Appendix

This section contains the Borland C++ compiled system software, “system.cpp”

```
--------------------------------------------------------------------------
#include <dos.h>  // Header files available with C++ compiler.
#include <conio.h>
#include <bios.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#define BASE 0x300      // Base address for the Single Board Computer.
#define CSR BASE        // Control register for the A/D Converter
#define ASR BASE+2      // A/D setup register         R/W
#define DTR BASE+4      // Data transfer register     R/W
#define TLR BASE+6      // Trigger counter (0~15)     R/W
#define TMR BASE+8      // Trigger counter (16~23)    R/W
#define GR1 BASE+10     // Gain register1 (1~8)       R/W
#define GR2 BASE+12     // Gain register2 (9~16)      R/W
#define DIOR BASE+14    // Digital I/O register       R/W
#define ADR BASE+16     // Data register              RO

#define LCDCTRL 0x260+3       // LCD control register
#define LPTDATA 0x260+2       // Data register

#define INTR 0X1C             // The clock tick interrupt

#ifdef __cplusplus
            #define __CPPARGS ...
#else
            #define __CPPARGS
#endif

void initad(void);                          // System file, DO NOT MODIFY
void initlcd(void);                         // System file, DO NOT MODIFY
void writelcd(unsigned char);               // System file, DO NOT MODIFY
void writectrl(unsigned char);              // System file, DO NOT MODIFY
void gotolcd(void);                         // System file, DO NOT MODIFY
void putslcd(char *);                       // System file, DO NOT MODIFY
void clearlcd(void);                        // System file, DO NOT MODIFY
void interrupt (*oldhandler)(__CPPARGS);  // System file, DO NOT MODIFY

// Defining variables
unsigned char x,y;
unsigned int csr,asr,dtr,tlr,tmr,gr1,gr2,dior,adr;
float ch[8],c[8],initvals[8],delVx1,delVx2,delVy1,delVy2,delVz1,delVz2;
float Xden,Yden,Zden;
unsigned int count,flag,i,firstval=0;
char *string;
float xcalibpos=1,xcalibneg=1;              // ← Make changes here
float ycalibpos=1,ycalibneg=1;              // ← Make changes here
```

```
float zcalibpos=1,zcalibneg=1;            // ← Make changes here
// Step 5 of calibration procedure.

void interrupt handler(__CPPARGS)
{

// Disable interrupts during the handling of the interrupt
      disable();
// Increase the global counter
      count++                 // 18 Hz clock
      count=count%36;         // 0.5 Hz display refresh rate

      for(i=0;i<8;i++)
      {
            outpw(CSR,0x1);         // Send S/W trigger to A/D Converter
            outpw(CSR,0x3);         // to begin conversion

            do{csr=inpw(CSR);}
            while(~csr&(1<<7));     // Wait for data ready

            c[i]=c[i]+(inpw(ADR)>>4);
      }

/* ch[0] is output from laser source. ch[1] through [7] are outputs from
the sensors. The following operation is performed every 2  seconds.*/

if(count==0)
{
      for (i=0;i<8;i++) ch[i]=c[i]*10/0xffe/36;
//Convert data into voltage and average over two seconds.

            for (i=1;i<8;i++) ch[i]=ch[i]/ch[0];
//Normalize to source.

            for (i=0;i<8;i++) c[i]=0;

            if (firstval==0)
            {
                  for (i=1;i<8;i++) initvals[i]=ch[i];
                  firstval=1;
            }
/* Obtain the initial values of each sensor output for use in
differentiation. Initvals[1] through [7] are the reference values of the
sensor outputs.*/

/* Calculate the field density for X, Y and Z axes using  Quadrature Phase
shift signal demodulation scheme.*/

            delVx1= ch[2]-initvals[2];
            delVx2= ch[3]-initvals[3];
            Xden= xcalibpos*sqrt(delVx1*delVx1+delVx2*delVx2);
```

```
            delVy1= ch[4]-initvals[4];
            delVy2= ch[5]-initvals[5];
            Yden= ycalibpos*sqrt(delVy1*delVy1+delVy2*delVy2);

            delVz1= ch[6]-initvals[6];
            delVz2= ch[7]-initvals[7];
            Zden= zcalibpos*sqrt(delVz1*delVz1+delVz2*delVz2);

/* Determining direction of external field. MAKE THE NECESSARY CHANGES
BELOW THIS LINE DURING CALIBRATION OF THE SYSTEM. ONLY CHANGE NEEDED IN THE
PROGRAM. Step 4 of calibration procedure.*/

/* Apply +ve and –ve calibration field to each sensor pair. Find the sensor
which shows opposite variation in output on application of these fields . If
output of sensor ‘I’ increases above reference value on application of – ve
calibration field, change the marked command line to;
iF (ch[I] > initvals[I])
If output of sensor ‘I’ decreases below reference value on application of –
ve calibration field, change command line to ;
iF (ch[I] < initvals[I])*/

// Make changes below this line only.
            if (ch[3] < initvals[3])      // ← MAKE CHANGE HERE.
