APPENDIX A
TABLE OF TRANSDUCER CHARACTERISTICS

This table lists the transducer characteristics for each transducer used in the experiment. Included in the table are nominal parameters, measured parameters, and calculated parameters along with the settings used on the Panametrics 5900 pulser/receiver.

Serial No.	98C164	00064	98C151	00068	00059	98C160	V380
Date Data Taken	6/18/98	3/22/99	7/27/98	2/23/99	2/22/99	4/12/98	9/1/99
Nominal Parameters							
Center Frequency (MHz)	9	9	6	6	3	3	7.5
Diameter (mm)	19.05	19.05	19.05	19.05	19.05	19.05	19.05
Focal Length (mm)	57.15	38.1	38.1	19.05	38.1	19.05	85.73
f/#	3	2	2	1	2	1	4.5
Measured Parameters							
Center Frequency (MHz)	8.37	8.23	5.61	5.58	2.83	2.82	6.55
~3dB Bandwidth (MHz)	1.71	1.75	0.73	0.67	0.27	0.24	4.60
Fractional Bandwidth (%)	20.39	21.28	12.93	12.02	9.55	8.55	70.23
Wavelength in Water (mm)	177.75	180.93	265.10	266.94	526.32	523.36	227.20
~20dB Pulse Duration (ns)	610.00	621.80	1414.20	2740.00	3600.00	4569.00	515.00
~6dB Beamwidth (mm)	526.70	436.70	541.40	325.80	420.30	213.10	859.10
~6dB Depth of Focus (mm)	9456.90	5465.80	6897.90	2501.10	5173.90	1890.70	15940.00
Focal Length (mm)	51.54	39.35	40.24	20.88	42.97	21.14	70.69
f1 (MHz)	7.52	7.35	5.25	5.24	2.69	2.71	4.25
f2 (MHz)	9.23	9.10	5.98	5.91	2.96	2.95	8.85
Calculated Parameters							
~6dB Beamwidth (mm)	494.38	192.09	575.72	300.77	610.18	597.13	866.69
~6dB Depth of Focus (mm)	9212.16	1366.26	8376.51	2270.38	4739.25	4564.33	22149.69
Panametrics 5900 Settings							
PRF (kHz)	2	2	2	2	2	1.25	2
Energy (mJ)	16	16	16	16	16	50	16
Damping (W)	26	26	26	26	26	15	26
High Pass Filter (MHz)	1	1	1	1	1	100	1
Low Pass Filter (MHz)	20	20	20	20	20	35	20
Attenuation (dB)	0	0	0	0	0	0	0
Gain (dB)	26	26	26	26	26	20	26
Water Temperature (deg C)	22.4	22	22.4	22.4	22	19.5	22

APPENDIX B
DATA ACQUISITION PROGRAM

This program acquires all the data needed for this thesis. It was written in Visual C++ 5.0.

```
/* -------------------
This is a program which finds the beam
axis of a transducer and then collects
RF signals along that path.  It was
created from the existing 2D program
used for developing transducer beam
scans. The program can be divided into
two parts: the first finds the beam
axis of the transducer, and the second
follows along the beam axis collecting
signals.

To do the first part, the program scans
Axis 2,finds the position of the
maximum PII, and then returns to the
start position. It then scans Axis 3
for the position of the max PII and
then returns to the start position. The
program then moves the hydrophone a
user- specified distance from the
transducer and then repeats the Axis 2
and Axis 3 scans for the max PII in
each direction.

This is done 5 times, and then a best-
fit line is calculated from the
positions of the max PII (5 each for
Axis 2 and Axis 3). This best-fit line
is the beam axis for the transducer.

With the beam axis calculated, the
hydrphone is moved back to the start
and then started on a point on the beam
axis.  The hydrophone increments at
user- specified points along the beam
axis and the RF signal is collected.

Modified by Jason Sempsrott from
programs written by Karen Topp, based
on ROWC.C by Kay Raum, and using ideas
from Nadine Smith, Kate Frazier,
and Dudley Swiney.

Two files will be created: a binary
file with extension .bin contains the
unscaled integer signal points.
The second file is an ASCII file, which
contains waveform and scan information.
-----------------------------------*/

#include <conio.h>
#include <stdio.h>
#include <string.h>

#include <stdlib.h>
#include <math.h>
#include <windows.h>
//#include "c:\qc2\include\msgraph.h"
#include <time.h>
#include "decl-32.h"

#define ERR     (1<<15)      /* Error
detected*/

#define TIMO    (1<<14)
/*Timeout*/

#define RQS     (1<<11)      /* Device
needs service  */

#define N 5/*number of points to fit*/

/* function prototypes */
void introduction(void);
void init_parameter(char [20]);
void init_files(void);
void init_device(void);
void measure(char [20]);

void line_calc(double *coord1, double
*coord2, double *line);
void scan(void);
void printMatrix(double *matrix, int
n);
//void measure2(char [20]);

FILE
       *fopen(),
       *ptr_Dat,
       *ptr1_Dat,
       *ptr2_Dat,
       *ptr_Bin;

double line2[2],    /* will contain
coefficients for axis1-axis2 line*/
       line3[2],   /* will contain
coefficients for axis1-axis3 line*/
       coordd1[N],
          coordd2[N],
          coordd3[N];

char

fileDat[50],file1Dat[50],file2Dat[50],
        fileBin[50],
        fileout[6],
        filenam[50]="d:\\data\\",
        filename[50],filename1[50],
        go1[40],
        go2[40],
        go3[40],
        gostep[30],
        comeback1[40],
        comeback2[40],
```

```
        comeback3[40],
        distance[20],
        wait1[20],
        wait2[20],
        wait3[20],
        wfm[2250],
        scopestat[100],
        scopesettle[10],
needshift1[5],  /* Do we need to shift
the time window -- y/n */
needshift2[5],  /*  for 2nd scan axis
*/
shiftdir[5],   /* Which direction to
shift window */
shiftstring[50],  /* How much do we
need to shift the time */
needavg[5],     /* For averaging trace
A */
avgstring[50],
sweeps[5],      /* How many sweeps for
averaging */
nomath[50],     /* For no math on trace
A */
        op,
        asr1[4],
        asr2[4],
        inter[20],
        ax1incr[5],    // axis1 step
size used for scanning along beam axis
        ax1tot[5],  //axis 1 total
distance for beam scan
        interval1[5];

char ax_number1,
     ax_number2;

unsigned long int   inter1,

       ax1incr_int,

       inter2,

       scopeset;

float              sos,
shifttime,shifttime1;

int     gpib0,
       dev0,
       dev1,
       dev2,
       dev3;


struct data
  {
  unsigned long int
number_point,numscan1,numscan2,numscan3
;
  float
xincr,xmult,xzero,ymult,yzero,timediv,a
xincr;
  };

struct data info;
```

```
main()
  {
  ibrsc(gpib0,1);
  introduction();
  init_parameter(distance);
  init_device();
  init_files();
  measure(distance);
  //measure2(distance);

  // compute line, and scan along it
  line_calc(coordd1, coordd2, line2);
// gives y=Ax+B
  line_calc(coordd1, coordd3, line3);
// gives z=Cx+D
  //how_many();
              // How many scans are to
be done?
