dt.set.dio (PGA202 PORT1)")

' The following function, dt.ct31.gate.delay, is used here to send
'   a pulse from clock CLK1.  The pulse will force the address decoder to
'   enable the gain latches, which will then set the gain muxes.

  istat = dt.ct31.gate.delay(iunit, 0, 0, 1, 2)
  IF istat <> 0 THEN CALL exit.error(istat, "dt.ct31.gate.delay (PGA202)")

'****************** START OF DATA ACQUISITION LOOP ******************

  PRINT "Acquiring projections...": PRINT

'  OPEN "d:\data\diagnose.dat" FOR OUTPUT AS #2 '*** diagnostic

  FOR projloop% = 1 TO proj%

' Get clock time at start of projection routine

    starttime! = TIMER

' 'icurrent' is the array index for the injection electrode, while
'   'icelec' is the associated electrode number; in the case of eight
'   electrodes, 'icurrent' ranges from 1 to 8, while 'icelec' runs
'   from 9 to 16.  See note under "Rules to set electrodes" above.

    FOR icurrent = 1 TO elec% - 1
      icelec = 16 - elec% + icurrent

'  'iground' and 'igelec' have the same relationship as 'icurrent'
'    and 'icelec'

      FOR iground = icurrent + 1 TO elec%
       igelec = 16 - elec% + iground
```

```
' In the statement (digitalio.diovalue=icelec-1+16+0), icelec is the
'   electrode that is the current injector; '-1' sets the bit pattern
'   correctly, e.g., for current electrode 8, a 7 in binary needs to be
'   sent to the digitalio channels.  The '+16' sets bit 04 to high,
'   enabling the latches.  '+0' is a placeholder which represents
'   the selection of current muxes; see the later statements for
'   voltage muxes, where '+32' appears.

' Write address of injection electrode to port 0

      digitalio.dioport = PORT0
      digitalio.diovalue = icelec - 1 + 16 + 0
      istat = dt.set.dio(iunit, ihandle, digitalio)
      IF istat <> 0 THEN CALL exit.error(istat, "dt.set.dio (current mux PORT 0)")

' Write address of ground electrode to port 1

      digitalio.dioport = PORT1
      digitalio.diovalue = igelec - 1 + 16
      istat = dt.set.dio(iunit, ihandle, digitalio)
      IF istat <> 0 THEN CALL exit.error(istat, "dt.set.dio (current mux PORT 1)")

      istat = dt.ct31.gate.delay(iunit, 0, 0, 1, 2)
      IF istat <> 0 THEN CALL exit.error(istat, "dt.ct31.gate.delay (current mux)")

' The next command is used to set the reference voltage electrode,
'   ivolt1.  The reference electrode is chosen to be the ground electrode.
'   '+32' selects the voltage muxes.

      ivolt1 = igelec

      digitalio.dioport = PORT0
      digitalio.diovalue = ivolt1 - 1 + 16 + 32
      istat = dt.set.dio(iunit, ihandle, digitalio)  'sends word to di/o.
      IF istat <> 0 THEN CALL exit.error(istat, "dt.set.dio (voltage mux (PORT0)")

' This loop is used to choose the electrode for voltage measurement,
'   ivelec.  This electrode goes through all possible values.  'ivolt2'
'   and 'ivelec' have the same relationship as 'icurrent' and 'icelec'.

      FOR ivolt2 = 1 TO elec%
        ivelec = 16 - elec% + ivolt2

        digitalio.dioport = PORT1
        digitalio.diovalue = ivelec - 1 + 16
        istat = dt.set.dio(iunit, ihandle, digitalio)  'sends word to di/o.
        IF istat <> 0 THEN CALL exit.error(istat, "dt.set.dio (voltage mux PORT1)")

        istat = dt.ct31.gate.delay(iunit, 0, 0, 1, 2)
        IF istat <> 0 THEN CALL exit.error(istat, "dt.ct31.gate.delay (voltage mux)")

' Now data must be read from the A/D channels.  A/D Channel 0 is the
'   carrier signal, A/D channel 1 is the quadrature signal.
'
' Reset buffer

        istat = dt.reset.buffer(ibhandle)
        IF istat <> 0 THEN CALL exit.error(istat, "dt.reset.buffer")

' Take multiple measurements and average output values for in-phase
'   and quadrature signals.  Acquire blength%/2 samples on each channel.
```

```
        istat = dt.start.acq(iunit, ihandle, isection)
        IF istat <> 0 AND istat <> 1 THEN
          CALL exit.error(istat, "dt.start.acq")
          istat = dt.stop.acq(iunit, ihandle, isection)
        END IF

'*** Diagnostic check of buffer added --- dlg, 8/27/97
'recheck:
'          istat = dt.check.buffer(iunit, isection, ibhandle, ibufstat)
'          IF ibufstat <> 194 THEN PRINT "Buffer status = "; ibufstat
'          IF ibufstat = 194 THEN GOTO recheck

' Check that buffer has been processed
' Buffer reset commented out to ensure data survives until it is
'   read --- dlg, 8/27/97

        istat = dt.wait.buffer(iunit, isection, ibhandle)
        IF istat <> 0 THEN CALL exit.error(istat, "dt.wait.buffer")

