ACCES I/O PRODUCTS INC
10623 Roselle Street San Diego CA 92121-1506
Tel. (619)550-9559 FAX (619)550-7322

# ANALOG OUTPUT CARD

# D/A-02A

# USER MANUAL

file:MDA-02A.C2c.wpd

# NOTICES

The information in this document is provided for reference only. ACCES does not assume any liability arising out of the application or use of the information or products described herein. This document may contain or reference information and products protected by copyrights or patents and does not convey any license under the patent rights of ACCES, nor the rights of others.

IBM PC, PC/XT, and PC/AT are registered trademarks of the International Business Machines Corporation.

# TABLE OF CONTENTS

<< Table of Contents will generate here >>

# INSTALLATION

The software provided with this card is contained on either one CD or multiple diskettes and must be installed onto your hard disk prior to use. To do this, perform the following steps as appropriate for your software format and operating system. Substitute the appropriate drive letter for your CD-ROM or disk drive where you see D: or A: respectively in the examples below.

## CD INSTALLATION

### DOS/WIN3.x

1. Place the CD into your CD-ROM drive.
2. Type D : Enter to change the active drive to the CD-ROM drive.
3. Type I N S T A L L Enter to run the install program.
4. Follow the on-screen prompts to install the software for this card.

### WIN95/98/NT

1. Place the CD into your CD-ROM drive.
2. The CD should automatically run the install program after 30 seconds. If the install program does not run, click START | RUN and type D : I N S T A L L, click OK or press Enter.
3. Follow the on-screen prompts to install the software for this card.
4. Click the “Go to ACCES Web” button to check for software updates.

## 3.5-INCH DISKETTE INSTALLATION

As with any software package, you should make backup copies for everyday use and store your original master diskettes in a safe location. The easiest way to make a backup copy is to use the DOS DISKCOPY utility.

In a single-drive system, the command is:

D I S K C O P Y A : A : Enter

You will need to swap disks as requested by the system.
In a two-disk system, the command is:

D I S K C O P Y A : B : Enter

This will copy the contents of the master disk in drive A to the backup disk in drive B.

To copy the files on the master diskette to your hard disk, perform the following steps.

1. Place the master diskette into a floppy drive
2. Change the active drive to the drive that has the diskette installed. For example, if the diskette is in drive A, type A : Enter.
3. Type I N S T A L L Enter and follow the on-screen prompts.

## DIRECTORIES CREATED ON THE HARD DISK

The installation process will create several directories on your hard disk. If you accept the installation defaults, the following structure will exist.

[CARDNAME] Root or base directory containing the SETUP.EXE setup program used to help you configure jumpers and calibrate the card.

**DOS\PSAMPLES:** A subdirectory of [CARDNAME] that contains Pascal samples.

**DOS\CSAMPLES:** A subdirectory of [CARDNAME] that contains "C" samples.

**WIN32*language*** Subdirectories containing samples for Win95/98 and NT.

WinRisc.exe: A Windows dumb-terminal type communication program designed for RS422/485 operation. Used primarily with REMOTE ACCES Data Acquisition Pods and our RS422/485 serial communication product line. Can be used to say hello to an installed modem.

**ACCES32:** This directory contains the Windows 95/98/NT driver used to provide access to the hardware registers when writing 32-bit Windows software. Several samples are provided in a variety of languages to demonstrate how to use this driver. The DLL provides four functions (InPortB, OutPortB, InPort, and OutPort) to access the hardware.

This directory also contains the device driver for Windows NT, ACCESNT.SYS. This device driver provides register-level hardware access in Windows NT. Two methods of using the driver are available, through ACCES32.DLL (recommended) and through the DeviceIOControl handles provided by ACCESNT.SYS (slightly faster).

**SAMPLES:** Samples for using ACCES32.DLL are provided in this directory. Using this DLL not only makes the hardware programming easier (MUCH easier), but also one source file can be used for both Windows 95/98 and WindowsNT. One executable can run under both operating systems and still have full access to the

hardware registers. The DLL is used exactly like any other DLL, so it is compatible with any language capable of using 32-bit DLLs. Consult the manuals provided with your language's compiler for information on using DLLs in your specific environment.

**VBACCES:** This directory contains sixteen-bit DLL drivers for use with VisualBASIC 3.0 and Windows 3.1 only. These drivers provide four functions, similar to the ACCES32.DLL. However, this DLL is only compatible with 16-bit executables. Migration from 16-bit to 32-bit is simplified because of the similarity between VBACCES and ACCES32.