  scan();

  ibloc(dev0);/* put scope back to
local mode */
  ibrsc(gpib0,0);

  return 0;
  }

void introduction()
  {
//  _clearscreen(_GCLEARSCREEN);
  printf("\n\n             2D.C  ");
  printf("\n      Scans in 2 directions
");
  printf("\n\n      Data is saved in
C:\\DATA\\  -- 'erase' your data
afterwards!  ");
  printf("\n\n\n Trace A on the LeCroy
scope will be the signal collected");
  printf("\n\n\n\n");
  }


void init_parameter(char distance1[20])
  {
  static char       //interval1[5],
                      interval2[5],
                      //distance1[20],
                      distance2[20],
                      direction1[2],
                      direction2[2],
                      stemp[4];

  unsigned long   check1, check2;


  float    dist1, dist2,ax1totl;

  float   temp, t;

  /* ---------------------------------
Calculate speed of sound in water
------------------------- */

  printf("\n\n Enter water temperature
in degrees C:  ");
```

```
  scanf("%s",stemp);
  temp=(float)atof(stemp);
  printf("\n   Temp = %f degrees
C",temp);
  t = temp/100;
  sos = (float)(1402.7 + (488*t) -
(482*t*t) + (135*t*t*t));
  printf("\n  Speed of sound in water =
%f m/s",sos);


/* -------------------------
          Set up file names
------------------------------ */

  printf("\n\n\n\n Enter a data
filename, up to 8 characters    ");
  scanf("%s",fileout);
  strcat(filenam,fileout);

  strcpy(fileBin,filenam); /* COPY name
to file for binary data */
  strcat(fileBin,".bin");  /*  ADD
".bin" to indicate binary waveform
data*/
  strcpy(fileDat,filenam); /* COPY for
.dat file */
  strcat(fileDat,".dat");  /*  ADD
".dat" to indicate the scope data */
  strcpy(file1Dat,filenam);
  strcat(file1Dat,".dat1");
  strcpy(file2Dat,filenam);
  strcat(file2Dat,".dat2");
/* -------------------------
   Info for Daedal direction axes
------------------------------ */
/* ----------------------------------
             FIRST AXIS
  -------------------------- */

 /* printf(" Enter the first scanaxis
(1-3) : ");
  ax_number1=getche();
  op=getche();*/
  //printf("\n Moving in the positive
(p) or negative (n) direction for
Daedal?: ");
  //scanf("%s", direction1);
  strcpy(direction1,"p");
  printf("\n Axis 2 and 3 stepsize (in
um)    : ");
  scanf("%s",interval1);
  printf(" Axis 2 and 3 total length
(in MM -- can use decimal)  : ");
  scanf("%s",distance1);
  check1 = atoi(distance1);
  if (check1 > 300)
     {
     printf("Cannot travel over 300
millimeters!");
     exit(0);
     }

  strcpy(inter,interval1);
  inter1=atoi(interval1);
  printf("interval1=%s\n",interval1);
  dist1=(float)atof(distance1);
/* ascii to float! */

info.numscan1=(long)(1000*dist1/inter1+
1);  /* truncates to integer */

ltoa(info.numscan1*inter1,distance1,10)
; /* for motor commands in um */

 /* switch(ax_number1)
        {
        case'1':
             dev1=ibfind("axis1");
             strcpy(go1, "MN A5 1V1
D");
             strcpy(comeback1, "MN A5
1V3 D");
             strcpy(wait1," 1R ");
             break;
        case'2':*/
             dev1=ibfind("axis2");
             strcpy(go1, "MN A5 2V1
D");
             strcpy(comeback1, "MN A5
2V3 D");
             strcpy(wait1," 2R ");
       /*    break;
           case '3':
             dev1=ibfind("axis3");
             strcpy(go1, "MN A5 3V1
D");
             strcpy(comeback1, "MN A5
3V3 D");
             strcpy(wait1," 3R ");
             break;
        default:
             printf("\n unknown
axis1");
             op=getche();
//
_clearscreen(_GCLEARSCREEN);
             exit(0);
        }*/


     {
     strcat(go1, "+");
     strcat(go1, interval1);
     strcat(go1, " G ");
     strcat(comeback1,"-");
     strcat(comeback1, distance1);
     strcat(comeback1, " G ");
     }


  /*  ---------------------------------
----------------------------
           Is the first scan axis in
the axial direction?
  -----------------------------------
--------------------------- */

  printf("\n Is the first scan axis in
the axial direction? ");
```

```
  printf("\n    i.e. Do you need the
window to shift in time (y/n) ?:  ");
  scanf("%s", needshift1);
  /* printf(" You said %s.",
needshift1); */
  if (needshift1[0] == 'y')    /*
calculate amount to shift window */
     {
     printf("\n    What direction must
the time window shift? ");
     printf("\n    i.e. positive (p) or
negative (n) ?:  ");
     scanf("%s", shiftdir);
     shifttime = (float)(2 *
((float)inter1/1e6) / sos);
     printf("\n Time shift of window
for each step is %g", shifttime);
     }


  /* ---------------------------------
------------------------------
                        SECOND AXIS
     ---------------------------------
---------------------------- */


  //printf("\n Moving in the positive
(p) or negative (n) direction for
Daedal?: ");
  //scanf("%s", direction2);
  printf("Transducer should face in the
negative axis 1 direction");
  strcpy(direction2,"n");
  printf("\n Distance between maximum
planes   (in um)      : ");
  scanf("%s",interval2);

  printf(" Axis 1 total length (in MM -
- can use decimal)  : ");
  scanf("%s",distance2);

  printf("\nEnter increment for beam
axis scan with respect to axis 1(in
um): ");
  scanf("%s",ax1incr);
  printf("\nEnter total distance for
beam axis with respect to axis 1(in mm)
");
  scanf("%s",ax1tot);


  check2 = atoi(distance2);
  if (check2 > 300)
     {
     printf("Cannot travel over 300
millimeters!");
     exit(0);
     }
  inter2=atoi(interval2);
            /*interval for finding
max*/
  ax1incr_int=atoi(ax1incr);
                   /*interval for
finding beam axis*/
  dist2=(float)atof(distance2);
/* ascii to float! */
  ax1totl=(float)atof(ax1tot);

info.numscan2=(long)(1000*dist2/inter2+
1);  /* truncates to integer */

info.numscan3=(long)(1000*ax1totl/ax1in
cr_int+1);


ltoa(info.numscan2*inter2,distance2,10)
;

   /* switch(ax_number2)
         {
         case'1': */
              dev2=ibfind("axis1");
              strcpy(go2, "MN A5 1V1
D");
              strcpy(comeback2, "MN A5
1V3 D");
              strcpy(wait2," 1R ");
              /*break;
         case'2':
              dev2=ibfind("axis2");
              strcpy(go2, "MN A5 2V1
D");
              strcpy(comeback2, "MN A1
2V3 D");
              strcpy(wait2," 2R ");
       /*     break;
         case '3':
              dev2=ibfind("axis3");
              strcpy(go2, "MN A5 3V1
D");
              strcpy(comeback2, "MN A5
3V3 D");
              strcpy(wait2," 3R ");
              break;
         default:
              printf("\n unknown
axis2");
              op=getche();
//
_clearscreen(_GCLEARSCREEN);
              exit(0);
         }*/


     strcat(go2, "-");
     strcat(go2, interval2);
     strcat(go2, " G ");
     strcat(comeback2,"+");
     strcat(comeback2, distance2);
     strcat(comeback2, " G ");


  /*  ---------------------------------
------------------------------
            Is the second scan axis in
the axial direction?
  ---------------------------------
---------------------------- */
```

```
  printf("\n Is the second scan axis in
the axial direction? ");
  printf("\n    i.e. Do you need the
window to shift in time (y/n) ?:  ");
  scanf("%s", needshift2);
  /* printf(" You said %s.",
needshift1); */
  if (needshift2[0] == 'y')    /*
calculate amount to shift window */
     {
     printf("\n    What direction must
the time window shift? ");
     printf("\n    i.e. positive (p) or
negative (n) ?:  ");
     scanf("%s", shiftdir);
     shifttime = (float)(2 *
((float)inter2/1e6) / sos);

shifttime1=(float)(2*((float)ax1incr_in
t/1e6)/sos);//shifttime for beam axis
scan
     printf("\n Time shift of window
for each step is %g", shifttime);
     }

  /* ---------------------------------
------------------------------
                how long for scope to
wait?
  -----------------------------------
------------------------------ */
  printf("\n\n Enter a 'wait time'
integer value for the scope to
settle.");
  printf("\n   (between 0 and a few
1000 --- 1000 is about 1 second):   ");
  scanf("%s",scopesettle);
  scopeset=atoi(scopesettle);

  /* ---------------------------------
------------------------------
                   Average of Trace A?
  -----------------------------------
------------------------------ */

  printf("\n\n Do you want averages of
trace A (y/n) ?: ");
  scanf("%s", needavg);
  if (needavg[0] == 'y')
     {
     printf("\n   Enter number of
sweeps for average (integer up to
1000): ");
     scanf("%s",sweeps);
     }

  }


void init_device(void)
  {
  static char            quest1[50],
                      quest2[50],
                      quest3[50];

  gpib0=ibfind("gpib0");

/* ---------------------------------
------------------------------
                  Initialize scope
  -----------------------------------
------------------------------ */

  dev0=ibfind("LECROY");   /* ID:
Lecroy */

  ibwrt(dev0,"*CLS",4);  /*  clear
status registers on scope */
  ibwrt(dev0,"WAVEFORM_SETUP SP, 0, NP,
0, FP, 0",36);
                        /* SParse=0 =>
no interval betw. pts.
                           NumPts=0 =>
send all pts.
                           FirstPoint=0
=> start reading at 1st point */

  ibwrt(dev0,"Comm_Format OFF,WORD,BIN
",25);
  ibwrt(dev0,"Comm_ForMat?",12);
  ibrd(dev0,quest1,45);
/*      printf("\n Waveform :
%s",quest1);         */

  ibwrt(dev0,"Comm_Order LO ",14);
/* LeastSigByte first */
  ibwrt(dev0,"Comm_Order?",11);
  ibrd(dev0,quest2,45);
/*      printf("\n Byte Order :
%s",quest2);           */

  ibwrt(dev0,"Comm_Header OFF ",16);
/* Header omitted */
  ibwrt(dev0,"Comm_Header?",12);
  ibrd(dev0,quest3,45);
/*      printf("\n Header :
%s",quest3);          */


/* ---------------------------------
------------------------------
               Get info for TRACE A
(Transmit signal)
   -----------------------------------
------------------------------*/

  ibwrt(dev0,"TA:INSP?
