# SAMPLE PROGRAMS

Two Sample Programs are provided on diskette with the AOB2-P. Sample Program #1 demonstrates general use of the card. This program prompts you for a voltage, calculates the closest actual voltage based on the 12-bit resolution of the DAC, and then programs the card to output this voltage. Sample Program #1 is provided in QuickBASIC, C, and Pascal.

Sample Program #2 will generate a sine, triangle, or sawtooth output waveform. This program is provided in QuickBASIC, C, and Pascal. A commented listing of the C language version is as follows:

```
/*****************************************************************
*                                SAMPLE 2.C                      *
*                                                                *
*     This sample program will generate three different waveforms;*
*     sine, triangle, and sawtooth.  You have the choice of base  *
*     address, DAC number, and the number of points per cycle.    *
*                                                                *
*     The base address entered during program execution should    *
*     correspond to that setup on the card.                       *
*                                                                *
*****************************************************************/

#include <math.h>
#include <conio.h>
#include <stdio.h>
#include <dos.h>

#define  PI 3.1415927

unsigned     counts;              /* number of points per cycle    */
unsigned     baseadrs;            /* card base address             */
unsigned     dacnum;              /* DAC used for output           */
unsigned     progstruct[20000];   /* buffer to hold points         */

/*****************************************************************
*                                                                *
*    FUNCTION: setparams() - local routine                       *
*                                                                *
*    PURPOSE: Prompts the user for DAC number, base address and the*
*    number of points per cycle.                                 *
*                                                                *
*    INPUT: None                                                 *
*                                                                *
*    CALLS: None                                                 *
*                                                                *
*    OUTPUT: None                                                *
*                                                                *
*****************************************************************/
```

```
void setparms()
{
clrscr();
printf("Enter the base address of your card (in hex)\n");
printf("(Example: 300 : ");
scanf("%x",&baseadrs);
printf("Enter the DAC number you wish to output to (0 or 1):");
scanf("%u",&dacnum);
dacnum% = 2;
printf("Enter the number of points that you wish to calculate per cycle,\n");
printf("(20000 maximum, program will use modulus if needed);");
scanf("%u",&counts);
counts%=20001;
}  /end setparms*/

/*********************************************************************
*                                                                    *
*   FUNCTION: sendtoport() - local routine                           *
*                                                                    *
*   PURPOSE: Writes point buffer to the DAC until a key is pressed   *
*                                                                    *
*   INPUT: None                                                      *
*                                                                    *
*   CALLS: None                                                      *
*                                                                    *
*   OUTPUT: None                                                     *
*                                                                    *
*********************************************************************/

void sendtoport()
{
int                 i,temp;
long                j;
unsigned char  lowbyte,hibyte;

/*Each point is broken into the high byte and low byte, and then
  written to the DAC in two separate bytes. */
do
  {
  for(i = 0; i <counts,i++)
      {
      temp = progstruct{i} % 256;
      lowbyte = (unsigned char)temp;
      temp = progstruct{i} / 256;
      hibyte = (unsigned char)temp;
outp  outportb(baseadrs+(dacnum*2),lowbyte);
      outportb(baseadrs+(dacnum*2+1),hibyte);
      }
   }
while (!kbhit());
outportb(baseadrs+(dacnum*2),0);     /*set DAC to 0 output */
outportb(baseadrs+(dacnum*2+1),0);
}           /*end sendtoport */
```

```
/*********************************************************************
*                                                                    *
*   FUNCTION: sinecurve() - local routine                            *
*                                                                    *
*   PURPOSE: Calculate the points to create a sine wave.             *
*                                                                    *
*   INPUT: None                                                      *
*                                                                    *
*   CALLS: None                                                      *
*                                                                    *
*   OUTPUT: None                                                     *
*                                                                    *
*********************************************************************/

void sinecurve()
{
int          i;
double       rads,sine;

if (counts == 0) return;              /*no point -- no curve */

clrscr();
printf("Calculating sine wave points....");

rads = (double) 2 * PI / (counts - 1);     /* rad per count */

for(i = 0;i <counts;i++)
   {
   sine = (sin(rads * i) + 1.0) * 2047;
   progstruct[i] = (unsigned) sine * 16;
   }
clrscr();