**PCI:** This directory contains PCI-bus specific programs and information. If you are not using an ACCES PCI card, this directory will not be installed.

**SOURCE**: A utility program is provided with source code you can use to determine allocated resources at run-time from your own programs in DOS.

PCIFind.exe A utility for DOS and Windows to determine what base address and IRQ are allocated to installed PCI cards. This program runs two versions, depending on the operating system. Windows 95/98/NT displays a GUI interface, and modifies the registry. When run from DOS or Windows3.x, a text interface is used. For information about the format of the registry key, consult the card-specific samples provided with the hardware. In Windows NT, NTioPCI.SYS runs each time the computer is booted, thereby refreshing the registry as PCI hardware is added or removed. In Windows 95/98/NT PCIFind.EXE places itself in the boot-sequence of the OS to refresh the registry on each power-up.

This program also provides some COM configuration when used with PCI COM ports. Specifically, it will configure compatible COM cards for IRQ sharing and multiple port issues.

**WIN32IRQ:** This directory provides a generic interface for IRQ handling in Windows 95/98/NT. Source code is provided for the driver, greatly simplifying the creation of custom drivers for specific needs. Samples are provided to demonstrate the use of the generic driver. Note that the use of IRQs in near-real-time data acquisition programs requires

considered an intermediate to advanced programming topic. Delphi, C++ Builder, and Visual C++ samples are provided.

DOS utility to determine an available base address for ISA bus , non-Plug-n-Play cards. Run this program once, before the hardware is

the card. Once the address has been determined, run the setup program provided with the hardware to see instructions on setting the

Poly.exe A generic utility to convert a table of data into an $n^{th}$ order polynomial.

thermocouples and other non-linear sensors.

A batch file demonstrating the command line parameters of

RISCTerm.exe

RS422/485 operation. Used primarily with REMOTE ACCES Data

line. Can be used to say hello to an installed modem. RISCTerm

## INSTALLING THE CARD

Before installing the card carefully read the Address Selection and Option Selection sections of this manual and configure the card according to your requirements. Be especially careful with address selection. If the addresses of two installed functions overlap you will experience unpredictable computer behavior. If unsure what locations are available, you can use the FINDBASE program provided on our CD to locate blocks of available addresses.

To install the card:

1. Turn off computer power.

2. Remove the computer cover.

3. Remove the blank I/O backplate.

4. Set switches for selected options. See manual Section 3.

5. Select the base address on the card. See manual Section 4.

6. Install the card in an I/O expansion slot. *Important: Make sure that the card mounting bracket is properly screwed into place and that there is a positive chassis ground.*

7. Install the I/O cable.

8. Inspect for proper fit of the card and cables, tighten screws.

9. Replace the computer cover and apply power.

To ensure that there is minimum susceptibility to EMI and minimum radiation, it is important that there be a positive chassis ground. Also, proper EMI cabling techniques (cable connect to chassis ground at the I/O connector, twisted-pair wiring, and, in extreme cases, ferrite level EMI protection) must be used for input/output wiring.

CE-marked versions of D/A-02A meet the requirements of EN50081-1:1992 (Emissions), EN50082-1:1992 (Immunity), and EN60950:1992 (Safety).

# FUNCTIONAL DESCRIPTION

The D/A-02A is a half-size card that can be installed in short I/O slots of PC/XT/AT class computers. It contains two digital-to-analog converters (DAC) and provides two independent analog output channels of 12-bit resolution. Each analog output channel can be configured for ranges of:

0V to +5V
0V to +10V
-5V to +5V
-10V to +10V
4mA to 20mA

Voltage ranges may be setup one of two ways; either by settings on two on-board jumpers or by jumpers at the output connector. The Option Selection section of this manual contains a description of how to make these selections.

Both analog output channels have a double-buffered input for single-step update and each is addressed at its own I/O location. The DACs have a two-byte (4LSB's+8MSB's) loading structure. The card is designed for left-justified data format. The analog outputs can be updated either independently or simultaneously.

Finally, D/A-02A contains automatic reset circuits which reset both D/A inputs to all zeroes at system power-on. On D/A channels set up for unipolar ranges, this automatic reset results in 0V output and on D/A channels set up for 4-20mA output, it results in 4mA output. On any D/A channels set up for bipolar ranges, this results in + Full Scale output of either +5VDC or +10VDC depending on the range selected.