'WAVE_ARRAY_COUNT'",27);  /* TA=trace A
*/
  ibrd(dev0,wfm,1);    /* read one char
to get rid of opening quotes */
  ibrd(dev0,wfm,100);
/*      printf("\n %s",wfm);         */
  sscanf(wfm,"WAVE_ARRAY_COUNT    : %lu
",&info.number_point);

  ibwrt(dev0,"TA:INSP?
'HORIZ_INTERVAL'",25);
  ibrd(dev0,wfm,1);
```

```
  ibrd(dev0,wfm,100);
/*      printf("\n %s",wfm);         */
  sscanf(wfm,"HORIZ_INTERVAL      : %g
",&info.xincr);

  ibwrt(dev0,"TA:INSP?
'HORIZ_OFFSET'",23);
  ibrd(dev0,wfm,1);
  ibrd(dev0,wfm,100);
/*      printf("\n %s",wfm);         */
  sscanf(wfm,"HORIZ_OFFSET        : %g
",&info.xzero);

  ibwrt(dev0,"TIME_DIV?",9);
  ibrd(dev0,wfm,100);       /* note
this doesn't come back in quotes */
/*     printf("\n %s",wfm);
*/
  sscanf(wfm,"%g", &info.timediv);

  ibwrt(dev0,"TA:INSP?
'VERTICAL_GAIN'",24);
  ibrd(dev0,wfm,1);
  ibrd(dev0,wfm,100);
/*      printf("\n %s",wfm);         */
  sscanf(wfm,"VERTICAL_GAIN       : %g
",&info.ymult);

  ibwrt(dev0,"TA:INSP?
'VERTICAL_OFFSET'",26);
  ibrd(dev0,wfm,1);
  ibrd(dev0,wfm,100);
/*      printf("\n %s",wfm);         */
  sscanf(wfm,"VERTICAL_OFFSET     : %g
",&info.yzero);

  printf("\n\n DATA FOR TRACE A ");
  printf("\n\n Sample points per A-scan
: %lu",info.number_point);
  printf("\n  Xincr : %g",info.xincr);
  printf("    Xzero : %g",info.xzero);
  printf("\n  Ymult : %g",info.ymult);
  printf("    Yzero : %g",info.yzero);
  printf("    Time/div :
%g",info.timediv);

  printf("\n\n Number of scan points:
");
  printf("\n Axis[%c] scan points
:%lu",ax_number1,info.numscan1);
  printf("\n Axis[%c] scan points
:%lu",ax_number2,info.numscan2);
  printf("\n\n");


  ibwrt(dev1," F ",3);
  ibwrt(dev2," F ",3);
  }

void init_files()
  {


  if ((ptr_Bin = fopen(fileBin,"wb"))
== NULL)
       {
       printf(" error opening binary
file \n");
       exit(0);
       }

  fclose(ptr_Bin);
  }


void measure(char distance1[20])
  {
  unsigned long int
i,j,k,t,w,u,waitscope,count,

NewDistance,newinter,newdist1,blah1,bla
h2,blah3,

       blah4,extra,imax1,burn,total2,tot
al3;
  int inr,
result,imax,newimax,dummy,couch,//maxva
lue,newmaxval,

pause,data[5000],temp,temp2,temp3,temp4
,temp5[20000],coord1[2000],coord2[2000]
,coord3[2000],temp6 ;            /* for
status result from LeCroy */
  char bugme,
comeback[40],tempos1[40],tempos2[40],ne
wdist3[40],newdist[40],

newdistance1[20],start2[40],start3[40],
go3[40],
         newdistance2[20];
  float  winpos, winpost[20],
posdiv,dist1;// NewDistance;
  static char    cmd[100],      /* for
status response string */
               curv[20000],   /* MAKE
SURE ENOUGH ROOM!!! */
               posstring[60],
name[4],interval1[5]; /* string for
current window position */


  count=1;

  winpos = info.xzero +
(info.number_point*info.xincr)/2 ;
      /*  for moving time window, if
needed (window uses center of trace) */

  /* ---------------------------------
-----------------------
       If we want an average of Trace
A...
  ---------------------------------
----------------------- */
  /*if (needavg[0] == 'y')  /* then
define trace A to be an average */
  /*   {
     strcpy(avgstring,"TA:DEF
EQN,'AVGS(C1)',SWEEPS,");
     strcat(avgstring,sweeps);
```

```
ibwrt(dev0,avgstring,strlen(avgstring))
;
     }
  else                       /* no
averaging -- transmit raw trace A */
     {
     ibwrt(dev0,"CLSW",4);
     strcpy(nomath,"TA:DEF
EQN,'ZOOMONLY(C1)'");
     ibwrt(dev0,nomath,strlen(nomath));
     }

        dev3=ibfind("axis3");
        strcpy(go3, "MN A5 3V1 D");
        strcat(go3, "+");
        strcat(go3, inter);
        strcat(go3, " G ");


  for(k=0;k<=(info.numscan2-1);k++)  /*
Rows */
        {
        ibwrt(dev1," E ",3);
        ibwrt(dev3," E ",3);
       /* if ((ptr_Bin =
fopen(fileBin,"ab")) == NULL)
             {
             printf(" error appending to
transmit binary file \n");
             exit(0);
             }*/
       strcpy(newdistance1,distance1);
       strcpy(newdistance2,distance1);
       blah1=0;
       blah2=0;
       blah3=0;
       blah4=0;


        for(i=0;i<=(info.numscan1-
1);i++)  /* Scans in each row (columns)
*/

             {

for(waitscope=0;waitscope<=scopeset;wai
tscope++)
              {
              printf("\rPlease
wait...");
              }
          /* printf(" Done with one
capture ");  */
          printf(" [%lu of
%lu]",count,(info.numscan1)*(info.numsc
an2));
          count=count+1;

         /*  if (needavg[0] == 'y')  /*
then trace A an average */
         /*     {
              ibwrt(dev0,"CLSW",4);
/* clear sweeps to start average */
         /*     result = 0;
             inr = 0;
             while( (result=0x0100&inr)
!= 256)  /* 256 means math on TA is
done */
             /*   {
                strcpy(cmd,"INR?");

ibwrt(dev0,cmd,strlen(cmd));  /* what
is scope status */
             /*
ibrd(dev0,scopestat,100);
                inr = atoi(scopestat);
                /* printf("\n Waiting
for scope: string is %s", scopestat);
*/
              /* }
             }*/

          ibwrt(dev0,"TA:WF? DAT1",11);
/* Transmit wave */

ibrd(dev0,curv,(info.number_point*2));

          /* ------------------------
-----------------------------
              Go to next step and write
last curve to file
             ------------------------
-------------------------- */

             ibwrt(dev1, go1,
strlen(go1));


             temp4=0;

          for
(j=1;j<=(info.number_point*2);j=j+2)
             {
                    count=(j-1)/2;
                    // The following
four lines contain code
                    temp=(int)curv[j];
             // that changes the
character string "curv"

      temp2=(int)curv[j+1];
      // into an integer value for
manipulation.
                    temp2=temp2 << 8;
             // "curv" is eight bits
and integers are 16 bits.
                    data[count]=temp +
temp2;
      //
      fprintf(ptr1_Dat,"%d\n",data[coun
t]);   //Storing the curv in a file.

      //data[count]=data[count]/1000;


      temp3=(data[count])*(data[count])
;

                    temp4=temp3+temp4;
```

```
      //
      fprintf(ptr2_Dat,"%d\n",temp4);
// Storing the square of the curv in a
file.



      //fputc(curv[j],ptr_Bin);


             }


                   temp5[i]=temp4;
      //           printf("\nThe max
value for curve %d is
%d.\n",i,temp5[i]);



      //           fclose(ptr1_Dat);
      //           fclose(ptr2_Dat);

       }

            while (asr1[1] != 'R')
             {


ibwrt(dev1,wait1,strlen(wait1));
              ibrd(dev1,asr1,3);
              }
          asr1[1] = 'B';
          printf("\r            ");



             /* ---------------  end of
i-loop ---------------*/


temp6=temp5[0];
             for(i=1;i<=(info.numscan1-
1);i++)  /* Scans in each row (columns)
*/
             {
                   if (temp5[i]>temp6)
                   {
                   temp6=temp5[i];
                   imax=i-1;
                   }
                   newimax=imax;
             }

      dist1=(float)atof(newdistance1);


      blah1=(long)dist1; //total
distance from start to finish

      blah2=newimax*inter1;
      coord2[k]=blah2;
      extra=blah1-blah2;
```

```
      ltoa(extra,newdist,10);

      strcpy(comeback1, "MN A5 1V3 D");

      strcat(comeback1,"-");
    strcat(comeback1, newdist);
    strcat(comeback1, " G ");

      strcpy(tempos1,comeback1);



       ibwrt(dev1, comeback1,
strlen(comeback1));

      strcpy(comeback1, "MN A5 1V3 D");

      ltoa(blah2,newdistance1,10);

      strcat(comeback1,"-");
      strcat(comeback1,newdistance1);
      strcat(comeback1," G ");

      ibwrt(dev1,comeback1,strlen(comeb
ack1));



       while (asr1[1] != 'R')
         {

ibwrt(dev1,wait1,strlen(wait1));
         ibrd(dev1,asr1,3);
         }
       asr1[1] = 'B';



  /* ---------------------------------
----------------------
      If we want an average of Trace
A...