            {
                  Xden=-xcalibneg*Xden/xcalibpos;
            }

            if (ch[5] > initvals[5])      // ←MAKE CHANGE HERE.
            {
                  Yden=-ycalibneg*Yden/ycalibpos;
            }

            if (ch[7] > initvals[7])      // ←MAKE CHANGE HERE.
            {
                  Zden=-zcalibneg*Zden/zcalibpos;
            }
// Make changes above this line only.

            flag=1;     // Start main function.

      }

// Reenable interrupts at the end of the handler
            enable();
// Call the old routine
            oldhandler();
}

// System functions, DO NOT CHANGE BELOW THIS LINE.
void clearlcd(void)
{
      writectrl(0x1);
```

```
	delay(5);
}

void initlcd(void)
{
delay(500);

	writectrl(0x38);
	delay(7);
	writectrl(0x38);
	delay(1);
	writectrl(0x38);
	writectrl(0x38);
	writectrl(0x0e);
	writectrl(0x6);
	writectrl(0x1);
	delay(5);
}

void putslcd(char *str)
{
	int len;
	len=strlen(str);
	for(int i=0;i<len;i++)
	{
		writelcd(str[i]);
		x++;
	}
}

void gotolcd(void)
{
	unsigned char pos;
	if((x<1||x>20)||(y<1||y>4)){x=1;y=1;}
	if(y==1){pos=x-1;}
	if(y==2){pos=(1<<6)+x-1;}
	if(y==3){pos=x+0x13;}
	if(y==4){pos=(1<<6)+x+0x13;}

	outportb(LPTDATA,pos|0x80);
	outportb(LCDCTRL,0x00);
	outportb(LCDCTRL,0x01);
	for(int i=0;i<10;i++);
	outportb(LCDCTRL,0x00);
	for(i=0;i<100;i++);
}

void writectrl(unsigned char c)
{
	outportb(LCDCTRL,0x00);
	outportb(LPTDATA,c);
	outportb(LCDCTRL,0x01);
	for(int i=0;i<10;i++);
	outportb(LCDCTRL,0x00);
```

```
      for(i=0;i<200;i++);
}

void writelcd(unsigned char ch)
{
      gotolcd();
      outportb(LCDCTRL,0x10);
      outportb(LPTDATA,ch);
      outportb(LCDCTRL,0x11);
      for(int i=0;i<10;i++);
      outportb(LCDCTRL,0x10);
      for(i=0;i<100;i++);
}

void initad(void)
{
      outpw(CSR,1<<6);
      delay(10);
      outpw(CSR,0);           // reset board

      outpw(GR1,0);
      outpw(GR2,0);           // set gain
      outpw(TLR,0xff9c);
      outpw(TMR,0xff);

      outpw(DTR,0);
      outpw(ASR,0x70);
      delay(1);
}
// System functions, DO NOT CHANGE ABOVE THIS LINE.

// Main function

int main(void)
{
      count=0;
      initad();         // Initialize A/D Converter
      initlcd();        // Initialize LCD Controller

// Save the old interrupt vector
      oldhandler = getvect(INTR);

// install the new interrupt handler
      setvect(INTR, handler);

      for(;;)                       // Set up an infinite loop.
      {
            do{}while(flag==0);     //wait for flag to be set

            clearlcd();
            x=1;y=1;
            gotolcd();
            putslcd("X: ");
            gcvt(Xden,6,string);
```

```
            putslcd(string);

            x=1;y=2;
            gotolcd();
            putslcd("Y: ");
            gcvt(Yden,6,string);
            putslcd(string);


            x=1;y=3;
            gotolcd();
            putslcd("Z: ");
            gcvt(Zden,6,string);
            putslcd(string);
            flag=0;
      }

// Reset the old interrupt handler
      setvect(INTR, oldhandler);

      return 0;
}
```