Software provided with D/A-02A includes setup and calibration programs and two sample programs. The setup and calibration program provides pictorial representation and menu selection on the computer monitor. For setup, of course, it is not necessary that the card be plugged into the computer. Sample Program #1 prompts for a user-desired output voltage and then programs the board to output this voltage. Sample Program #2 will generate sine, triangle, or sawtooth waveforms.

# D/A-02A BLOCK DIAGRAM

# OPTION SELECTION

output ranges determined either jumper installation the output
or jumper placement s described in the following paragraphs. Also, the method of updating D/A is programmable described on next page in the Programming section of this manual.

A and easy to select voltage ranges unipolar or is to
jumpers (D/A channel and S2 channel #1). These jumpers are located at the

Placing jumpers in " 5V" selects the to 5V output at 24 and
18 respectively, the -5V to +5V range output at pins 23 and 17 respectively, and the

Placing and S2 the " position selects 0 to range output pins 24
18 respectively and the -10V to +10V range output at pins 23 and 17 respectively.

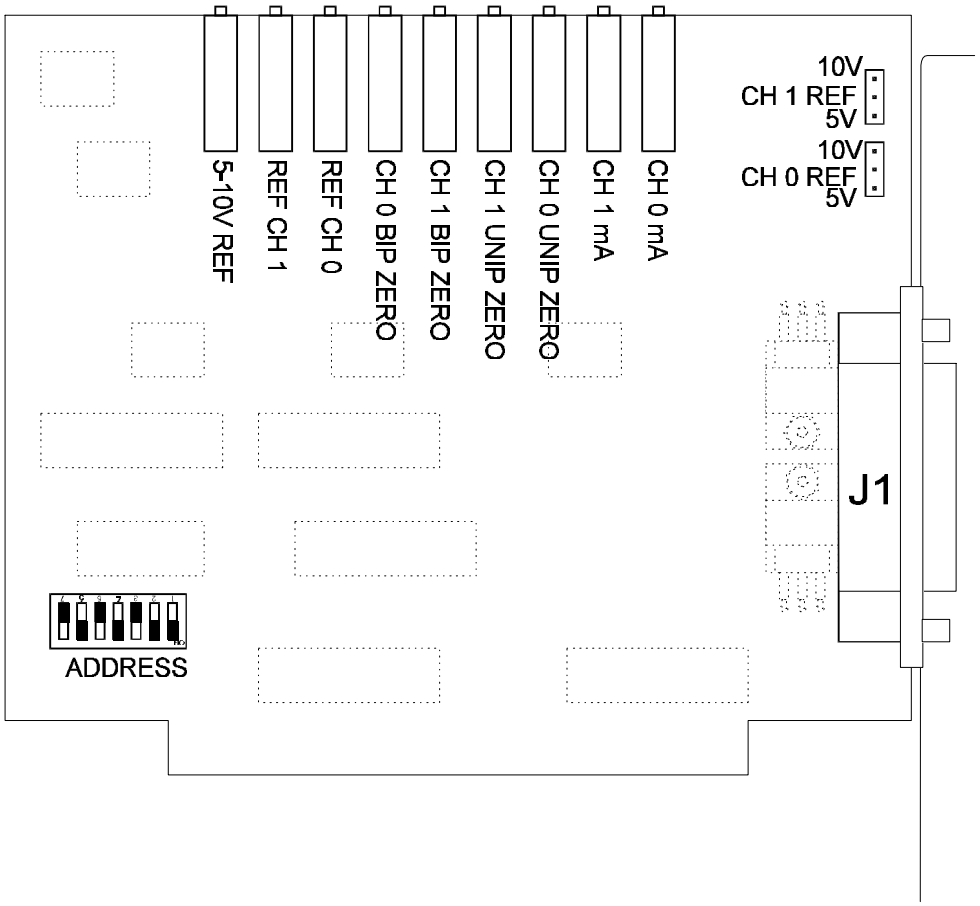
(NOTE: If select "none" jumper selection the setup you are to