  -------------------------------------
---------------------- */
 /* if (needavg[0] == 'y')  /* then
define trace A to be an average */
 /*    {
     strcpy(avgstring,"TA:DEF
EQN,'AVGS(C1)',SWEEPS,");
     strcat(avgstring,sweeps);

ibwrt(dev0,avgstring,strlen(avgstring))
;
     }
  else*/                    /* no
averaging -- transmit raw trace A */
     {
     ibwrt(dev0,"CLSW",4);
     strcpy(nomath,"TA:DEF
EQN,'ZOOMONLY(C1)'");
     ibwrt(dev0,nomath,strlen(nomath));
     }
printf("hello");
```

```
  //for(k=0;k<=(info.numscan2-1);k++)
/* Rows */
        //{



//printf("info.numscan1=%d",info.numsca
n1);
        for(i=0;i<=(info.numscan1-
1);i++)  /* Scans in each row (columns)
*/
          {

for(waitscope=0;waitscope<=scopeset;wai
tscope++)
              {
              printf("\rPlease
wait...");
              }
           /* printf(" Done with one
capture ");  */
           printf(" [%lu of
%lu]",count,(info.numscan1)*(info.numsc
an2));
           count=count+1;

          /* if (needavg[0] == 'y')  /*
then trace A an average */
          /*    {
              ibwrt(dev0,"CLSW",4);
/* clear sweeps to start average */
          /*    result = 0;
              inr = 0;
              while( (result=0x0100&inr)
!= 256)  /* 256 means math on TA is
done */
              /*   {
                 strcpy(cmd,"INR?");

ibwrt(dev0,cmd,strlen(cmd));  /* what
is scope status */
              /*
ibrd(dev0,scopestat,100);
                 inr = atoi(scopestat);
                 /* printf("\n Waiting
for scope: string is %s", scopestat);
*/
                 /*}
              }*/

           ibwrt(dev0,"TA:WF? DAT1",11);
/* Transmit wave */

ibrd(dev0,curv,(info.number_point*2));

           /* ------------------------
-----------------------------
              Go to next step and write
last curve to file
              ------------------------
------------------------- */


             ibwrt(dev3, go3,
strlen(go3));

       strcpy(filename1,file1Dat);
              strcpy(filename,file2Dat);
              itoa(i,name,10);

              strcat(filename,name);
              strcat(filename1,name);
              temp4=0;
            for
(j=1;j<=(info.number_point*2);j=j+2)
              {
                     count=(j-1)/2;
// The following four lines contain
code
                     temp=(int)curv[j];
// that changes the character string
"curv"

       temp2=(int)curv[j+1];
// into an integer value for
manipulation.
                     temp2=temp2 << 8;
// "curv" is eight bits and integers
are 16 bits.
                     data[count]=temp +
temp2;
       //
       fprintf(ptr1_Dat,"%d\n",data[coun
t]);   //Storing the curv in a file.

       //data[count]=data[count]/1000;


       temp3=(data[count])*(data[count])
;

                     temp4=temp3+temp4;

       //fputc(curv[j],ptr_Bin);
        }
                     temp5[i]=temp4;
        }
             while (asr1[1] != 'R')
              {

ibwrt(dev3,wait1,strlen(wait1));
              ibrd(dev3,asr1,3);
              }
           asr1[1] = 'B';
           printf("\r            ");

     /* ---------------  end of i-loop
---------------*/

temp6=temp5[0];
imax1=0;
             for(i=1;i<=(info.numscan1-
1);i++)  /* Scans in each row (columns)
*/
             {

                     if (temp5[i]>temp6)
                     {
```

```
      temp6=temp5[i];
                  imax1=i-1;
                  }

                  newimax=imax1;
            }

      dist1=(float)atof(newdistance2);

      blah3=(long)dist1; //total
distance from start to finish
      blah4=newimax*inter1;
      coord3[k]=blah4;
      extra=blah3-blah4;
      ltoa(extra,newdist3,10);
      strcpy(comeback3, "MN A5 3V3 D");
      strcat(comeback3,"-");
    strcat(comeback3, newdist3);
    strcat(comeback3, " G ");
       ibwrt(dev3, comeback3,
strlen(comeback3));

      strcpy(comeback3, "MN A5 3V3 D");
      ltoa(blah4,newdistance2,10);
      strcat(comeback3,"-");
      strcat(comeback3,newdistance2);
      strcat(comeback3," G ");
      ibwrt(dev3,comeback3,strlen(comeb
ack3));



       while (asr1[1] != 'R')
         {

ibwrt(dev3,wait1,strlen(wait1));
          ibrd(dev3,asr1,3);
          }
       asr1[1] = 'B';

/*     if (needshift1[0]=='y')  /* then
put time window back */
/*          {
            winpos = info.xzero +
(info.number_point*info.xincr)/2 ;  /*
orig. position */
/*          posdiv = winpos /
(10*info.timediv) * 10;

gcvt((double)posdiv,5,posstring);
            strcpy(shiftstring, "TA:HPOS
");
            strcat(shiftstring,
posstring);
            ibwrt(dev0, shiftstring,
strlen(shiftstring) );
            }  */

       ibwrt(dev3," F ",3);

       ibwrt(dev1," F ",3);

       //fclose(ptr_Bin);
```

```
////////////////////////////////////
////////////////////////////////////
//////



       ibwrt(dev2," E ",3);

       /*while (asr2[1] != 'R')
             {
             printf("asr2=%c
\r",asr2[1]);

ibwrt(dev2,wait2,strlen(wait2));
             ibrd(dev2,asr2,3);
             }*/

       asr2[1] = 'B';

       /* ------------------------------
---------------------------
            If time shift of window is
needed for second scan axis
       ------------------------------
--------------------------- */

       if (needshift2[0]=='y')
          {
          if (shiftdir[0]=='p')
             {
                   winpos = winpos +
shifttime/2;
             }
               else
             winpos = winpos -
(shifttime);
               posdiv = winpos /
(10*info.timediv) *10;  /* pos. in div.
*/
                 /* note LeCroy needs
position in terms of divisions */

//printf("posdiv=%f\n",posdiv);

gcvt((double)posdiv,5,posstring);  /*
convert position to string */
          strcpy(shiftstring, "TA:HPOS
");
          strcat(shiftstring,
posstring);
          ibwrt(dev0, shiftstring,
strlen(shiftstring) );
          }

       ibwrt(dev2, go2, strlen(go2));
        for(w=1;w<=1000;w++)
  {
        for(i=1;i<=10;i++)
        {
              printf("i=%d\r",i);
        }
  }
```

```
        //printf("inter2=%d\n",inter2);
        coord1[k]=k*inter2;
        ibwrt(dev2," F ",3);


        }  /* --------------- end of k-
loop --------------*/



      if (needshift2[0]=='y')  /* then
put time window back */
     {
     winpos = info.xzero +
(info.number_point*info.xincr)/2 ;  /*
orig. position */
     posdiv = winpos /
(10*info.timediv) *10;
     gcvt((double)posdiv,5,posstring);
     strcpy(shiftstring, "TA:HPOS ");
     strcat(shiftstring, posstring);
     ibwrt(dev0, shiftstring,
strlen(shiftstring) );
     }

  ibwrt(dev2," E ",3);
  ibwrt(dev2, comeback2,
strlen(comeback2));
  printf("comeback2=%s",comeback2);
   for(w=1;w<=1000;w++)
  {
        for(i=1;i<=10;i++)
        {
              printf("i=%d\r",i);
        }
  }
  /*while (asr2[1] != 'R')
       {
       ibwrt(dev2,wait2,strlen(wait2));
       ibrd(dev2,asr2,3);
       }*/
  asr2[1] = 'B';

  ibwrt(dev2," F ",3);

  ibwrt(dev0,"CLSW",4);  /* clear
sweeps */
  ibwrt(dev0,"*CLS",4);  /* clear
status registers */

  total2=0;total3=0;
  for(k=0;k<=(info.numscan2-1);k++)
  {
        total2=coord2[k]+total2;
        total3=coord3[k]+total3;

  }

   total2=(long)total2/k;
   total3=(long)total3/k;

ltoa(total2,start2,10);
ltoa(total3,start3,10);

     dev1=ibfind("axis1");
        dev2=ibfind("axis2");
        dev3=ibfind("axis3");

        ibwrt(dev1," E ",3);

        ibwrt(dev3," E ",3);

        ibwrt(dev2," E ",3);

strcpy(go1,"MN A5 1V1 D");
strcat(go1,"+");
strcat(go1,comeback2);
strcat(go1," G ");

 for(w=1;w<=1000;w++)
  {
        for(i=1;i<=10;i++)
        {
              printf("i=%d\r",i);
        }
  }

strcpy(go2,"MN A5 2V1 D");
strcat(go2,"+");
strcat(go2,start2);
strcat(go2," G ");

 for(w=1;w<=1000;w++)
  {
        for(i=1;i<=10;i++)
        {
              printf("i=%d\r",i);
        }
  }
strcpy(go3,"MN A5 3V1 D");
strcat(go3,"+");
strcat(go3,start3);
strcat(go3," G ");

       ibwrt(dev1," F ",3);

       ibwrt(dev3," F ",3);

        ibwrt(dev2," F ",3);

/* cast coord data to double */
       for (i=0; i<N; i++)
       {
              coordd1[i] = (double)
coord1[i];
              coordd2[i] = (double)
coord2[i];
              coordd3[i] = (double)
coord3[i];
       }
}

/* This function calculates the
coefficients for a best fit line to
 M data points in the least squares
sense. The coefficients are for
 an equation of the form: y=Ax+B, where
A and B are constants.