printf("Generating sine wave, press any key to stop....");
sendtoport();
}            /* end sinecurve */

/*********************************************************************
*                                                                    *
*   FUNCTION: trianglecurve() - local routine                        *
*                                                                    *
*   PURPOSE: Calculate the points to create a triangle wave.         *
*                                                                    *
*   INPUT: None.                                                     *
*                                                                    *
*   CALLS: None                                                      *
*                                                                    *
*   OUTPUT: None.                                                    *
*                                                                    *
*********************************************************************/
```

```
void trianglecurve(void)
{
int         i;
double      slope,temp;

if (counts == 0) return;       /* no counts -- no curve */

clrscr();
Printf("Calculating triangle wave points....");

slope = 4095.0 / counts * 2.0;
                               /* waveform slope */
for(i=0;i <counts/2;i++)
   {
   temp = slope * i;
   progstruct[i] = (int)temp * 16;
   temp = 4095 - temp;
   progstruct[i+counts/2+1] = (int)temp * 16;
   }
clrscr();
printf("Generating triangle wave, press any key to stop....");
sendtoport();
}           /* end triangle curve */

/*****************************************************************
*                                                                *
*   FUNCTION: sawcurve() - local routine                         *
*                                                                *
*   PURPOSE: Calculate the points to create a sawtooth wave.     *
*                                                                *
*   INPUT: None.                                                 *
*                                                                *
*   CALLS: None.                                                 *
*                                                                *
*   OUTPUT: None.                                                *
*                                                                *
*****************************************************************/

void sawcurve()
{
int         i;
double      slope,temp;

if (counts == 0) return;

clrscr();
printf("Calculating sawtooth wave points....");

slope = 4095.0 / counts; /* sawtooth slope*/

for(i = 0,i <counts;i++)
   {
   temp = slope * i;
   progstruct[i] = (int) temp;
   progstruct[i] %= 4096;
   progstruct[i] *= 16;
   }
clrscr();
printf("Generating sawtooth wave, press any key to stop....");
sendtoport();
}           /* end sawcurve */
```

```
/*********************************************************************
*                                                                    *
*   FUNCTION: menulist() - local routine                             *
*                                                                    *
*   PURPOSE: Display the menu choice on the screen.                  *
*                                                                    *
*   INPUT: None.                                                     *
*                                                                    *
*   CALLS: None.                                                     *
*                                                                    *
*   OUTPUT: None.                                                    *
*                                                                    *
*********************************************************************/

void menulist(void)
{
clrscr();
printf("\n\n\n");
printf("Your menu selections are:\n");
printf("1. Input Card Data (do this first.)\n"):
printf("2. Sine Curve\n");
printf("3. Triangle Curve\n");
printf("4. Sawtooth Curve\n");
printf("5. End Program, Return to DOS\n");
printf("Input Choice;");
}           /* end menulist */

/*********************************************************************
*                                                                    *
*   FUNCTION: main() - local routine                                 *
*                                                                    *
*   PURPOSE: Controls program execution.                             *
*                                                                    *
*   INPUT: None                                                      *
*                                                                    *
*   CALLS: None                                                      *
*                                                                    *
*   OUTPUT: None                                                     *
*                                                                    *
*********************************************************************/

void main(void)
{
char        menuchoice;

clrscr();
do
   {
   memset(progstruct, 0, sizeof(int);      /* clear buffer          */
   menulist();                             /* display the menu      */
   menuchoice=getch();                     /* fetch the menu choice   */
   switch(menuchoice)                      /* execute menu selection  */
      {
      case '1':      setparms();     /* fetch system parameters    */
                     break;
      case '2':      sinecurve();    /* generate a sine wave       */
                     break;
      case '3':      trianglecurve(); /* generate a triangle wave       */
                     break;
      case '4':      sawcurve();     /* generate a sawtooth wave   */
                     break;
      case '5':      return;         /* exit to operating system   */
      };
   }
while(1== 1);
}                    /* end main */
```