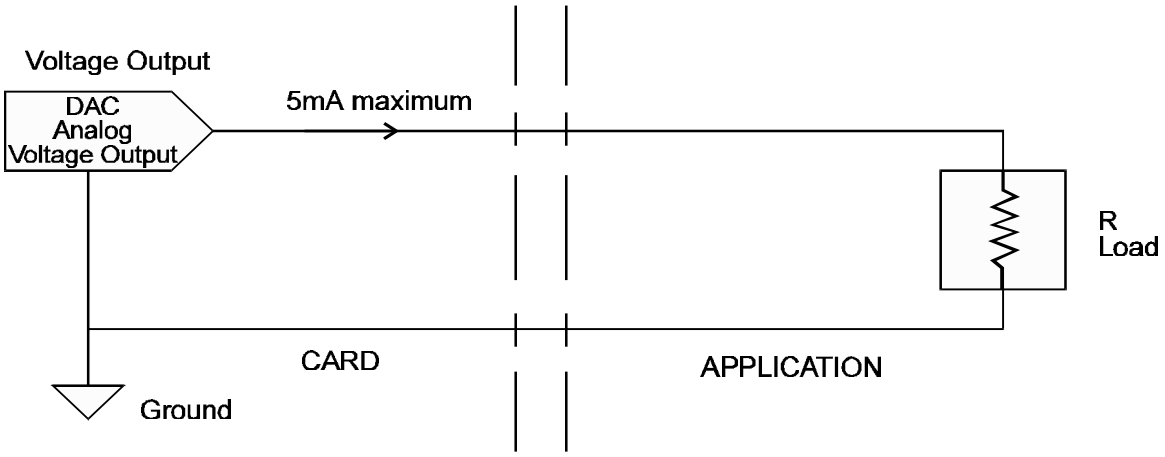
An way to output
jumpers pins on mating half the output connector. The various ranges

| | JUMPER BETWEEN PINS | OUTPUT PIN |
|---|---|---|
| D/A #1 | 21 and 22 | 24<br>18 |
| D/A #1 | 20 and 22 | 24<br>18 |
| D/A #1 | 21 and 22 | 23*<br>17* |
| D/A #1 | 20 and 22 | 23*<br>17* |
| D/A #1 | 21 and 22 | 25<br>19 |

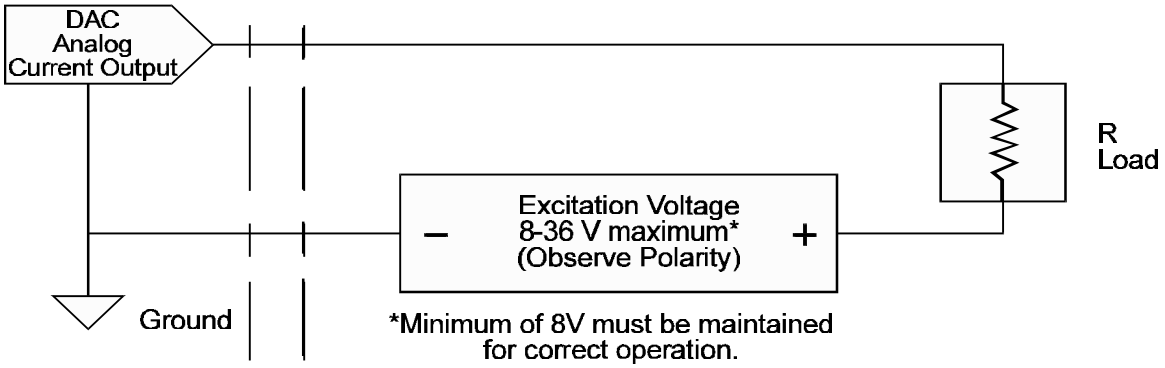
Due to analog inversion the bipolar range, data is complementary binary.
zero digital corresponds to +full scale analog and 4095 digital corresponds to -full scale analog.

## OPTION SELECTION MAP






Current Output (Jumpers placed in "5V" Position)




Minimum voltage of 8V must be maintained for correct operation.

**CAUTION**

Do not connect current loops in a DAC that is set to voltage mode. The loop supply can destroy the DAC.

## ANALOG OUTPUTS UPDATE

Analog outputs may be updated under program control in either of two ways:

(a) Each D/A output is updated when new data are written to its related high byte base address.

(b) The outputs of both D/A's may be updated simultaneously. This is done by first writing the low and high bytes for D/A 0 to base address +4 and base address +5 respectively and the low byte for D/A 1 to base address +6. Then, when the high byte for D/A 1 is written to base address +7, both D/A outputs are updated.