 The coefficients are stored in a 2-
dim. array where the
```

```
 first element is A, and the second is
B. Arguments
 are: pointers to the arrays containing
the data points, and a pointer to
 the array which will serve as the
destination of the results.
 N is assumed to be defined as the
number of data points.
written by Adam Wunderlich, 11/1/98
*/
void line_calc(double *coord1, double
*coord2, double *line)
{

   /* local variables */
         double A[N][2],
          At[2][N],     /* transpose of
A */
          B[N],
          C[2][2] = {0},   /* C = At*A
*/
          D[2] = {0},      /* D = At*B
*/
          m,     /* multiplier */
    /* pointers to matrices */
       *aptr,
       *bptr,
       *atptr,
       *cptr,
       *dptr,

   /* temp. variables */
       ctemp,
       dtemp;

    /* counter variables */
   int i,
       j,
       k,
       r,
          l;

   aptr = &A[0][0];
   bptr = B;
   atptr = &At[0][0];
   cptr = &C[0][0];
   dptr = D;


   /* fill A */
   for (i=0; i<N; i++)
   {
      A[i][0] = coord1[i];
      A[i][1] = 1;
   }

   /* fill A transpose */
   for (i=0; i<N; i++)
   {
      for (j=0; j<2; j++)
         At[j][i] = A[i][j];
   }

   /* fill B */
   for (i=0; i<N; i++)
      B[i] = coord2[i];

   /* get At*A and put the result in C
*/
   for (k=0; k<2; k++)  /* increment
row in array At */
   {
      for (i=0; i<2; i++)  /* increment
column in array A */
    {
         for (j=0; j<N; j++) /*
increment column in At */
         {
            /* dot rows of matrix At
with columns of matrix A using */
            /* address arithmetic */
            *(cptr+ i+ k*2) +=
(*(atptr+ k*N +j)) * (*(aptr+ i +
j*2));
         }
     }
   }

   /* get At*B and put the result in D
*/
   for (k=0; k<2; k++)  /* increment
row in array At */
   {
          for (j=0; j<N; j++) /*
increment column in At */
      {
         /* dot rows of matrix At with
columns of matrix B using */
         /* address arithmetic */
         *(dptr+ k) += (*(atptr+ k*N
+j)) * (*(bptr+ j));
      }

   }
  // printf("matrix C:  \n");
   printMatrix(cptr,2);
  // printf("matrix D: \n");
   for (i=0; i < (2); i++)  /* loop
through the number of elements */
   {
   //   printf("%lf ", *(dptr+i)); /*
use address arithmetic, increment */
                                    /*
address  */
   //   printf("\n");
   }

   /* solve the normal equations Cx=D
using gaussian elimination with
      partial pivoting -- this is a 2x2
system, the result is placed
      in line */

   for (k=0; k<1; k++)
   {
      for (r=(k+1); r<2; r++)
      {
         /* partial pivoting */
         if (C[k][k] < C[r][k])
         {
```

```
            /* switch row k and row r
*/
            for (l=0; l<2; l++)
            {
               ctemp = C[r][l];
               C[r][l] = C[k][l];
               C[k][l] = ctemp;
               dtemp = D[r];
               D[r] = D[k];
               D[k] = dtemp;
            }
         }

         /* substract multiple of kth
row from rth row */
         m = (C[r][k])/(C[k][k]);
         for (j=0; j<2; j++)
                        {
            C[r][j] = (C[r][j] -
m*C[k][j]);
         }
            D[r] = D[r] - m*D[k];
      }
   }

printMatrix(cptr,2);

   for (i=0; i < (2); i++)  /* loop
through the number of elements */
   {
    //  printf("%lf ", *(dptr+i)); /*
use address arithmetic, increment */
                                  /*
address  */
   //   printf("\n");
   }

line[1] = D[1] / C[1][1];
line[0] = (D[0] - line[1]*C[0][1]) /
C[0][0];

//printf("coefficients: %lf %lf \n",
line[0], line[1]);

return;
}

/*-------------------------------------
----------------------*/
/* This function takes two arguments.
The first is a pointer to the    */
/* first element of an array.  The
second argument tells how many rows  */
/* and columns are in the array.  The
function will print the matrix to */
/* the screen, one row per line.
*/

void printMatrix(double *matrix, int n)
{
   /* declare local variables */
   int i;  /* counter variable for loop
*/

   /* print the matrix */
   printf("         ");
   for (i=0; i < (n*n); i++)  /* loop
through the number of elements */
   {
      //printf("%lf ", *(matrix+i)); /*
use address arithmetic, increment */
                                  /*
address  */
              if (((i+1)%n) == 0)  /*
if a multiple of N elements have */
                           /* been
printed, line return */
         printf ("\n         ");
   }
   printf("\n");
   return;  /* void return */
}  /* end function */

/*-------------------------------------
----------------------*/
/* This function moves the hydrophone
from the origin to the intercept
   of the line with the axis2-axis3
plane.  Then the hydrophone scans
   along the line with stepsizes and
total length as specified by the user
   with respect to axis1.  Note that I
often refer to axis1, axis2, and
   axis3, as x, y, and z respectively.
The line is defined as:
   x = (y-B)/A = (z-D)/C , where line2
= [A,B] and line3 = [C,D].
   written by Adam Wunderlich, 11/21/98
*/
void scan(void)
{

      // local variables

      long int pos[3] = {0},  //
coordinates of current position
            ax1, ax2, ax3,      //
increments for axis1, axis2, and axis3
            ax1long,
      //total beam axis 1 distance
            temp;

      unsigned long int
i,j,k,w,u,waitscope,count;
    int inr, result, axincr;
/* for status result from LeCroy */
    float  winpos, posdiv, posf[3],
magnitude, ax1f;

      char
            ax2c[5], ax3c[5],  //
increments as strings, note that these
// must be positive

      axis1[10],axis2[10],axis3[10];


    static char    cmd[100],    /* for
status response string */
```

```
                    curv[20000],    /*
MAKE SURE ENOUGH ROOM!!! */
                  posstring[60];    /*
string for current window position */

      count=1;

      winpos = info.xzero +
(info.number_point*info.xincr)/2;

      // open binary file
     if ((ptr_Bin =
fopen(fileBin,"wb")) == NULL)
            {
            printf(" error opening
binary file \n");
            exit(0);
            }

      // cast parameters specified by
user to long int
      ax1 = atol(ax1incr);
      // axis1 increment, assume
positive value
      ax1long=atol(ax1tot);
      // axis1 length
      ax1long=ax1long*1000;

      // update numscan2 for number of
scan points
      info.numscan3=ax1long/ax1 +1;
      printf("numscan3 = %ld \n",
info.numscan3);

      //  move to intercept of line
with y-z plane

      // move to y intercept

      ax2 = (long)line2[1];

      // printf("ax2=%ld\n",ax2);
      // increment is an absolute value
      temp = abs(ax2);
      ltoa(temp, ax2c, 10); // change
long to string

      ibwrt(dev1," E ",3);
      ibwrt(dev2," E ",3);
      ibwrt(dev3," E ",3);
      dev1=ibfind("axis1");
      dev2 = ibfind("axis2");
      strcpy(go2,"MN A5 2V1 D");
      strcpy(wait2," 2R ");

      if (line2[1] >= 0)
            strcat(go2,"+");
      else
            strcat(go2,"-");

      strcat(go2, ax2c);
      strcat(go2, " G ");
      ibwrt(dev2, go2, strlen(go2));

for(w=1;w<=1000;w++)
  {
        for(i=1;i<=100;i++)
        {
              printf("i=%d\r",i);
        }
  }

      // move to z intercept
      ax3 = (long)line3[1];
      //printf("ax3=%ld\n",ax3);

      temp = abs(ax3);
      ltoa(temp, ax3c, 10);
      // long to ascii
      dev3 = ibfind("axis3");
      strcpy(go3,"MN A5 3V1 D");
      strcpy(wait2," 2R ");

      if (line3[1] >= 0)
            strcat(go3,"+");
      else
            strcat(go3,"-");

      strcat(go3, ax3c);
      strcat(go3, " G ");
      ibwrt(dev3, go3, strlen(go3));


for(w=1;w<=1000;w++)
  {
        for(i=1;i<=10;i++)
        {
              printf("i=%d\r",i);