'          istat = dt.reset.buffer(ibhandle)
'          PRINT "Handle of reset buffer = "; ibhandle
'          IF istat <> 0 THEN CALL exit.error(istat, "dt.reset.buffer")

' Oversampled voltage measurements have been sent to the buffer, alternating
'   between carrier and quadrature values, blength%/2 of each; measurements
'   are now stripped from the buffer and placed in the 'carrsum&' and
'   'quadsum&' arrays

        ocave& = 0: oqave& = 0

        FOR ib = 1 TO blength% / 2
'*** diagnostic prints added
'            PRINT #2, USING "& &"; HEX$(ibuff(2 * ib - 2) + 2048); HEX$(ibuff(2 * ib
      - 1) + 2048)'***
'            PRINT "carrier value = "; ibuff(2 * ib - 2), '***
'            PRINT "quadrature value = "; ibuff(2 * ib - 1) '***
          ocave& = ocave& + ibuff(2 * ib - 2)
          oqave& = oqave& + ibuff(2 * ib - 1)
        NEXT ib

'          WRITE #2, projloop%, icurrent, iground, ivolt2, (ocave& / (blength% / 2)),
      (oqave& / (blength% / 2))'***diagnostic
'          WRITE #2, icurrent, iground, ivolt1, ivolt2, icelec, igelec, ivelec,
      (ocave& / (blength% / 2)), (oqave& / (blength% / 2))'***diagnostic

        carrsum&(icurrent, iground, ivolt2) = ocave& + carrsum&(icurrent, iground,
      ivolt2)
        quadsum&(icurrent, iground, ivolt2) = oqave& + quadsum&(icurrent, iground,
      ivolt2)

' Calcuate average voltages for each projection for each case where
'   the current injection and ground are 180 degrees opposed. There are
'   (elec%/2) cases per projection and the voltages are recorded and
'   averaged for each electrode.

        IF iground = (icurrent + (elec% / 2)) THEN
          k = ivolt2 - icurrent + 1
          IF k < 1 THEN k = elec% + 1 - icurrent + ivolt2
          Vsum!(projloop%, k) = Vsum!(projloop%, k) + SQR(CSNG(ocave& ^ 2 + oqave& ^
      2))

'            PRINT USING "ocave(##) = ######   oqave(##) = ######"; k; ocave& /
      (blength% / 2); k; oqave& / (blength% / 2) '***diagnostic
```

```
        END IF

      NEXT ivolt2
     NEXT iground
   NEXT icurrent

   FOR k = 1 TO elec%
     Vsum!(projloop%, k) = Vsum!(projloop%, k) / CSNG((elec% / 2) * (blength% / 2))
   NEXT k

' Record time at which projection work ends

   endtime = TIMER

   PRINT USING " Projection ### of ### acquired in ##.## seconds"; projloop%; proj%;
     endtime - starttime!
' proj% printout on previous line added by TJO 11/6/96

   PRINT USING " Mean Cross Electric Voltage, injection at 1 = #####.## ";
     Vsum!(projloop%, 1)
   PRINT

  NEXT projloop%

'  CLOSE #2 '*** diagnostic file

' Set bit 04 to zero to disable muxes and shut off current injection at
'   electrodes --- dlg, 11/6/97

  digitalio.dioport = PORT0
  digitalio.diovalue = 0
  istat = dt.set.dio(iunit, ihandle, digitalio)
  IF istat <> 0 THEN CALL exit.error(istat, "dt.set.dio (current mux PORT 0)")

  digitalio.dioport = PORT1
  digitalio.diovalue = 0
  istat = dt.set.dio(iunit, ihandle, digitalio)
  IF istat <> 0 THEN CALL exit.error(istat, "dt.set.dio (current mux PORT 1)")

  istat = dt.ct31.gate.delay(iunit, 0, 0, 1, 2)
  IF istat <> 0 THEN CALL exit.error(istat, "dt.ct31.gate.delay (current mux)")

' Delete buffers and terminate communication with board
' 'dt.delete.buffer' and 'dt.ct31.terminate' moved outside electrode
'   loop, into subroutine where they can be called under normal or
'   abnormal conditions (in 'exit.error') --- dlg, 9/11/97

  CALL cleanup(dummy)

  istat = dt.terminate(ihandle)
  IF istat <> 0 THEN CALL exit.error(istat, "dt.terminate")

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

'****************** END OF DATA ACQUISITION LOOP *********************

' Call subroutines to compute voltage statistics and output results

  CALL eistats(elec%, proj%, (blength% / 2), carrsum&(), quadsum&(), Vsum!(), resp$)

' DO...LOOP WHILE added to eliminate bug in output options -- dlg, 5/28/97
```

```
LOOP WHILE resp$ = "C" OR resp$ = "c"

PRINT : PRINT " Program stop.": PRINT

CLOSE
END

REM $STATIC
SUB cleanup (dummy)
SHARED iunit, isection, ibhandle, ihandle

istat = dt.delete.buffer(iunit, isection, ibhandle)
IF istat <> 0 THEN PRINT "dt.delete.buffer, istat = "; istat

istat = dt.ct31.terminate(ihandle)
IF istat <> 0 THEN PRINT "dt.ct31.terminate, istat = "; istat

END SUB
```