Refer to the Programming section of this manual for more detail.

# ADDRESS SELECTION

The D/A-02A card requires eight consecutive address locations in I/O space. The starting, or base address, can be selected anywhere within an I/O address range 100-3FF hex (except 1F0 through 1F8) in AT's and 200- 3FF in XT's, providing that the address does not overlap with other functions. If in doubt refer to Table 1 below for a list of standard address assignments. The Base Address Locator program FINDBASE provided by ACCES will assist you in selecting a base address that will avoid this conflict.

**STANDARD ADDRESS ASSIGNMENTS FOR 286/386/486 COMPUTERS**

| Hex Range | Usage |
|---|---|
| 000-01F | DMA Controller 1 |
| 020-03F | INT Controller 1, Master |
| 040-05F | Timer |
| 060-06F | 8042 (Keyboard) |
| 070-07F | Real Time Clock, NMI Mask |
| 080-09F | DMA Page Register |
| 0A0-0BF | INT Controller 2 |
| 0C0-0DF | DMA Controller 2 |
| 0F0 | Clear Math Coprocessor Busy |
| 0F1 | Reset Coprocessor |
| 0F8-0FF | Arithmetic Processor |
| 1F0-1F8 | Fixed Disk |
| 200-207 | Game I/O |
| 278-27F | Parallel Printer Port 2 |
| 2F8-2FF | Asynchronous Comm'n (Secondary) |
| 300-31F | Prototype Card |
| 360-36F | Reserved |
| 378-37F | Parallel Printer Port 1 |
| 380-38F | SDLC or Binary Synchronous Comm'n 2 |
| 3A0-3AF | Binary Synchronous Comm'n 1 |
| 3B0-3BF | Monochrome Display/Printer |
| 3C0-3CE | Local Area Network |
| 3D0-3DF | Color/Graphic Monitor |
| 3F0-3F7 | Floppy Diskette Controller |
| 3F8-3FF | Asynchronous Comm'n (Primary) |

The D/A-02A base address bits A3 through A9 are set by DIP switch S1. The setup program provided with your card includes an interactive base-address selection program. The computer monitor presents a pictorial display of the DIP switch and, when you enter your desired hex base address, the display changes to show proper switch settings for that address.

To understand how this works, consider the following. In order to select the base address, convert the desired address to binary form. Then *for each "1" of binary address set the corresponding DIP switch to OFF, and for each "0" of binary address set the corresponding switch to ON.*

Here's an example showing how to program the base address to hex 300:

1. Convert hex 300 to binary
   300 (hex) = 11 0000 0000 (binary)

2. Set the Address Selection DIP Switches

   The D/A-02A card occupies eight bytes of I/O address space. Address lines A3 through A9 are used to select the base address via DIP switches marked with the same names. Address lines A0, A1, and A2 are used to address registers at the digital-to-analog converters and there are no DIP switches for these three lines.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Address | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Switch | A9 | A8 | A7 | A6 | A5 | A4 | A3 | None | | |
| Setting | OFF | OFF | ON | ON | ON | ON | ON | None | | |

# PROGRAMMING

The D/A-02A card uses eight consecutive I/O addresses. The I/O address map is as follows:

| Address | Write | Comment |
|---|---|---|
| Base Address | AO 0 Low Byte | |
| Base Address +1 | AO 0 High Byte | Updates D/A #0 Output |
| Base Address +2 | AO 1 Low Byte | |
| Base Address +3 | AO 1 High Byte | Updates D/A #1 Output |
| Base Address +4 | AO 0 Low Byte | Data to Buffer Only |
| Base Address +5 | AO 0 High Byte | Data to Buffer Only |
| Base Address +6 | AO 1 Low Byte | Data to Buffer Only |
| Base Address +7 | AO 1 High Byte | Updates Both D/A Outputs |

Note that, if you wish to update both D/A's simultaneously, you can do so by using Base Address +4 through Base Address +7.

Data are written to the D/A in left-justified, binary format.