        }
  }

    // update pos[]
    pos[1] = ax2;   // update position
for axis2
      pos[2] = ax3;   // update
position for axis3

      ibwrt(dev0,"TA:WF? DAT1",11);
/* Transmit wave */
      ibrd(dev0,curv,(info.number_point
*2));

      // take curve for starting point
      for
(j=1;j<=(info.number_point*2);j++)
      {
            fputc(curv[j],ptr_Bin);
      }

      for (i=0; i<(info.numscan3 -1)
;i++)
      {
            strcpy(wait1," 1R ");
            strcpy(wait2," 2R ");
            strcpy(wait3," 3R ");
            // move to new position
            ibwrt(dev1," E ",3);
            ibwrt(dev2," E ",3);
            ibwrt(dev3," E ",3);
            strcpy(go1,"MN A5 1V1 D");
```

```
            strcpy(go2,"MN A5 2V1 D");
            strcpy(go3,"MN A5 3V1 D");
            strcat(go1,"-");
            // axis1 increment is in
negative direction, so choose other
            // increment directions
accordingly
            if (line3[0] >= 0)
                  strcat(go3,"+");
            else
                  strcat(go3,"-");

            if (line2[0] >= 0)
                  strcat(go2,"+");
            else
                  strcat(go2,"-");

            ax2=line2[0]*ax1;
            //printf("ax2=%ld\n",ax2);
            ax3=line3[0]*ax1;
            //printf("ax3=%ld\n",ax3);
            temp = abs(ax2);
            ltoa(temp, ax2c, 10);
            temp = abs(ax3);
            ltoa(temp, ax3c, 10);

            strcat(go2,ax2c);
            strcat(go2," G ");
            ibwrt(dev2, go2,
strlen(go2));  //increment axis 2

      ibwrt(dev2,wait2,strlen(wait1));
            strcat(go3,ax3c);
            strcat(go3," G ");
            ibwrt(dev3, go3,
strlen(go3)); //increment axis 3

      ibwrt(dev3,wait3,strlen(wait1));
            strcat(go1, ax1incr);
            strcat(go1," G ");
            ibwrt(dev1, go1,
strlen(go1));//increment axis 1

      ibwrt(dev1,wait1,strlen(wait1));


if (needshift2[0]=='y')
          {
          if (shiftdir[0]=='p')
             {
                  winpos = winpos +
shifttime1/2;
            }
              else
            winpos = winpos -
(shifttime1);
               posdiv = winpos /
(10*info.timediv) * 10;  /* pos. in
div. */
                 /* note LeCroy needs
position in terms of divisions */

//printf("posdiv=%f\n",posdiv);


gcvt((double)posdiv,5,posstring);  /*
convert position to string */
          strcpy(shiftstring, "TA:HPOS
");
          strcat(shiftstring,
posstring);
          ibwrt(dev0, shiftstring,
strlen(shiftstring) );
          }
    // update pos[]
       pos[0] = pos[0] - ax1;   //
update position for axis1
    pos[1] = pos[1] + ax2;  // update
position for axis2
       pos[2] = pos[2] + ax3;   //
update position for axis3

       //printf("pos[0]= %ld
       pos[1]=%ld
       pos[2]=%ld\n",pos[0],pos[1],pos[2
]);

  /*   strcpy(avgstring,"TA:DEF
EQN,'AVGS(C1)',SWEEPS,");
     strcat(avgstring,sweeps);

ibwrt(dev0,avgstring,strlen(avgstring))
;

             ibwrt(dev0,"CLSW",4);
/* clear sweeps to start average */
/*           result = 0;
             inr = 0;
             while( (result=0x0100&inr)
!= 256)  /* 256 means math on TA is
done */
/*              {
                strcpy(cmd,"INR?");

ibwrt(dev0,cmd,strlen(cmd));  /* what
is scope status */
/*
ibrd(dev0,scopestat,100);
                inr = atoi(scopestat);
                /* printf("\n Waiting
for scope: string is %s", scopestat);
*/
/*              }*/

          ibwrt(dev0,"TA:WF? DAT1",11);
/* Transmit wave */

ibrd(dev0,curv,(info.number_point*2));


      ibwrt(dev2,wait2,strlen(wait1));
      ibwrt(dev3,wait3,strlen(wait1));
      ibwrt(dev1,wait1,strlen(wait1));

            // take curve at this
position
      for
(j=1;j<=(info.number_point*2);j++)
             {
```

```
              fputc(curv[j],ptr_Bin);
              }

//     ibwrt(dev0,"CLSW",4);  /* clear
sweeps */
//    ibwrt(dev0,"*CLS",4);  /* clear
status registers */
for(w=1;w<=1000;w++)
  {
        for(u=1;u<=5;u++)
        {
              printf("u=%d\r",u);
        }
  }


       ibwrt(dev1," F ",3);
       ibwrt(dev2," F ",3);
       ibwrt(dev3," F ",3);
       }

       // put pos[] data into float form
       for (i=0; i<2 ; i++)
       {
              posf[i] = (float) pos[i];

       }

       // find magnitude of position
vector
       magnitude =
sqrt((posf[0])*(posf[0]) +
(posf[1])*(posf[1])
                          +
(posf[2])*(posf[2]));

       printf("magnitude = %f\n",
magnitude);


       // calculate actual increment
along beam axis
       // first, normalize position
vector to a unit vector
       for (i=0; i<2 ; i++)
       {
              posf[i] =
(pos[i])/magnitude;

       }

       ax1f = (float) ax1;
       printf("ax1f = %f \n",ax1f);

       // devide ax1 increment by axis1
component of posf[] to find
       // axincr
       info.axincr = ax1f / posf[0];
       printf("axincr = %f \n",
info.axincr);
////////////////////////////////////
////////////////////////////////
//Move back to beamscan start position
       ibwrt(dev1," E ",3);
       ibwrt(dev2," E ",3);
       ibwrt(dev3," E ",3);
       temp=fabs(pos[0]);
       ltoa(temp,axis1,10);
       // Axis 1 Final Position
       strcpy(go1,"MN A5 1V1 D");
       strcat(go1,"+");
       strcat(go1,axis1);
       strcat(go1," G ");
       ibwrt(dev1,go1,strlen(go1));
       //Move Axis 1 Back To Start
Position


for(w=1;w<=1000;w++)
  {
        for(i=1;i<=10;i++)
        {
              printf("i=%d\r",i);
        }
  }
       temp=fabs(pos[1]);
       ltoa(temp,axis2,10);
       strcpy(go2,"MN A5 2V1 D");
       if (pos[1]<0)
       {
             strcat(go2,"+");
       }
       else
             strcat(go2,"-");
       strcat(go2,axis2);
       strcat(go2," G ");
       ibwrt(dev2,go2,strlen(go2));

for(w=1;w<=1000;w++)
  {
        for(i=1;i<=10;i++)
        {
              printf("i=%d\r",i);
        }
  }
       temp=fabs(pos[2]);
       ltoa(temp,axis3,10);
       strcpy(go3,"MN A5 3V1 D");
       if (pos[2]<0)
       {
             strcat(go3,"+");
       }
       else
             strcat(go3,"-");
       strcat(go3,axis3);
       strcat(go3," G ");
       ibwrt(dev3,go3,strlen(go3));

for(w=1;w<=1000;w++)
  {
        for(i=1;i<=10;i++)
        {
              printf("i=%d\r",i);
        }
  }


       ibwrt(dev1," F ",3);
       ibwrt(dev2," F ",3);
       ibwrt(dev3," F ",3);
```

```
       if (needshift2[0]=='y')  /* then
put time window back */
     {
     winpos = info.xzero +
(info.number_point*info.xincr)/2 ;  /*
orig. position */
     posdiv = winpos /
(10*info.timediv) *10;
     gcvt((double)posdiv,5,posstring);
     strcpy(shiftstring, "TA:HPOS ");
     strcat(shiftstring, posstring);
     ibwrt(dev0, shiftstring,
strlen(shiftstring) );
     }

       // take absolute value of axincr
       info.axincr = (-1 * info.axincr);
  if ((ptr_Dat = fopen(fileDat,"w")) ==
NULL)
       {
       printf(" error opening data file
\n");
       exit(0);
       }
       fprintf(ptr_Dat,"%g %g %g %lu %lu
%lu %g %g %lu %g %g %lu",
    info.ymult, info.yzero, info.xincr,
info.number_point, info.numscan1,
    info.numscan2, info.xzero,
info.axincr, inter2, sos, shifttime,
       info.numscan3);
    fprintf(ptr_Dat,"\nymult yzero
xincr number_point numscan1 numscan2");
    fprintf(ptr_Dat," xzero stepsize1
stepsize2 sos timeshift numscan3\n");
    fclose(ptr_Dat);

       // note that the beam axis
increment (in microns) is in
info.axincr

       fclose(ptr_Bin);

       printf("pos[0] = %ld \n",
pos[0]);
       printf("pos[1] = %ld \n",
pos[1]);
       printf("pos[2] = %ld \n",
pos[2]);
}
```

APPENDIX C
DATA ANALYSIS PROGRAM

This program does the analysis described in Chapter 4. It was written in Matlab®.

```
% marconi.m
% computes PII, Pc, Pr and plots versus axial diastance,
% given beam axis measurements of a transducer made using a hydrophone
% this script loads both the bin and dat file
%
% written by Adam Wunderlich, Sept. 1998
% using complete5.m as a basis
% modified Dec. 1998 by Jason Sempsrott
% modified Aug. 1999 by Jason Sempsrott

clear all;
close all;

n      = input('Enter a filename: ','s');
fc     = input('Enter the center frequency of the transdcuer in MHz ');
dist   = input('Distance of hydrophone from transducer at start  (in cm)');
R      = input('Enter Ritec setting: ');
date   = input('Date of calibration: ','s');
who    = input('Person calibrating: ','s');
which  = input('Hydrophone that was used (type and Serial #):\n ','s');
factor = input('Enter the calibration factor in V/MPa: ');

bin = [n '.bin'];
dat = [n '.dat'];

fid    = fopen(dat, 'r');
A      = fscanf(fid,'%f',[12,1]);

ymult=A(1,1);       % y-scaling
yzero=A(2,1);       % DC offset
xincr=A(3,1);       % x-scaling, 1/(sampling rate)
number_point=A(4,1);       % # points in waveform
numscan1=A(5,1);    % # of scans for scan axis1
numscan2=A(6,1);    % # of scans for scan axis2
xzero=A(7,1);       % time for start of first wave form
stepsize1=A(8,1);   % step size for scan axis 1
stepsize2=A(9,1);   % stepsize for scan axis 2
sos=A(10,1);        % speed of sound
timeshift=A(11,1);  % window time shift
numscan3=A(12,1);   % number of scan points for beam axis

clear A;

% open and read .bin file
% assume LeCroy oscilloscope was used to take data

status = fclose(fid);
fid=fopen(bin,'rb','b');  % 'b' for LeCroy



%//////////////////////////////////////////////////////////////////////////
% compute PII
%//////////////////////////////////////////////////////////////////////////
PIItemp1=0;
```

```
PII=zeros(1,numscan3);
    for(j=1:numscan3)
        B=fread(fid,number_point,'short');  % B is a column vector
        B=B.*ymult;
        B=B-yzero;      % - for LeCroy
        PII(1,j)=xincr*sum(B .* B);
    end

x1 = ((0:numscan3-1)*stepsize1);     % calculates axis for axial distance
x1 = x1/10000;  % convert to cm from um
x = dist + x1;


figure
orient tall

% plot PII versus axial distance
subplot(3,1,1)
% want to apply smoothing function to curve
pPII   = polyfit(x,PII,6);
fPII   = polyval(pPII,x);
plot(x,fPII)
xlabel('Axial distance (mm)')
ylabel('PII')
grid
status = fclose(fid);
[max_fPII,x_fPII] = max(fPII);
x_fPII=dist+(x_fPII*stepsize1/10000); % axial distance of maximum Pc in cm

% also need derated water value for PII and position of maximum
PII3=fPII.*exp(-.069*fc*x);
[max_PII3,junk] = max(PII3);
x_PII3 = dist+(junk*stepsize1/10000); % axial position of max PII3


%//////////////////////////////////////////////////////////////////////////
%      relevent variables are:
%      x      = axial position from transducer
%      PII    = raw data obtained during scan
%      fPII   = smooth curve of PII
%      PII3   = derated PII values from smooth curve
%      x_PII3  = position of maximum PII3
%//////////////////////////////////////////////////////////////////////////

%//////////////////////////////////////////////////////////////////////////
% plot peak compressional pressure versus axial distance
%//////////////////////////////////////////////////////////////////////////

fid=fopen(bin,'rb','b');    % b for Lecroy

Pc=zeros(1,numscan3);
    for(j=1:numscan3)
        C=fread(fid,number_point,'short');  % C is a column vector
        C=C.*ymult - yzero;  % - for LeCroy
        Pc(1,j) = max(C);
    end

% Need to convert voltages to pressures
Pc=Pc/(factor);

subplot(3,1,2)
pPc    = polyfit(x,Pc,6);
fPc    = polyval(pPc,x);
plot(x,fPc)
xlabel('Axial distance (cm)')
```

```
ylabel('Pc (MPa)')
grid
status = fclose(fid);

%      need to find maximum Pc value and its position

[max_Pc,x_Pc]=max(fPc);
x_Pc=dist+(x_Pc*stepsize1/10000); % axial distance of maximum Pc in cm

% derated water value for Pc

Pc3    = fPc.*exp(-.0345*fc*x);

%      max value of Pc3 and its position
[max_Pc3,x_Pc3]=max(Pc3);
x_Pc3=dist+(x_Pc3*stepsize1/10000); % axial distance of maximum Pc3

Pc3_max_PII3=Pc3(1,junk); % derated Pc at the position of max PII3


%//////////////////////////////////////////////////////////////////////////
%      relevant variables include:
%            x      = axial distance
%            Pc     = original max compressional values
%            fPc    = smooth curve of Pc
%            x_Pc   = position of maximum Pc (cm)
%            max_Pc = value of maximum Pc (MPa)
%            Pc3    = derated water value for fPc (MPa)
%            x_Pc3  = position of maximum Pc3 (cm)
%            max_Pc3 = value of maximum Pc3 (MPa)
%            Pc3_max_PII3 = value of Pc3 located at max_PII3
%//////////////////////////////////////////////////////////////////////////

%//////////////////////////////////////////////////////////////////////////
% plot peak rarefractional pressure versus axial distance
%//////////////////////////////////////////////////////////////////////////

fid=fopen(bin,'rb','b');    % b for Lecroy

Pr=zeros(1,numscan3);
    for(j=1:numscan3)
        D=fread(fid,number_point,'short');  % D is a column vector
        D=D.*ymult - yzero;  % - for LeCroy
        Pr(1,j) = abs(min(D));
    end

% Need to convert voltages to pressures
Pr=Pr/(factor);

subplot(3,1,3)
pPr=polyfit(x,Pr,6);
fPr=polyval(pPr,x);
plot(x,fPr)
xlabel('Axial distance (cm)')
ylabel('Pr (MPa)')
grid
status = fclose(fid);

%      need to find maximum Pr value and its position

[max_Pr,x_Pr]=max(fPr);
x_Pr=dist+(x_Pr*stepsize1/10000); % axial distance of maximum Pr


%      derated water value for Pr
```

```
Pr3    = fPr.*exp(-.0345*fc*x);
%      max value of Pr3 and its position
[max_Pr3,x_Pr3]=max(Pr3);
x_Pr3=dist+(x_Pr3*stepsize1/10000); % axial distance of maximum Pr3

Pr3_max_PII3=Pr3(1,junk); % derated Pr at the position of max_PII3

MI = Pr3_max_PII3/(fc^.5);

%//////////////////////////////////////////////////////////////////////
%      relevant variables include:
%             x      = axial distance (cm)
%             Pr     = original max rarefactional values (MPa)
%             fPr    = smooth curve of Pr (MPa)
%             x_Pr   = position of maximum Pr (cm)
%             max_Pr = value of maximum Pr (MPa)
%             Pr3    = derated water value for fPr (MPa)
%             x_Pr3  = position of maximum Pr3 (cm)
%             max_Pr3 = value of maximum Pr3 (MPa)
%          Pr3_max_PII3 = value of Pr3 at position of max_PII3
%             MI     = mechanical index
%//////////////////////////////////////////////////////////////////////

%//////////////////////////////////////////////////////////////////////
% plot the time domain waveform at the maximum point in the PII plot
% (plots waveform at focus)
%//////////////////////////////////////////////////////////////////////

fid=fopen(bin,'rb','b');   % b for Lecroy

[max_PII,x_PII]=max(PII);     % finds axial location of focus
junky=dist+(x_PII*stepsize1/10000); % axial distance of maximum Pc in cm

temp=0;
for(j=1:x_PII-1)
    [wave]=fread(fid,number_point,'short'); % wave is col. vector

end




% check for zero points at end of data set, and count them
z=0;
for (i=1:number_point)
    if (wave(i,1) == 0)
        z = z+1;
    end
end


% plot the waveform at the focus centered around 0 V
figure

g = number_point-z;      % g is an intermediate variable

%twave=wave-mean(wave);
wavescaled=wave.*ymult - yzero;

% Want to develop the waveform for PII vs. Time
simple=0;
for(j=1:number_point)
       tempwave=wave(j,1).*ymult;
       tempwave1=tempwave.*tempwave;
       tempwave2=tempwave1+simple;
```

```
        simple=tempwave2;
        newwave(j,1)=tempwave2;
end



%twavescaled= wkeep(wavescaled,g,'l');  % truncate to get rid of zeros

timeaxis=(0:number_point-1)*xincr*1e6;
plot(timeaxis,wavescaled)
title('Waveform at Focus')
xlabel('Time (microseconds)')
ylabel('Amplitude (V)')
grid
focuspc=max(wavescaled);   %This gives the pc voltage value at the focus.