DATA FORMAT

| BIT | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| Low Byte | B3 | B2 | B1 | B0 | x | x | x | x |
| High Byte | B11 | B10 | B9 | B8 | B7 | B6 | B5 | B4 |

**For UNIPOLAR ranges:** For Unipolar ranges, data are in true binary form.

```
                0000 0000 0000 =   ZERO
                1000 0000 0000 =   1/2 SCALE
                1111 1111 1111 =   FULL SCALE
                |                  |
 MSB or B1<----|                  |----> B12 or LSB
```

**For BIPOLAR ranges:** For Bipolar ranges, data are in complementary offset binary form.

```
                0000 0000 0000 = + FULL SCALE
                1000 0000 0000 =   ZERO
                1111 1111 1111 = - FULL SCALE
                 |                 |
MSB or B1  <----|                 |----> B12 or LSB
```

# SOFTWARE

The D/A-02A card is straightforward to program. For example the following C procedure might be used:

To output an analog value with 12-bit resolution, a corresponding decimal number N between 0 and 4095 is calculated (2^12 = 4095).

N/4095 = V(out)/V(full scale)

Then the number is left-justified by multiplying by 16.

B = N * 16;

Next the data are written to the selected analog output channel. (See I/O Address Map.) In this example, we will assume analog output 0 (AO 0).

OutPort (BASE + 0, B);

For simplicity, it was assumed that the simultaneous-update capability was not used.

This is demonstrated in Sample Program #1 on the CD provided with your D/A-02A card. That program is provided in Interpreted BASIC, QuickBASIC, C, and Pascal.

# CALIBRATION

Periodic calibration of the D/A-02A card is recommended if it is used in extreme environmental conditions. The card uses very stable components but vibration, or high-low temperature cycles might result in slight analog output errors.

Factory calibration and periodic calibration of the card includes adjustment of the internal reference voltage unless you are using an external reference voltage.

The suggested sequence for calibration is:

a. Adjust internal reference.
b. Adjust Channel #0 reference voltage.
c. Adjust Channel #1 reference voltage
d. Adjust Unipolar range of Channel #0
e. Adjust Unipolar range of Channel #1
d. Adjust Bipolar range of Channel #0
e. Adjust Bipolar range of Channel #1
f. Adjust Current Output range of Channel #0
g. Adjust Current Output range of Channel #1

NOTE
The current output ranges
apply only for 5V excitation

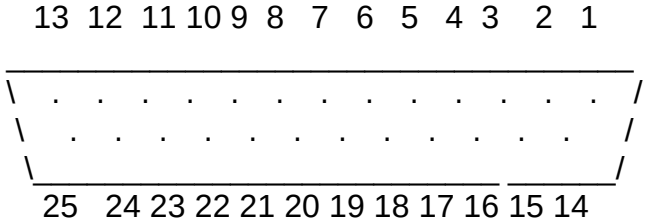
To calibrate the card, run the setup program and follow the screen prompts. No attempt at calibration should be made in noisy locations or with a noisy calibration setup.

# CONNECTOR PIN ASSIGNMENTS

The analog outputs are accessible via a female 25-pin D type connector that extends through the back of the computer case and a DB25P solder cup plug may be used to make connections. Usually only three or four wires are required (D/A outputs and ground) and multiwire flat cable is not necessary. Output range selection is performed by either jumpering pins on the plug body or by on-board switch selection as described in the Option Selection section of this manual. Pin assignments are as follows:

| Pin | Name | Function |
|---|---|---|
| 1 | D.GND | Digital Ground |
| 2 | A.GND | Analog Ground |
| 3 | A.GND | " " |
| 4 | A.GND | " " |
| 5 | A.GND | " " |
| 6 | A.GND | " " |
| 7 | A.GND | " " |
| 8 | A.GND | " " |
| 9 | A.GND | " " |
| 10 | A.GND | " " |
| 11 | A.GND | " " |
| 12 | A.GND | " " |
| 13 | +5V | +5V Power from Computer |
| 14 | -10V REF | -10V Reference for D/A's |
| 15 | -5V REF | -5V Reference for D/A's |
| 16 | REF IN | D/A #1 Reference Voltage Input |
| 17 | BIP OUT | Bipolar Analog Output, Channel 1 |
| 18 | UNIP OUT | Unipolar Analog Output, Channel1 |
| 19 | 4-20 MA | Current Output, Channel 1 |
| 20 | -10V REF | -10V Reference for D/A's |
| 21 | -5V REF | -5V Reference for D/A's |
| 22 | REF IN | D/A #0 Reference Voltage Input |
| 23 | BIP OUT | Bipolar Analog Output, Channel 0 |
| 24 | UNIP OUT | Unipolar Analog Output, Channel 0 |
| 25 | 4-20 MA | Current Output, Channel 0 |

NOTE: The figure below shows how pins are numbered on D type connectors.