focuspc=focuspc/(factor);  % Converted volts to MPa
focuspr=abs(min(wavescaled));   %This gives the pr voltage value at the focus.
focuspr=focuspr/(factor);  % Converted volts to MPa

%//////////////////////////////////////////////////////////////////////////
%             relevant variables include:
%                           junky   = position of max_PII
%                           focuspc = Pc located at max_PII
%                           focuspr = Pr located at max_PII
%//////////////////////////////////////////////////////////////////////////

%//////////////////////////////////////////////////////////////////////////
% A plot of PII vs Time
%//////////////////////////////////////////////////////////////////////////
figure
plot(timeaxis,newwave);
xlabel('Time (microseconds)');
ylabel('PII Value');
grid

% Need to find limits for calculating the pulse duration
% This calculation uses the 90% and 10% points of the PII curve
% First find the upper and lower limits according to FIgure 3

upperlimit=max(newwave)*.9;
lowerlimit=max(newwave)*.1;

% Second, find the time value at the upper and lower limits
for(j=1:number_point)
        if(newwave(j,1) > (upperlimit))
                        timeaxisupper=timeaxis(1,j-1);
                        break;
        end
end
for(j=1:number_point)
        if(newwave(j,1) > (lowerlimit))
                        timeaxislower=timeaxis(1,j-1);
                        break;
        end
end

% Third, calculate a time for the pulse duration
pulse=(timeaxisupper-timeaxislower)*1.25;

%//////////////////////////////////////////////////////////////////////////
```

```
rawdata             = [x;PII;Pc;Pr];
smoothdata    = [x;fPII;PII3;fPc;Pc3;fPr;Pr3];
interests1    = [max_Pc;x_Pc;max_Pr;x_Pr];
interests2    = [max_Pc3;x_Pc3;max_Pr3;x_Pr3];
interests3    = [junky;focuspc;focuspr];
interests4    = [x_PII3;Pc3_max_PII3;Pr3_max_PII3];
interests5    = [MI;pulse;R];
everything    =
[R;pulse;max_Pc;x_Pc;max_Pr;x_Pr;max_Pc3;x_Pc3;max_Pr3;x_Pr3;junky;focuspc;focuspr;x_P
II3;Pc3_max_PII3;Pr3_max_PII3;MI];

n=input('Enter filename to save all data: ','s');
txtfile=[n,'.txt'];
fid=fopen(txtfile,'a');
fprintf(fid,'\n/////\n');
fprintf(fid,'Raw\n');
fprintf(fid,'x(cm)\t PII\t Pc(MPa)\t Pr(MPa)\n');
fprintf(fid,'%1.4f\t %2.4e\t %2.4f\t  %2.4f\n',rawdata);
fprintf(fid,'/////\n');
fprintf(fid,'Best Fit \n');
fprintf(fid,'x(cm)\t PII\t PII.3\t Pc(MPa)\tPc.3(MPa)\tPr(MPa)\tPr.3(MPa)\n');
fprintf(fid,'%1.4f\t %2.4e\t %2.4e\t %2.4f\t%2.4f\t%2.4f\t%2.4f\n',smoothdata);
fprintf(fid,'/////\n');
fprintf(fid,'Points of Interest\n');
fprintf(fid,'\nGlobal Information from Best Fit Data:\n');
fprintf(fid,'\nMax Pc(MPa)\tPosition(cm)\t Max Pr(MPa)\tPosition(cm)\n');
fprintf(fid,'%2.4f\t\t %1.4f\t\t %2.4f\t %1.4f\n\n',interests1);
fprintf(fid,'Max Pc.3(MPa)\tPosition(cm)\t Max Pr.3(MPa)\tPosition(cm)\n');
fprintf(fid,'%2.4f\t\t %1.4f\t\t %2.4f\t\t %1.4f\n\n',interests2);
fprintf(fid,'Information from waveform at focus (max of PII):\n');
fprintf(fid,'Position of max PII\t Pc (MPa)\t\t Pr (MPa)\n');
fprintf(fid,'%1.4f\t\t\t%2.4f\t\t\t%2.4f\n\n',interests3);
fprintf(fid,'Position, Pc.3, and Pr.3 at maximum of PII.3:\n');
fprintf(fid,'Position of max PII.3\t Pc.3 (MPa)\t\t Pr.3 (MPa)\n');
fprintf(fid,'%1.4f\t\t\t%2.4f\t\t\t%2.4f\n\n',interests4);
fprintf(fid,'Mechanical Index\t Pulse Duration (microsecs)\t RITEC Setting\n');
fprintf(fid,'%1.4f\t\t\t%1.4f\t\t\t\t%1.2f\n\n',interests5);
fprintf(fid,'Date of calibration:\n');
fprintf(fid,date);
fprintf(fid,'\nPerson who calibrated:\n');
fprintf(fid,who);
fprintf(fid,'\nHydrophone that was used (type and serial #):\n');
fprintf(fid,which);
fprintf(fid,'\nHydrophone conversion factor is (V/MPa):\n');
fprintf(fid,'%1.4f',factor);
fprintf(fid,'\nTransducer frequency(MHz)       %1.4f\t',fc);
fprintf(fid,'\n/////\n');

status=fclose(fid);

n1=input('Enter filename for points of interest only: ','s');
txtfile1=[n1,'.txt'];
fid=fopen(txtfile1,'a');
fprintf(fid,'%1.2f\t%2.4f\t%2.4f\t%2.4f\t%2.4f\t%2.4f\t%2.4f\t%2.4f\t%2.4f\t%2.4f\t%2.
4f\t%2.4f\t%2.4f\t%2.4f\t%2.4f\t%2.4f\t%2.4f\n',everything);
status=fclose(fid)

disp('Max Pc in MPa is:');
max_Pc
disp('Max Pr in MPa is:');
max_Pr
```

APPENDIX D
FREQUENCY ANALYSIS PROGRAM

This program takes select information from the program in Appendix B and does a frequency spectrum analysis. This program was written in Matlab®.

```
% trans.m
% computes the frequency spectrum for the waveform
% at the focus and determines the fundamental, second
% and third harmonics as functions of distance from the
% transducer.
% written by Jason Sempsrott
% Nov. 1999

fid=fopen(bin, 'rb','b');

for(j=1:numscan3)
       beta=fread(fid,number_point,'short');
       beta=beta.*ymult;
       beta=beta-yzero;
       for(i=1:number_point)
       these(i,j)=beta(i,1); % 'these' is every time-domain collected wave
       end

end

fclose(fid);
Ts=(timeaxis(2)-timeaxis(1));%time in seconds
Fs=1/Ts;
N=length(wavescaled);
K=Fs*(0:N-1)/N;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             Want to find frequency spectrum
%             of time domain plots
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for(i=1:numscan3)
       y=fft(these(:,i),N);
       ymag=2*abs(y)/N;
       ymag(1)=ymag(1)/2;
       ymag((N/2)+1)=ymag((N/2)+1)/2;
       Mags(:,i)=ymag;     % 'Mags' is the frequency spectrum of each wave
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             Want to find harmonic components as
%             functions of distance from transducer.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for(i=1:numscan3)
       principalx=K(1:1251);
       principaly(:,i)=Mags((1:1251),i); % Want principal alias only
       fundmax(i)=max(principaly(:,i));  % Find max values of fundamental
end



for(i=1:number_point/2)
       if(principaly(i,100)==fundmax(100))
```

```
            xvalue=i;    % 'xvalue' is the array position of fundamental
      end
end

%%%%%%% Finding the second harmonics for each collected wave
lowside=(xvalue*2)-(round(xvalue/2));
highside=(xvalue*2)+(round(xvalue/2));

for(i=1:numscan3)
      junk_x=K(lowside:highside);
      junk_y=Mags((lowside:highside),i);
      secondharm(i)=max(junk_y);
end

%%%%%% Finding the third harmonic
lowside3=(xvalue*3)-(round(xvalue/2));
highside3=(xvalue*3)+(round(xvalue/2));

for(i=1:numscan3)
      junk_x=K(lowside3:highside3);
      junk_y=Mags((lowside3:highside3),i);
      thirdharm(i)=max(junk_y);
end

%%%%%%%      Normalize all harmonics to the maximum of the fundamental

fundamental=fundmax/(max(fundmax));
second=secondharm/(max(fundmax));
third=thirdharm/(max(fundmax));
figure(4)
h1=plot(x,fundamental);
hold
h2=plot(x,second);
h3=plot(x,third);
h4=gtext('1st');
h5=gtext('2nd');
h6=gtext('3rd');
set(h4,'fontsize',[13],'fontweight','bold')
set(h5,'fontsize',[13],'fontweight','bold')
set(h6,'fontsize',[13],'fontweight','bold')
set(gca,'fontsize',[13],'fontweight','bold')
xlabel('Distance from Transducer (cm)')
ylabel('Harmonic Magnitudes')
```