# SPECIFICATION

## ANALOG OUTPUTS

Resolution: 12 bits (0 to 4095 decimal).
Channels: Two.
Voltage Output Ranges at 5mA max.
0.0 to 5.0 VDC.
0.0 to 10.0 VDC.
-5.0 to +5.0 VDC.
-10.0 to +10.0 VDC
Current Output Range(with excitation voltage 8-36 VDC).
4 to 20 mA - 1 to 5 mA - 0 to 20 mA
Digital-to-Analog Converter:
AD-7548 monolithic chip, double buffered.
Relative Accuracy: +/-1 LSB (includes nonlinearity).
Monotonicity: Guaranteed over operating temperature range.
Settling Time: 4 usec to 0.01% for full-scale step input.
Offset Temperature Drift:
+/-1 ppm/$^{o}$C. typical.
+/-3 ppm/$^{o}$C. maximum.
Gain Temperature Drift:
+/-25 ppm/$^{o}$C. (with reference)
+/-5 ppm/$^{o}$C. (w/ext. ref)
Reference Voltage Input Range: +/-10V (2 or 4 quadrant)
Reference Input Resistance: 7 Kohm min., 11 Kohm typ, 20 Kohm max.
Data Format: Left-justified, two bytes (4 LSB's, then 8 MSB's).

## POWER REQUIREMENTS:

+5 VDC at 75 mA typical, 150 mA max.
+12 VDC at 15 mA typical, 25 mA max.
-12 VDC at 25 mA typical, 35 mA max.

## ENVIRONMENTAL:

Operating Temperature Range: 0 to +60 $^{o}$C.
Storage Temperature Range: -25 to +85 $^{o}$C.
Humidity: 5% to 95% non-condensing.
Weight: 4 oz.

**SIZE:** 5.0" long (127 mm). Can be used in half- or full-size slot.

# WARRANTY

Prior to shipment, ACCES equipment is thoroughly inspected and tested to applicable specifications. However, should equipment failure occur, ACCES assures its customers that prompt service and support will be available. All equipment originally manufactured by ACCES which is found to be defective will be repaired or replaced subject to the following considerations.

## TERMS AND CONDITIONS

If a unit is suspected of failure, contact ACCES' Customer Service department. Be prepared to give the unit model number, serial number, and a description of the failure symptom(s). We may suggest some simple tests to confirm the failure. We will assign a Return Material Authorization (RMA) number which must appear on the outer label of the return package. All units/components should be properly packed for handling and returned with freight prepaid to the ACCES designated Service Center, and will be returned to the customer's/user's site freight prepaid and invoiced.

## COVERAGE

First Three Years: Returned unit/part will be repaired and/or replaced at ACCES option with no charge for labor or parts not excluded by warranty. Warranty commences with equipment shipment.

Following Years: Throughout your equipment's lifetime, ACCES stands ready to provide on-site or in-plant service at reasonable rates similar to those of other manufacturers in the industry.

## EQUIPMENT NOT MANUFACTURED BY ACCES

Equipment provided but not manufactured by ACCES is warranted and will be repaired according to the terms and conditions of the respective equipment manufacturer's warranty.

## GENERAL

Under this Warranty, liability of ACCES is limited to replacing, repairing or issuing credit (at ACCES discretion) for any products which are proved to be defective during the warranty period. In no case is ACCES liable for consequential or special damage arriving from use or misuse of our product. The customer is responsible for all charges caused by modifications or additions to ACCES equipment not approved in writing by ACCES or, if in ACCES opinion the equipment has been subjected to abnormal use. "Abnormal use" for purposes of this warranty is defined as any use to which the equipment is exposed other than that use specified or intended as evidenced by purchase or sales representation. Other than the above, no other warranty, expressed or implied, shall apply to any and all such equipment furnished or sold by ACCES.

# APPENDIX A
# SAMPLE PROGRAMS

Two Sample Programs are provided with the D/A-02A. 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 set up 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 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;
unsigned       data;
long          j;

do
  {
  for(i = 0; i <counts,i++)
    {
     data = progstruct{i} * 16;
     outport(baseadrs+(dacnum*2),data);
     }
  }
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 trinagle 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 */
```